Документация TMS DoQA

Тест-кейсы, чек-листы, прогоны, автотесты и отчётность — в одном месте.

Актуально для	DoQA 4.3 Titanium от 17.09.2026
Дата сборки	19 октября 2026 г.

docs.doqa.app

Оглавление

Начало работы

- Руководство пользователя ... 1
- Быстрый старт ... 4
- Проекты ... 6
- Пространства ... 13

Тестовая документация

- Тест-кейсы ... 17
- Чек-листы ... 27
- Общие шаги ... 34
- Параметризация тестов ... 39
- Атрибуты ... 49
- Теги ... 53
- AI-функции ... 60
- История редактирования ... 71

Требования и покрытие

- Требования ... 75
- Коннектор требований ... 82
- Матрица покрытия ... 89

Выполнение тестов

- Прогоны ... 93
- Баг-репорты ... 126
- История запуска ... 132

Автоматизация

- Автотесты ... 134
- Скачать CLI ... 140
- Отправка результатов ... 141
- Адаптеры ... 153
  - Адаптер JUnit 5 ... 157
  - Адаптер JUnit 4 ... 169
  - Адаптер TestNG ... 180
  - Адаптер pytest ... 194
  - Переезд с Allure ... 204
- Плагины для IDE ... 207
  - Работа с тестами в IntelliJ ... 216
  - Помощь в редакторе и черновики тестов ... 246
  - Плагин для VS Code ... 259
- Утилита doqactl ... 293
- Autotest API ... 310
- Каталог автотестов ... 318
- Подключение CI-систем ... 331
- Запуск автотестов из DoQA ... 339

Автоматика запусков 352
Результаты и разбор падений 358
Нестабильные тесты и карантин 367
Публичный API 377
Сценарии
Что такое сценарии 382
Первый прогон автотестов 383
Разбор упавших автотестов 388
От требования до релиза 393
Корпоративный вход (SSO) 398
Организация и поиск
Папки 403
Фильтры 406
Глобальный поиск 410
Корзина 414
Аналитика и обмен данными
Дашборд 417
Экспорт 421
Импорт 424
Совместная работа
Комментарии 431
Уведомления 439
Справочник
Горячие клавиши 443
Глоссарий 447
Администрирование
Настройки администратора 451
Как устроены права 455
Создание проекта 458
Владелец организации 460
Люди
Пользователи 462
Добавление пользователей 469
Редактирование пользователей 470
Сброс пароля 472
Группы 473
Сервисные аккаунты 481
API-токены 484
Роли и доступ
Роли и права 487
Справочник прав 494
Доступ на всю организацию 509
Доступ к проектам и пространствам 511
Доступ к пространству 515

Закрытые проекты и пространства 517
Запреты 519
Запросы доступа 521
Пересмотр доступа 523
Проверка доступа 526
Мой доступ 528
Безопасность
Журнал аудита 530
Вход и SSO 532
Лицензии
Лицензии и оплата (облако) 534
Лицензии (серверная версия) 545
Настройка проекта
Атрибуты 548
Интеграции
Интеграции с трекерами 552
Jira Cloud 556
Jira Server 558
Yandex Tracker 560
YouTrack 564
Redmine 568
Bitrix24 569
Kaiten 571
GitLab 574
Интеграции с CI/CD 576
DoQA AI
Настройки DoQA AI 580
Настройка прокси для LLM 589
DoQA сервер
Серверная версия 593
Требования 594
Утилита управления сервером 597
Установка (install) 599
Запуск (start) 602
Остановка (stop) 605
Обновление (update) 606
Резервное копирование (backup) 610
Восстановление (restore) 611
Смена домена (domain) 613
SSL-сертификат (cert) 615
Параметры среды (.env) 618
Настройка сервера почты 626
Авторизация через LDAP 629
Корпоративный вход (SSO) 633

Переход на новую систему прав 640
Релизы DoQA
Облачная версия 645
Серверная версия 678

Руководство пользователя

Здесь описано, как работать с DoQA каждый день.

Порядок работы обычно такой: в **проекте** создаётся **пространство**, в нём заводятся **тест-кейсы** и **чек-листы**, из них собирается **прогон**, по итогам прогона появляются результаты и **баг-репорты**. Результаты автотестов попадают в те же прогоны.

Если вы работаете с DoQA впервые, начните с **Быстрого старта**: это путь от пустого проекта до первого отчёта. Незнакомый термин можно найти в **Глоссарии**.

С чего начать по вашей роли

- **Ручной тестировщик** — тест-кейсы, чек-листы, общие шаги, прогоны, баги.
- **Автоматизатор (AQA)** — автотесты, параметризация, публичный API, история запуска.
- **Тест-менеджер** — проекты, пространства, требования, матрица покрытия, фильтры.
- **Аналитик** — дашборд, экспорт, импорт, требования.

Начало работы

- **Проекты** — верхний уровень: участники, права, атрибуты.
- **Пространства** — рабочая область внутри проекта и карта её разделов.

Тестовая документация

- **Тест-кейсы** — редактор, шаги, поля, таблица.
- **Чек-листы** — когда нужен список проверок, а не подробный сценарий.
- **Общие шаги** — повторяющиеся шаги, которые правятся в одном месте.
- **Параметризация тестов** — один тест на набор значений вместо десятка копий.
- **Атрибуты** — свои поля для тестовой документации проекта.
- **Теги** — метки для поиска и фильтрации.

- **AI-функции** — генерация и анализ тестовой документации.
- **История редактирования** — кто и что менял.

Требования и покрытие

- **Требования** — требования из трекера и их связь с тестами.
- **Подключение трекера требований** — коннектор, вебхуки, синхронизация.
- **Матрица покрытия** — что покрыто тестами, а что нет.

Выполнение тестов

- **Прогоны** — создание, прохождение, отчёты.
- **Баги** — баг-репорты и их связь с трекерами.
- **История запуска** — как тест проходили раньше.

Автоматизация

- **Автотесты** — как результаты автотестов попадают в DoQA и хранятся в прогонах.
- **Публичный API** — Swagger и работа с API.

Организация и поиск

- **Папки** — структура хранения тестов.
- **Фильтры** — поиск по параметрам.
- **Корзина** — что можно восстановить после удаления.

Аналитика и обмен данными

- **Дашборд** — метрики пространства.
- **Экспорт** — выгрузка данных.
- **Импорт** — перенос тестов в DoQA.

Совместная работа

- **Комментарии** — обсуждение тестов и упоминания.
- **Уведомления** — о чём и как DoQA сообщает.

Справочник

- Горячие клавиши

Быстрый старт

Пять шагов, чтобы пройти путь от пустого проекта до первого отчёта. Каждый шаг ведёт на страницу с подробностями. Сюда можно возвращаться как к оглавлению.

Шаг 1. Создайте проект и пространство

Проект — верхний уровень: к нему относятся участники, права и атрибуты. Внутри проекта создаётся пространство — рабочая область, где лежит вся тестовая документация.

- Создание проекта
- Пространства

> Готово, когда: в проекте есть хотя бы одно пространство, и вы видите его вкладки: «Кейсы», «Чек-листы», «Прогоны».

Шаг 2. Пригласите команду

Пригласите коллег и раздайте роли: кто работает с документацией, кто только смотрит, кто управляет проектом и доступом.

- Добавление пользователей
- Права доступа к проекту

> Готово, когда: приглашённые видят пространство с нужным уровнем доступа.

Шаг 3. Наполните библиотеку тестов

Заведите тест-кейсы и чек-листы. Повторяющиеся шаги вынесите в общие, однотипные тесты не копируйте, а параметризуйте. Кейсы можно сгенерировать с помощью AI.

- Тест-кейсы · Чек-листы
- Общие шаги · Параметризация тестов

> Готово, когда: в пространстве есть тесты, готовые к прохождению.

Шаг 4. Соберите прогон и пройдите тесты

Объедините кейсы и чек-листы в прогон, пройдите их и заведите баг-репорты по найденным дефектам. Результаты автотестов приходят в эти же прогоны.

- Прогоны · Баги
- Автотесты

> Готово, когда: прогон завершён, а по проблемам заведены баг-репорты.

Шаг 5. Посмотрите результаты

Оцените итоги на дашборде, при необходимости выгрузите данные.

- Дашборд · Экспорт

> Готово, когда: вы видите метрики прогона и понимаете состояние тестирования.

Смотрите также

- **Глоссарий** — если встретился незнакомый термин
- **Требования и покрытие** — как связать тесты с требованиями из трекера
- **Горячие клавиши** — чтобы работать быстрее

Проекты

Проект — верхний уровень в DoQA: он содержит **пространства**, задаёт права доступа и служит границей для **атрибутов** и интеграций. Страница «Проекты» открывается при входе. С неё вы попадаете в любое пространство.

Страница «Проекты»

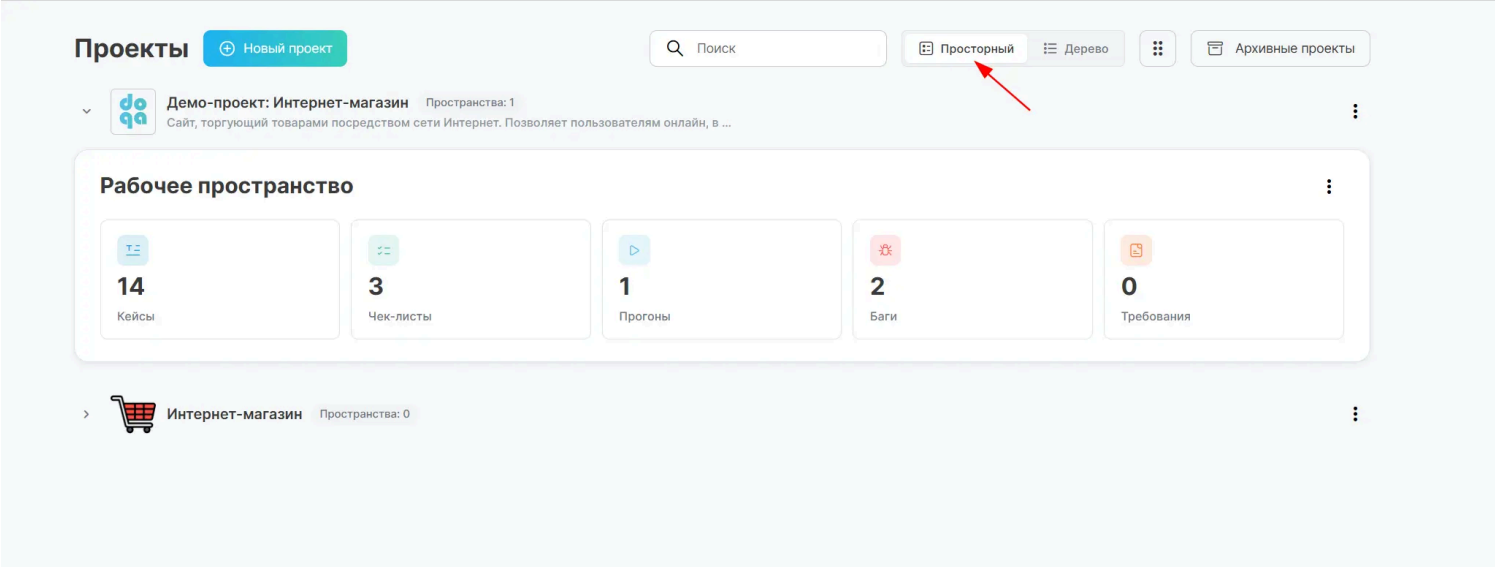
На странице показаны проекты, к которым у вас есть доступ, и пространства внутри них. Клик по названию пространства открывает его **тест-кейсы**. Название проекта — обычный текст: отдельной страницы проекта со статистикой в DoQA нет.

Режимы отображения

Переключатель в шапке страницы задаёт вид списка:

- «Просторный» — каждое пространство показано карточкой с плитками пяти сущностей: «Кейсы», «Чек-листы», «Прогоны», «Баги», «Требования». На плитке видно количество, клик по ней открывает соответствующий раздел пространства.
- «Дерево» — компактный список: проект, под ним пространства со счётчиками. Если пространств больше четырёх, остальные скрыты за кнопкой «Показать все».

Выбранный режим DoQA хранит в профиле пользователя на сервере, поэтому он переносится между браузерами.



Поиск

Строка поиска в шапке ищет одновременно по названиям и описаниям проектов и по названиям и описаниям пространств. Поиск нечёткий: опечатка в запросе не мешает найти нужное. Совпадения подсвечиваются в результатах. Если ничего не нашлось, DoQA показывает «Проектов с таким названием найдено не было».

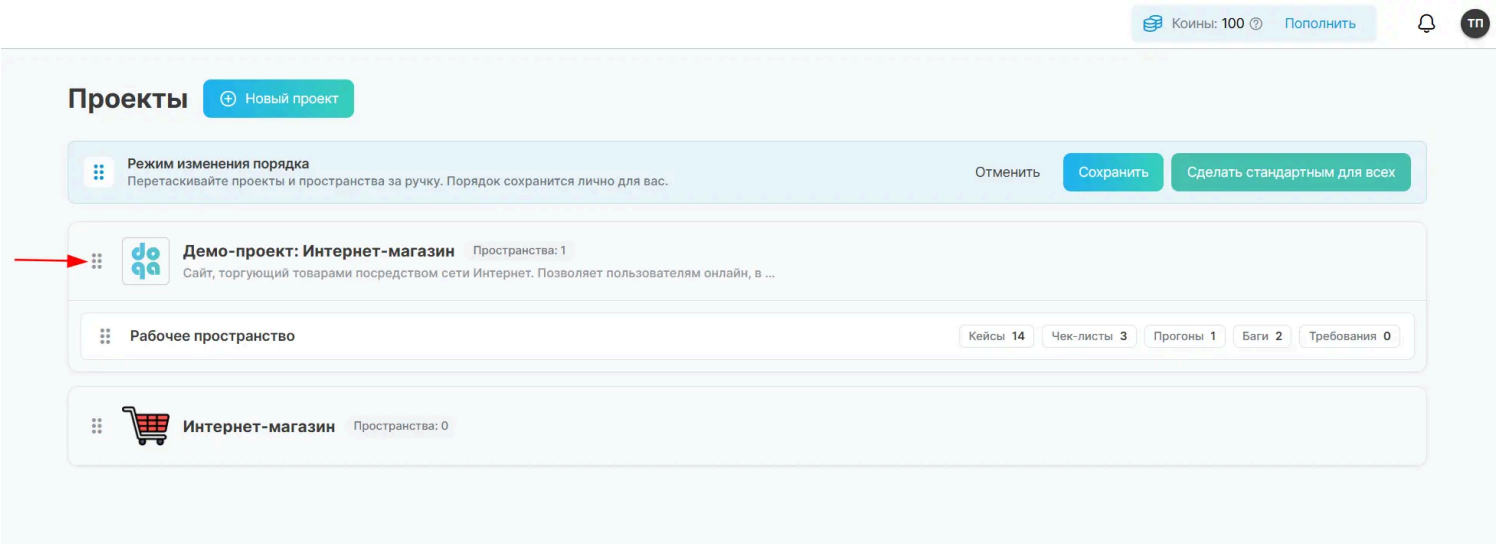
Порядок проектов и пространств

Ручную сортировку включает кнопка с иконкой из шести точек: она стоит в шапке между переключателем режима и кнопкой «**Архивные проекты**». Подписи у неё нет, во всплывающей подсказке — «Изменить порядок». Кнопка появляется, только если проектов больше одного или хотя бы в одном проекте больше одного пространства, и не работает, пока в строке поиска что-то введено.

1. Нажмите кнопку с шестью точками. Поиск сбросится, над списком появится панель «Режим изменения порядка», а у проектов и пространств — ручки перетаскивания (та же иконка из шести точек).
2. Перетащите проект или пространство за ручку. Пространство переставляется только внутри своего проекта.
3. Нажмите «**Сохранить**».

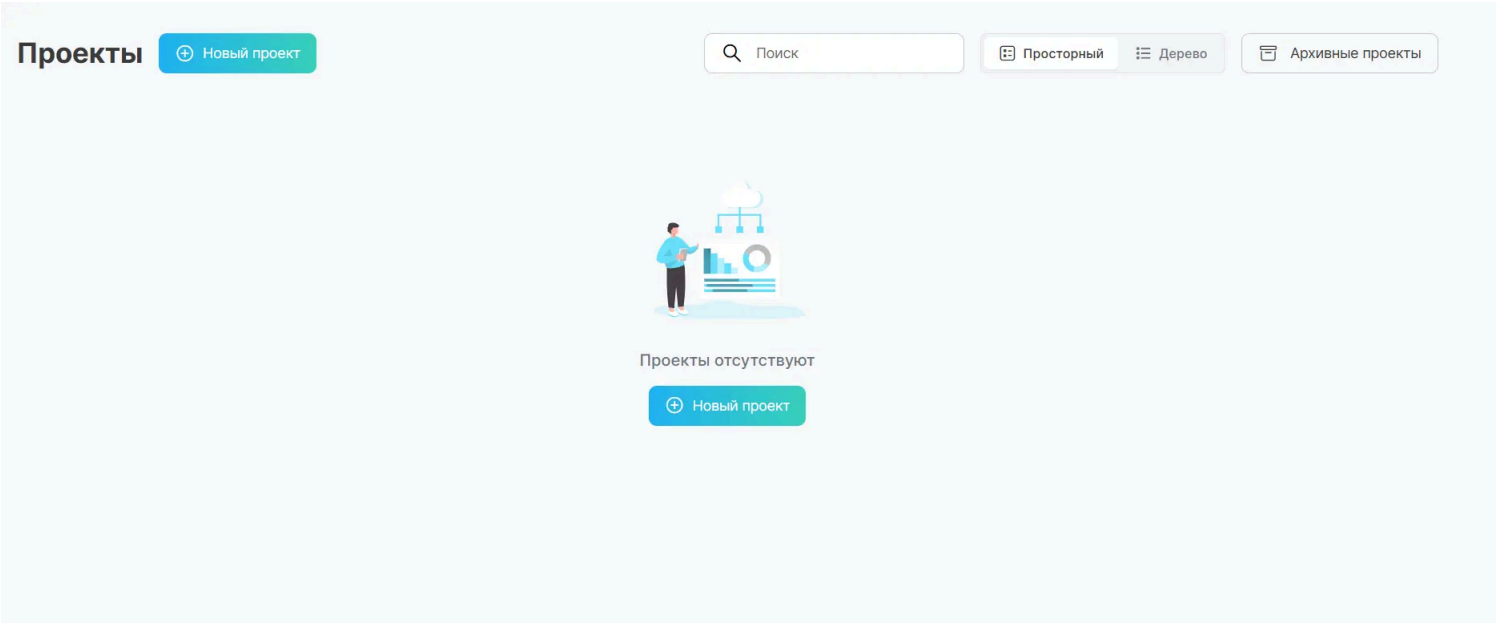
Сохранённый так порядок действует только для вас. Других пользователей он не затрагивает. В панели режима есть ещё две кнопки:

- «**Сделать стандартным для всех**» - сохраняет текущий порядок как порядок по умолчанию для всех пользователей. Кнопку видят только те, у кого есть право изменять проекты на всю организацию, например с ролью «Администратор проектов» на **всей организации**.
- «**Сбросить настройки**» — возвращает вас к порядку по умолчанию. Появляется, только если личный порядок уже сохранён.



Создание проекта

Нажмите кнопку «**Новый проект**». Она доступна тем, у кого есть право создавать проекты (роли «Создатель проектов» и «Администратор организации»).



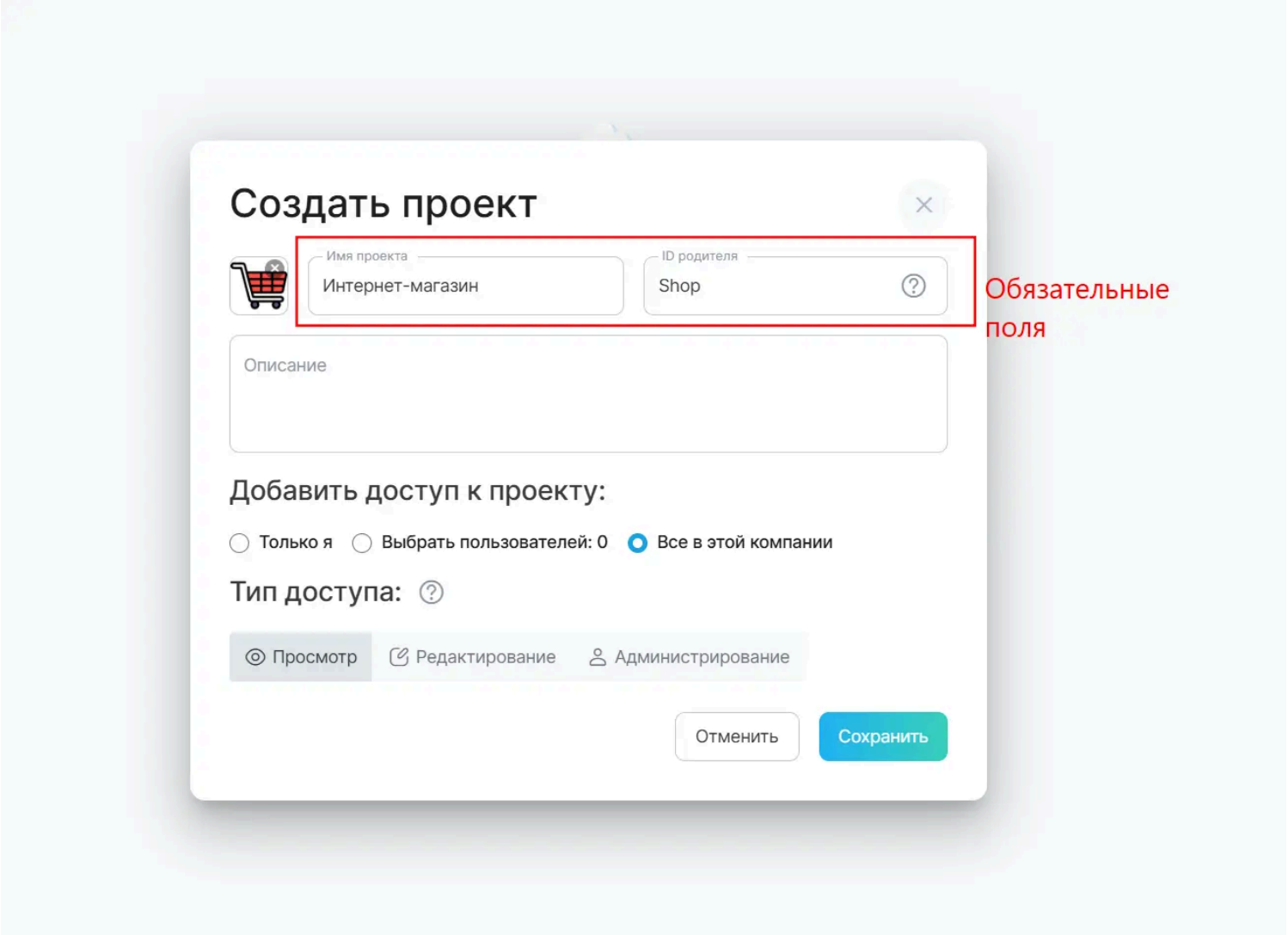
Поля проекта

- «Имя проекта» — обязательное, от 1 до 100 символов. Сочетание «//» недопустимо.
- «ID родителя» — обязательное, от 3 до 4 символов, только латинские буквы. Значение должно быть уникальным среди всех проектов. Подсказка у поля называет его префиксом.
- «Описание» — до 500 символов.

- Аватар — изображение до 10 МБ (JPEG, PNG, GIF, SVG, WEBP). Нажмите на область аватара, чтобы выбрать файл, или перетащите файл на неё.

Внимание

«ID родителя» задаётся только при создании: в форме редактирования проекта этого поля нет.



Участники проекта

После создания проекта открывается второй шаг «Участники проекта»: панель выдачи доступа. Выберите людей, группы или сервисные аккаунты и роль, которую они получат на этом проекте. Шаг показывается, если у вас есть право управлять доступом на новом проекте (по умолчанию создатель проекта получает на нём роль «Тест-лид»). Шаг можно пропустить и выдать доступ позже на вкладке «Доступ» проекта в администрировании.

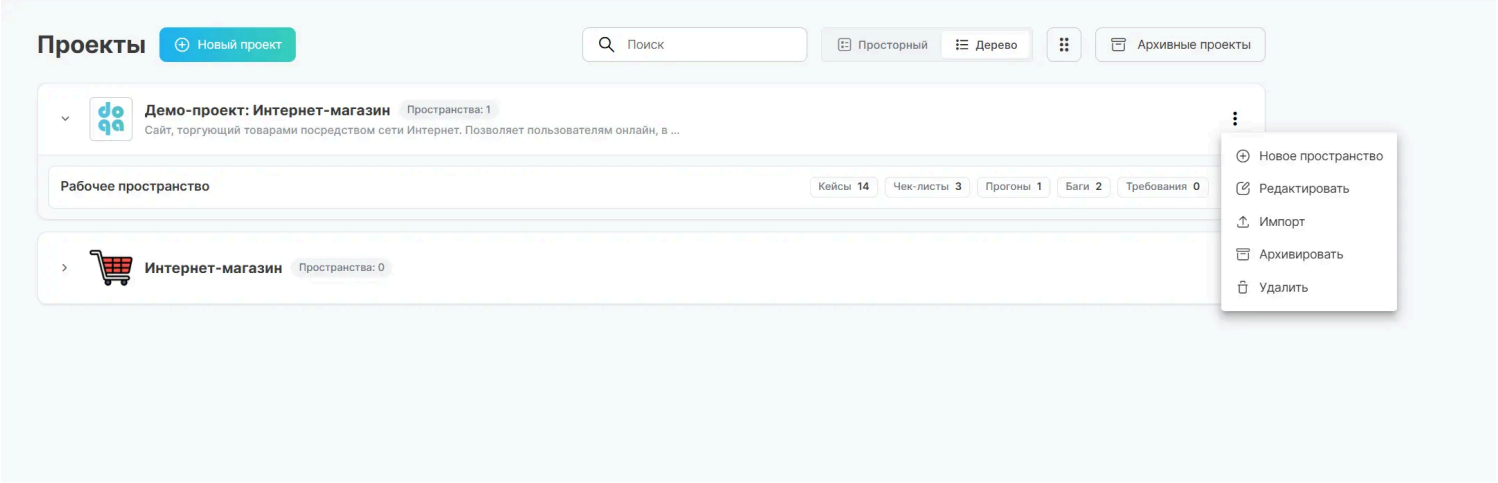
Что даёт каждая роль, описано в разделах **Роли и права** и **Доступ к проектам и пространствам**.

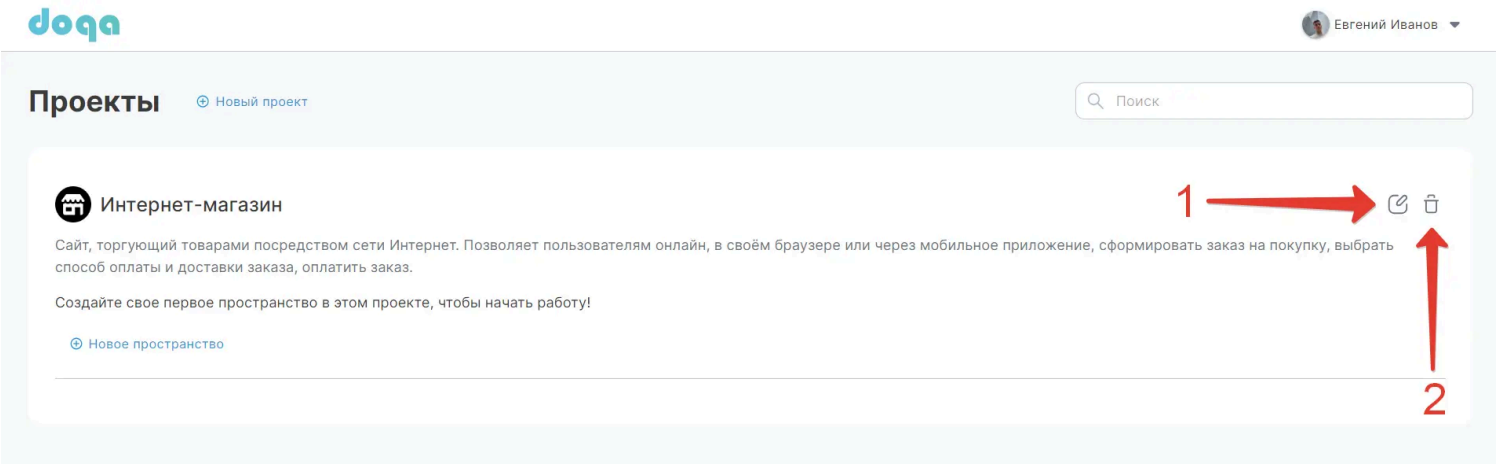
После заполнения полей нажмите «**Сохранить**». Проект появится на странице «Проекты».

Действия с проектом

Меню «⋮» на карточке проекта видят те, у кого есть хотя бы одно из прав на проект: изменение, архивирование, удаление, создание пространств (роли «Тест-лид» и «Администратор проектов»). Пункты меню доступны по правам. В меню пять пунктов:

- «Новое пространство» — создать **пространство** в проекте.
- «Редактировать» - изменить имя, описание и аватар. «ID родителя» в этой форме не отображается; показаны «Дата создания» и «Создатель».
- «Импорт» — перенести в проект данные из Qase или Test IT: импортированный проект становится новым пространством внутри этого, см. **Импорт проекта**.
- «Архивировать» — убрать проект из списка активных.
- «Удалить» — удалить проект.





Внимание

Удаление проекта необратимо: вместе с проектом удаляются все его пространства, тест-кейсы, прогоны и остальные связанные данные. В **корзину** они не попадают. Если проект может понадобиться позже, архивируйте его вместо удаления.

Архивные проекты

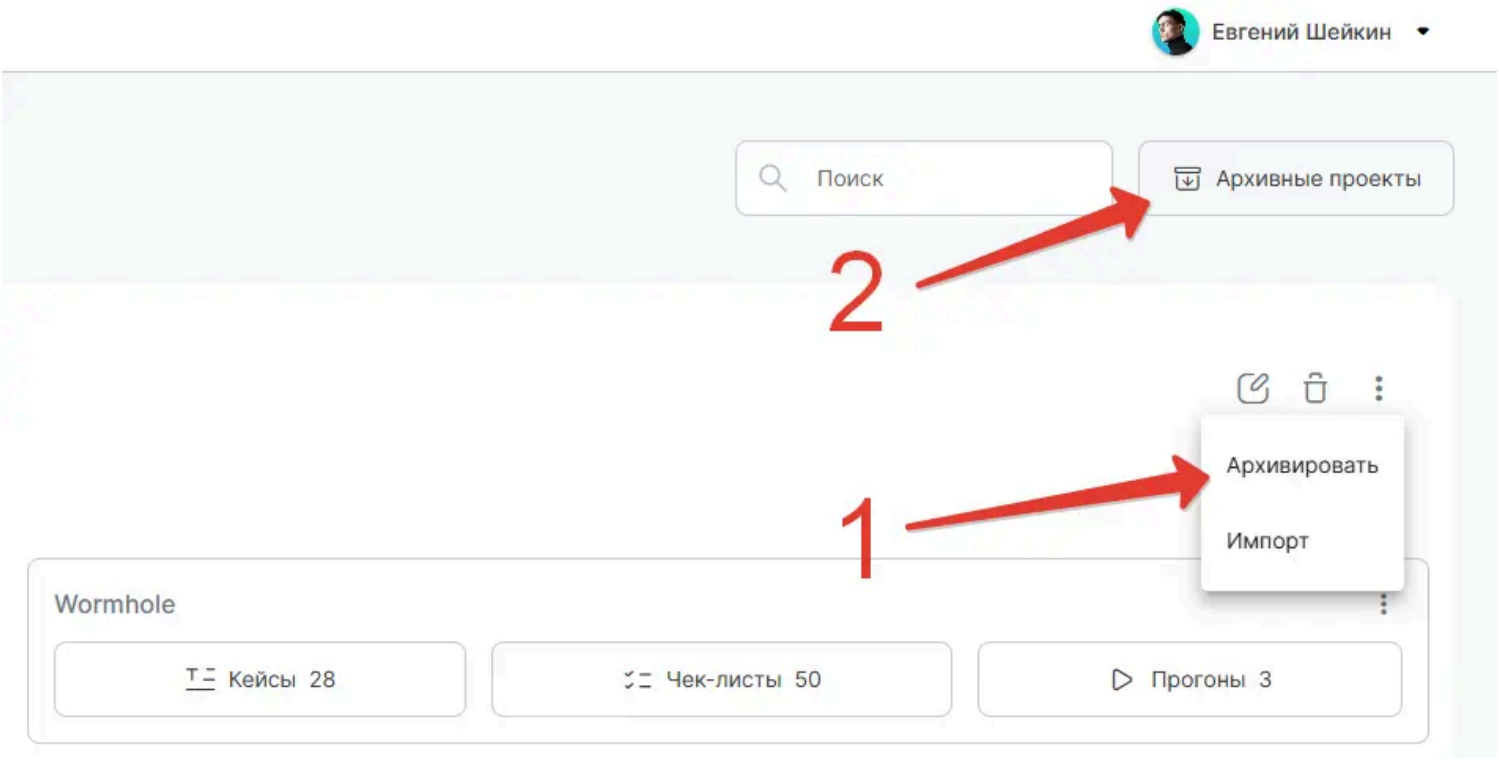
Архив — способ убрать неактуальный проект из списка, не теряя данные. Архивирование обратимо, в отличие от удаления.

Кнопка «**Архивные проекты**» в шапке открывает экран архива. И кнопка, и само архивирование доступны тем, у кого есть право `project.archive` хотя бы на одном проекте (роли «Тест-лид» и «Администратор проектов»).

Экран архива работает только на просмотр: открыть пространства архивного проекта нельзя. По каждому проекту видно дату архивации, число пространств, суммарные счётчики сущностей и названия пространств. Есть своя строка поиска: она ищет по названиям проектов и пространств.

Для каждого проекта в архиве доступны две кнопки:

- «**Разархивировать**» — возвращает проект в список активных.
- «**Удалить**» — удаляет проект насовсем, вместе со всеми данными.


Евгений Шейкин
Поиск
Архивные проекты
2
Архивировать
Импорт
1
Wormhole
Кейсы 28
Чек-листы 50
Прогоны 3

Пространства

Пространство — рабочая единица внутри **проекта**: в нём находятся тест-кейсы, чек-листы, общие шаги, прогоны, требования, автотесты, баги, свой дашборд и своя корзина. Проект только группирует пространства; вся работа идёт внутри пространства. Число пространств в проекте не ограничено.

Что находится в пространстве

Разделы в боковом меню пространства — в том же порядке, что и в интерфейсе.

Раздел	О чём он
Кейсы	Тест-кейсы: шаги, ожидаемые результаты, атрибуты, вложения.
Чек-листы	Списки проверок — когда пошаговый тест-кейс избыточен.
Общие шаги	Блоки шагов, переиспользуемые в нескольких тест-кейсах.
Прогоны	Выполнение тестов и его результаты.
Требования	Требования из внешнего трекера и их покрытие тестами.
Автотесты	Результаты автотестов и их запуск из CI/CD.
Баги	Баг-репорты, заведённые по результатам прогонов.
Дашборд	Сводная статистика по пространству.
Настройки	Привязка к проектам CI/CD и коннектор требований.
Корзина	Удалённые кейсы, чек-листы и прогоны — до окончательного удаления.

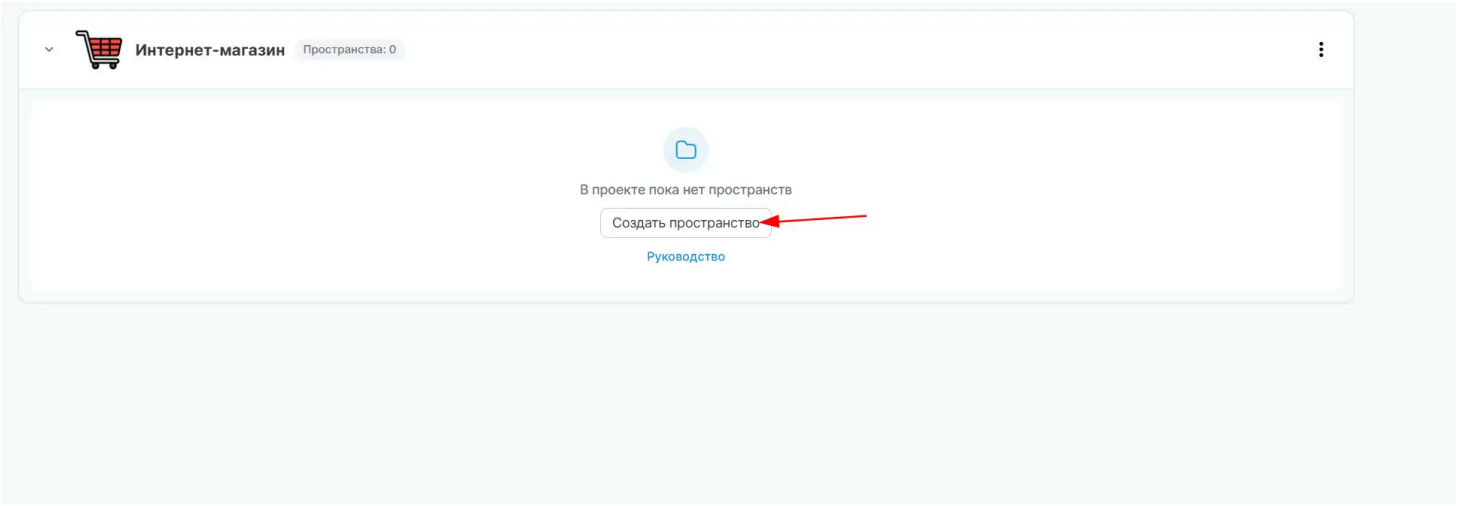
«Настройки» и «Корзина» доступны не всем: их видимость зависит от **прав в проекте**.

Сквозные механики: **папки**, **фильтры**, **теги**, **атрибуты**, **параметры**, **комментарии** — работают внутри пространства и описаны на своих страницах.

Создание пространства

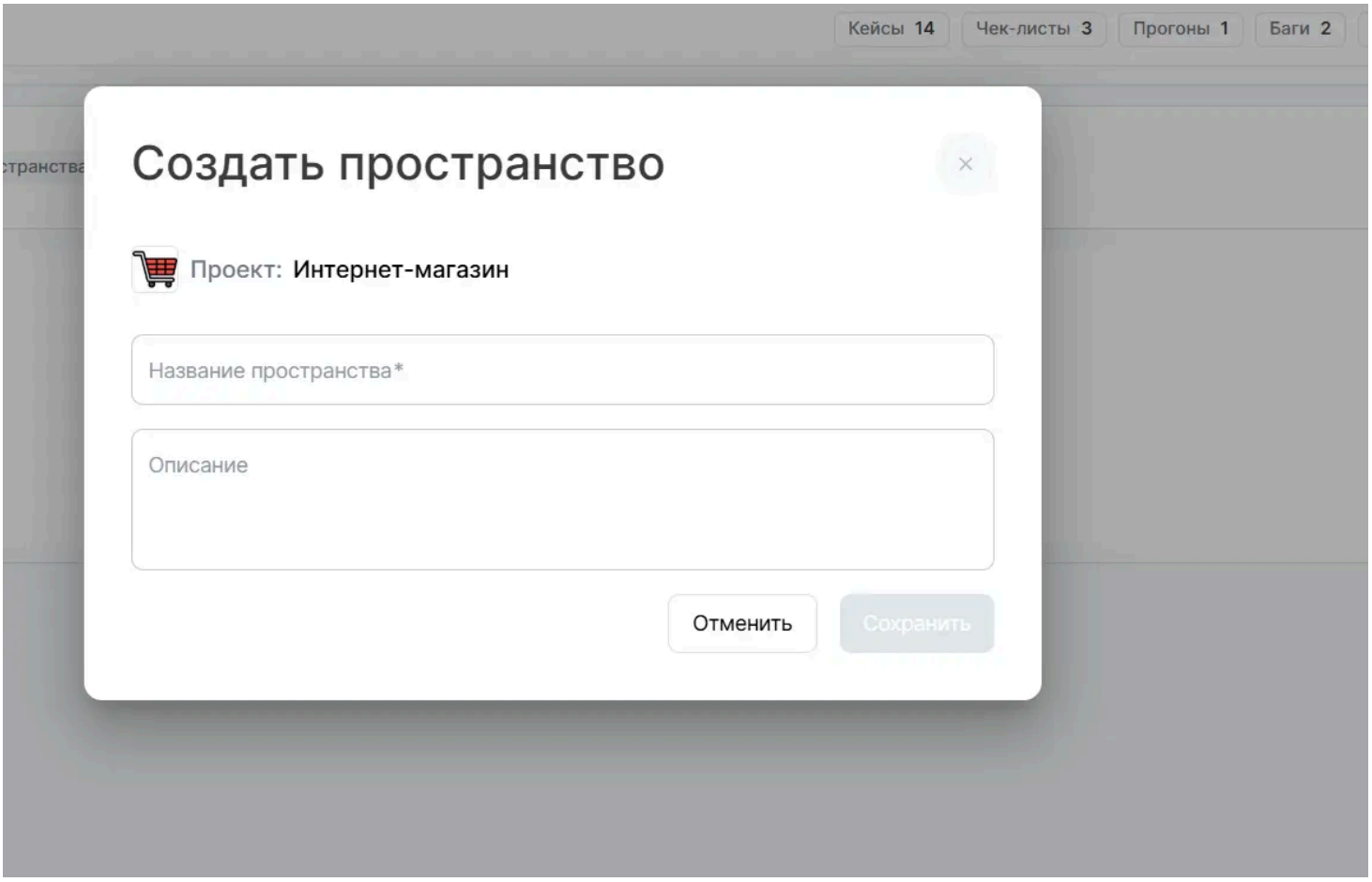
Пространства создаёт тот, у кого есть права на редактирование проекта.

1. На странице «Проекты» наведите курсор на проект, нажмите «⋮» в его шапке и выберите «**Новое пространство**». Если в проекте ещё нет пространств, нажмите «**Создать пространство**» в центре карточки проекта.



2. Заполните поля и нажмите «**Сохранить**»:

- «Название пространства» — от 1 до 100 символов, сочетание `//` запрещено;
- «Описание» — до 500 символов, необязательное. Описание видно в карточке пространства и участвует в поиске на странице проектов.

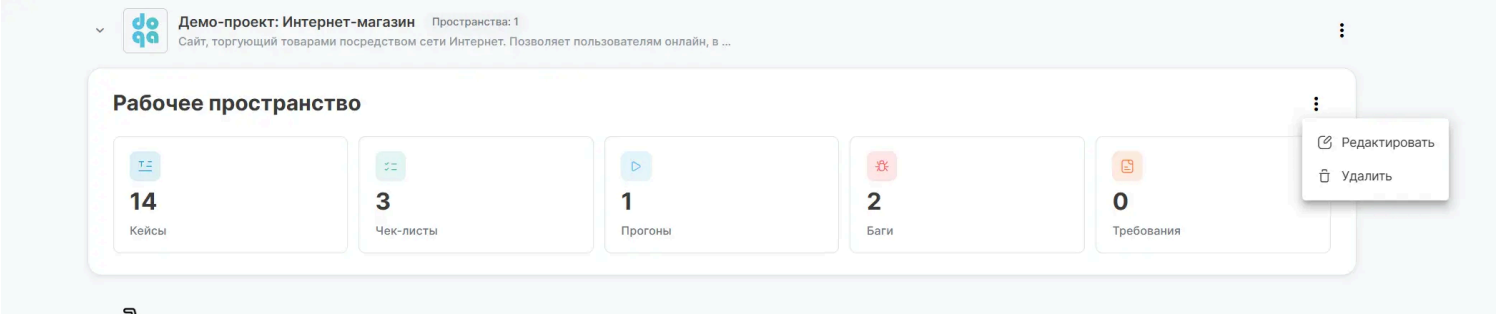


Карточка пространства на странице проектов

В карточке пространства пять плиток со счётчиками. Каждая ведёт в соответствующий раздел: «Кейсы», «Чек-листы», «Прогоны», «Баги», «Требования». Клик по названию пространства открывает его тест-кейсы.

Меню «⋮» в карточке доступно при правах на редактирование проекта и содержит два пункта:

- «**Редактировать**» — изменить название и описание;
- «**Удалить**» — удалить пространство вместе со всеми его данными.



Настройки пространства

Боковое меню пространства → «Настройки». Здесь две вкладки.

Настройки CI/CD

Привязывает пространство к проектам в вашей CI/CD-системе: результаты **автотестов** из привязанных проектов попадут в это пространство.

Отметьте нужные проекты чек-боксами и нажмите «**Сохранить**». Верхний чек-бокс отмечает и снимает все сразу.

Что можно увидеть вместо списка:

- «Интеграция CI/CD не настроена» и ссылка «Перейдите в раздел Интеграции» — CI/CD ещё не подключена администратором. См. **Настройка CI/CD**.
- «Список доступных проектов из CI/CD пуст. Проверьте настройки интеграции и валидность токена.» — интеграция есть, но не возвращает ни одного проекта. Чаще всего дело в истёкшем токене.

Коннектор требований

Подключает пространство к трекеру, из которого загружаются **требования**, и настраивает двустороннюю синхронизацию: ID проверок и статус покрытия отправляются в трекер, а изменения требований возвращаются в DoQA. На той же вкладке — «Лента операций» со всеми обменами с трекером.

Настройка подробно описана на странице **Коннектор требований**.

Тест-кейсы

Тест-кейс — сценарий проверки: шаги с ожидаемым результатом, ответственным и статусом автоматизации. Если расписывать проверки по шагам не нужно, используйте **чек-листы**.

Переход на вкладку «Кейсы»

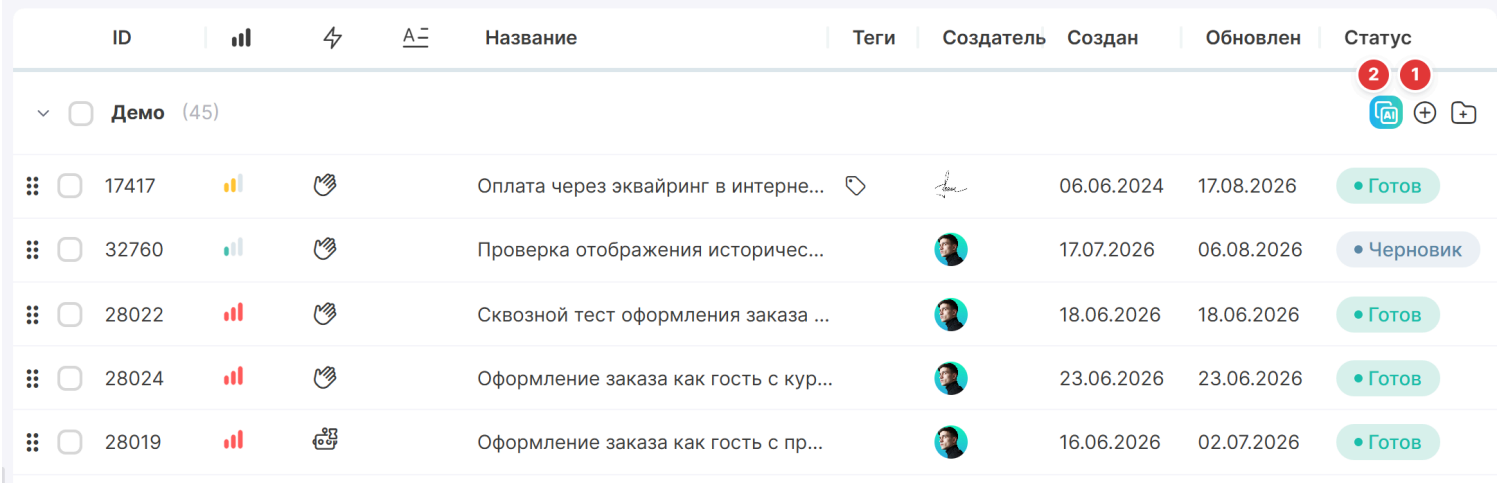
Перейти на вкладку «Кейсы» можно двумя способами:

- Со страницы «Проекты». Для этого нажмите на кнопку «**Кейсы**» в одном из пространств.
- Из вкладок «Прогоны» или «Чек-листы» при помощи левого бокового меню.

Создание тест-кейса

Для создания тест-кейса нажмите на кнопку «+» (цифра 1). Кнопка создания нового тест-кейса отображается в шапке каждой папки.

Также вы можете сгенерировать тест-кейс с помощью DoQA AI в выбранной папке (цифра 2).

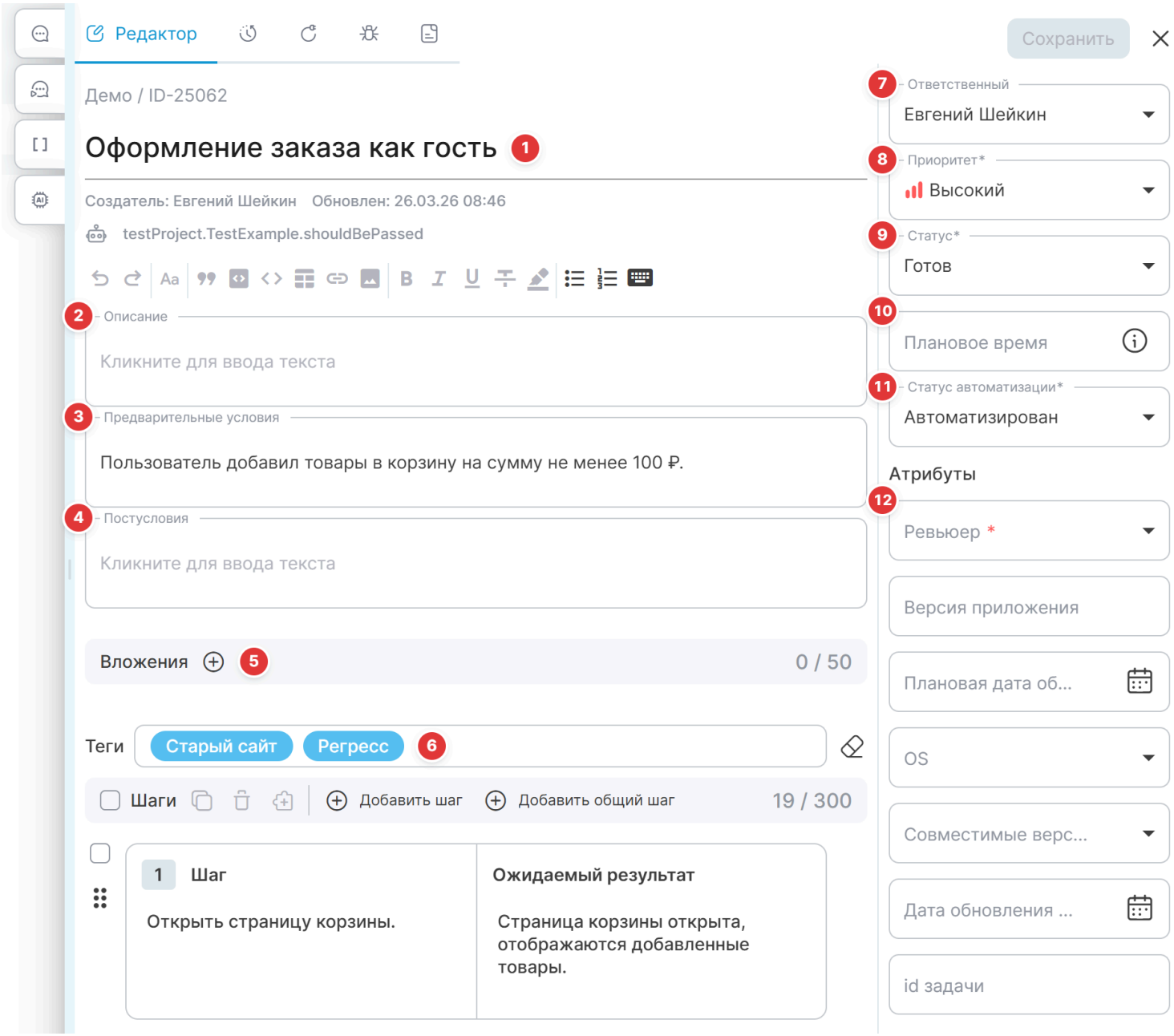


Заполнение полей тест-кейса

При создании тест-кейса обязательно заполнить поле «Название» (цифра 1 на скриншоте). Также можно указать:

- «Описание» (цифра 2) — любая дополнительная информация о тест-кейсе

- «Предварительные условия» (цифра 3) — состояние, в котором должна быть система, или действия, которые нужно выполнить перед началом прохождения тест-кейса
- «Постусловия» (цифра 4) — действия, которые нужно выполнить после прохождения шагов тест-кейса. Например, удаление сущностей, созданных при тестировании
- «Вложения» (цифра 5) — любые файлы размером до 10 mb, не более 50 файлов на тест-кейс. Счетчик рядом с заголовком показывает, сколько файлов уже приложено
- «Теги» (цифра 6) — метки, по которым можно найти определенный тест-кейс. Подробнее о **тегах**
- «Ответственный» (цифра 7) — пользователь, отвечающий за актуальное состояние тест-кейса. По умолчанию — создатель тест-кейса
- «Приоритет» (цифра 8) — по умолчанию указан «Средний», можно выбрать также «Низкий» и «Высокий»
- «Статус» (цифра 9) — по умолчанию указан «Готов». Также доступны статусы «Ревью», «Черновик», «В архиве» и «Сломан»
- «Плановое время» (цифра 10) — ожидаемая длительность прохождения тест-кейса. Формат: `ч` (часы), `м` (минуты), `с` (секунды). Например, `1ч 30м`. Латинские `h`, `m` и `s` тоже принимаются, но смешивать латиницу и кириллицу в одном значении нельзя. Такое значение поле не примет
- «Статус автоматизации» (цифра 11) — по умолчанию указан «Ручной». Если вы планируете автоматизировать этот тест-кейс, можно изменить статус на «Подлежит автоматизации». Статус «Автоматизирован» может быть проставлен только автоматически, когда DoQA получает отчет об **автотесте**, который связан с тест-кейсом через Allure ID. Связь с автотестом сбрасывается при любом ручном изменении статуса автоматизации, а не только при уходе со статуса «Автоматизирован». У связанного тест-кейса над полями редактора показан путь автотеста
- «Атрибуты» — дополнительные поля, созданные пользователем для тестовой документации определенного проекта. Обязательные для заполнения атрибуты отмечены звездочкой (цифра 12). Подробнее об **атрибутах**



Структура шага

Шаг состоит из двух полей:

- «Шаг» — действие, которое нужно выполнить
- «Ожидаемый результат» — что должно произойти после этого действия

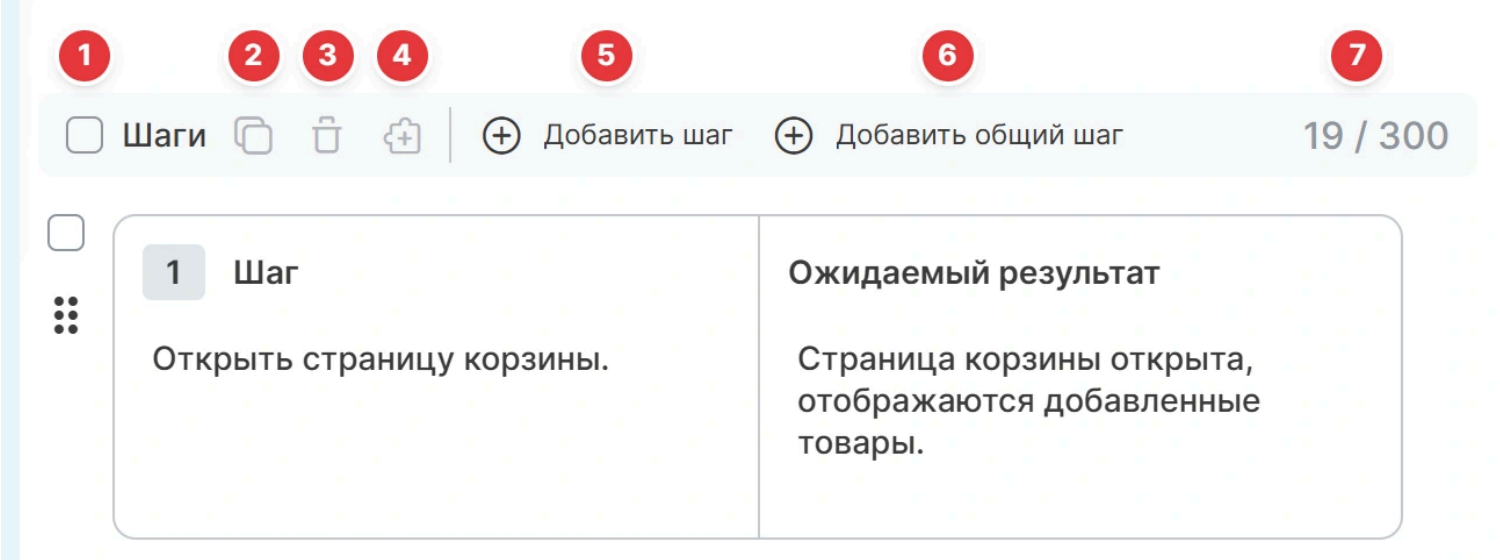
Дополнительно можно раскрыть поле «Тестовые данные» кнопкой в боковом меню шага. Крестик в заголовке поля убирает его обратно.

Нумерация шагов сквозная: шаги внутри общих шагов входят в общую нумерацию и учитываются счетчиком. Сам общий шаг за отдельный шаг не считается.

Панель действий с шагами

Панель содержит:

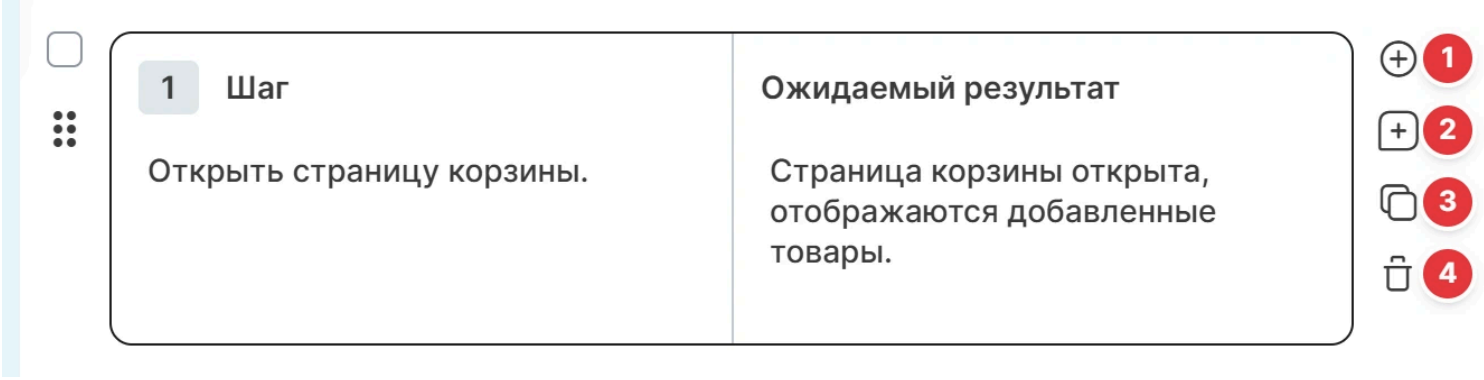
- Чек-бокс выбора шагов (цифра 1)
- Кнопка копирования шагов (цифра 2). Активна, если отмечен как минимум 1 чек-бокс обычного шага. Копии встают в конец тест-кейса. Общие шаги скопировать нельзя: если среди выбранных есть общий шаг, кнопка блокируется с подсказкой «Общие шаги не могут быть скопированы. Вы можете добавить их повторно через интерфейс добавления шагов». Кнопка блокируется и в том случае, если копии не помещаются в лимит 300 шагов
- Кнопка удаления шагов (цифра 3). Активна, если отмечен как минимум 1 чек-бокс шага
- Кнопка создания общего шага из выбранных шагов (цифра 4). Объединять можно только обычные шаги: если среди выбранных есть общий шаг, кнопка блокируется с подсказкой «Выбран общий шаг. Можно объединять только обычные шаги». Максимальное количество шагов в одном общем шаге — 100, поэтому при выборе более 100 шагов кнопка также блокируется. Подробнее — в разделе **Общие шаги**
- Кнопка добавления нового шага в конец тест-кейса (цифра 5). Это же действие можно выполнить с помощью сочетания клавиш `Ctrl + Shift + D` (Windows) или `Command + Shift + D` (MacOS), см. **Горячие клавиши**
- Кнопка добавления общего шага в конец списка (цифра 6)
- Счетчик шагов (цифра 7). Максимальное количество шагов 300



Боковое меню шага

Меню появляется при наведении курсора на шаг и содержит:

- Кнопку добавления шага или общего шага (цифра 1). Добавленный шаг появляется сразу после текущего
- Кнопку добавления поля с тестовыми данными (цифра 2)
- Кнопку копирования шага (цифра 3). Копия встает сразу после исходного шага
- Кнопку удаления шага (цифра 4)



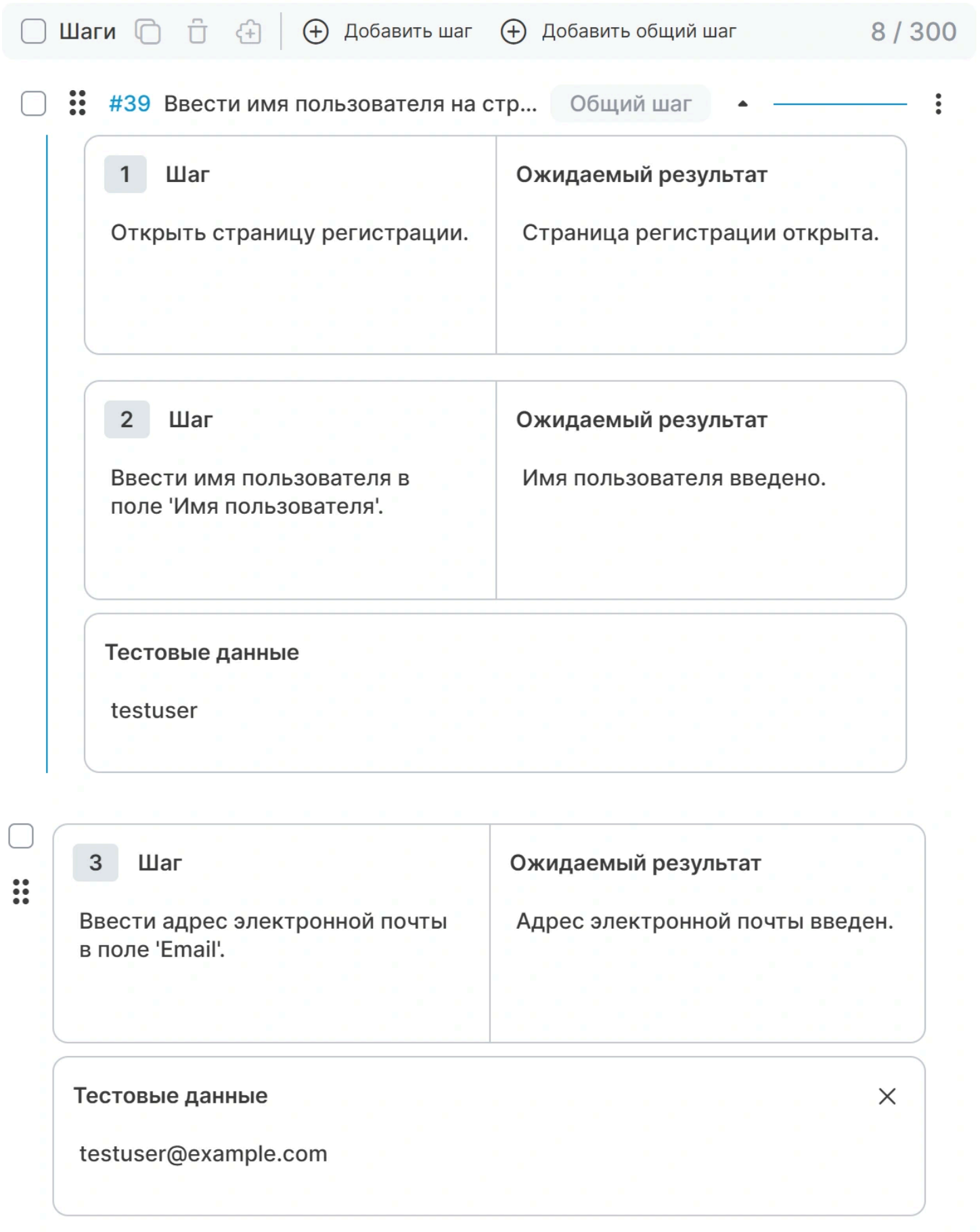
Использование общих шагов в кейсах

Помимо использования обычных шагов вы можете добавлять в тест-кейсы **общие шаги**. Вертикальная голубая полоса показывает границы общего шага.

Шаги внутри общего шага в редакторе тест-кейса только для чтения. При наведении на них показывается подсказка «Для редактирования общего шага перейдите в его редактор». Для редактирования перейдите в редактор общего шага по ссылке в ID: он откроется в новой вкладке. После сохранения изменений общий шаг обновится во всех тест-кейсах, в которых он используется.

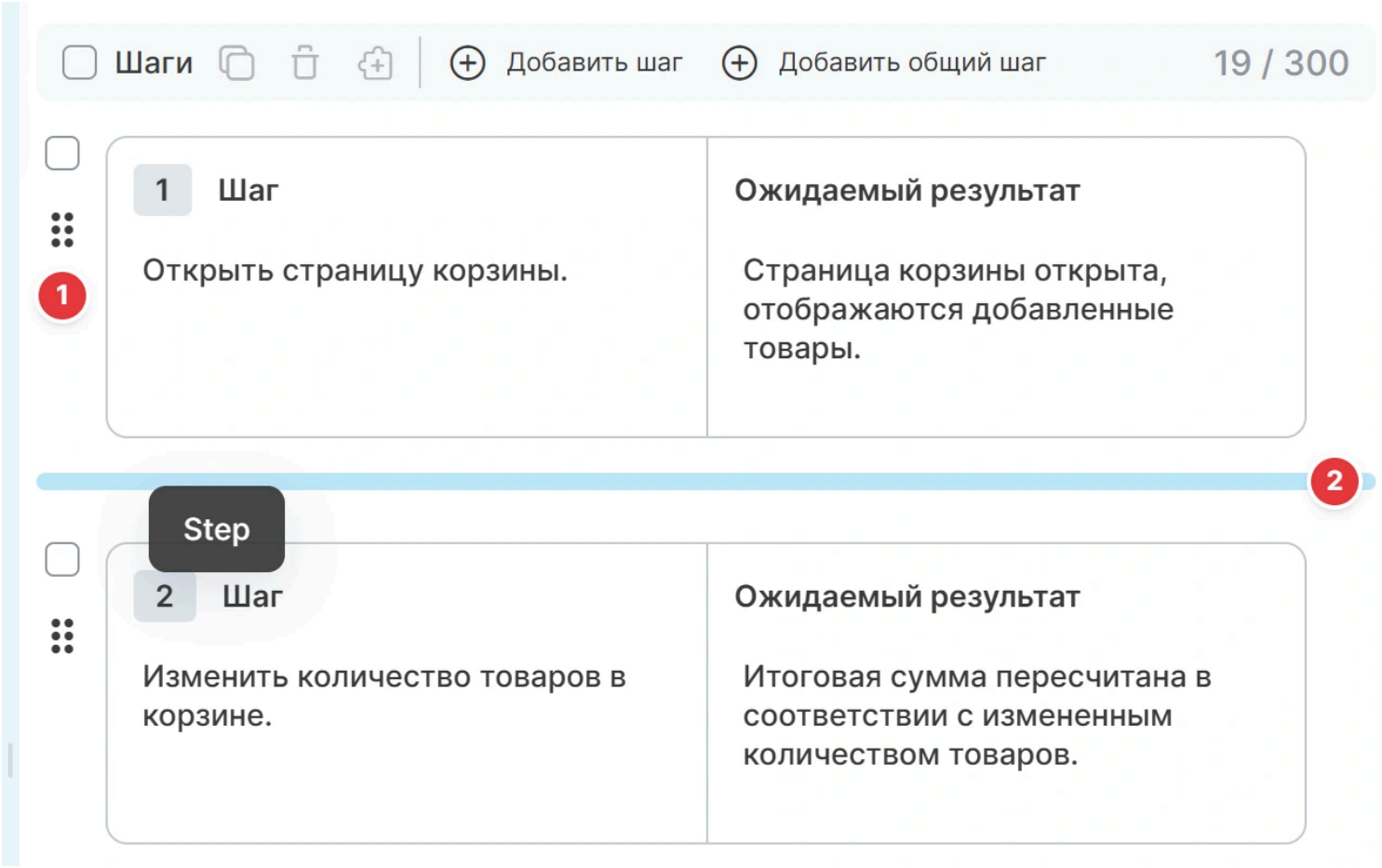
Стрелка в заголовке сворачивает и разворачивает содержимое общего шага. Кнопка «...» открывает меню с действиями:

- «Разбить на простые шаги» — общий шаг заменяется своими шагами как обычными. Связь с библиотекой общих шагов теряется: последующие правки общего шага в этот тест-кейс уже не придут
- «Добавить шаг» — новый обычный шаг после общего шага
- «Добавить общий шаг» — еще один общий шаг после текущего
- «Удалить» — убрать общий шаг из тест-кейса



Перетаскивание шагов

Кликните и удерживайте иконку перетаскивания шага (цифра 1) и переместите шаг на нужную позицию (цифра 2).



Завершение создания тест-кейса

После внесения всех данных нажмите кнопку «**Сохранить**». При сохранении новый тест-кейс появится в таблице.

Сохранить изменения также можно с помощью сочетания клавиш `Ctrl + S` (Windows) или `Command + S` (MacOS).

Вкладки редактора тест-кейса

В редакторе тест-кейса пять вкладок:

- «Редактор» — поля тест-кейса и шаги. Открыта по умолчанию
- «История редактирования» — версии тест-кейса, что и кем менялось, откат к предыдущей версии. Подробнее — в разделе **История редактирования**

- «История запуска» — результаты прохождения тест-кейса в прогонах. Подробнее — в разделе **История запуска**
- «Баги» — баг-репорты, заведенные по этому тест-кейсу. Подробнее — в разделе **Баги**
- «Требования» — требования из трекера, с которыми связан тест-кейс

У несохраненного тест-кейса доступна только вкладка «Редактор», остальные включаются после первого сохранения. Кнопка «**Сохранить**» показывается только на вкладке «Редактор».

Боковые панели

Рядом с редактором расположен ряд закладок. Каждая открывает свою панель:

- «Комментарии» — обсуждение тест-кейса. Подробнее — в разделе **Комментарии**
- «Комментарии прогонов» — комментарии, оставленные при прохождении тест-кейса в **прогонах**
- «Параметры» — параметры тест-кейса и стратегия их комбинирования. Подробнее — в разделе **Параметризация тестов**
- AI-ревью — проверка тест-кейса силами DoQA AI. Подробнее — в разделе **AI анализ тест-кейса**

Закладка AI-ревью показывается только у сохраненного тест-кейса, только на вкладке «Редактор» и только при праве на редактирование. У чек-листов ее нет.

Требования

На вкладке «Требования» видно, с какими требованиями из трекера связан тест-кейс.

Кнопка «**Привязать требование**» открывает поиск по ключу или заголовку требования. Один тест-кейс можно связать с несколькими требованиями. Чтобы разорвать связь, наведите курсор на строку требования и нажмите иконку разрыва связи.

Если требование изменилось в трекере, под его строкой появляется полоса «Требование изменилось в трекере — обновите кейс и подтвердите.» с кнопкой «**Актуализировано**». Кнопка не меняет тест-кейс. Она подтверждает, что вы его уже привели в соответствие с

новой версией требования. Требования, удаленные или закрытые в трекере, показываются зачеркнутыми.

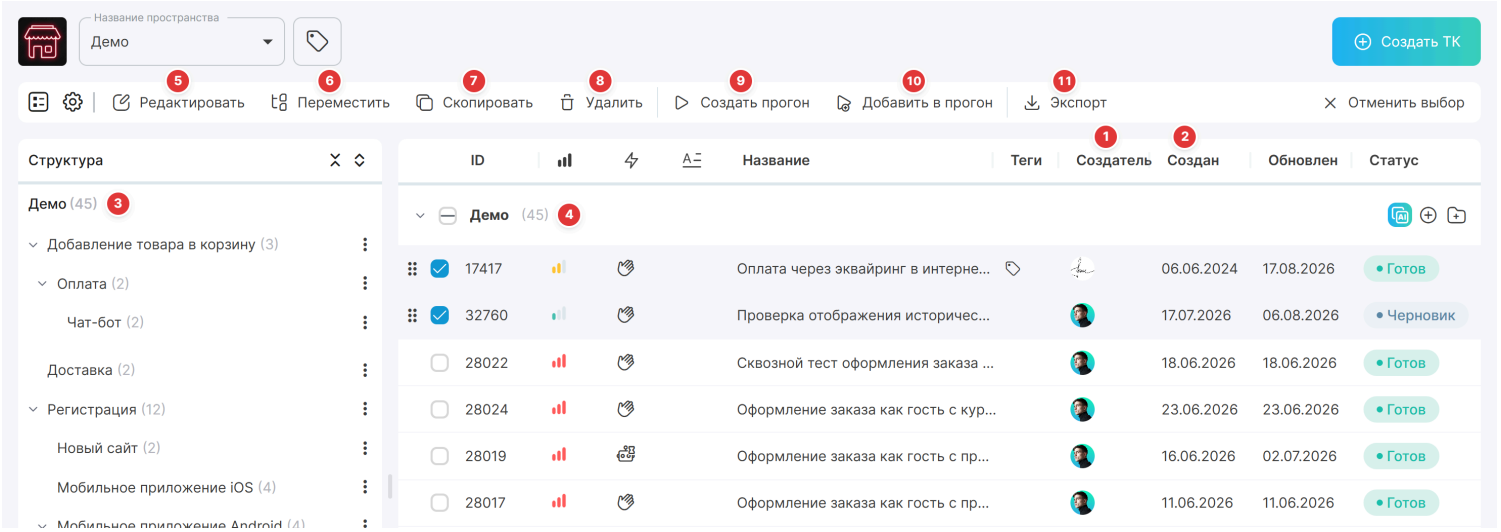
Подробнее — в разделе **Требования**. Требования попадают в DoQA из трекера через **коннектор требований**.

Таблица тест-кейсов

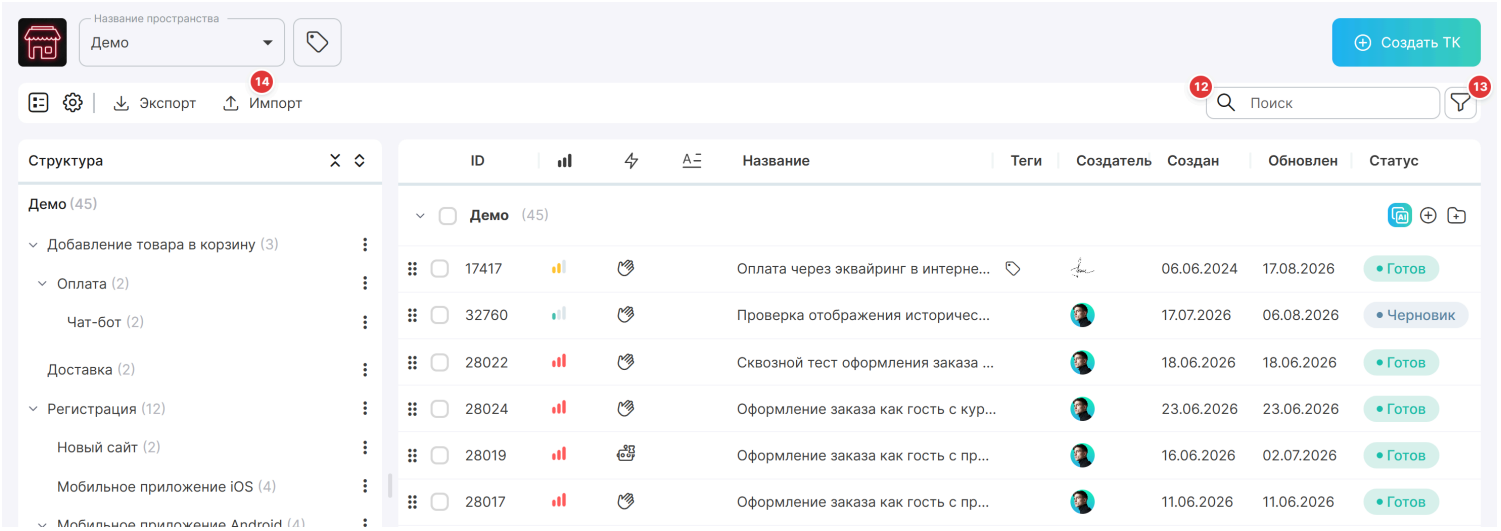
Колонки таблицы: «ID», «Приоритет», «Статус автоматизации», «Название», «Теги», «Создатель» (цифра 1), «Создан» (цифра 2), «Обновлен», «Статус». Счетчик показывает общее количество тест-кейсов в корневой папке пространства (цифры 3 и 4) и во вложенных папках.

Доступные функции таблицы

- Сортировка по графам таблицы
- Настройка колонок. Шестеренка на панели открывает список колонок: колонку можно скрыть чек-боксом или изменить ее порядок перетаскиванием. Там же настройку колонок можно сохранить как пресет и переключаться между пресетами. Пресеты личные: у каждого пользователя свои, и хранятся они отдельно для каждой таблицы
- Показать или скрыть структуру папок — кнопка слева на панели. Подробнее о **папках**
- «**Редактировать**» (цифра 5) — массовое редактирование выбранных тест-кейсов
- «**Переместить**» (цифра 6)
- «**Скопировать**» (цифра 7). Копия получает название исходного тест-кейса с суффиксом `[COPY]`, создателем и ответственным становится пользователь, выполнивший копирование. Связь с автотестом не переносится: статус автоматизации «Автоматизирован» у копии сбрасывается в «Ручной»
- «**Удалить**» (цифра 8). Тест-кейсы перемещаются в **корзину**, а не удаляются сразу
- «**Создать прогон**» (цифра 9) — объединить выбранные тест-кейсы в новый **прогон**
- «**Добавить в прогон**» (цифра 10) — добавить выбранные тест-кейсы в существующий прогон
- «**Экспорт**» (цифра 11) — выгрузка в формате xlsx. Подробнее — в разделе **Экспорт**



- Поиск по названию или ID тест-кейса (цифра 12)
- **Фильтры** (цифра 13)
- «**Импорт**» (цифра 14) — загрузка тест-кейсов в DoQA. Подробнее — в разделе **Импорт**



Тест-кейсы можно перетаскивать из одной папки в другую, а также менять их порядок внутри папок.

Чек-листы

Чек-лист — дерево проверок без шагов и ожидаемых результатов. Подходит там, где достаточно списка «что проверить», а расписывать сценарий по шагам не нужно.

Чем чек-лист отличается от **тест-кейса**:

	Тест-кейс	Чек-лист
Тело	линейные шаги с ожидаемым результатом и тестовыми данными	дерево чеков и сабчеков, до 8 уровней вложенности
Предварительные условия и постусловия	есть	нет
Статус автоматизации	есть	нет
Общие шаги	можно использовать	нет
AI-ревью	есть	нет

Остальное совпадает: поля, вкладки редактора, боковые панели, функции таблицы, сохранение по `Ctrl + S`.

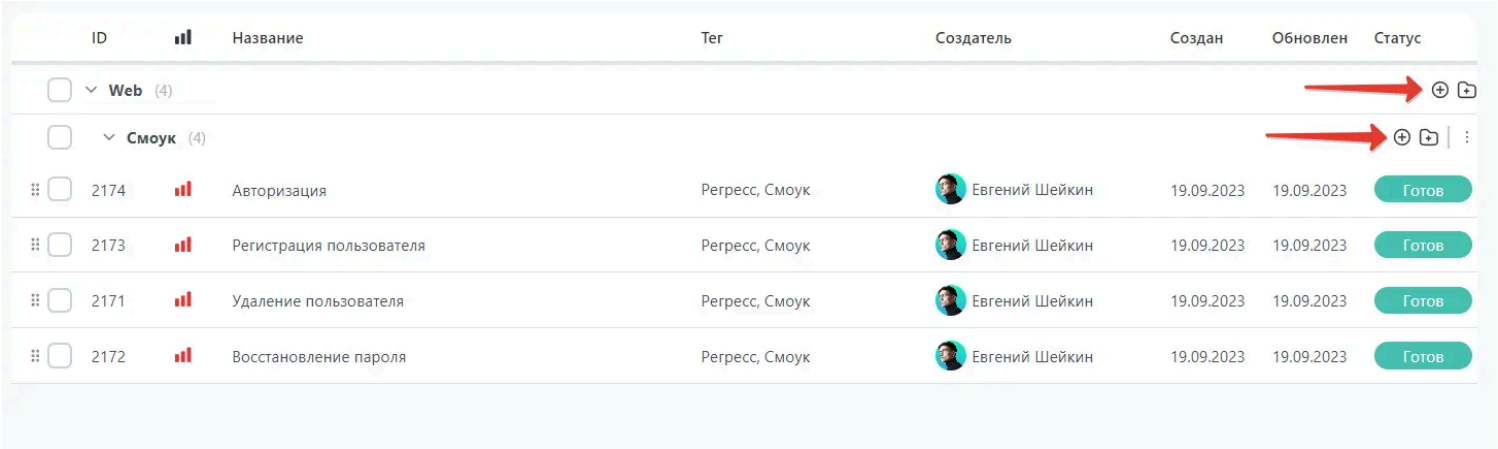
Переход на вкладку «Чек-листы»

Перейти на вкладку «Чек-листы» можно несколькими способами:

- Со страницы «Проекты». Для этого нажмите на кнопку «**Чек-листы**» в одном из пространств.
- Из вкладок «Кейсы» или «Прогоны» при помощи левого бокового меню.

Создание чек-листа

Для создания нового чек-листа нажмите на кнопку «**+**» в шапке любой папки. Если пользователь не создавал папки, отображается только корневая папка пространства. Рядом находятся кнопка создания папки и кнопка генерации чек-листа с помощью **DoQA AI**.

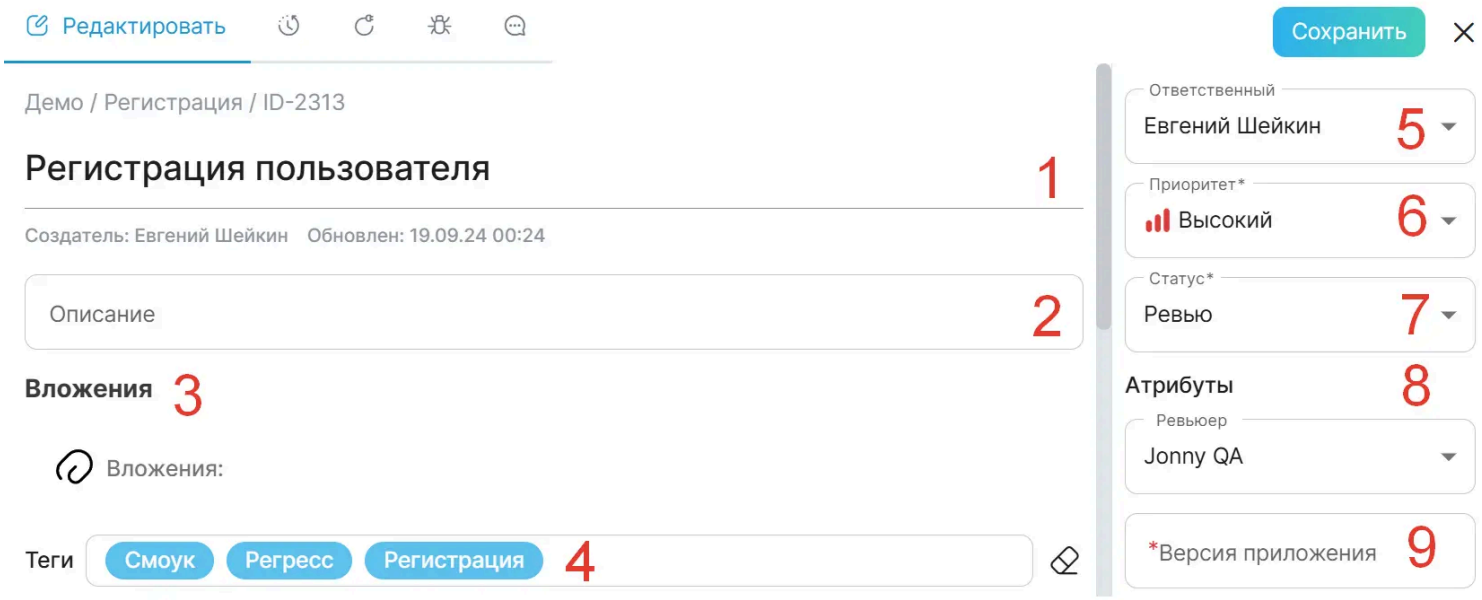


Заполнение полей чек-листа

Поля те же, что у тест-кейса, кроме предварительных условий, постусловий и статуса автоматизации, которых у чек-листа нет. Обязательное поле одно: «Название» (цифра 1 на скриншоте). Также можно указать:

- «Описание» (цифра 2)
- «Вложения» (цифра 3) — любые файлы размером до 10 mb, не более 50 файлов на чек-лист
- «Теги» (цифра 4) — подробнее о **тегах**
- «Ответственный» (цифра 5) — по умолчанию создатель чек-листа
- «Приоритет» (цифра 6) — по умолчанию «Средний», можно выбрать также «Низкий» и «Высокий»
- «Статус» (цифра 7) — по умолчанию «Готов». Также доступны статусы «Ревью», «Черновик», «В архиве» и «Сломан»
- «Плановое время» — ожидаемая длительность прохождения чек-листа. Формат: `ч` (часы), `м` (минуты), `с` (секунды). Например, `1ч 30м`. Латинские `h`, `m` и `s` тоже принимаются, но смешивать латиницу и кириллицу в одном значении нельзя
- «Атрибуты» (цифра 8) — дополнительные поля, созданные пользователем для тестовой документации определенного проекта. Обязательные для заполнения атрибуты отмечены звездочкой (цифра 9). Подробнее об **атрибутах**

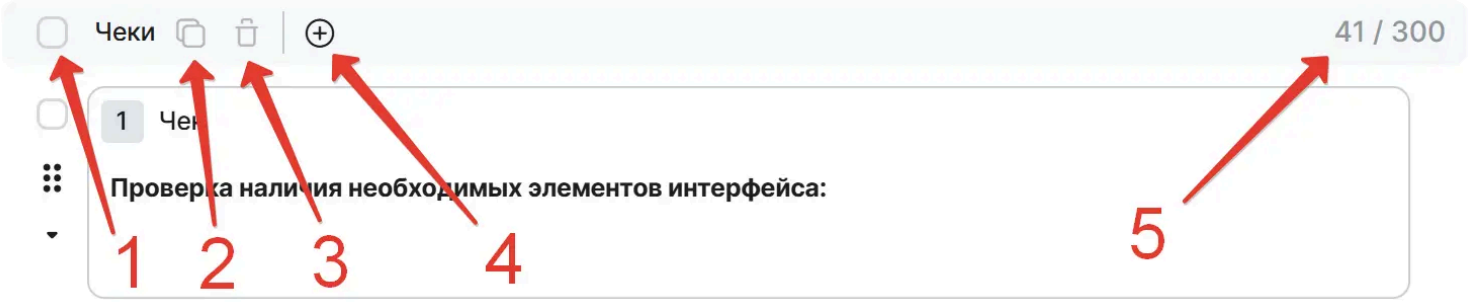
Подробнее о полях — в разделе **Заполнение полей тест-кейса**.



Панель действий с чеками

Панель содержит:

- Чек-бокс выбора чеков (цифра 1)
- Кнопка копирования чеков (цифра 2). Активна, если отмечен как минимум 1 чек-бокс чека. Копии встают в конец чек-листа
- Кнопка удаления чеков (цифра 3). Активна, если отмечен как минимум 1 чек-бокс
- Кнопка добавления нового чека в конец чек-листа (цифра 4). Это же действие можно выполнить с помощью сочетания клавиш `Ctrl + Shift + D` (Windows) или `Command + Shift + D` (MacOS), см. **Горячие клавиши**
- Счетчик чеков (цифра 5). Максимальное количество чеков и сабчеков 300



Боковое меню чека

Меню появляется при наведении курсора на чек и содержит:

- Кнопку копирования чека (цифра 1). Новый чек появляется сразу после текущего, на соответствующем уровне вложенности. Чек копируется вместе со всеми вложенными

сабчеками

- Кнопку удаления чека (цифра 2). Чек удаляется вместе со всеми вложенными сабчеками
- Кнопку добавления сабчека текущему чеку (цифра 3). Допускается создание 8 уровней вложенности (1 чек + 7 уровней сабчеков)
- Кнопку добавления чека того же уровня вложенности (цифра 4). Новый чек встает сразу после исходного, на соответствующем уровне вложенности



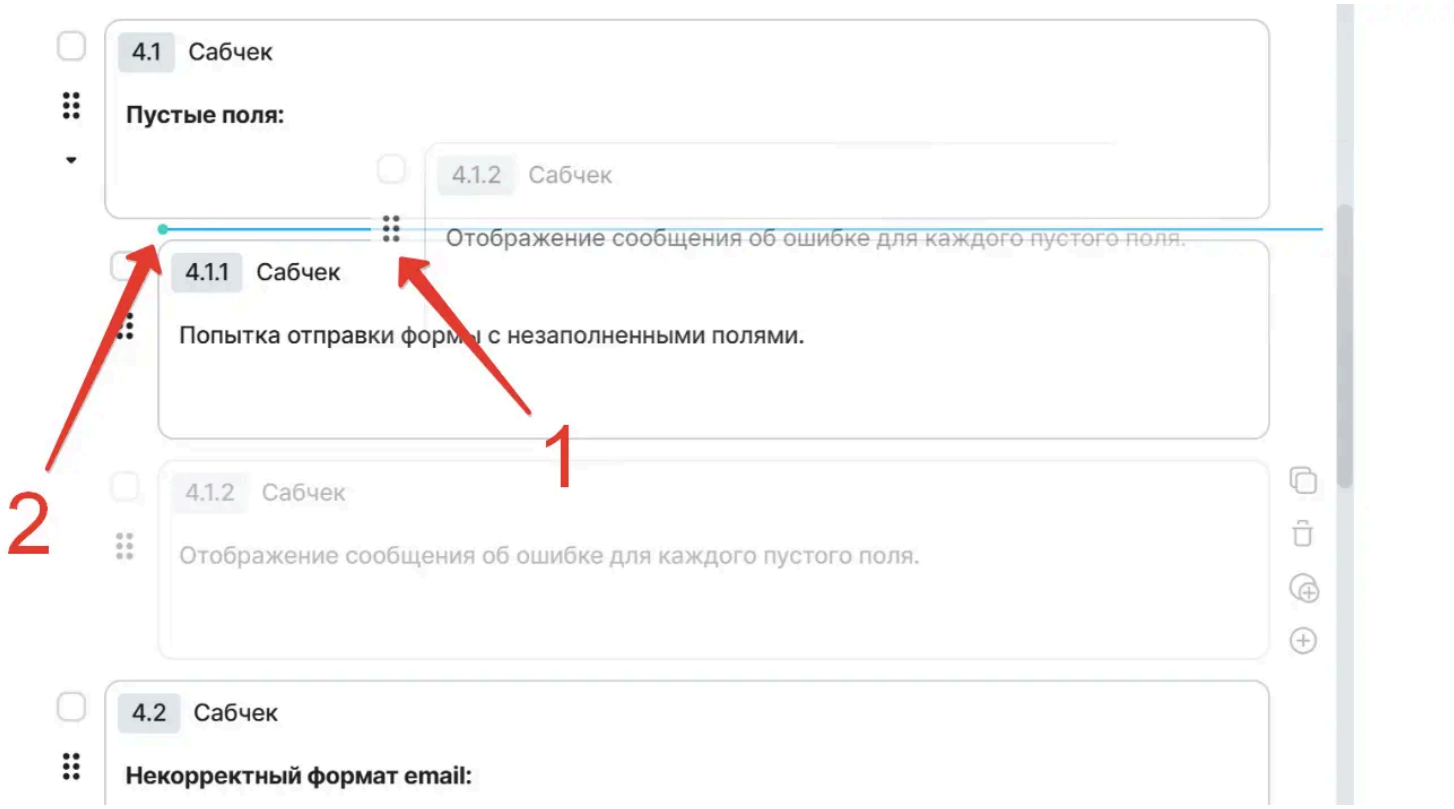
Дерево чеков

Чеки нумеруются иерархически: сабчек первого чека получает номер `1.1`, его сабчек — `1.1.1` и так далее.

У чека, к которому добавлены сабчеки, слева появляется стрелка: она сворачивает и разворачивает всю вложенную ветку. У чеков без сабчеков стрелки нет.

Перетаскивание чеков

Кликните и удерживайте иконку перетаскивания чека (цифра 1) и переместите его на нужную позицию (цифра 2). Чеки перемещаются вместе со всеми вложенными сабчеками.



Завершение создания чек-листа

После внесения всех данных нажмите кнопку «**Сохранить**». При сохранении новый чек-лист появится в таблице.

Сохранить изменения также можно с помощью сочетания клавиш `Ctrl + S` (Windows) или `Command + S` (MacOS).

Вкладки редактора чек-листа

Вкладки те же, что у тест-кейса: «Редактор», «История редактирования», «История запуска», «Баги», «Требования». У несохраненного чек-листа доступна только вкладка «Редактор». Подробнее — в разделе **Вкладки редактора тест-кейса**.

Боковые панели

У чек-листа три закладки боковых панелей: «Комментарии», «Комментарии прогонов» и «Параметры». Подробнее — в разделе **Боковые панели**.

AI-ревью для чек-листов недоступно: эта закладка есть только у тест-кейсов.

Требования

Чек-лист можно связать с требованиями из трекера так же, как тест-кейс: вкладка «Требования», кнопка «**Привязать требование**», полоса актуализации при изменении требования в трекере. Подробнее — в разделах **Требования** и **Требования в карточке тест-кейса**.

Таблица чек-листов

Колонки таблицы: «ID», «Приоритет», «Название», «Теги», «Создатель» (цифра 1), «Создан» (цифра 2), «Обновлен», «Статус». Колонки «Статус автоматизации» у чек-листов нет, в отличие от таблицы тест-кейсов.

Счетчик показывает общее количество чек-листов в корневой папке пространства (цифра 3) и во вложенных папках (цифра 4).

Доступные функции

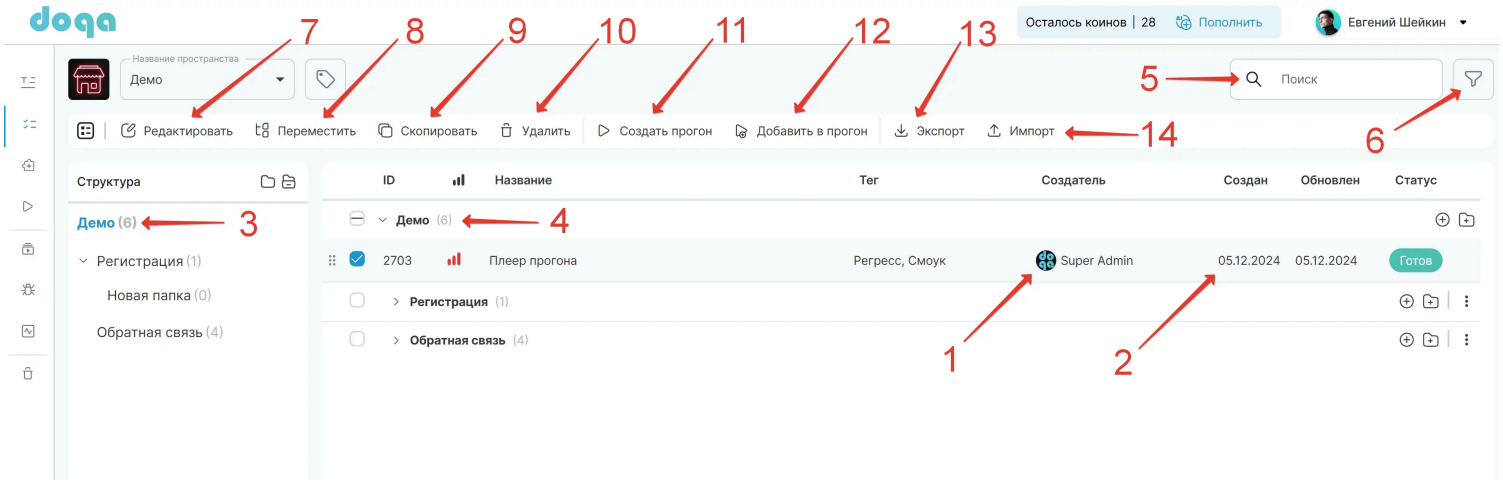
- Сортировка по графам таблицы
- Настройка колонок. Шестеренка на панели открывает список колонок: колонку можно скрыть или изменить ее порядок
- Показать или скрыть структуру папок — кнопка слева на панели. Подробнее о **папках**
- Поиск по названию или ID чек-листа (цифра 5)
- **Фильтры** (цифра 6)

Чек-листы можно:

- «**Редактировать**» (цифра 7) — массово, для всех выбранных чек-листов
- «**Переместить**» (цифра 8)
- «**Скопировать**» (цифра 9). Копия получает название исходного чек-листа с суффиксом `[COPY]`, создателем становится пользователь, выполнивший копирование. Ответственный остается прежним

- «**Удалить**» (цифра 10). Чек-листы перемещаются в **корзину**, а не удаляются сразу
- «**Создать прогон**» (цифра 11) — объединить выбранные чек-листы в новый **прогон**
- «**Добавить в прогон**» (цифра 12) — добавить выбранные чек-листы в существующий прогон
- «**Экспорт**» (цифра 13) — выгрузка в формате xlsx. Подробнее — в разделе **Экспорт**
- «**Импорт**» (цифра 14) — загрузка чек-листов в DoQA. Подробнее — в разделе **Импорт**

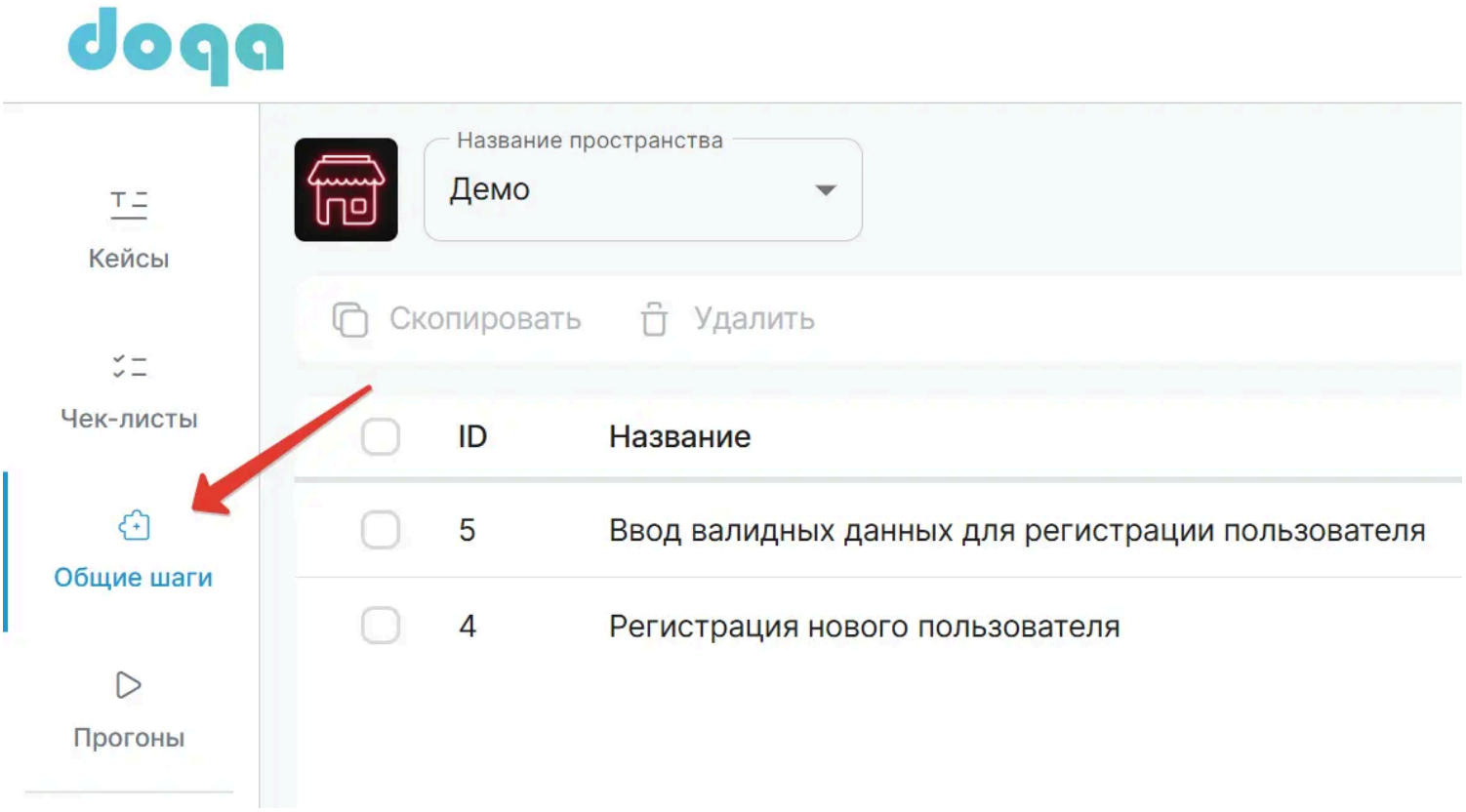
Порядок чек-листов в таблице можно менять, перетаскивая элементы на нужные позиции.



Общие шаги

Общий шаг — набор шагов, который хранится отдельно от тест-кейсов и переиспользуется в них. Общие шаги привязаны к пространству и доступны во всех его **тест-кейсах**. При изменении общего шага правка применяется во всех тест-кейсах, где он используется.

Иконка для перехода во вкладку «Общие шаги» отображается в левом боковом меню.



Создание общего шага

Создать общий шаг можно двумя способами.

В редакторе общих шагов

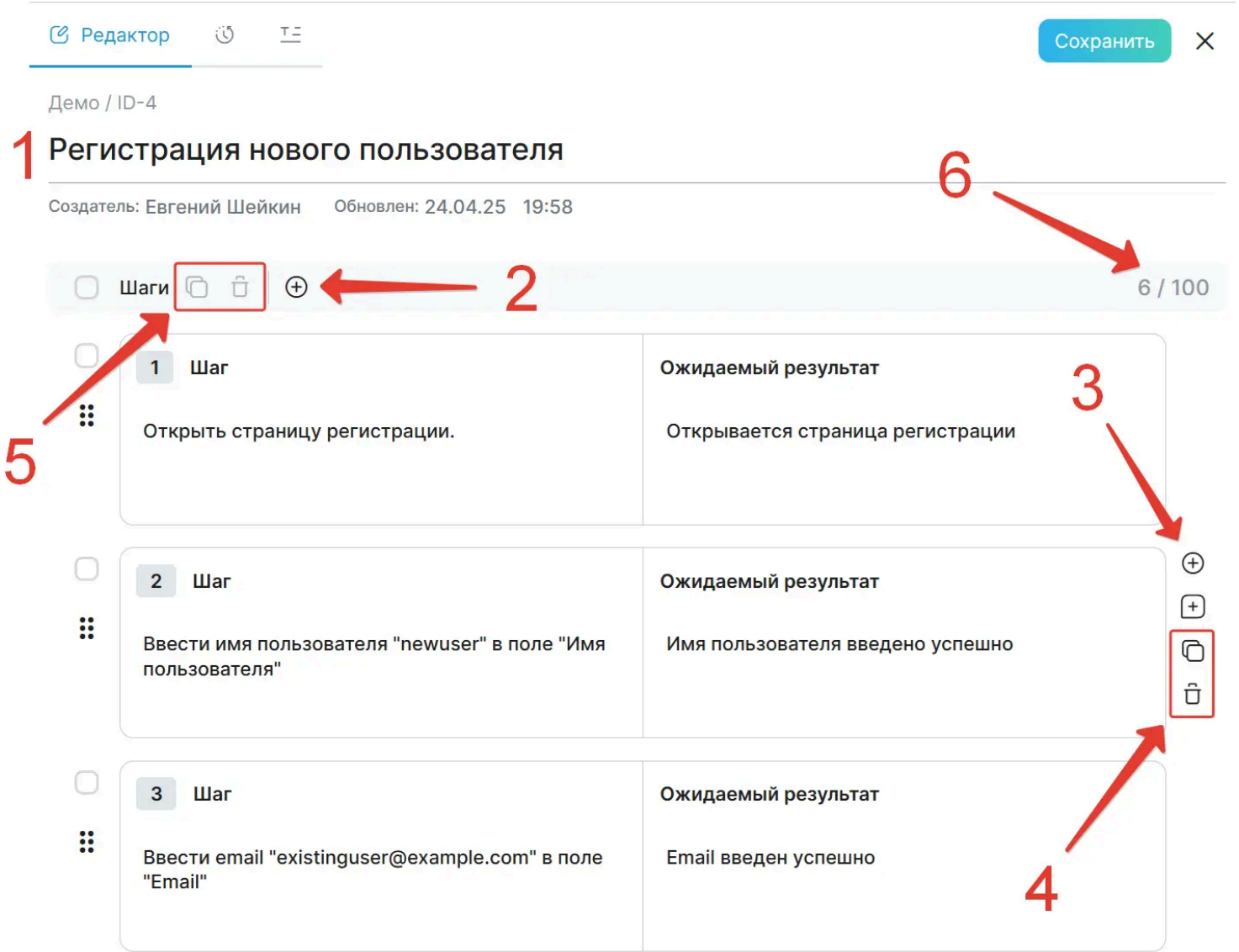
Во вкладке «Общие шаги» нажмите на кнопку «**Создать общий шаг**». Откроется редактор общего шага.

Совет

Не путайте с кнопкой «**Добавить общий шаг**» в редакторе тест-кейса: она не создает новый общий шаг, а вставляет в тест-кейс уже существующий.

1. Введите название общего шага (цифра 1).
2. Добавьте шаги кнопкой в верхней панели (цифра 2) или через меню каждого шага (цифра 3). Шаг в конец списка добавляется и сочетанием `Ctrl + Shift + D` (Windows) или `Command + Shift + D` (MacOS), см. **Горячие клавиши**.
3. Чтобы скопировать или удалить шаги, используйте кнопки в меню шага (цифра 4) либо отметьте чек-боксы шагов и нажмите кнопки в верхней панели (цифра 5).
4. Нажмите кнопку «**Сохранить**».

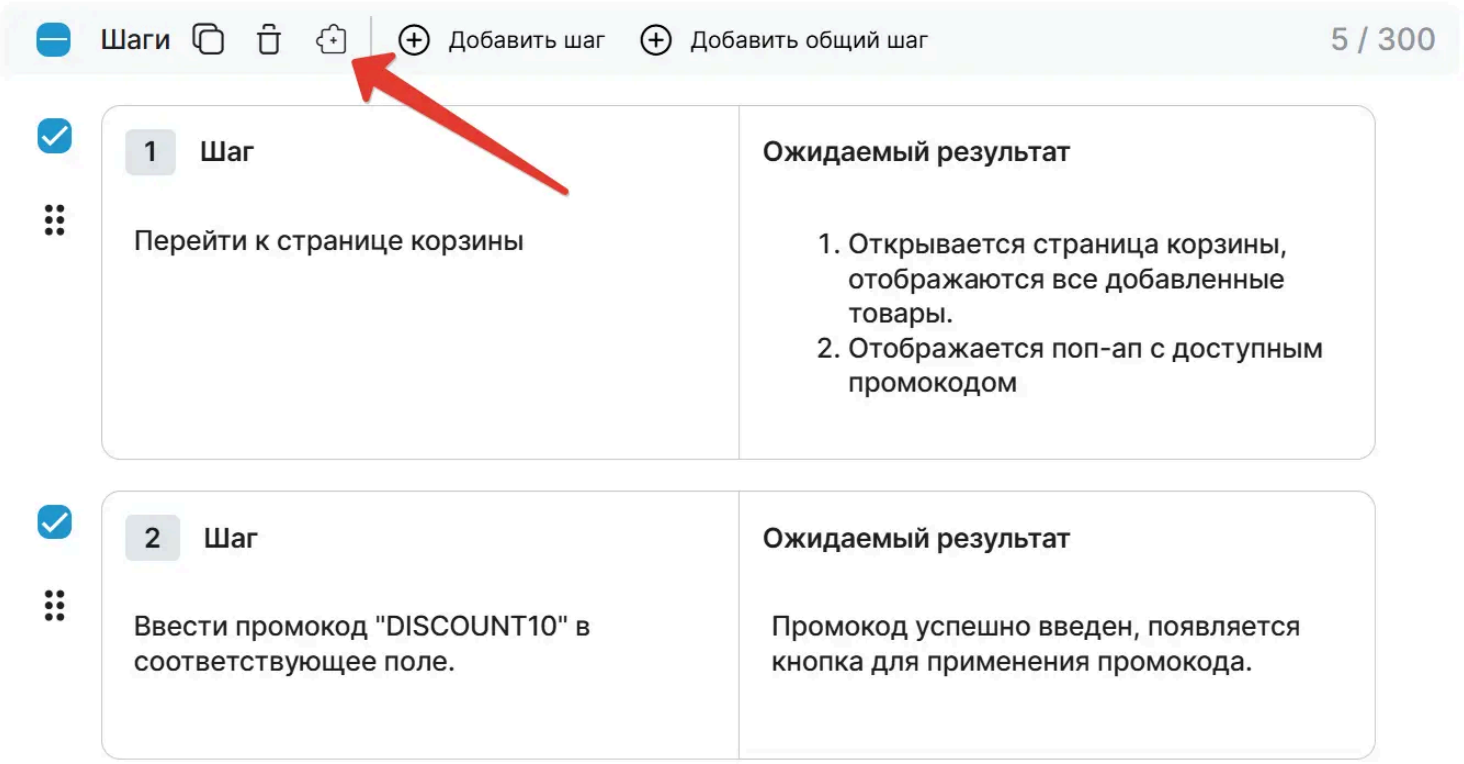
Счетчик показывает количество добавленных шагов и их лимит (цифра 6). Максимальное количество шагов в одном общем шаге — 100. У тест-кейса лимит другой — 300 шагов, см. **Панель действий с шагами**.



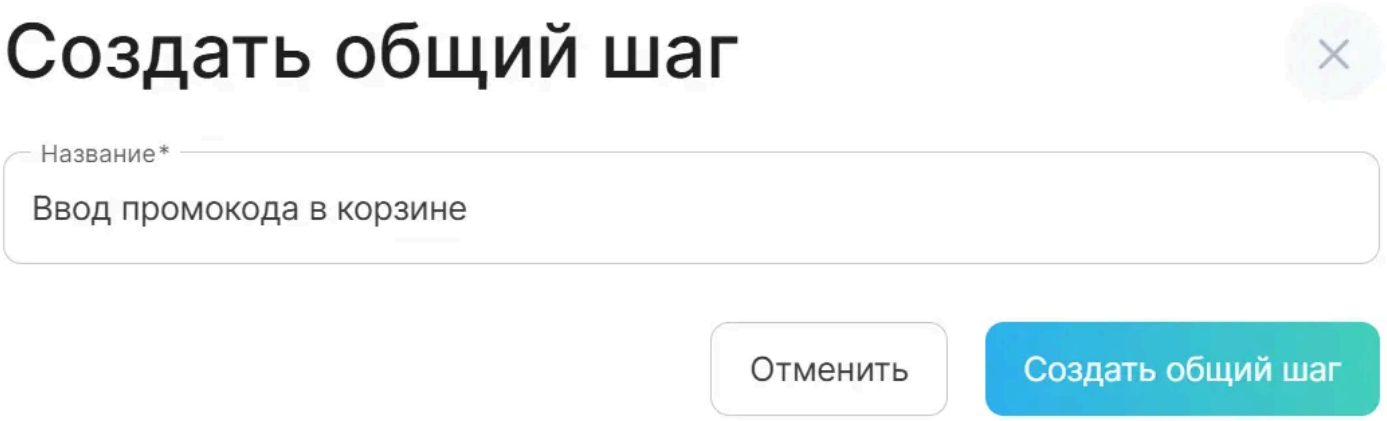
Объединение обычных шагов в редакторе кейсов

Откройте редактор тест-кейса, отметьте чек-боксы шагов, которые вы хотите объединить в общий шаг, и нажмите на кнопку создания общего шага.

Объединять можно только обычные шаги: если среди выбранных окажется общий шаг, кнопка заблокируется. Выбрать больше 100 шагов тоже нельзя — это лимит шагов в одном общем шаге.



В диалоговом окне введите имя общего шага и подтвердите его создание.

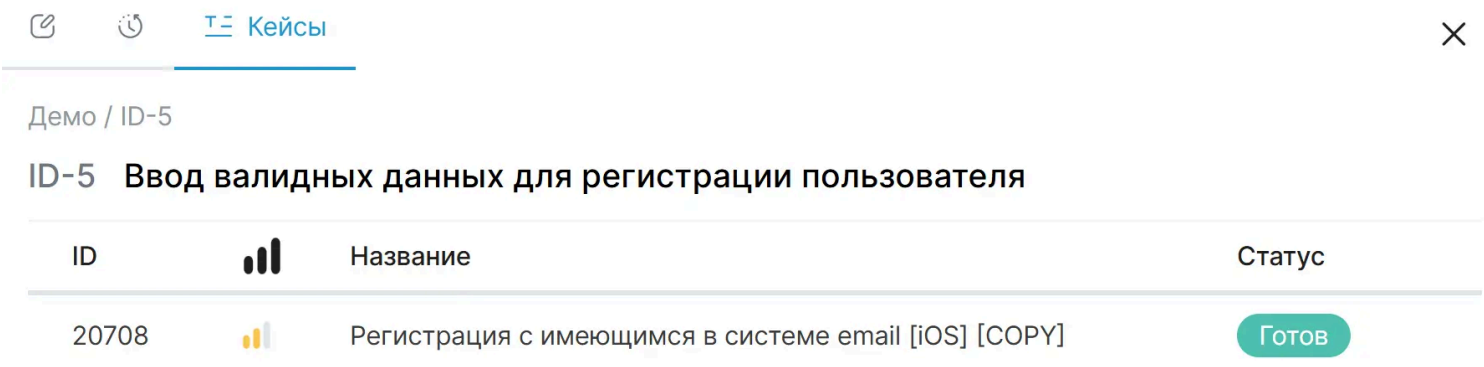


Вкладки редактора общего шага

В редакторе общего шага три вкладки:

- «Редактор» — название и шаги общего шага
- «История редактирования» — версии общего шага, просмотр и откат к предыдущей версии. Подробнее — в разделе **История редактирования**
- «Кейсы» — список тест-кейсов, в которых используется этот общий шаг. Если общий шаг будет изменен, это повлияет на все связанные тест-кейсы

У несохраненного общего шага доступна только вкладка «Редактор», остальные включаются после первого сохранения.



В редакторе тест-кейса шаги общего шага доступны только для чтения: править их можно лишь здесь, в редакторе общего шага. Подробнее — в разделе **Использование общих шагов в кейсах**.

Таблица общих шагов

Колонки таблицы: «ID», «Название», «Создатель», «Создан», «Обновлен». По каждой колонке доступна сортировка. Поиск ищет по названию общего шага.

Отметьте чек-боксы общих шагов, чтобы выполнить с ними массовые действия:

- «**Скопировать**» — копии появляются в той же таблице
- «**Удалить**»

Если общих шагов больше, чем помещается на страницу, внизу таблицы появляется кнопка «**Показать еще**».

Внимание

При удалении общие шаги перемещаются в **корзину** и хранятся там 30 дней. По истечении этого срока они удаляются без возможности восстановления.

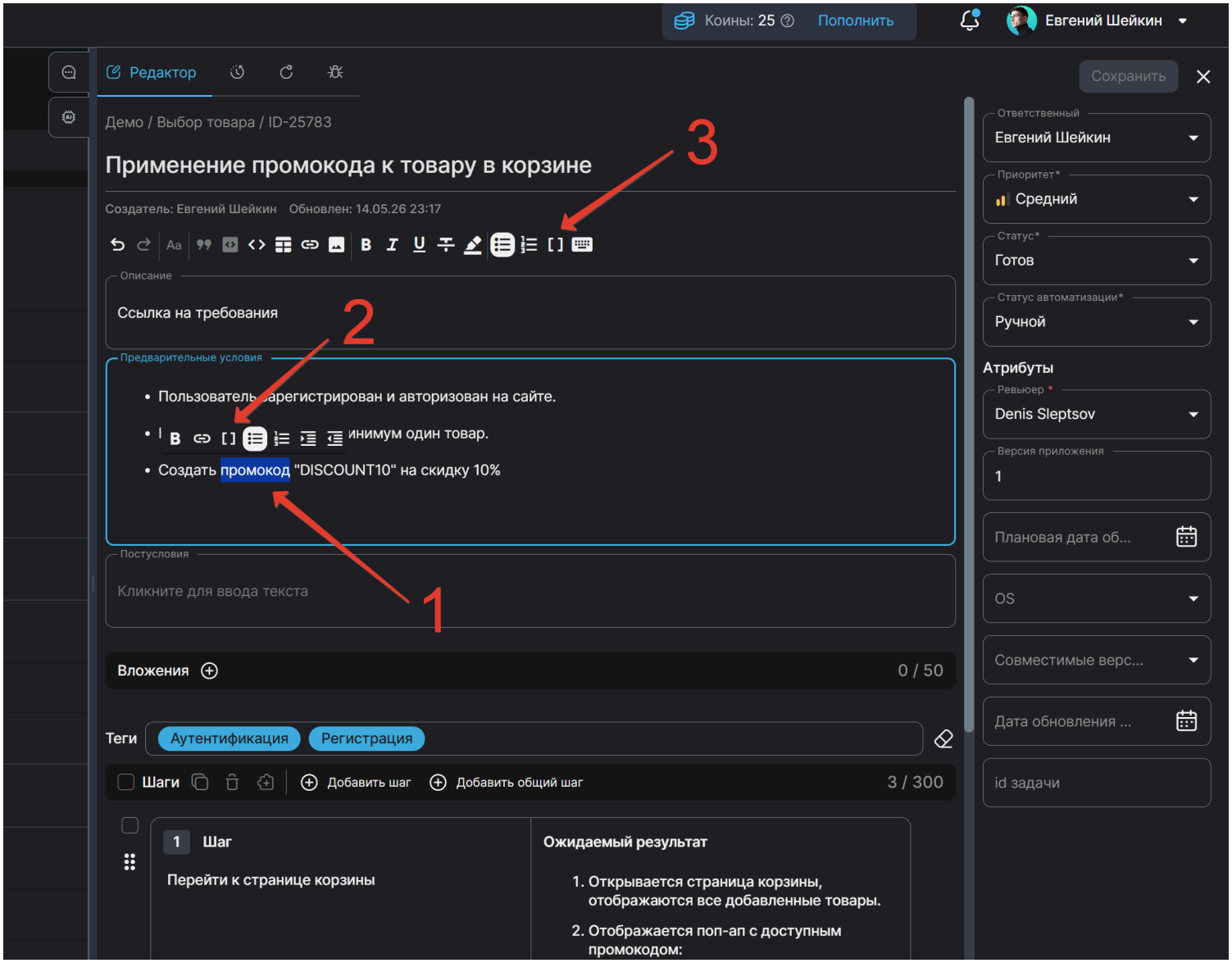
Параметризация тестов

Параметр — это именованный набор значений, который подставляется в текст тест-кейса или чек-листа: например, «Браузер» со значениями Chrome, Firefox и Safari. При добавлении параметризованного теста в **прогон** DoQA создаёт отдельный слепок на каждую комбинацию значений. Сколько именно будет слепков, задаёт **стратегия комбинаций**.

Создание параметра

В редакторе тест-кейса или чек-листа параметр можно создать тремя способами.

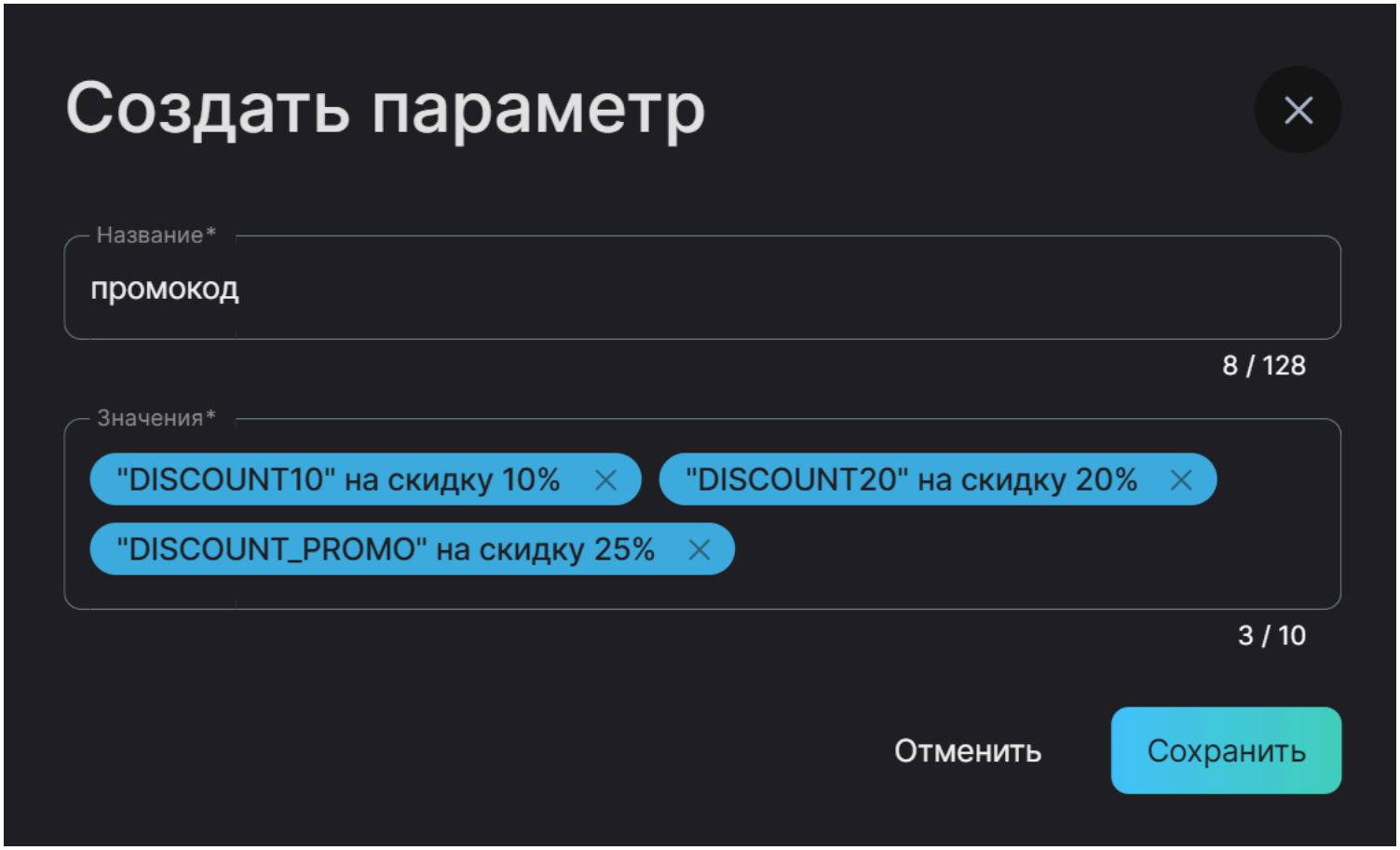
Из выделенного текста. Выделите текст, который станет именем параметра (цифра 1), и нажмите кнопку параметра во всплывающем меню (цифра 2) или на панели редактора (цифра 3).



Вводом `[]`. Наберите `[]` в тексте. Откроется список: сначала параметры этого теста, ниже — глобальные параметры пространства. Продолжайте печатать, чтобы найти нужный по имени, или выберите «Создать параметр».

Горячей клавишей. `Ctrl + Shift + K` (Windows) или `Command + Shift + K` (macOS) вставляет `[]` и открывает тот же список.

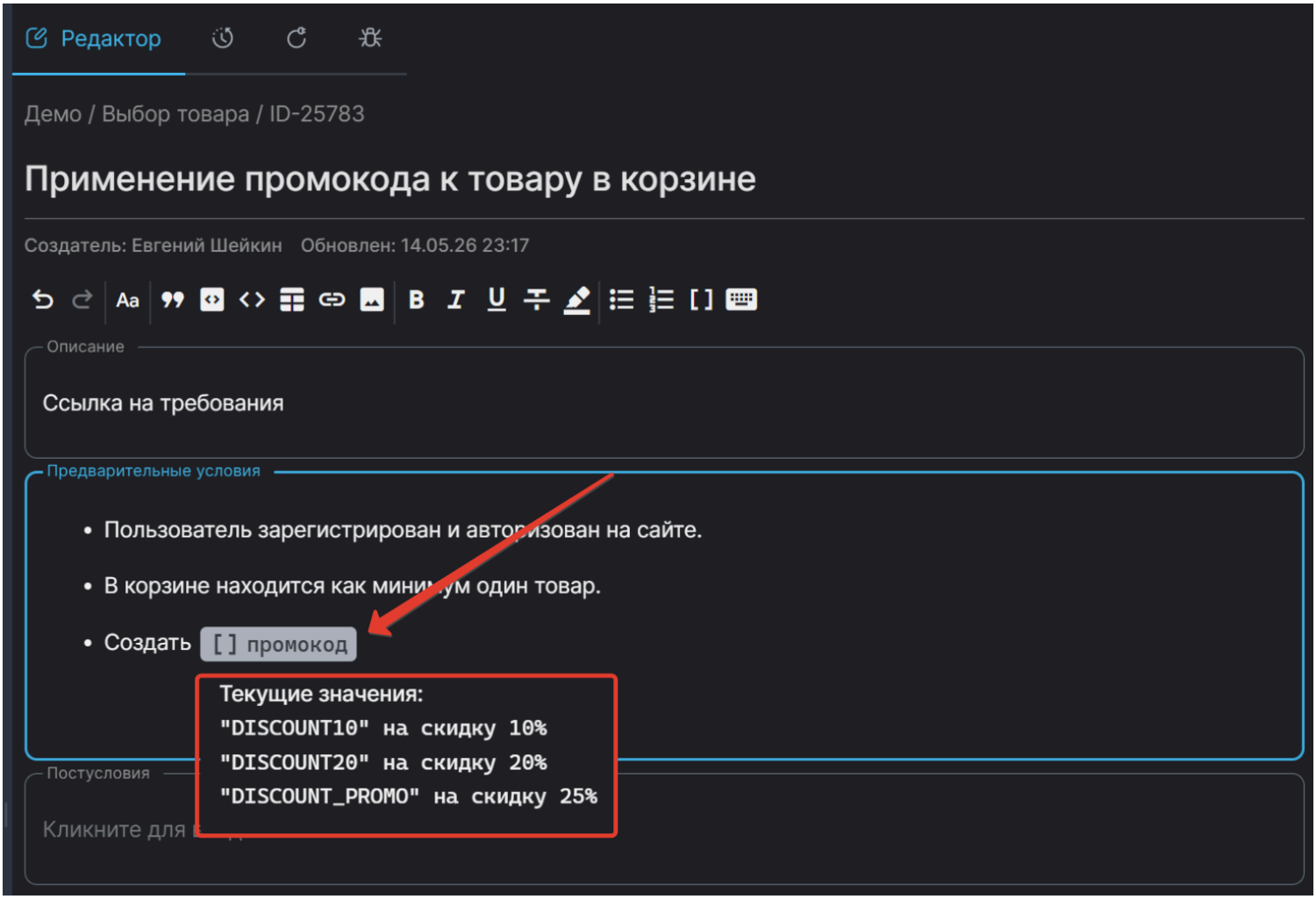
Задайте имя параметра, добавьте значения и нажмите «**Сохранить**»:



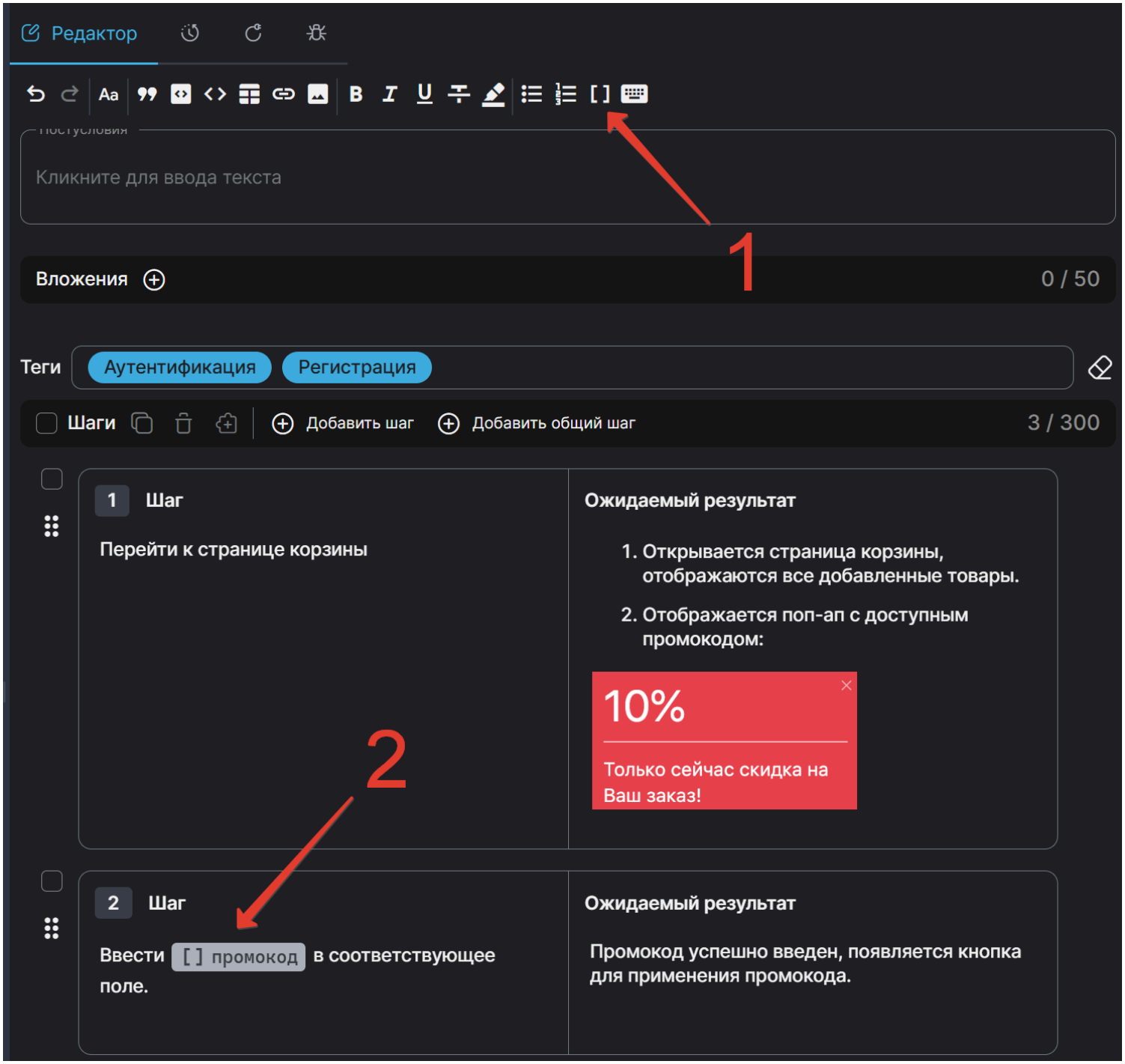
Значения добавляются по Enter, запятой или точке с запятой. Сохранить окно можно сочетанием `Ctrl + Enter` (Windows) или `Command + Enter` (macOS), закрыть без сохранения — клавишей Esc.

Ограничения: имя параметра — до 128 символов, значение — до 128 символов, до 100 значений на параметр. Хотя бы одно значение обязательно. Значения внутри параметра не должны повторяться, а имя параметра должно быть уникальным в пределах теста, в том числе между локальным и глобальным параметром.

При наведении курсора на параметр в тексте отображаются его значения:



Чтобы добавить в текст ещё одно вхождение уже созданного параметра, наберите `[]` на нужной позиции и выберите параметр из списка.



Локальные и глобальные параметры

Тип параметра выбирается в первом поле окна создания:

- «Локальный» («Только для этого тест-кейса») — параметр существует только внутри этого тест-кейса или чек-листа.
- «Глобальный» («Доступен в пространстве») — параметр хранится на уровне **пространства** и предлагается в списке по `[]` в других тест-кейсах и чек-листах этого пространства.

Глобальный параметр хранится в одном экземпляре на всё пространство: если изменить его имя или значения в одном тест-кейсе и сохранить тест, изменения увидят все тест-кейсы и чек-листы пространства, где этот параметр используется.

Глобальные параметры помечены отдельной иконкой: в списке по `[]`, в панели «Параметры» и в таблице «Комбинации значений».

Тип можно изменить при редактировании параметра. Если имя при этом не меняется, параметр не дублируется, а перепривязывается к выбранному типу.

> **Совет**
>
> Сколько слепков получится в прогоне, зависит не от параметра, а от **стратегии комбинаций**, выбранной для теста целиком.

Стратегии комбинаций

Стратегия задаётся для всего тест-кейса или чек-листа и определяет, какие сочетания значений попадут в прогон.

Откройте панель «Параметры» — закладка справа в редакторе тест-кейса или чек-листа. В панели три элемента:

- «Стратегия комбинаций» — выбор одной из четырёх стратегий. Поле неактивно, пока в тесте нет ни одного параметра со значениями.
- «**Показать комбинации**» — предпросмотр будущих слепков.
- «Используемые параметры» — список параметров теста со значениями. Клик по карточке открывает окно редактирования параметра.

Стратегия сохраняется вместе с тест-кейсом или чек-листом. По умолчанию — «Все комбинации».

Дальше в примерах используются два параметра: «Браузер» со значениями Chrome, Firefox, Safari и «ОС» со значениями Windows, Linux.

Все комбинации

Декартово произведение: каждое значение каждого параметра сочетается с каждым значением остальных. Стратегия по умолчанию. Шесть слепков:

№	Браузер	ОС
1	Chrome	Windows
2	Chrome	Linux
3	Firefox	Windows
4	Firefox	Linux
5	Safari	Windows
6	Safari	Linux

Число слепков растёт как произведение: три параметра по три значения дают уже 27 слепков.

Каждый выбор

Каждое значение каждого параметра попадает хотя бы в одну комбинацию. Число комбинаций равно длине самого длинного списка значений; более короткие списки зацикливаются с начала. Три слепка:

№	Браузер	ОС
1	Chrome	Windows
2	Firefox	Linux
3	Safari	Windows

Windows повторяется в третьей строке, потому что список «ОС» короче и начался заново.

Линейное сопоставление

Значения сопоставляются строго по порядку: первое с первым, второе со вторым. Списки не зацикливаются: если у параметра значения кончились, в комбинации будет прочерк. Три слепка:

№	Браузер	ОС
1	Chrome	Windows
2	Firefox	Linux
3	Safari	—

Стратегия подходит, когда значения разных параметров связаны попарно и порядок в списках осмыслен (например, логин и соответствующий ему пароль).

Попарное тестирование (pairwise)

Каждая пара значений любых двух параметров встречается хотя бы в одной комбинации. Полного перебора при этом не происходит.

На двух параметрах pairwise совпадает со «Всеми комбинациями»: все шесть строк и есть все пары. Выигрыш появляется начиная с трёх параметров. Если к «Браузеру» и «ОС» добавить параметр «Язык» со значениями RU и EN, «Все комбинации» дадут 12 слепков, а pairwise — 6:

№	Браузер	ОС	Язык
1	Chrome	Windows	RU
2	Chrome	Linux	EN
3	Firefox	Windows	EN
4	Firefox	Linux	RU
5	Safari	Windows	RU
6	Safari	Linux	EN

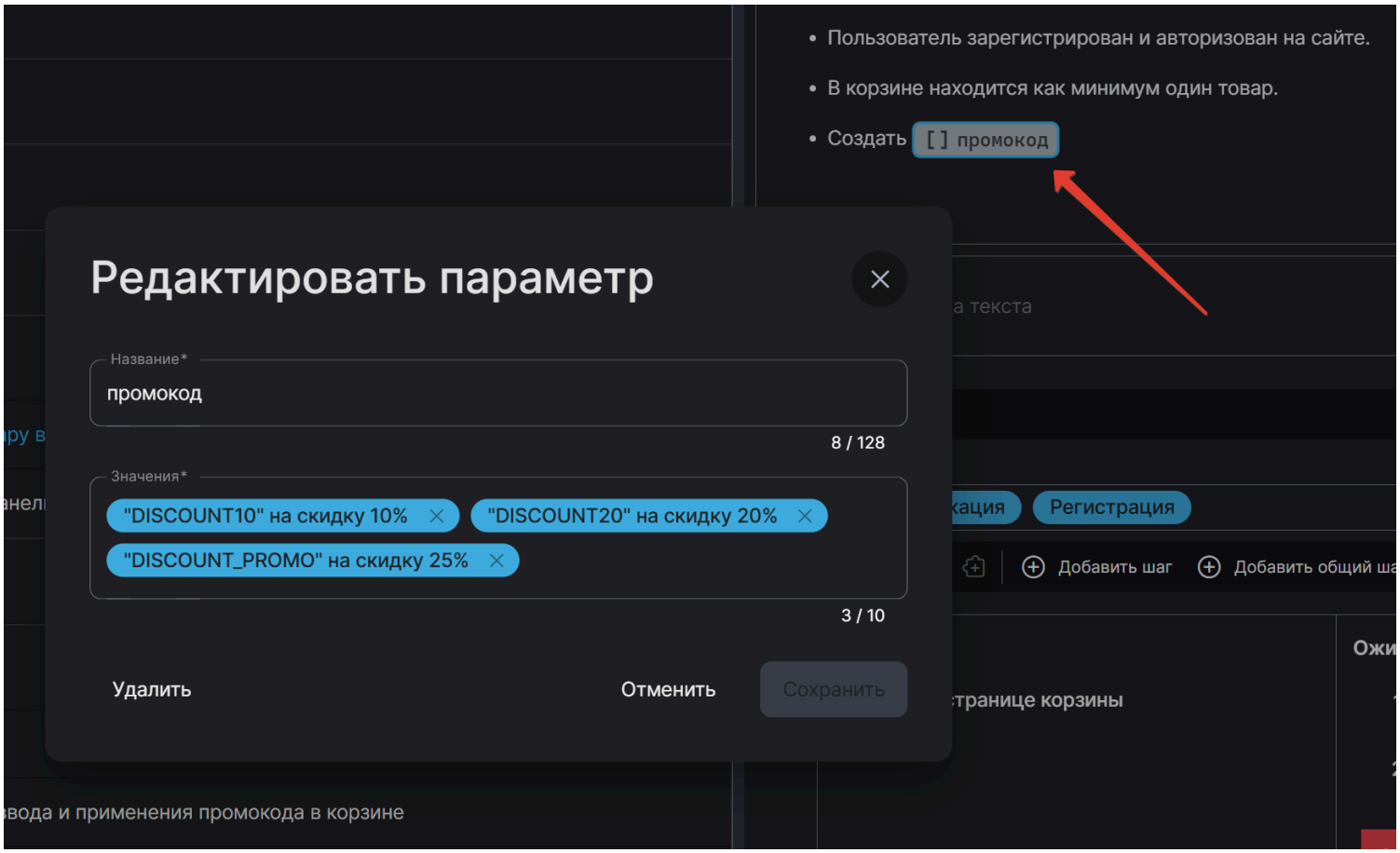
Показать комбинации

Кнопка «**Показать комбинации**» открывает окно «Комбинации значений»: строка таблицы — один будущий слепок, колонки — параметры теста. Прочерк означает, что для этой комбинации у параметра нет значения. У глобальных параметров рядом с именем колонки выводится иконка.

Таблица считается тем же алгоритмом, что и разворачивание теста в прогоне, поэтому по ней можно оценить число слепков до того, как тест будет добавлен в прогон.

Изменение параметра и его значений

Кликните по параметру в тексте левой кнопкой мыши или по его карточке в панели «Параметры». Откроется окно редактирования. Изменения затрагивают все вхождения параметра в тесте.



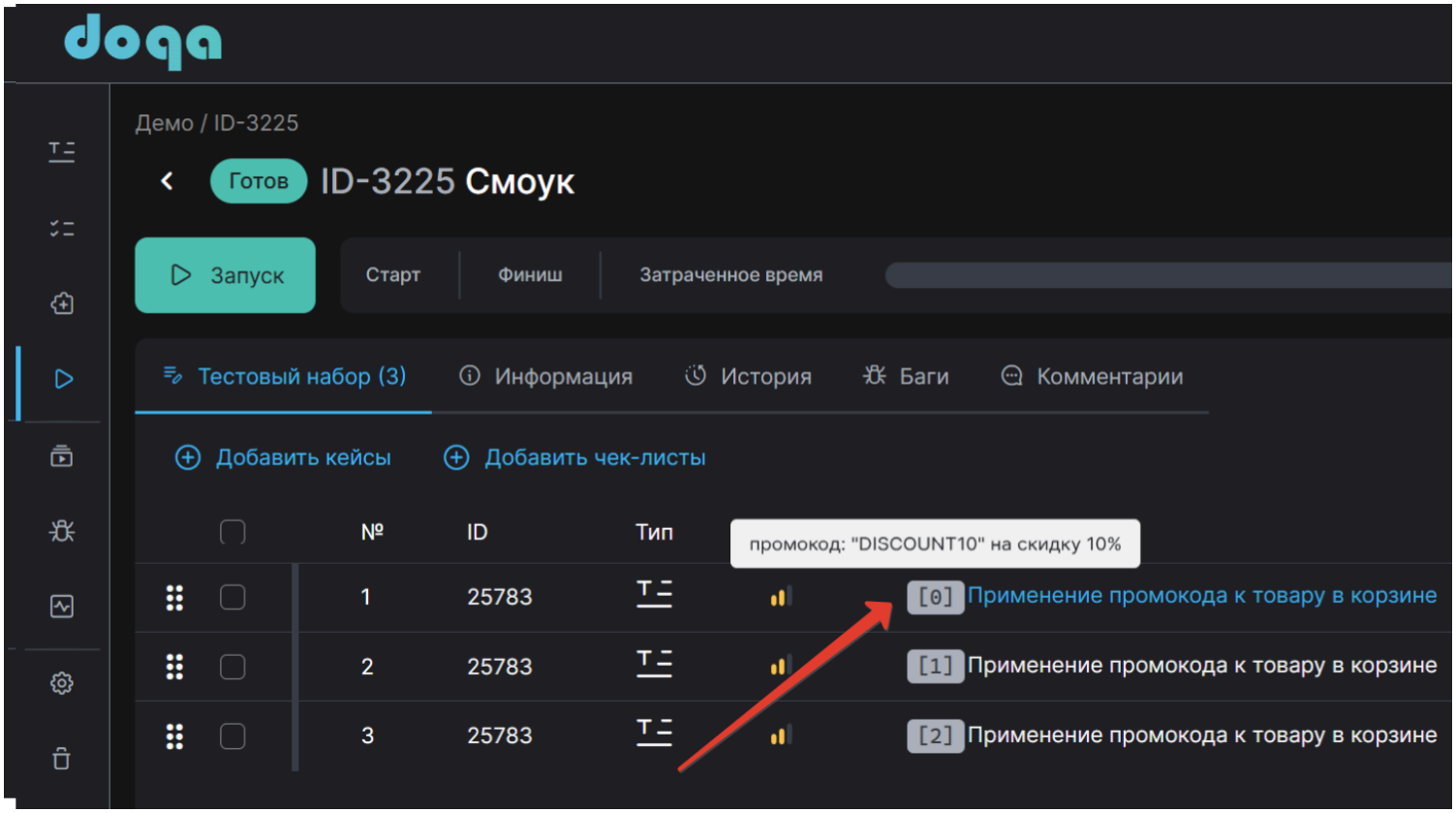
Кнопка «**Удалить**» в окне редактирования удаляет параметр вместе со всеми его вхождениями в тексте. DoQA запросит подтверждение.

Если удалить из текста последнее вхождение параметра, DoQA также запросит подтверждение: вместе с вхождением удаляется сам параметр.

Прохождение параметризованного теста в прогоне

При добавлении параметризованного теста в прогон DoQA создаёт отдельный слепок на каждую комбинацию значений, рассчитанную по стратегии, которая сохранена в самом тест-кейсе или чек-листе. В описании, предусловиях и ожидаемом результате каждого слепка параметры заменены на значения своей комбинации.

Слепки помечены бейджем вида `[N]` — это номер комбинации. При наведении курсора отображается состав комбинации: «имя параметра: значение» для всех параметров теста.



Тот же бейдж выводится в **истории запуска** тест-кейса и в кратком отчёте прогона.

Атрибуты

Атрибут — дополнительное поле тест-кейса или чек-листа: версия приложения, ОС, ревьюер. Набор атрибутов свой у каждого проекта: их создаёт администратор, он же задаёт порядок полей в редакторе и обязательность.

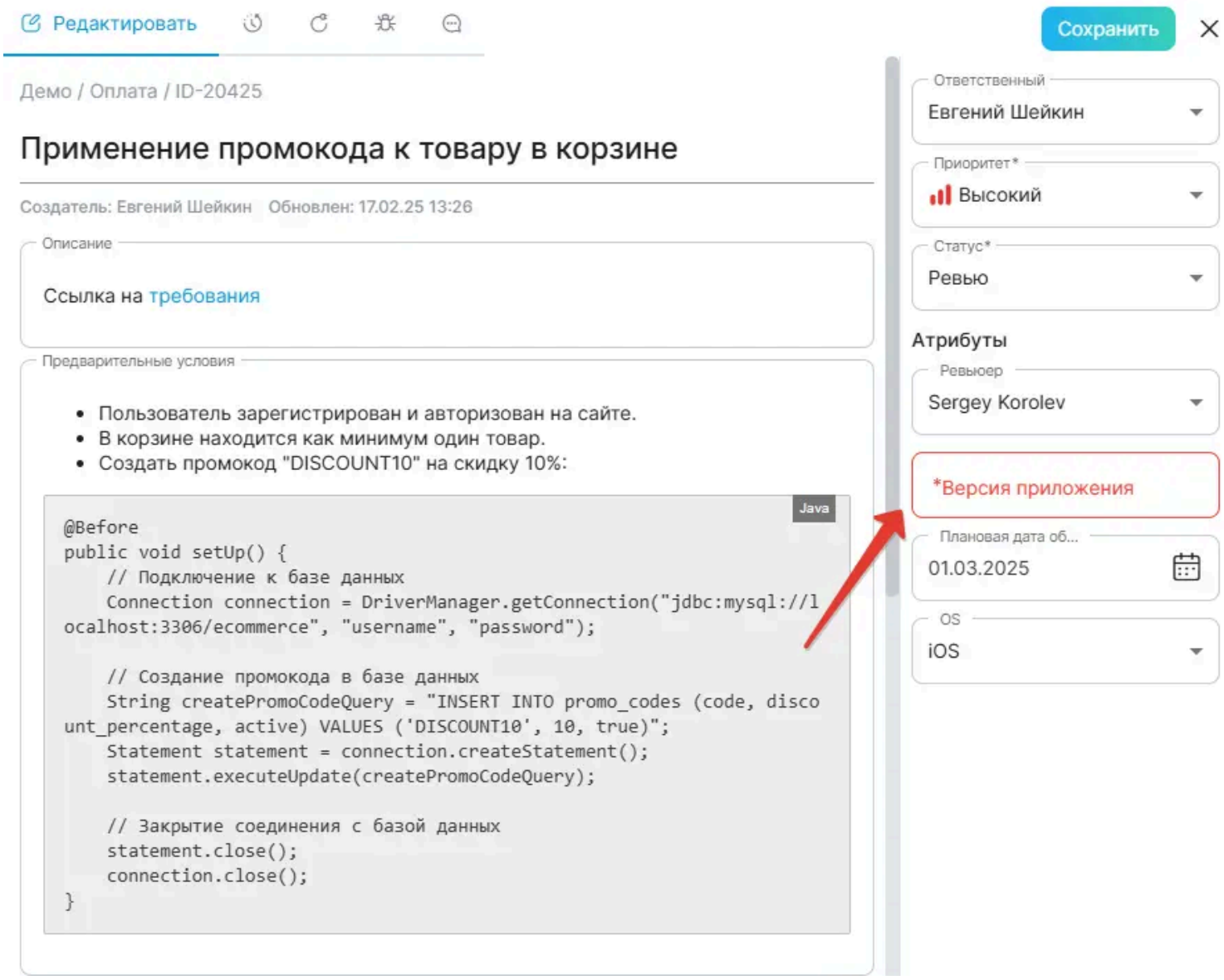
В DoQA пять типов атрибутов:

Тип	Что принимает
«Строка»	Произвольный текст, до 255 символов
«Дата»	Дату из календаря
«Вариант на выбор»	Один из заранее заданных вариантов
«Множественный вариант на выбор»	Несколько заранее заданных вариантов
«Пользователь»	Пользователя DoQA из списка

Создание, порядок и обязательность атрибутов описаны в разделе **Атрибуты в настройках администратора**.

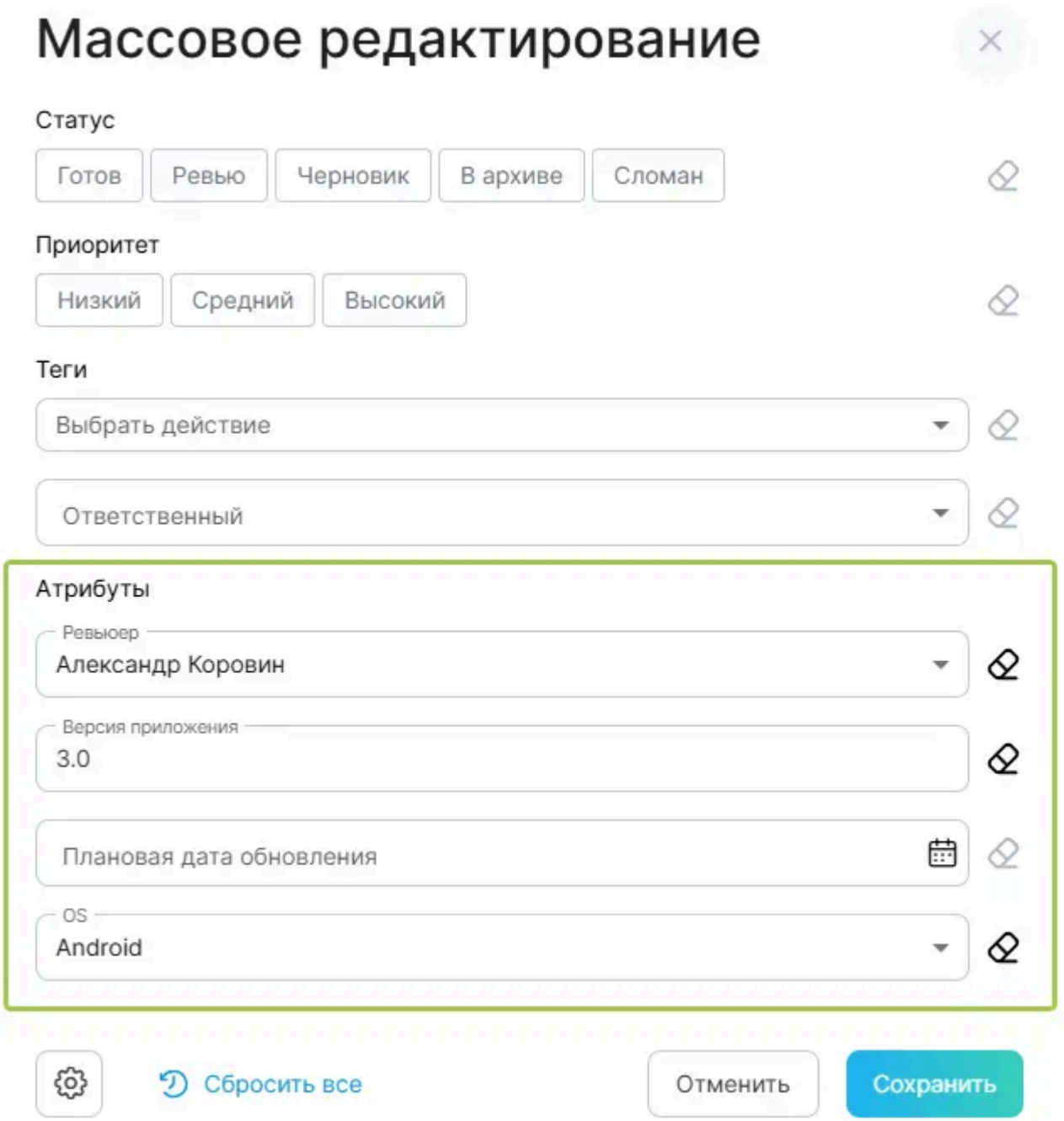
Заполнение атрибутов

Если в проекте созданы атрибуты, при создании и редактировании тест-кейсов и чек-листов их значения выбираются в редакторе, в блоке «Атрибуты». Без заполнения обязательного атрибута тест-кейс или чек-лист сохранить не получится.



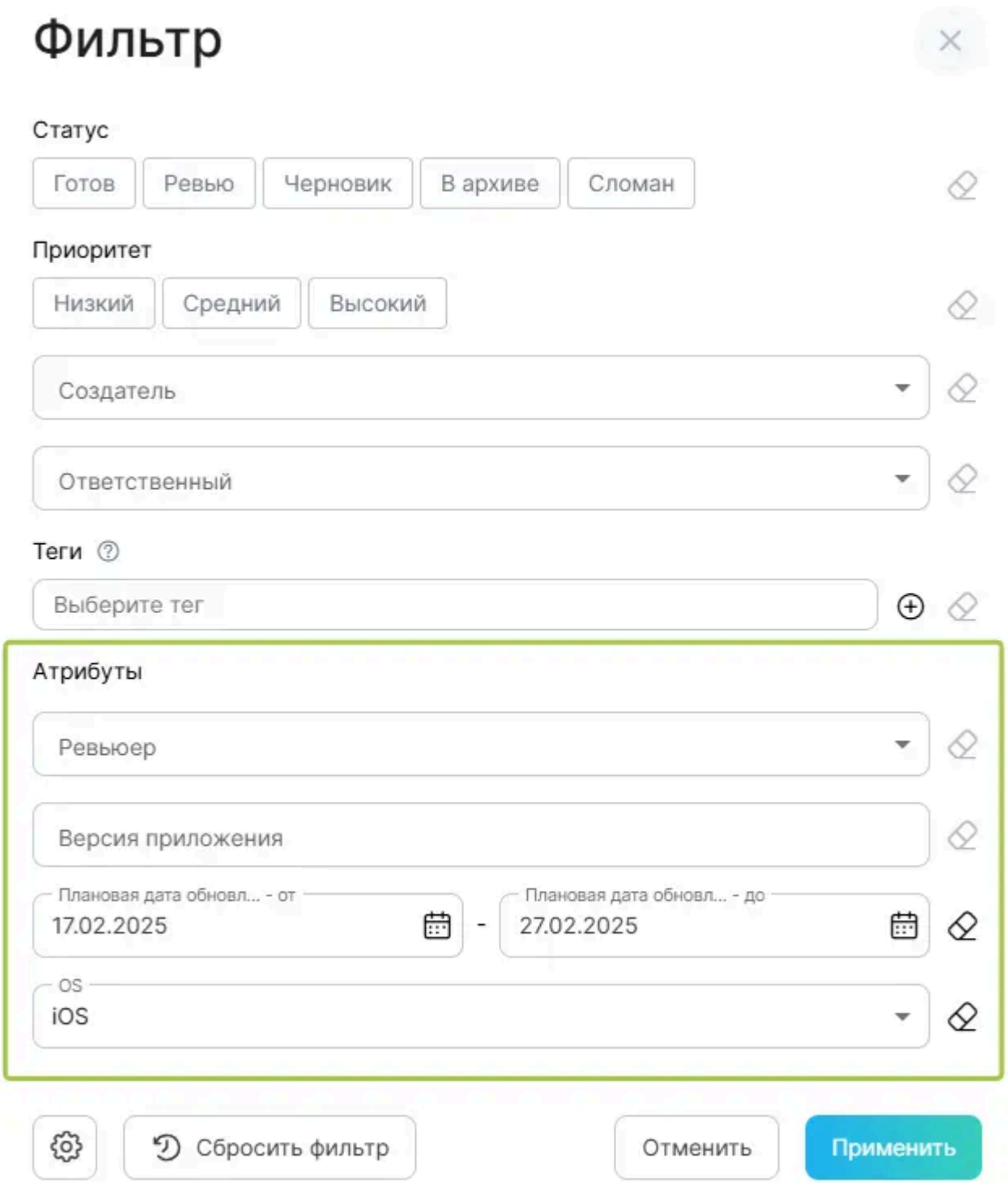
Массовое редактирование

Назначить атрибуты нескольким тест-кейсам или чек-листам сразу можно через окно массового редактирования:



Фильтрация по атрибутам

Атрибуты проекта выводятся в окне **фильтра** отдельным блоком «Атрибуты», ниже стандартных полей. Тип поля зависит от типа атрибута: «Строка» — текстовое поле, «Дата» — диапазон дат, остальные — выбор из списка.


Фильтр
Статус
Готов
Ревью
Черновик
В архиве
Сломан
Приоритет
Низкий
Средний
Высокий
Создатель
Ответственный
Теги
Выберите тег
Атрибуты
Ревьюер
Версия приложения
Плановая дата обновл... - от
17.02.2025
-
Плановая дата обновл... - до
27.02.2025
OS
iOS
Сбросить фильтр
Отменить
Применить

Теги

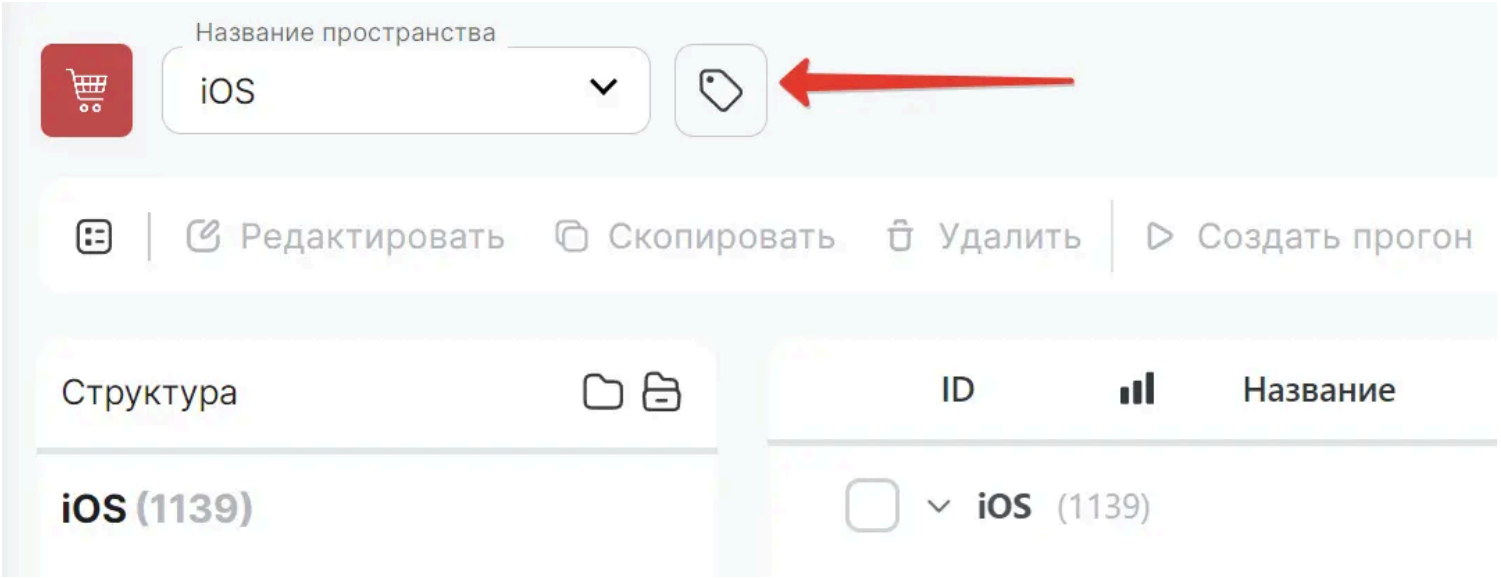
К каждому тест-кейсу, чек-листу или прогону можно добавить теги. Это помогает классифицировать написанную документацию и облегчает ее поиск.

Ключи и теги создаются в пределах **пространства**: в другом пространстве будет свой набор ключей.

Создание ключей и тегов

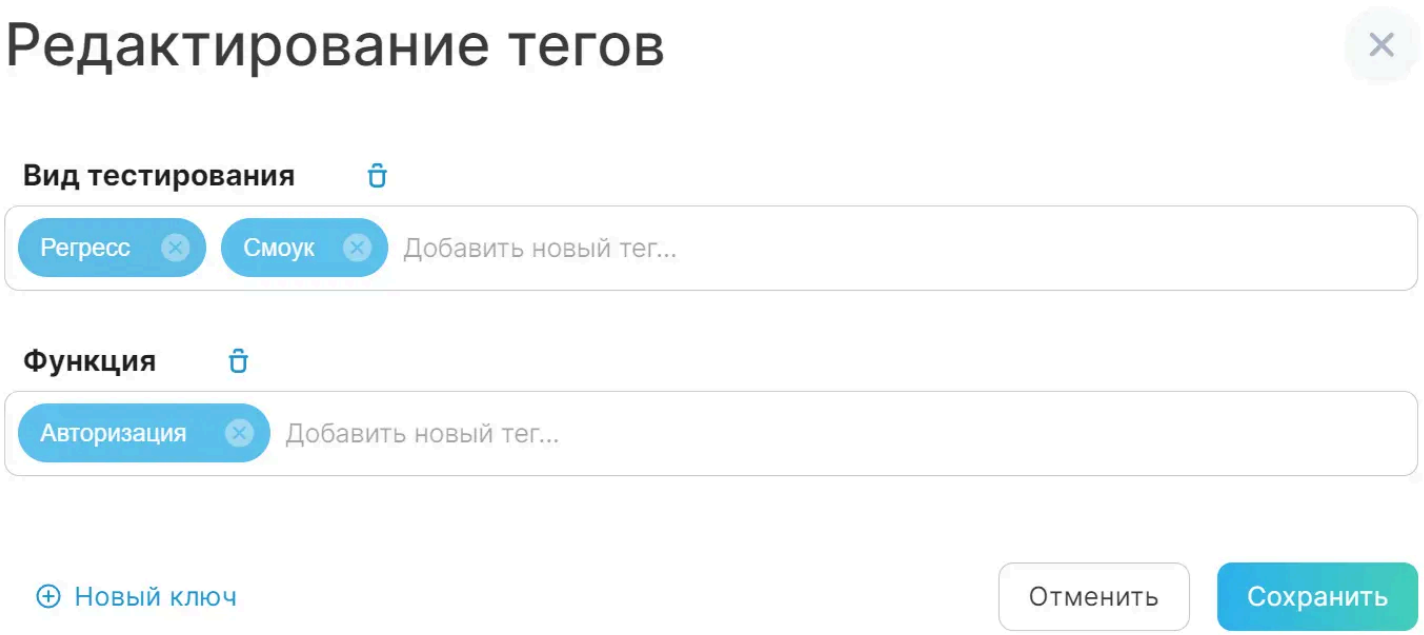
Создание ключа

Чтобы создать тег, сначала нужно создать ключ. Для этого откройте редактор: его иконка есть во вкладках «Кейсы», «Чек-листы» и «Прогоны».



Откроется окно «Редактирование тегов». Ключ добавляется кнопкой «**Новый ключ**». Она доступна пользователям с правом на редактирование ключей.

Один ключ может содержать несколько тегов.



Ограничения: имя ключа и имя тега — до 255 символов, и ни то, ни другое не может содержать символы `:` `,` `|` `&`. Имена ключей уникальны в пространстве, имена тегов — внутри своего ключа. Ключ без единого тега сохранить нельзя.

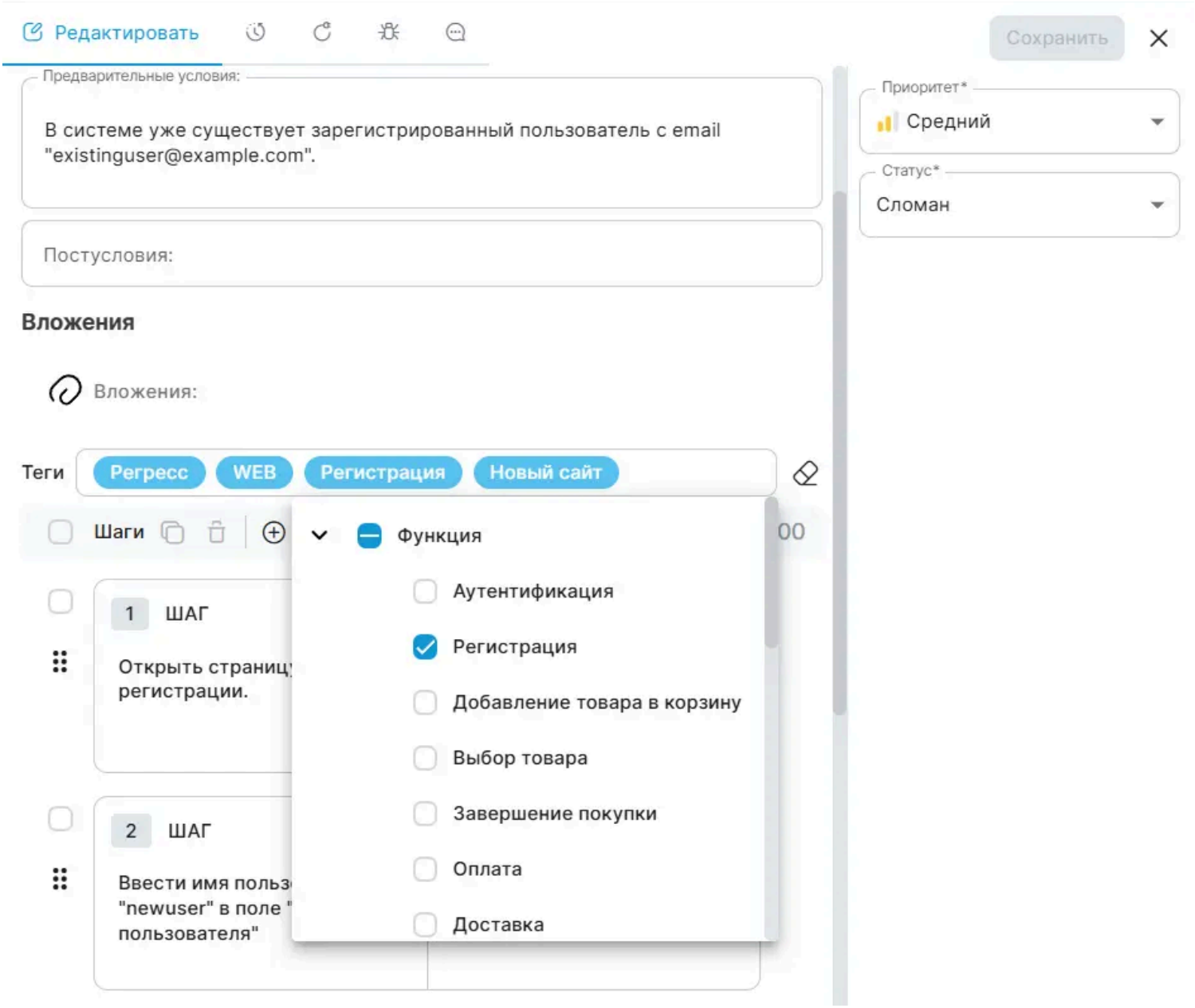
Присвоение тегов

После создания тегов их можно присваивать кейсам, чек-листам и прогонам.

Способ 1: Присвоение тега в редакторе

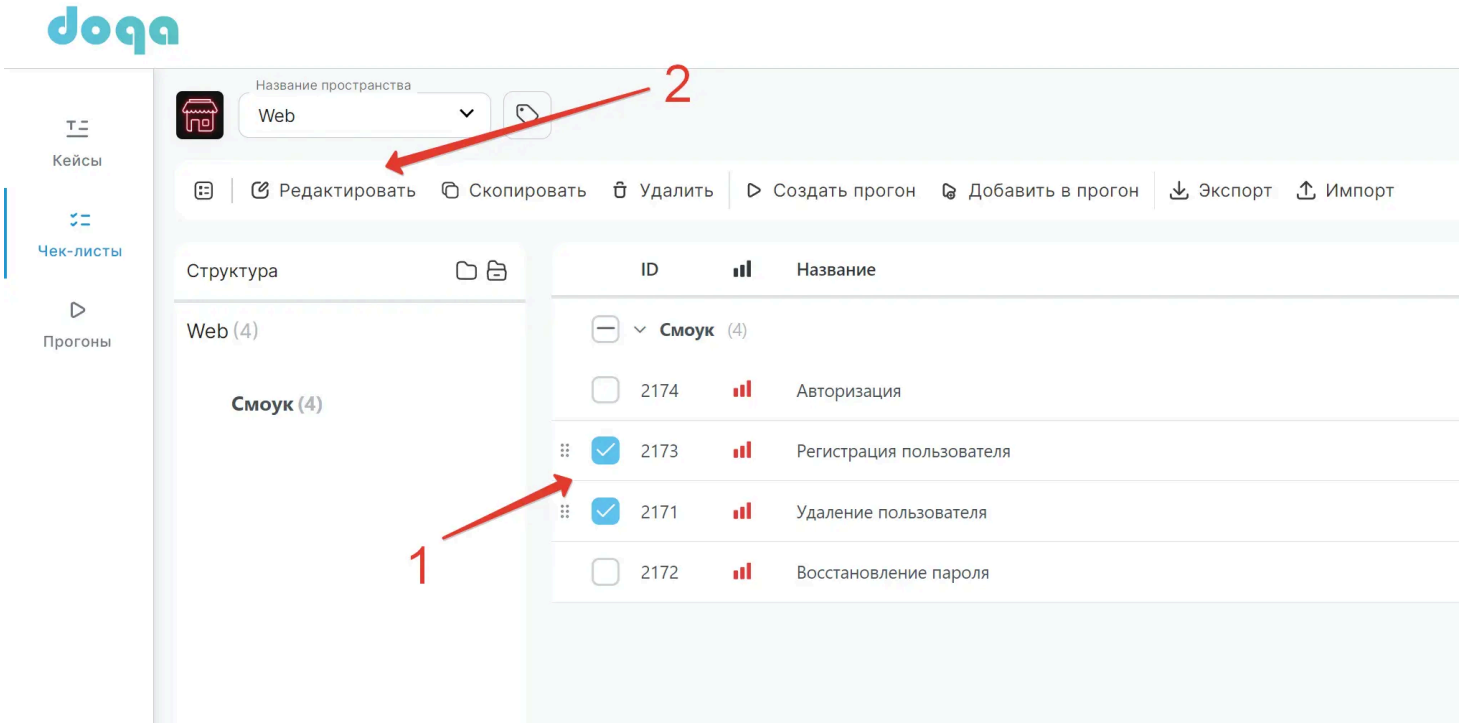
Ниже показано, как добавить тег к тест-кейсу. Чек-листам и прогонам теги можно присвоить так же.

Нажмите на поле выбора тегов и отметьте чек-боксы.

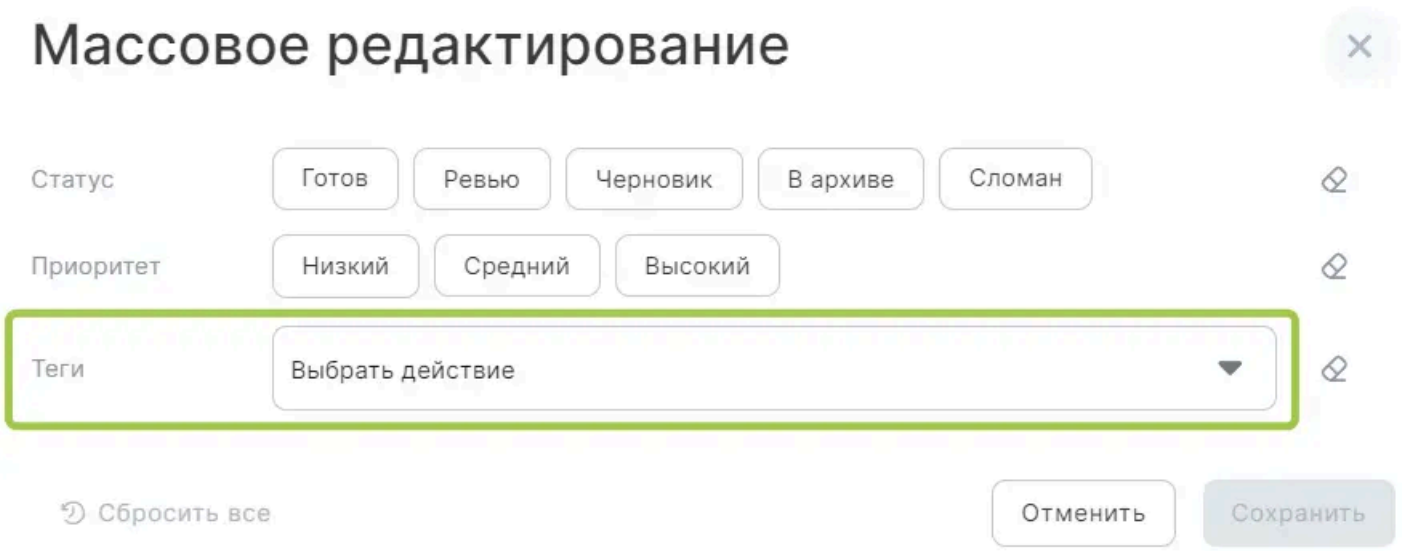


Способ 2: Присвоение тегов через массовое редактирование

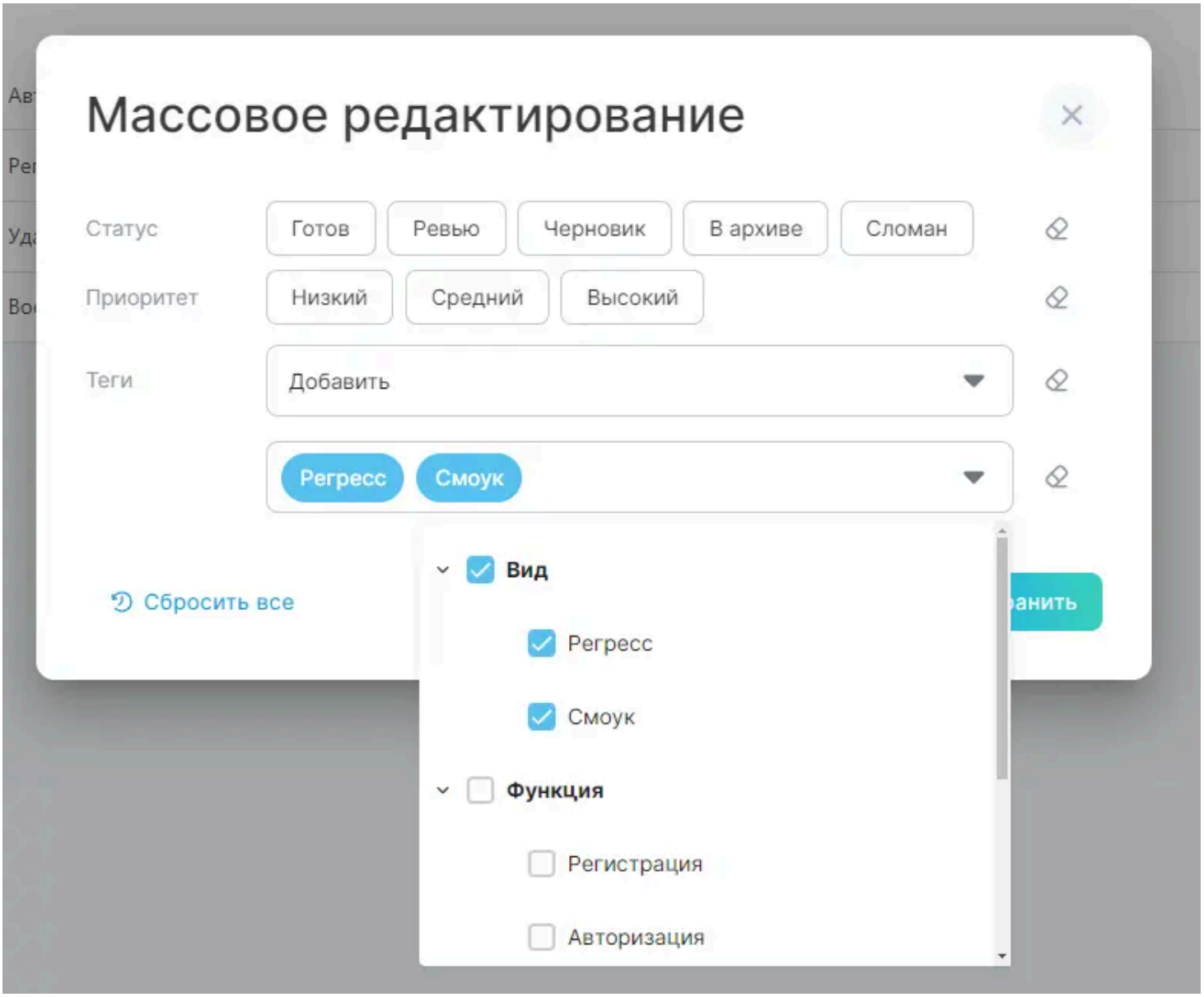
1. В таблице кейсов или чек-листов отметьте чек-боксы нужных элементов (цифра 1) и нажмите на кнопку «**Редактировать**» (цифра 2).



2. В окне массового редактирования выберите действие с тегами (добавление новых тегов, удаление ранее присвоенных тегов или очистка от всех тегов).



3. При выборе действий «Добавить» или «Удалить» откроется дополнительное поле с деревом тегов.



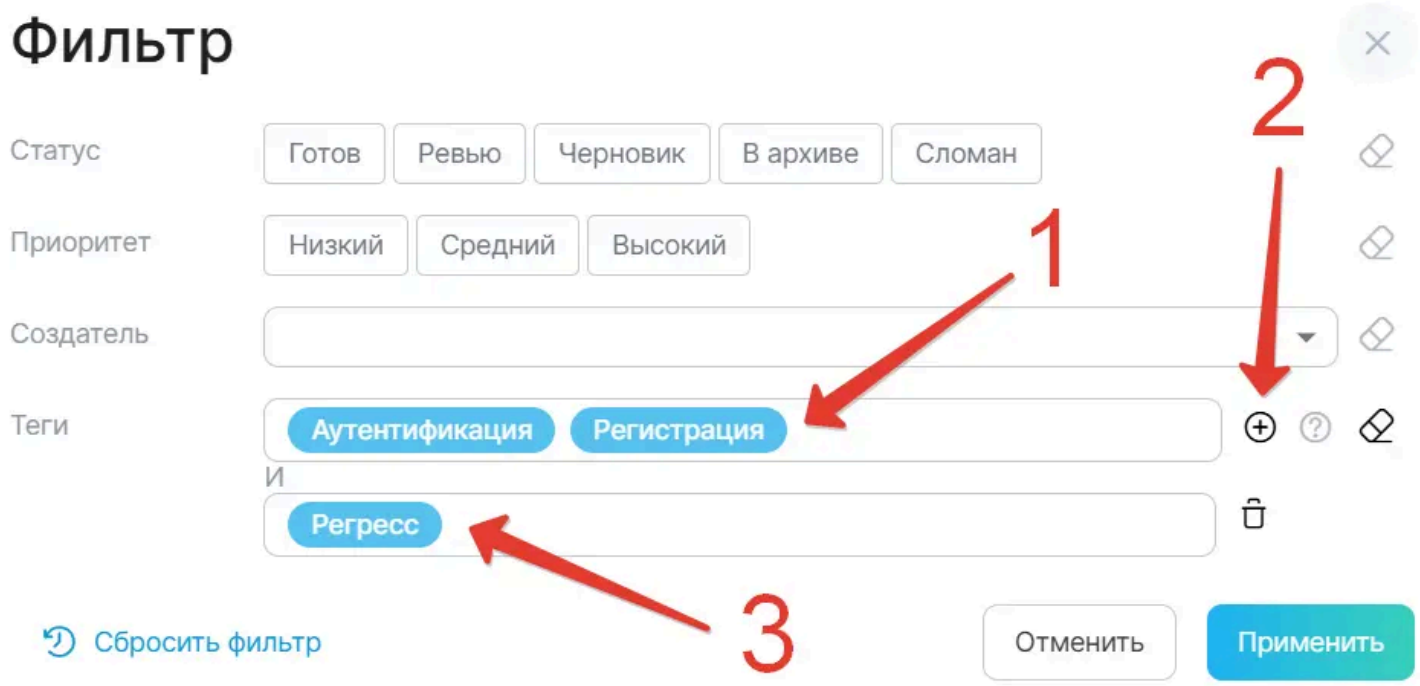
4. Для действий «Добавить» и «Удалить» отметьте нужные теги в дереве, затем нажмите «**Сохранить**». Для действия «Очистить» дерево тегов не открывается: нажмите «**Сохранить**» сразу, дополнительного подтверждения не запрашивается.

Фильтр по тегам

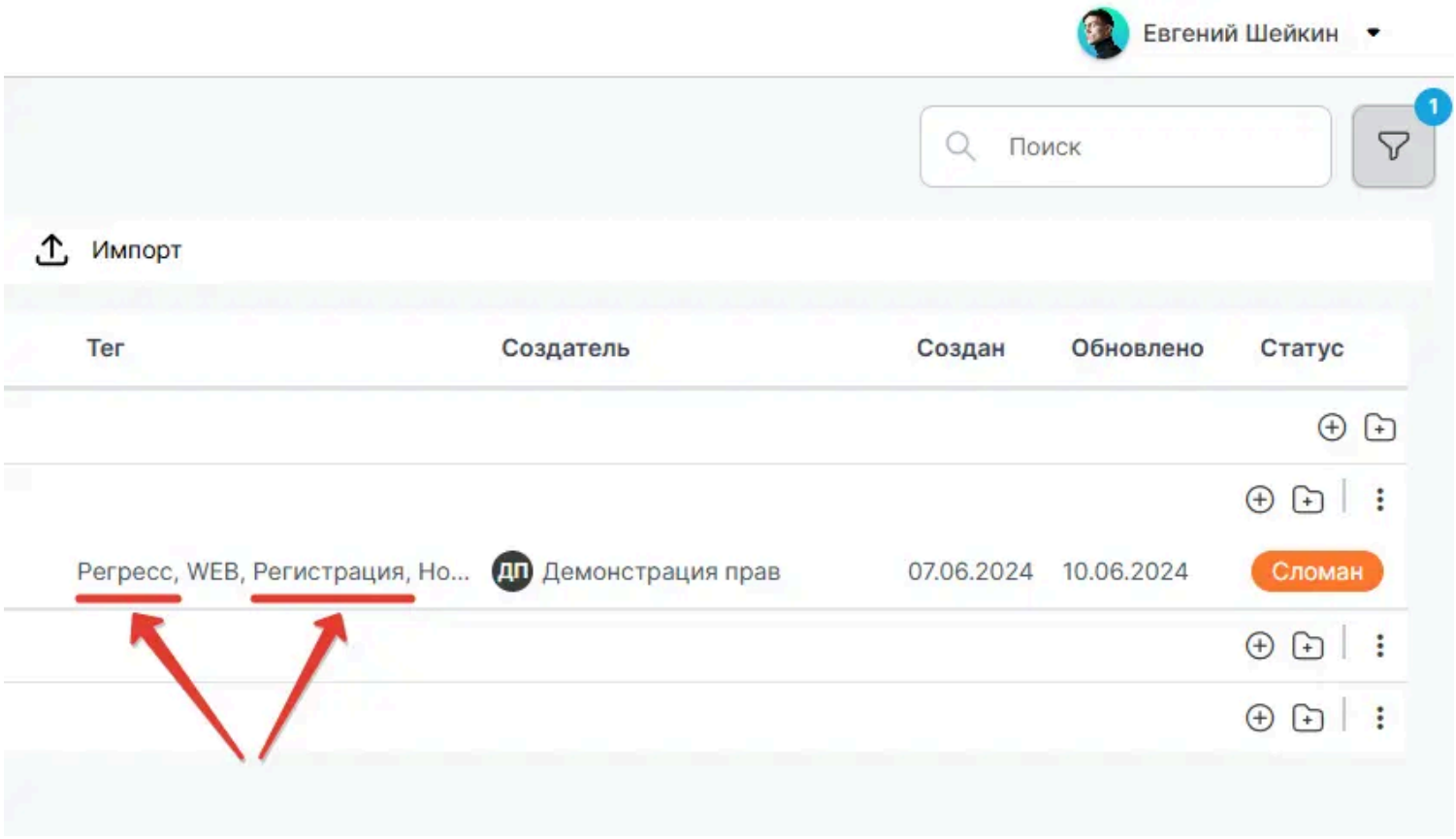
В окне **фильтра** теги задаются строками, и логика поиска зависит от того, как они разложены по строкам:

- Несколько тегов **в одной строке** — логика «ИЛИ»: в выборку попадут элементы, имеющие любой из выбранных тегов (цифра 1).
- Теги **в разных строках** — логика «И»: в выборку попадут только элементы, имеющие сочетание тегов из всех строк.

Чтобы искать по сочетанию тегов, добавьте дополнительную строку кнопкой «+» (цифра 2) и выберите теги (цифра 3). Лишнюю строку можно удалить кнопкой с корзиной. Ту же подсказку про «ИЛИ» и «И» показывает иконка со знаком вопроса рядом с полем «Теги» в окне фильтра.



В этом примере в выборке отображается элемент, у которого одновременно есть теги «Регистрация» и «Регресс».


Евгений Шейкин
Поиск
1
Импорт
Тег
Создатель
Создан
Обновлено
Статус
Регресс, WEB, Регистрация, Но...
ДП
Демонстрация прав
07.06.2024
10.06.2024
Сломан

AI-функции

DoQA умеет создавать тестовую документацию по текстовому описанию и проверять уже написанные тест-кейсы.

Все AI-функции платные. И в облаке, и в серверной версии можно подключить своего LLM-провайдера: тогда функции расходуют его токены и оплачиваются по его тарифу. В облаке есть альтернатива — встроенный провайдер DoQA. Он работает по умолчанию и списывает DoQA-коины: их покупают пакетами, отдельно от лицензий.

Доступны шесть функций (цены указаны в коинах встроенного провайдера):

Функция	Где вызывается	Стоимость
Генерация тест-кейса	список тест-кейсов	6 коинов
Генерация набора тест-кейсов	список тест-кейсов	30 коинов
Генерация чек-листа	список чек-листов	6 коинов
AI анализ тест-кейса	редактор тест-кейса	13 коинов
Генерация названия баг-репорта	форма баг-репорта в прогоне	2 коина
Генерация тестов по требованию	список требований	6 или 30 коинов

Выбор LLM-провайдера, цены и история запросов описаны в разделе **Настройки DoQA AI**.

Что нужно для работы AI-функций

- **Права.** Генерация тест-кейсов, чек-листов и AI-анализ доступны тем, у кого есть право `ai.use` и право на изменение соответствующей документации в пространстве, а также лицензия. Генерация названия баг-репорта - тем, кто может создавать баг-репорты в прогоне. Подробнее - в разделе **Роли и права**.
- **Встроенный провайдер DoQA.** На балансе должны быть коины. Текущий баланс показан в шапке приложения. Если коинов не хватает, операция прерывается и в **истории** появляется запись со статусом «Прервано».
- **Свой провайдер.** Администратор выбирает провайдера в настройках AI и заполняет поля подключения — см. **Подключение своего провайдера**. Коины не списываются: запросы

оплачиваются по тарифу провайдера. Пока провайдер не настроен, все кнопки AI неактивны с подсказкой «AI недоступен: не настроен провайдер в настройках администратора».

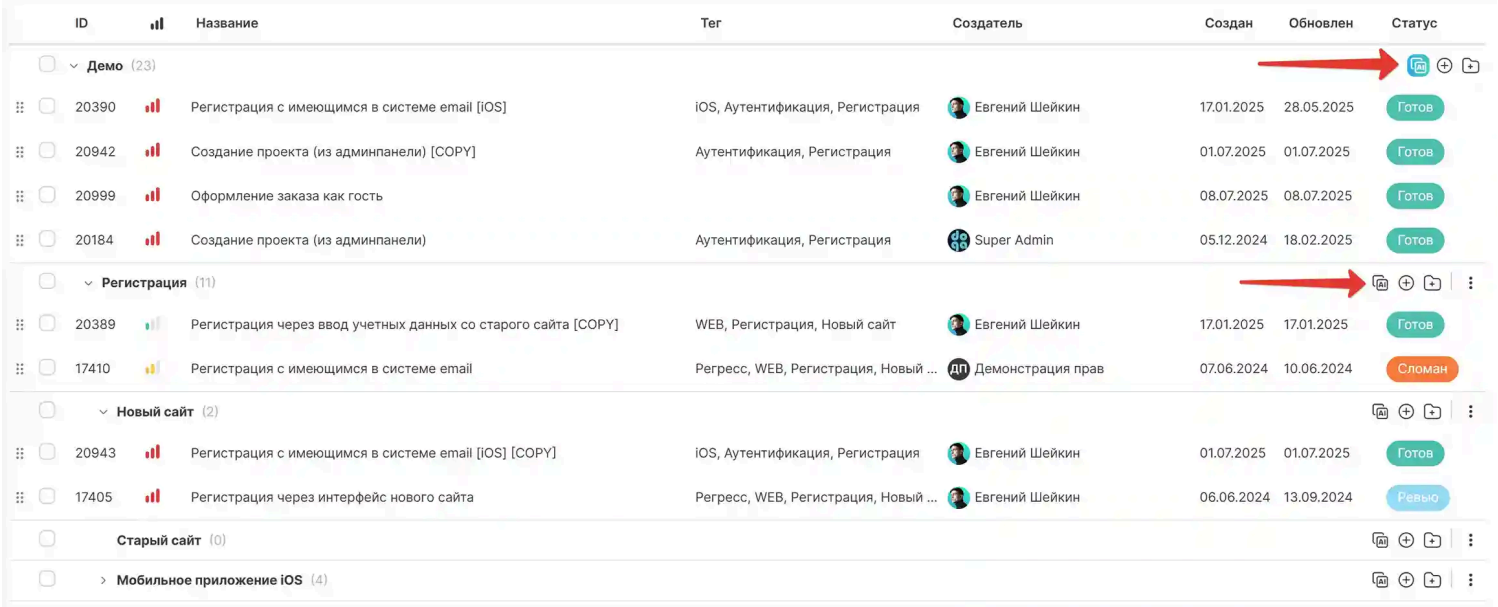
Администратор может изменить промпты AI-функций под ваши стандарты ведения документации — см. **Промпты**.

Генерация тест-кейса

DoQA создаёт один **тест-кейс** по вашему описанию: заполняет название, описание, предусловия и шаги с ожидаемыми результатами.

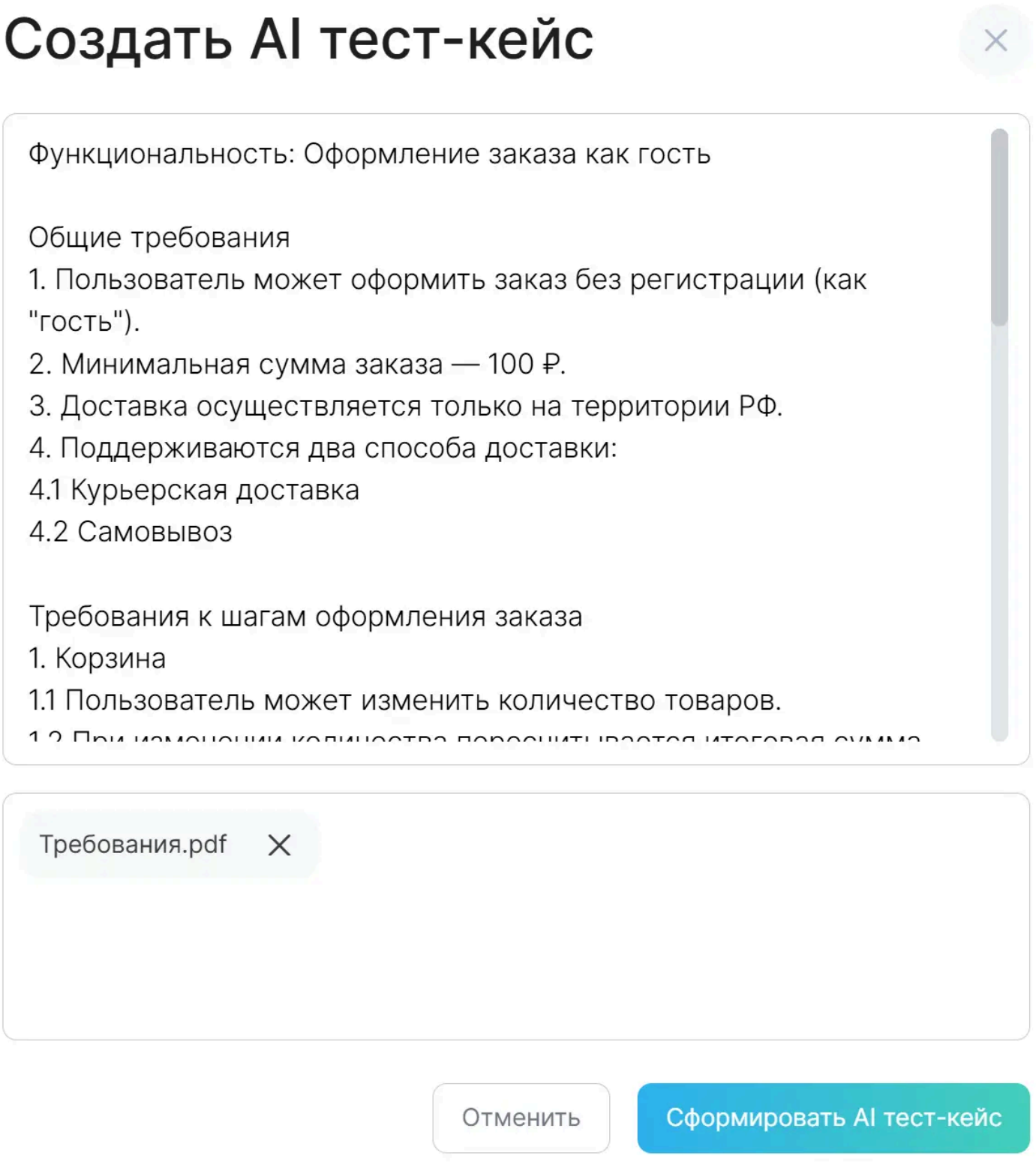
Вызвать генерацию можно двумя способами:

1. Нажмите «**Создать ТК**» в правом верхнем углу списка тест-кейсов и выберите в меню «С помощью AI». Кейс будет создан в текущей **папке**.
2. Либо наведите курсор на строку нужной папки и нажмите иконку AI.



В диалоге «AI-генерация» опишите в поле контекста, что нужно проверить. Поле обязательное: пока оно пустое, кнопка генерации неактивна. Дополнительно можно приложить один файл в формате PDF или DOCX размером до 10 МБ. Нажмите «**Сформировать AI тест-кейс**».

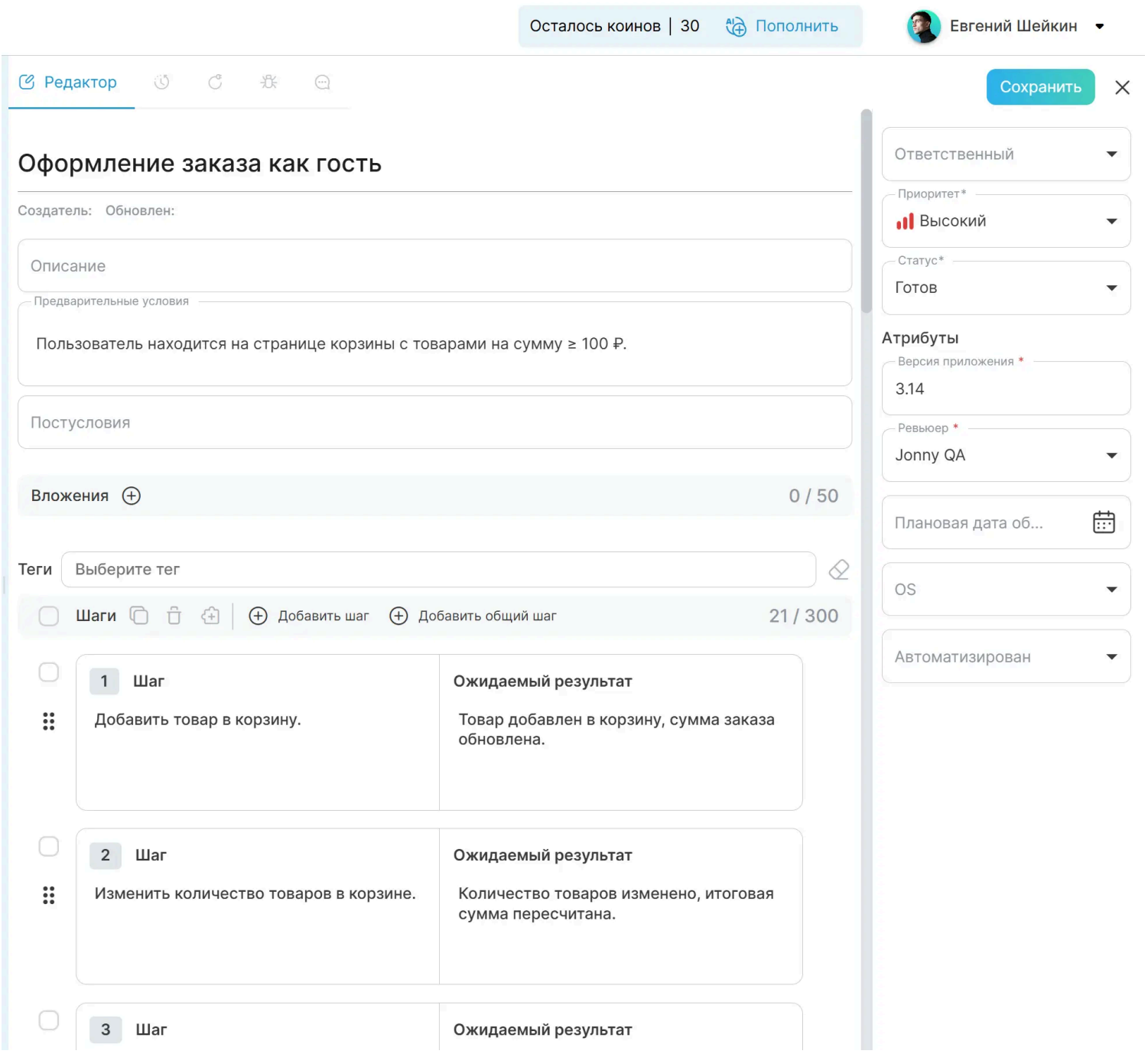
Чем полнее и точнее описание, тем ближе результат к тому, что вам нужно.



Редактор нового тест-кейса откроется сразу, поля будут заполняться по мере генерации. Дождитесь, пока заполнение закончится, проверьте результат и сохраните кейс.

Внимание

Сгенерированный тест-кейс не сохраняется автоматически. Если закрыть редактор, не нажав «**Сохранить**», результат генерации будет потерян и восстановить его нельзя.



Генерация набора тест-кейсов

Второй режим того же диалога создаёт не один кейс, а сразу набор. Переключатель «Один тест-кейс» / «Набор тест-кейсов» показывается только в списке тест-кейсов и только тем, кто может их редактировать; для чек-листов такого режима нет.

Выберите «Набор тест-кейсов», заполните поле контекста, при необходимости приложите файл и нажмите «**Сформировать AI тест-кейс**».

Массовая генерация идёт в фоне: ждать её не нужно, можно уйти на другую страницу. Кейсы создаются в той папке, из которой вы запустили генерацию.

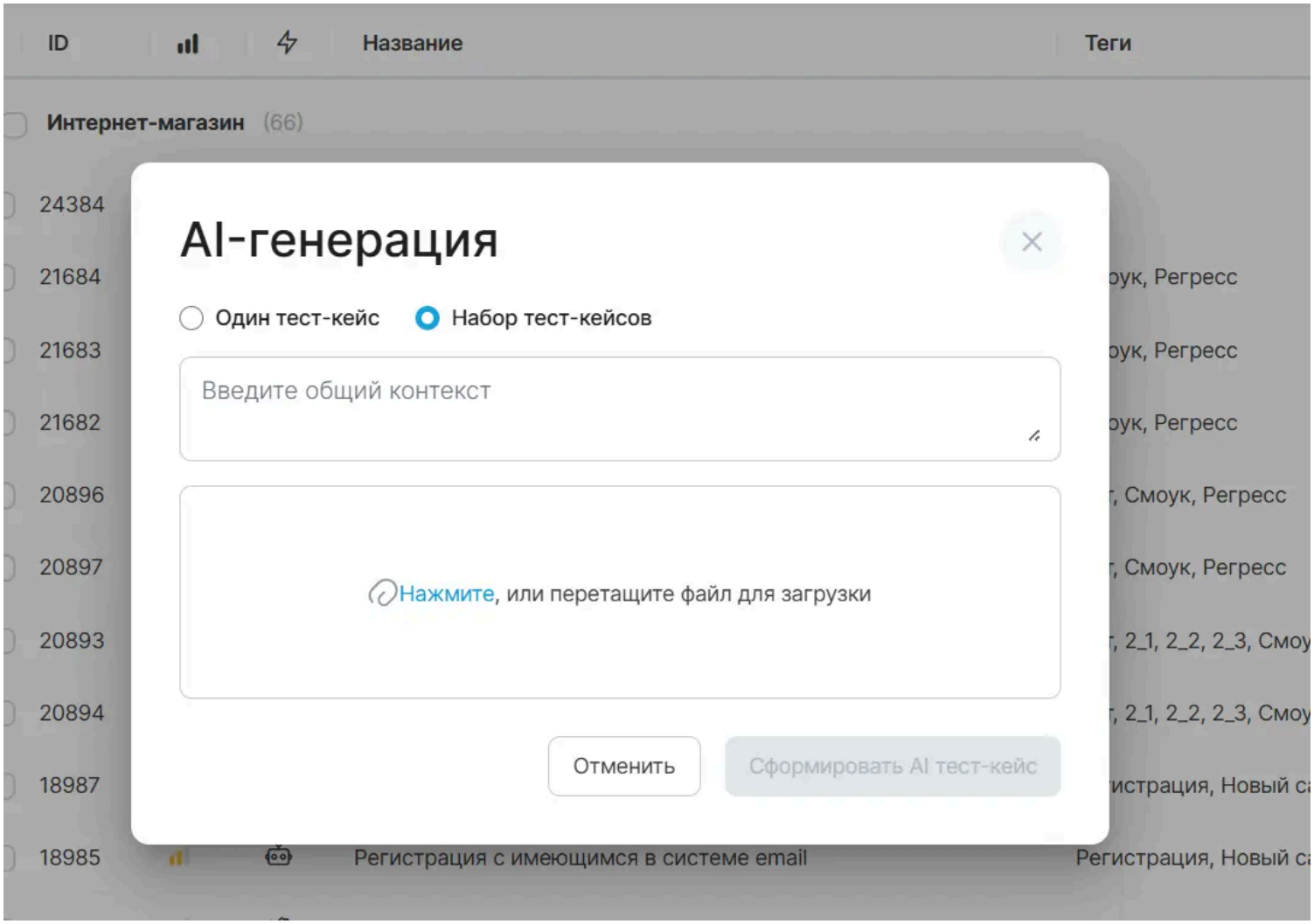
Прогресс показывает виджет «AI-генерация» в шапке списка тест-кейсов: статус («Идет AI-генерация», «AI-генерация завершена», «AI-генерация не выполнена») и сколько кейсов уже создано из запланированных.

Наведите курсор на виджет или нажмите на него. Раскроется панель со всеми генерациями пространства: кто запустил каждую из них, статус, количество созданных кейсов и время постановки в очередь («В очереди») или старта («Старт»). Кнопка «**Результаты**» открывает папку с созданными кейсами.

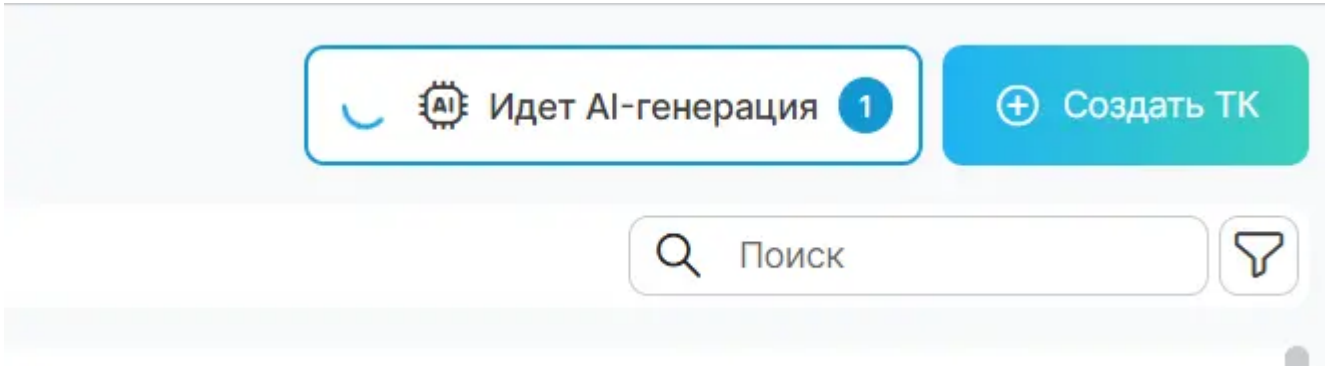
Ограничения на пространство: одновременно выполняется одна массовая генерация, всего в работе и в очереди — не больше пяти.

Генерация набора тест-кейсов

DoQA помогает подготовить набор тест-кейсов по требованиям или User Story.



После запуска на вкладке отобразится уведомление со статусом генерации. Когда генерация завершится, информация в уведомлении обновится. Наведите курсор на уведомление, чтобы посмотреть подробности завершенной генерации и количество созданных тест-кейсов.

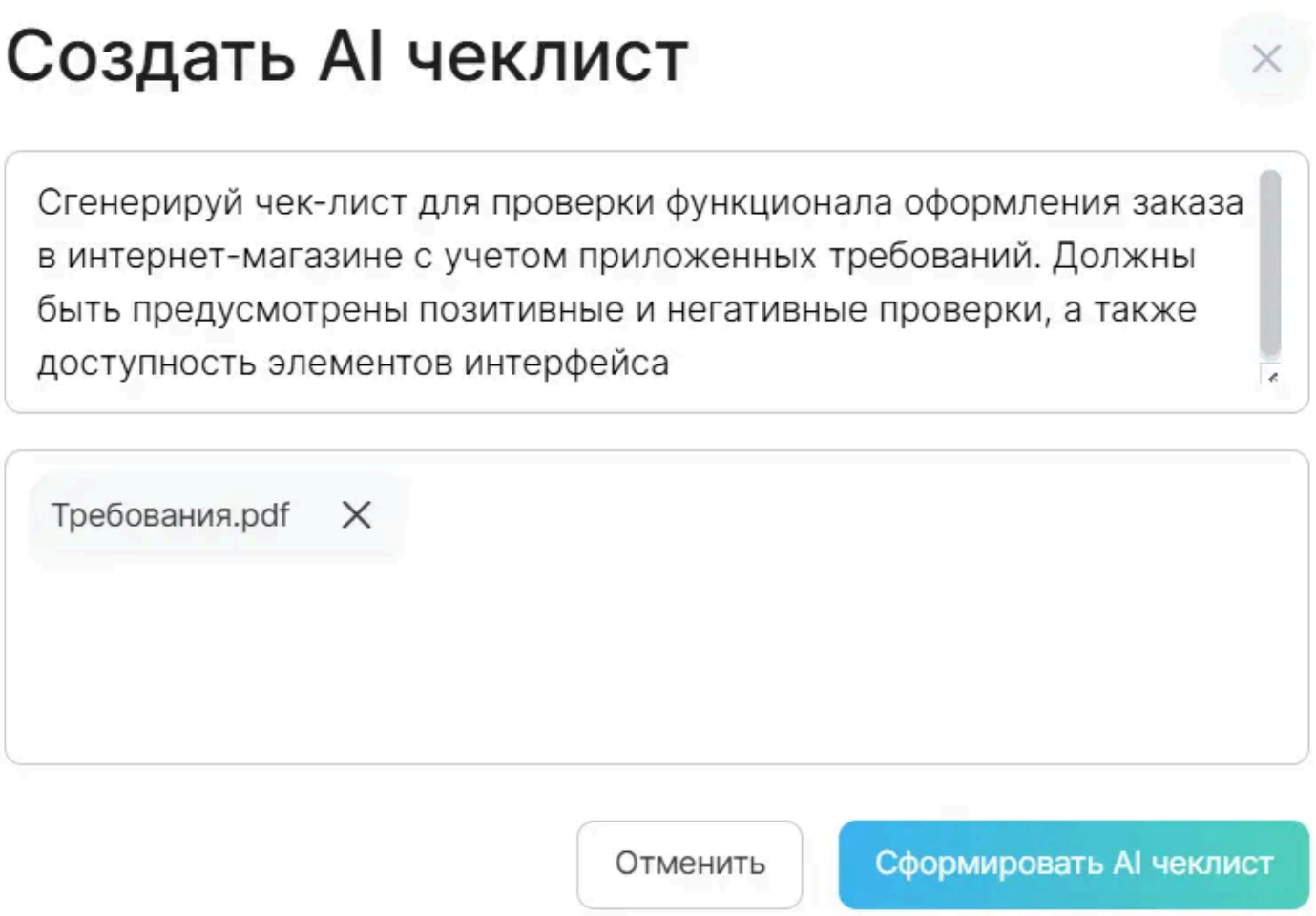


Генерация чек-листа

DoQA создаёт **чек-лист** по текстовому описанию — набор проверок в одном документе.

Порядок тот же, что и для тест-кейса: в списке чек-листов нажмите «**Создать ЧЛ**» и выберите «С помощью AI» либо нажмите иконку AI в строке папки. В диалоге «AI-генерация» заполните поле контекста, при необходимости приложите один файл PDF или DOCX до 10 МБ и нажмите «**Сформировать AI чеклист**».

Редактор чек-листа откроется сразу и будет заполняться по мере генерации. Результат так же нужно сохранить вручную.

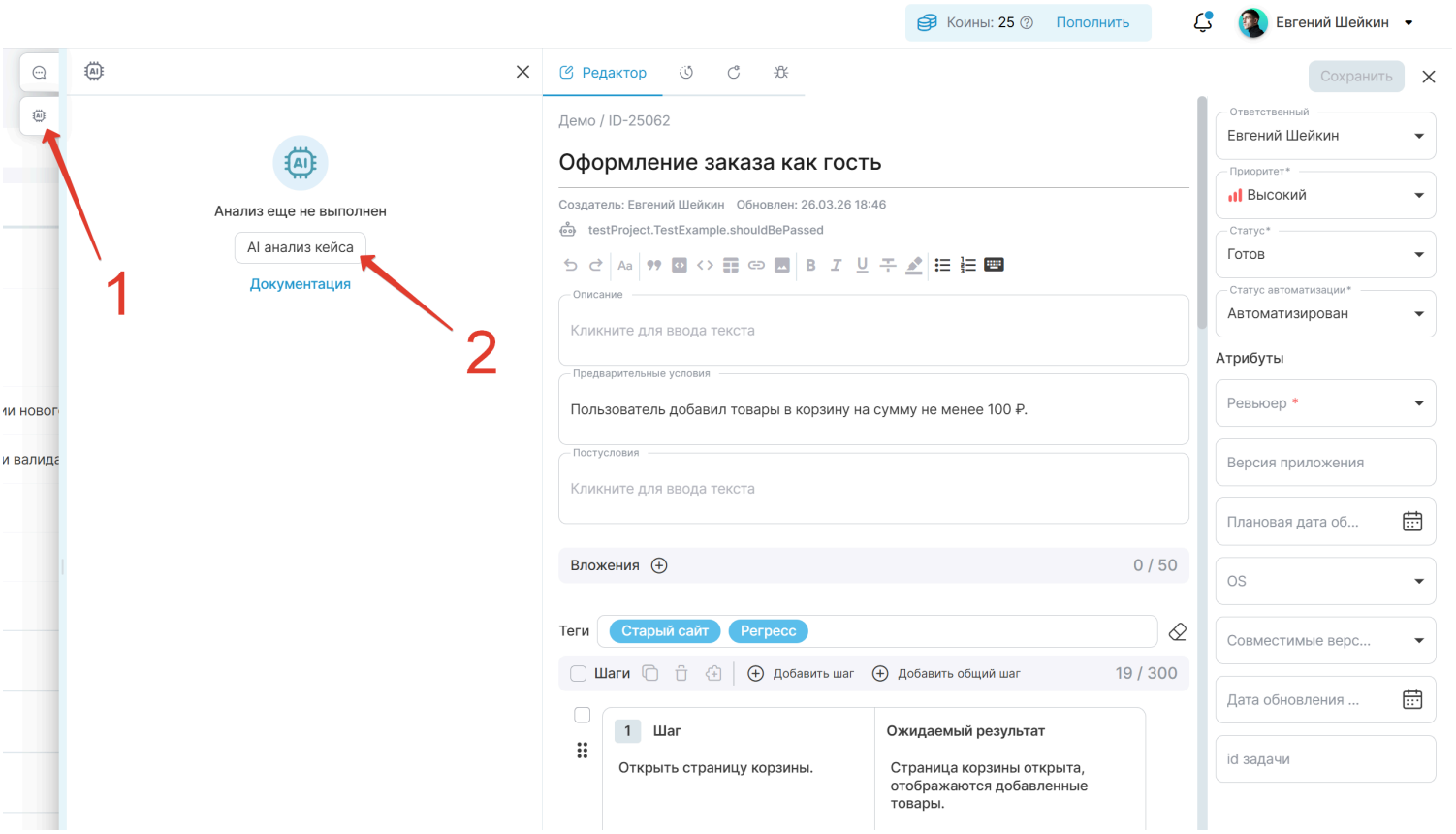


AI анализ тест-кейса

AI-анализ проверяет уже сохранённый тест-кейс и предлагает исправленные формулировки, которые можно подставить в поля кейса.

Анализ доступен только для сохранённого тест-кейса с заполненными шагами и только тем, кто может редактировать тест-кейсы. Для чек-листов AI-анализ не поддерживается.

Откройте тест-кейс и нажмите кнопку с иконкой AI на левой кромке редактора (цифра 1). Откроется панель AI-анализа. Нажмите «**AI анализ кейса**» (цифра 2).



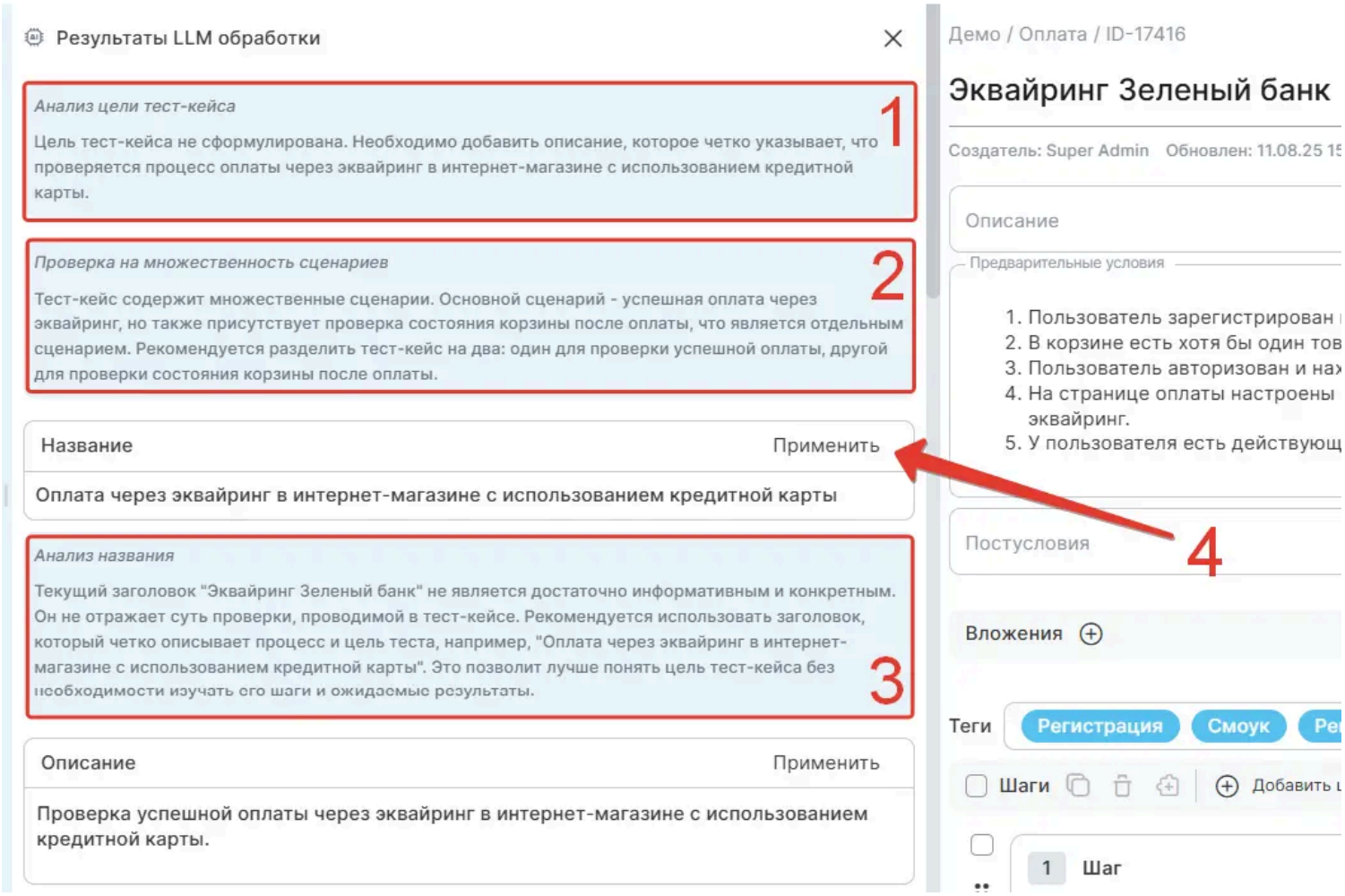
Если кейс не сохранён или в нём есть несохранённые правки, вместо анализа появится предупреждение «Сохраните тест-кейс, чтобы провести ревью». Если шаги не заполнены — «Заполните шаги, чтобы провести ревью».

Результат появляется в той же панели под заголовком «Результаты LLM обработки» и содержит блоки:

- «Анализ цели тест-кейса» — насколько понятно, что именно проверяет кейс (цифра 1).
- «Проверка на множественность сценариев» — есть ли смысл разделить кейс на несколько (цифра 2).
- «Анализ названия» (цифра 3), «Анализ описания», «Анализ предусловия», «Анализ постусловия» — комментарий по каждому полю и предложенный вариант текста.
- «Анализ шагов» — комментарий к каждому шагу с предложенным вариантом шага, ожидаемого результата и тестовых данных. Рядом с заголовком «Шаги» указано количество разобранных шагов из максимальных 300.

Кнопка «**Применить**» (цифра 4) есть у названия, описания, предварительных условий и постусловий: она подставляет предложенный вариант в соответствующее поле тест-кейса.

Шаги применить одной кнопкой нельзя. Их переносят вручную. После применения правок сохраните тест-кейс.



Генерация названия баг-репорта

При создании **баг-репорта** в плеере **прогона** DoQA может сформировать название по описанию проблемы.

1. Заполните поля «Фактический результат» и «Ожидаемый результат». Если хотя бы одно из них пустое, генерация не запустится: DoQA подсветит незаполненное поле и ничего не сгенерирует.
2. Нажмите кнопку с иконкой AI справа от поля «Имя дефекта» (подсказка «Сгенерировать название баг-репорта»).

Сгенерированное название подставляется в поле «Имя дефекта». Его можно отредактировать вручную перед сохранением баг-репорта.

Баг-репорт

Имя дефекта*

Поле 'Email' неактивно при попытке регистрации нового пользователя

Приоритет

Средний

Статус

Открыт

Фактический результат*

Невозможно ввести email, поле не активно

Ожидаемый результат*

Email введен успешно

Содержание

Предусловие:

В системе уже существует зарегистрированный пользователь с email "existinguser@example.com".

Шаги:

Шаг 1: Открыть страницу регистрации.

Шаг 2: Ввести имя пользователя "newuser" в поле "Имя пользователя"

Шаг 3: Ввести email "existinguser@example.com" в поле "Email"

Отменить Сохранить

Генерация тестов по требованию

Если в пространстве настроен **коннектор требований**, тестовую документацию можно сгенерировать прямо из **требования**: DoQA возьмёт его текст из трекера, и вводить контекст вручную не нужно.

Функция доступна, только пока к требованию не привязана ни одна проверка. Как только к требованию привязан хотя бы один тест-кейс или чек-лист, кнопка генерации пропадает.

1. В списке требований раскройте строку требования. Откроется панель связанных проверок.
2. Нажмите «**Сгенерировать тесты с помощью ИИ**».
3. В диалоге выберите вид тестовой документации: «Тест-кейс», «Чек-лист» или «Набор тест-кейсов».
4. Выберите папку для сохранения или создайте новую кнопкой «**Создать папку**».
5. Нажмите «**Сгенерировать**».

Генерация идёт в фоне и занимает несколько минут. Созданные тест-кейсы или чек-листы попадают в выбранную папку и автоматически привязываются к требованию. Стоимость такая же, как у обычной генерации: тест-кейс и чек-лист — 6 коинов, набор тест-кейсов — 30 коинов.

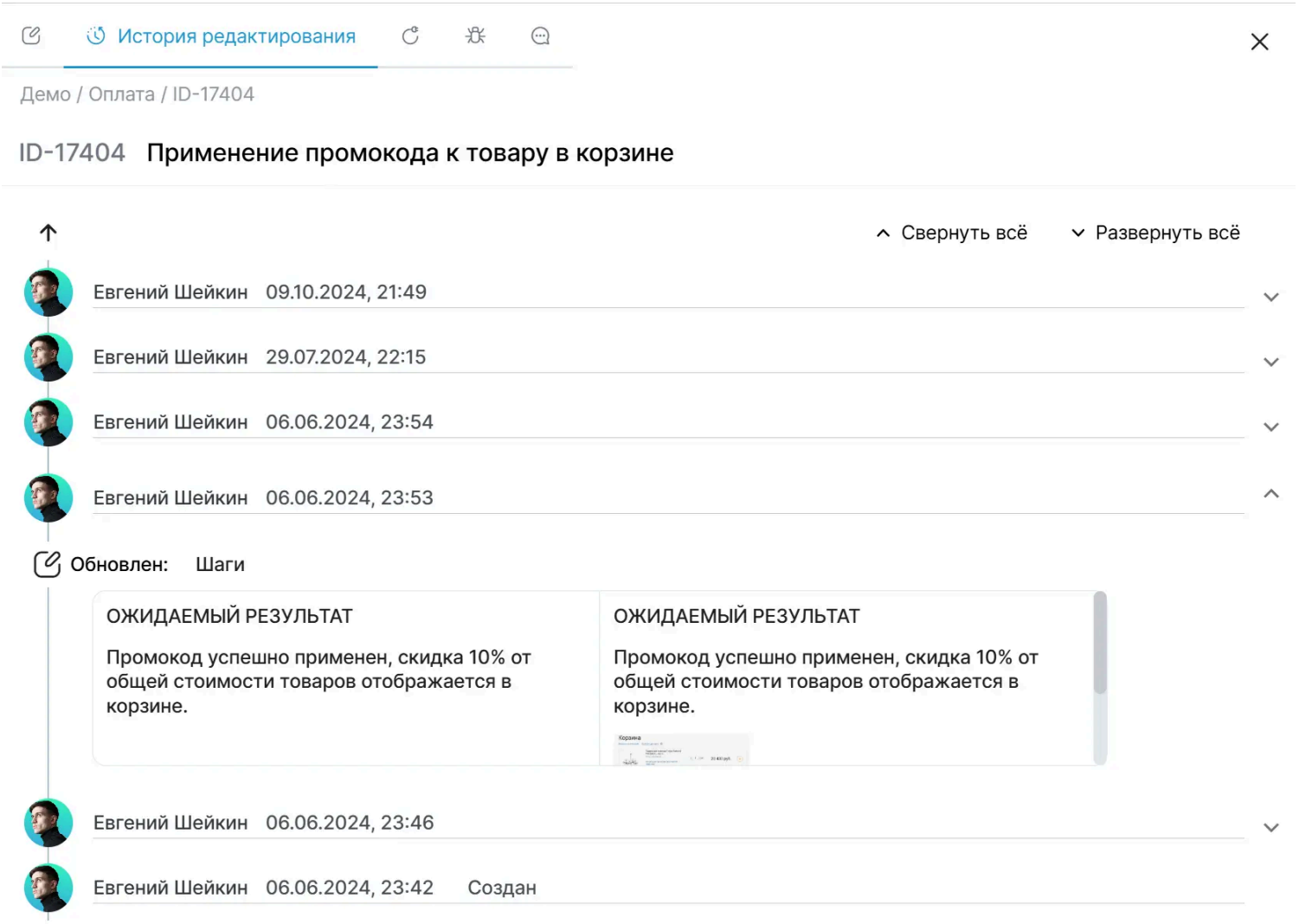
История редактирования

DoQA сохраняет историю редактирования тест-кейсов, чек-листов и прогонов: кто, когда и что изменил. По истории можно найти автора правки и откатить тест-кейс или чек-лист к предыдущей версии.

Доступ к истории редактирования

Для тест-кейсов и чек-листов откройте редактор нужного элемента и нажмите на вкладку «**История редактирования**».

Для **прогонов** откройте плеер прогона и перейдите на вкладку «**История**».



Информация об изменениях

Записи сгруппированы по авторам и времени. В заголовке записи отображается:

- **Автор изменений**.
- **Дата и время изменений**.
- **Событие**: «Создан», «Удалён», «Восстановлен» (возвращен из **Корзины**) или «Скопирован с ID ...» — ID исходного элемента, по которому можно перейти к нему.
- **Бейдж версии** (для тест-кейсов и чек-листов). По нажатию открывается окно просмотра версии.

Запись раскрывается по нажатию: внутри — описание изменений.

Историей можно управлять:

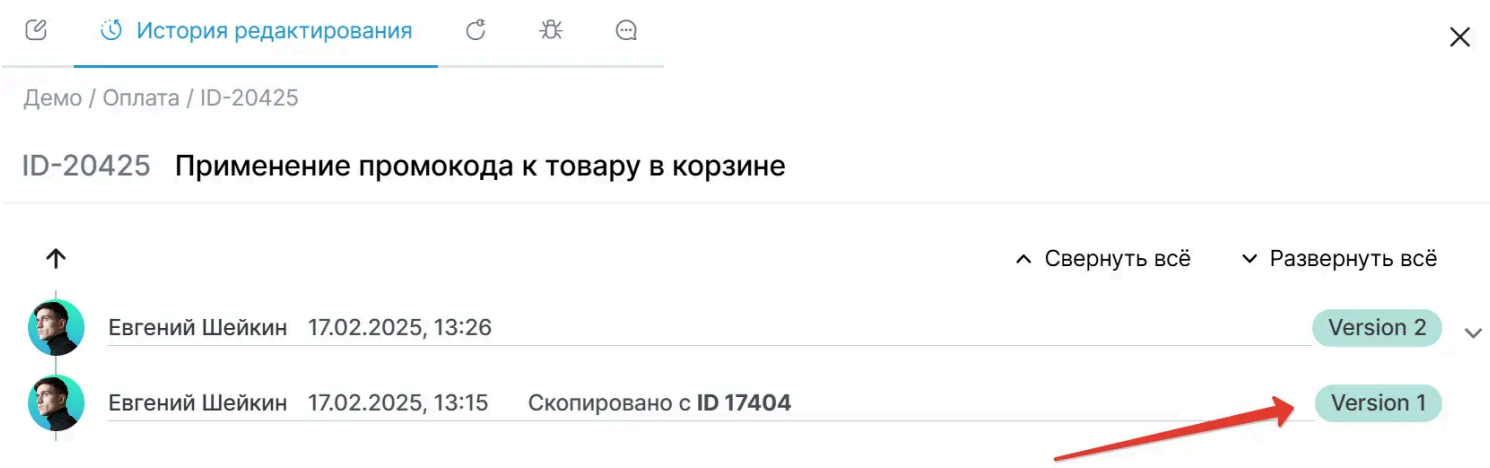
- иконка-стрелка над списком переключает порядок сортировки по дате: по возрастанию или по убыванию;
- кнопки «**Свернуть всё**» и «**Развернуть всё**» сворачивают и раскрывают все записи сразу.

Что можно отслеживать

- Добавление или удаление шагов или чеков в редакторах кейсов и чек-листов.
- Добавление или удаление тестов из прогона.
- Любые изменения в существующих тест-кейсах, чек-листах или прогонах, включая изменения в названиях, описаниях, статусах, **тегах**.
- В тест-кейсах и чек-листах также отслеживается изменение приоритетов, версий, ответственных, **атрибутов** и вложений.

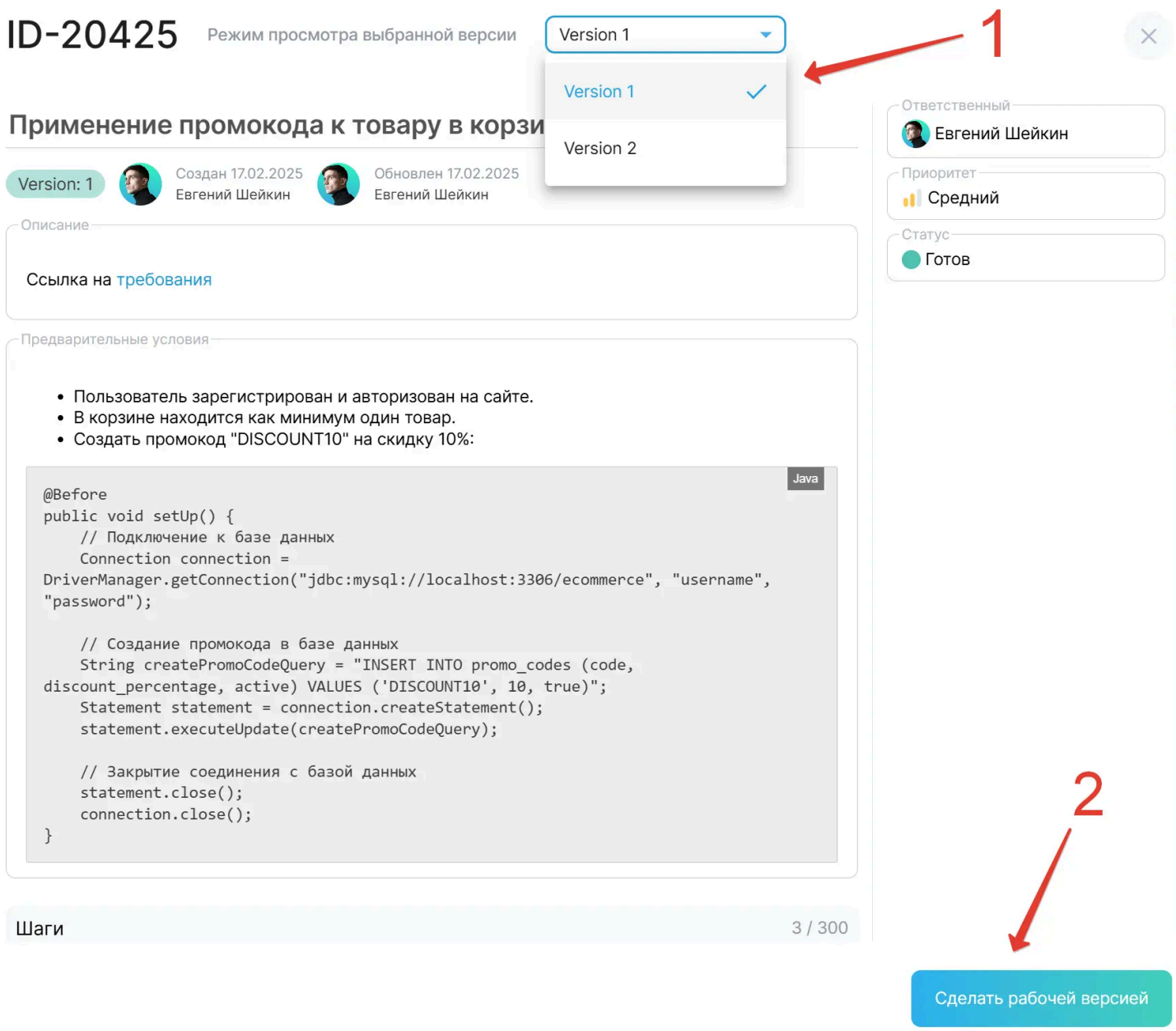
Версии тест-кейсов и чек-листов

Начиная с DoQA 3.0 при создании или изменении тест-кейсов и чек-листов создаются версии — слепки тестовой документации по состоянию на определенную дату. Для просмотра определенной версии нажмите на нее:



Откроется окно в режиме просмотра выбранной версии: в нем видно состояние тест-кейса или чек-листа после определенных событий (создание, копирование, изменение). Версию можно переключить выпадающим списком в шапке окна (цифра 1).

Чтобы вернуться к выбранной версии, нажмите «**Сделать рабочей версией**» (цифра 2). Кнопка недоступна для текущей рабочей версии и для пользователей без прав на редактирование.


ID-20425
Режим просмотра выбранной версии
Version 1
Version 1
Version 2
1
Применение промокода к товару в корзи
Version: 1
Создан 17.02.2025
Евгений Шейкин
Обновлен 17.02.2025
Евгений Шейкин
Описание
Ссылка на требования
Предварительные условия
Пользователь зарегистрирован и авторизован на сайте.
В корзине находится как минимум один товар.
Создать промокод "DISCOUNT10" на скидку 10%:
Java
@Before
public void setUp() {
// Подключение к базе данных
Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/ecommerce", "username",
"password");
// Создание промокода в базе данных
String createPromoCodeQuery = "INSERT INTO promo_codes (code,
discount_percentage, active) VALUES ('DISCOUNT10', 10, true)";
Statement statement = connection.createStatement();
statement.executeUpdate(createPromoCodeQuery);
// Закрытие соединения с базой данных
statement.close();
connection.close();
}
Шаги
3 / 300
Ответственный
Евгений Шейкин
Приоритет
Средний
Статус
Готов
2
Сделать рабочей версией

Требования

Требования не создаются в DoQA. Они загружаются из внешнего трекера и хранятся в пространстве как локальные копии: ключ, заголовок, статус в трекере и ссылка на исходный тикет. Раздел отвечает на два вопроса: какие требования покрыты тестами, а какие нет, и где тесты устарели после того, как требование изменили.

Источником требований может быть Jira Server или YouTrack. Пока администратор не настроил **коннектор требований**, раздел пуст.

> **Проверка = тест-кейс или чек-лист**
>
> В этом разделе «проверка» — это **тест-кейс** или **чек-лист**. Они равноправны: оба можно привязать к требованию, оба считаются в покрытии и оба попадают в **матрицу покрытия**.

Как открыть

Откройте пространство и выберите «Требования» в боковом меню. Кнопка «**Матрица покрытия**» в шапке ведёт на отдельную страницу — **матрицу покрытия**.

Что показывает список

Колонка	Что в ней
«Ключ»	Ключ тикета в трекере. Рядом — значок трекера; клик по ключу открывает тикет в трекере.
«Заголовок»	Заголовок требования, как он записан в трекере.
«Статус в трекере»	Статус тикета. DoQA его не меняет, только показывает.
«Покрытие»	Одно из пяти состояний, см. **ниже**.
«Проверки»	Число связанных проверок и полоса с разбивкой по последним результатам. Наведите курсор — увидите, сколько из них кейсов и сколько чек-листов.

Колонка	Что в ней
«Актуализация»	Сколько связей уже актуализировано из скольких. Колонка появляется, только когда в списке есть требования с пометкой «Требуется актуализация».

Внизу списка — счётчик «Показано N из M». Следующая страница подгружается по мере прокрутки.

Раскрытие строки

Клик по строке раскрывает панель связанных проверок. Для каждой проверки видно её ID, тип (кейс или чек-лист), дату привязки, последний результат прогона и кнопку отвязки. Клик по проверке открывает её в DoQA, клик средней кнопкой мыши — в новой вкладке браузера.

Здесь же кнопки «**Добавить кейс**» и «**Добавить чек-лист**». У архивных требований этих кнопок нет.

Если связей ещё нет, панель показывает «Нет связанных проверок. Привяжите кейс или чек-лист, чтобы появилось покрытие.»

Состояния покрытия

Состояние	Когда возникает
«Нет тестов»	С требованием не связана ни одна проверка.
«Не прогонялись»	Связи есть, но ни у одной проверки нет засчитанного результата прогона.
«Все прошли»	У всех связанных проверок последний результат — «Пройден».
«Не все прошли»	Есть проверки с результатом «Провален», «Сломан», «Заблокирован» или «Пропущен», либо часть проверок вообще не прогонялась.
«Требуется актуализация»	Хотя бы одна связь помечена как требующая актуализации.

Три правила, которые объясняют неожиданные значения в колонке:

1. **«Требуется актуализация» перебивает всё остальное.** Пока пометка висит хотя бы на одной связи, требование показывает именно это состояние, даже если все тесты пройдены.
2. **Последним результатом считается последний прогон, в котором статус проверки был реально выставлен.** Если проверку добавили в новый прогон, но ещё не прошли (статус не проставлен), DoQA берёт результат из предыдущего прогона, а не считает, что прогона не было.
3. **Прогоны, сделанные до подтверждения актуализации, не засчитываются.** После того как вы нажали «**Актуализировано**», старый результат перестаёт учитываться, и требование возвращается в «Не прогонялись», пока проверку не прогонят заново.

Метрики: Покрытие и Успешность

В шапке списка — два показателя.

- «Покрытие» — доля требований, у которых есть хотя бы одна связанная проверка: (все требования − требования в состоянии «Нет тестов») ÷ все требования. Подпись в интерфейсе: «Процент требований, к которым привязаны тест-кейсы или чек-листы. Показывает полноту тестового покрытия.»
- «Успешность» — доля **покрытых** требований в состоянии «Все прошли», от числа покрытых. Знаменатель здесь — покрытые требования, а не все.

Из-за разных знаменателей цифры расходятся: если из 100 требований покрыто 20 и все 20 в состоянии «Все прошли», «Покрытие» покажет 20%, а «Успешность» — 100%. Это не ошибка.

Обе метрики и счётчики на чипах состояний считаются по **всем активным требованиям пространства**. Поиск, фильтр «Связь» и выбранный чип на них не влияют, архивные записи в них не входят. Если требований нет (или нет покрытых), вместо процента стоит прочерк.

Срезы и фильтры

Список делится на три вкладки.

- «Все требования» — активные требования, загруженные из трекера.
- «Архивные записи» — требования, которых больше нет в выгрузке трекера: удалённые или ушедшие из-под фильтра коннектора. DoQA их не удаляет, а архивирует, чтобы не потерять

связи и историю; в списке они отображаются зачёркнутым заголовком. Фильтров у этой вкладки нет. Пустое состояние — «Архивных записей нет».

- «Несвязанные проверки» — обратный срез: не требования, а проверки без единой связи. Подпись в интерфейсе: «Ручные проверки (кейсы и чек-листы) без единого связанного требования — кандидаты на привязку». Для каждой строки доступны «**Открыть**» (в новой вкладке) и «**Привязать требование**». Есть свой поиск (по названию или ID проверки) и фильтр «Тип» — «Все», «Кейсы», «Чек-листы».

Фильтры вкладки «Все требования»:

- строка поиска — ищет по ключу и заголовку требования;
- фильтр «Связь» — «Все», «Связанные», «Несвязанные»;
- чипы состояний покрытия со счётчиками. Клик по чипу оставляет в списке только требования в этом состоянии, повторный клик снимает фильтр.

Когда фильтры активны, рядом появляется «**Сбросить фильтры**». Выбранный срез и фильтры остаются в адресе страницы. Ссылку можно отправить коллеге.

Связывание требований с проверками

Связь двусторонняя. При её создании DoQA дописывает ID проверки в поле тикета в трекере. Какое именно поле, задаёт администратор в настройке «Поле для ID кейсов» **коннектора**. Связь M:N: одну проверку можно привязать к нескольким требованиям и наоборот.

Привязать можно из четырёх мест.

1. **Из списка требований.** Раскройте строку и нажмите «**Добавить кейс**» или «**Добавить чек-лист**». Откроется тот же диалог выбора, что и при добавлении в прогон. Можно выбрать сразу набор. Результат: «Привязано проверок: N». Если всё выбранное уже было привязано, появится сообщение «Новых привязок нет — выбранные проверки уже были привязаны».
2. **Из карточки тест-кейса или чек-листа.** Секция «Требования» → «**Привязать требование**» → поиск по ключу или заголовку, можно отметить несколько требований сразу.
3. **Из среза «Несвязанные проверки».** Кнопка «**Привязать требование**» в строке проверки.
4. **Отвязка.** Кнопка с разорванной связью в панели связанных проверок или в секции «Требования» карточки. DoQA спросит подтверждение и предупредит, что проверка перестанет учитываться в покрытии требования.

Удаление проверки и связь

Помещение тест-кейса или чек-листа в **корзину** связь **не** разрывает. Она переживает восстановление. Связь рвётся только при окончательном удалении; тогда же DoQA переписывает список ID в поле тикета.

Актуализация: что делать, когда требование изменилось

Это основной сценарий раздела. Он работает так.

1. Аналитик в трекере переводит требование в статус, сопоставленный состоянию «Требуется актуализация». Это единственное состояние, которое DoQA читает **из** трекера, а не пишет в него.
2. DoQA получает вебхук (или узнаёт об изменении при обновлении из трекера), помечает все связанные проверки, и требование переходит в «Требуется актуализация». Ответственные за связанные тест-кейсы и чек-листы получают **уведомление**.
3. Пока пометка висит, DoQA не пишет вычисленный статус покрытия в трекер. В «Ленте операций» такие записи помечены как «Заморожено до актуализации».
4. Тестировщик правит проверку и подтверждает актуализацию — кнопкой «**Актуализировано**» в карточке кейса или чек-листа, либо «**Актуализировать**» / «**Актуализировать все**» в панели связанных проверок.
5. После подтверждения проставляется дата актуализации. Прогоны, сделанные до неё, перестают учитываться: требование покажет «Не прогонялись», пока проверку не прогонят заново.

Пометку снимает только человек. Правка тест-кейса сама по себе её не снимает.

Прогресс виден в колонке «Актуализация» списка (сколько актуализировано из скольких) и в шапке раскрытой панели: «N из M тестов требуют актуализации», а после подтверждения всех — «Все тесты актуализированы».

Генерация тестов с помощью ИИ

DoQA может забрать текст требования из трекера и создать по нему тестовую документацию.

Кнопка «**Сгенерировать тесты с помощью ИИ**» находится в панели связанных проверок и показывается, только пока у требования нет ни одной связи.

В диалоге «Генерация тестов с помощью ИИ» выберите:

1. «Вид тестовой документации» — «Тест-кейс», «Чек-лист» или «Набор тест-кейсов».
2. «Папка для сохранения» — существующая **папка** или новая, её можно создать прямо в диалоге кнопкой «**Создать папку**».

Нажмите «**Сгенерировать**». Генерация асинхронная: в строке требования крутится индикатор, панель показывает «ИИ создаёт тестовую документацию...» и предупреждает, что это может занять несколько минут. Со страницы можно уйти. По завершении появится «Тестовая документация сгенерирована», а созданные проверки будут автоматически привязаны к требованию.

Кнопки не будет, если:

- требование архивное;
- коннектор выключен или требование загружено не той интеграцией, которая указана в коннекторе сейчас;
- у вас нет доступа к **AI-функциям**.

Генерация расходует AI-коины, как и остальные AI-функции; при ошибке списание возвращается. Подробнее — в **настройках ИИ**.

Создать прогон из требования

Нажмите «**...**» в строке требования и выберите «**Создать прогон**». В прогон попадут все связанные с требованием проверки — и тест-кейсы, и чек-листы одним смешанным набором. Откроется обычная форма создания **прогона**; после создания DoQA сразу перейдёт на страницу нового прогона.

Пункт неактивен, если с требованием не связана ни одна проверка.

Обновление из трекера

Кнопка «**Обновить из трекера**» в шапке списка запускает синхронизацию: DoQA перечитывает требования по фильтру коннектора, обновляет заголовки и статусы, добавляет новые и архивирует те, которых в выгрузке больше нет. По завершении — «Требования обновлены из трекера».

Пока коннектор не настроен, кнопка заблокирована.

Вебхуки доставляют изменения сами. Ручное обновление нужно, когда вебхук не настроен или события были пропущены.

Если требований нет

- «Требований пока нет» и «Настройте коннектор трекера, чтобы загрузить требования» — коннектор ещё не настроен. Кнопка «**Настроить коннектор**» ведёт в настройки пространства; она видна тем, у кого есть доступ к настройкам.
- «Требований пока нет» и подсказка про проверку фильтра (JQL или проект) в настройках коннектора — коннектор настроен, но выгрузка пуста. Скорее всего, фильтр коннектора не выбирает ни одного тикета.
- «Не удалось загрузить требования» и «Трекер недоступен или вернул ошибку» — проблема на стороне трекера или подключения. Состояние подключения видно на вкладке «Коннектор требований», подробности — в «Ленте операций».

Коннектор требований

Коннектор связывает пространство с проектом во внешнем трекере: загружает оттуда **требования**, пишет обратно в тикет ID связанных проверок и вычисленный статус покрытия, принимает события об изменении требований. Один коннектор на пространство.

Какие трекеры подходят

Модуль требований умеет работать с двумя трекерами: **Jira Server** и **YouTrack**. Остальные трекеры, которые DoQA подключает для баг-репортов — Jira Cloud, Yandex Tracker, Bitrix 24, Redmine, GitLab, Kaiten, — источником требований быть не могут и в списке «Трекер» не появятся. Если такой трекер всё же попадёт в коннектор, DoQA откажется его сохранить: «Этот трекер не поддерживается модулем требований».

Коннектор не хранит логин и токен: он ссылается на уже существующую интеграцию-трекер. Сначала администратор подключает **Jira Server** или **YouTrack** в разделе интеграций. Только после этого трекер появится в списке коннектора. Подсказка в интерфейсе так и говорит: «Из уже подключённых интеграций-трекеров DoQA».

В списке предлагается по одной интеграции на тип трекера. Если в DoQA заведено несколько подключений к Jira Server, коннектор возьмёт первое из них.

YouTrack в списке «Трекер»

YouTrack показывается в списке «Трекер», только если коннектор этого пространства уже привязан к интеграции YouTrack. При настройке коннектора с нуля в списке будет Jira Server. Дальнейшие разделы этой страницы описывают оба трекера — они относятся к уже подключённому YouTrack.

Где настраивается

Пространство → «Настройки» в боковом меню → вкладка «Коннектор требований». Страница настроек открывается при **правах** на редактирование проекта или пространств либо при правах на редактирование тест-кейсов, чек-листов или прогонов в этом пространстве. Из пустого списка требований туда же ведёт кнопка «**Настроить коннектор**».

Рядом с заголовком «Коннектор требований» — состояние подключения:

Значок	Что означает
«Не настроено»	Коннектор ещё не сохранён.
«Подключено»	Последний обмен с трекером прошёл без ошибок.
«Поле не найдено»	Указанного кастомного поля в трекере нет (трекер ответил HTTP 400). Исправьте поле и сохраните заново.
«Трекер недоступен»	Подключиться к трекеру не удалось; DoQA показывает данные из кэша.

Настройка

Трекер и проект-источник

1. «Трекер» — выберите подключённую интеграцию. Пока трекер не выбран, остальные настройки не показываются: набор полей зависит от интеграции.
2. «Проект-источник» — выберите проект трекера. DoQA подтягивает список проектов из трекера сама.
3. Запрос-фильтр под списком проектов — «JQL-запрос» для Jira Server или «Запрос YouTrack» для YouTrack. Работает как дополнительный фильтр **внутри** выбранного проекта: например, `status = Open`. Требования, не попавшие под фильтр, в пространство не загружаются, а уже загруженные уходят в «Архивные записи».

Если трекер не умеет отдавать схему полей, DoQA переключается в ручной режим и просит ввести ключ проекта и идентификаторы полей строками (появится сообщение «Трекер не поддерживает выбор полей — введите значения вручную»).

Поле для ID кейсов

Кастомное поле трекера, в которое DoQA списком пишет ID связанных проверок. Именно оно делает связь двусторонней: открыв тикет, аналитик видит, какие проверки его покрывают.

Для YouTrack рекомендуемый тип поля — «текст» (одиночное значение).

Статус покрытия

DoQA вычисляет **состояние покрытия** и отражает его в трекере. Настройка «Куда писать статус покрытия» определяет способ:

- «Кастомное поле» (рекомендуется) — статус пишется в отдельное select-поле трекера. Не требует настройки workflow.
- «Системный статус (workflow)» — DoQA двигает тикет по статусам/переходам проекта. Подходит, если в проекте трекера уже настроены нужные статусы и переходы.

Переключатель показывается, только если трекер поддерживает оба варианта. Для YouTrack доступно только кастомное поле; рекомендуемый тип поля — «перечисление» (одиночное значение).

При выборе «Кастомное поле» задайте «Поле статуса покрытия» — это **отдельное** поле, не то же самое, что «Поле для ID кейсов». Без него коннектор не сохранится.

Маппинг статусов покрытия

Таблица сопоставляет пять состояний покрытия DoQA со значениями в трекере. Что подставлять в правую колонку, зависит от выбранной цели. При «Кастомном поле» это опция выбранного поля, при «Системном статусе» — статус или переход workflow. Состояние без сопоставления («— не задано —») в трекер не пишется.

Четыре состояния DoQA **пишет** в трекер: «Нет тестов», «Все прошли», «Не все прошли», «Не прогонялись». У них в таблице стрелка вправо.

Состояние «Требуется актуализация» — единственное, которое DoQA **читает** из трекера. У него стрелка влево и подсказка «Этот статус выставляет аналитик в трекере — DoQA слушает его и снимает пометку». Как только значение в трекере совпадает с этим маппингом, DoQA помечает все связанные проверки требования и замораживает запись статуса в трекер до подтверждения актуализации. Подробнее — **цикл актуализации**.

Если маппинг пуст

Без маппинга «Требуется актуализация» цикл актуализации не запустится: DoQA не поймёт, какой статус в трекере означает «требование изменилось».

Вебхуки

Вебхук доставляет в DoQA события «требование изменилось» из трекера, не дожидаясь ручного обновления. Тумблер «Принимать вебхуки» включает приём.

Тумблер выключает коннектор целиком

Несмотря на название, «Принимать вебхуки» — это выключатель всего коннектора, а не только приёма событий. Пока он выключен, DoQA не загружает требования по кнопке «**Обновить из трекера**», не пишет в трекер ID проверок и статус покрытия и не даёт генерировать тесты по требованию. В «Ленте операций» пропущенные операции будут помечены причиной «Коннектор выключен».

Настраивается вебхук на стороне трекера, и у Jira Server и YouTrack это делается по-разному.

Jira Server

DoQA даёт готовый URL вебхука с секретом внутри. Секрет на экране скрыт. Кнопка «**Скопировать**» кладёт в буфер полный URL вместе с ним. Кнопка «**Перевыпустить**» выдаёт новый секрет; старый URL сразу перестаёт работать.

Подсказка со шагами открывается кнопкой «?» рядом с полем «Вебхук». В самой Jira: Администрирование → Система → Веб-перехватчики (WebHooks) → Создать.

1. Имя — любое; Статус — Включён.
2. URL — вставьте скопированный из DoQA.
3. События → Проблема: отметьте «обновлено» (и «удалено», если нужно архивировать требования).
4. Не ставьте галочку «Исключить основу»: DoQA принимает JSON-тело.
5. Сохраните.

Внимание

URL содержит секрет. Не публикуйте его. Адрес DoQA должен быть доступен с сервера Jira: для внешней Jira нужен публичный домен или туннель, `localhost` не подойдёт.

YouTrack

У YouTrack нативных вебхуков нет. Вместо них ставится workflow-правило. Кнопка «**Настройка вебхука YouTrack**» открывает окно с готовым скриптом: URL, секрет и имя поля статуса покрытия в него уже подставлены. Секрет в URL не передаётся: скрипт отправляет его заголовком.

Кнопка доступна только после сохранения коннектора: до этого DoQA ещё не выдала URL и секрет.

1. Откройте Administration → Workflows.
2. Нажмите New workflow (или выберите существующий) и привяжите его к проекту-источнику требований.
3. Создайте правило (rule) и вставьте скопированный скрипт.
4. Сохраните workflow и убедитесь, что правило включено для нужного проекта.
5. Нажмите «**Обновить из трекера**» в DoQA, чтобы загрузить текущие требования.

Если поле статуса покрытия не выбрано, скрипт не будет отслеживать изменение покрытия. Окно об этом предупредит.

Без workflow-правила модуль остаётся рабочим: данные синхронизируются по кнопке «**Обновить из трекера**», вебхук лишь ускоряет обновление.

Ручное обновление

Кнопка «**Обновить из трекера**» в блоке «Ручное обновление» перечитывает требования по фильтру коннектора. Во время работы виден прогресс («Прочитано N из M»), по завершении — сколько требований обновлено. Рядом показывается время последнего обновления или число загруженных требований.

Та же кнопка есть в шапке списка требований.

Лента операций

Под формой коннектора — «Лента операций»: что DoQA отправляла в трекер и что получала оттуда. Индикатор справа показывает «синк в норме» либо число ошибок.

Записи фильтруются по «Статусу» («Все», «OK», «Ошибки», «Пропущено») и «Операции»:

- «Получение» — «Получение требования» (загрузка требований из трекера);
- «Отправка» — «Отправка ID проверок в тикет» и «Отправка статуса покрытия»;
- «Вебхук» — «Принят вебхук об изменении».

Отдельным цветом выделены записи «Актуализация».

Строки со статусом «Пропущено» объясняют, почему DoQA ничего не сделала:

Причина	Что это значит
«Коннектор не настроен»	В пространстве нет коннектора.
«Коннектор выключен»	Приём вебхуков выключен тумблером.
«Заморожено до актуализации»	Требование ждёт подтверждения актуализации. Статус покрытия в трекер не пишется.
«Статус не сопоставлен или переход недоступен»	В маппинге нет значения для этого состояния либо трекер не даёт нужный переход.
«Повторное событие»	Вебхук с тем же событием уже обработан.

Очистка данных требований

Внизу вкладки — «Очистка данных требований». Кнопка «**Очистить данные**» безвозвратно удаляет из базы **все** загруженные в это пространство требования и **все** связи тест-кейсов и чек-листов с ними.

Действие необратимо

Восстановить связи после очистки нельзя. Их придётся расставлять заново вручную. В трекер DoQA при этом ничего не пишет: очистка убирает локальные данные, а не отвязывает проверки в тикетах.

Не удаляются: настройки коннектора, сама интеграция и сами тест-кейсы и чек-листы. После очистки можно синхронизироваться заново.

Кнопка доступна только тем, у кого есть право `requirement.purge` (роль «Администратор проектов»), остальным она заблокирована. В диалоге подтверждения кнопка «**Удалить**» разблокируется через несколько секунд. По завершении DoQA сообщит, сколько требований и связей удалено.

Очистка нужна, если синхронизировались не с тем проектом или делали синхронизацию для теста. Чтобы просто перестать загружать часть требований, поменяйте фильтр коннектора: лишние записи уйдут в «Архивные записи», а связи и история сохранятся.

Матрица покрытия

Матрица показывает пересечение требований и проверок: строки — **требования**; столбцы — тест-кейсы и чек-листы; ячейки — последний результат прогона. Так видно, какие требования покрыты единственной проверкой, какие проверки покрывают сразу несколько требований и где в покрытии пробелы.

Откройте раздел «Требования» в пространстве и нажмите «**Матрица покрытия**» в шапке. Кнопка «**Дерево требований**» возвращает к списку.

В матрицу попадают только активные требования. Архивные записи в ней не показываются. Их видно на одноимённой вкладке списка.

Как читать таблицу

Слева три закреплённых колонки: «Ключ», «Название» и «Покрытие» — **состояние покрытия** требования. Дальше идёт по колонке на каждую проверку среза; в шапке колонки — значок типа (кейс или чек-лист) и ID. Клик по ID открывает проверку в новой вкладке.

Границу между «Название» и остальной таблицей можно потянуть мышью, двойной клик по ней сбрасывает ширину. Ширина запоминается в браузере.

Наведение подсвечивает строку и колонку целиком. Клик по ячейке открывает полосу под шапкой: результат, тип и ID проверки, её название, ключ требования и кнопка «**Открыть кейс**» либо «**Открыть чек-лист**».

Строки подгружаются по мере прокрутки. Внизу — счётчик размера среза вида «N × M».

Легенда

Заливка ячейки — последний результат прогона проверки по этому требованию.

Значок	Результат
«Пройден»	Последний засчитанный прогон пройден.

Значок	Результат
«Сломан»	Проверка сломана.
«Пропущен»	Проверка пропущена.
«Заблокирован»	Проверка заблокирована.
«Провален»	Проверка провалена.
«Нет прогона»	Связь есть, но засчитанного результата нет.

Пустая ячейка — проверка с этим требованием не связана.

Про то, какой прогон считается последним и почему после актуализации результат обнуляется, — в разделе **Состояния покрытия**.

Фильтры

- **Чипы состояний покрытия** со счётчиками. Клик оставляет в матрице только требования в этом состоянии, повторный клик снимает фильтр. Счётчики на чипах от выбранного чипа не зависят.
- **«Только связанные требования»** — тумблер. Включён по умолчанию: в матрице только требования, у которых есть хотя бы одна связь. Выключите его, чтобы увидеть непокрытые требования: они добавятся пустыми строками.

Когда фильтр активен, появляется «**Сбросить фильтры**» — над таблицей и в её подвале.

Метрики

В шапке — те же «Покрытие» и «Успешность», что и в списке требований, плюс полоса с разбивкой по пяти состояниям. Формулы описаны в разделе **Метрики: Покрытие и Успешность**.

«Покрытие» здесь считается по срезу

Метрики матрицы учитывают тумблер «Только связанные требования». Пока он включён (а это состояние по умолчанию), в срез попадают только требования со связями, поэтому «Покрытие»

показывает 100%. Чтобы увидеть реальное покрытие пространства, выключите тумблер или смотрите метрику в списке требований.

Выбранный чип состояния на метрики, наоборот, не влияет.

Экспорт

Кнопка «**Экспорт**» предлагает два формата: «**Экспорт в CSV**» и «**Экспорт в Excel**».

Ведут они себя по-разному:

- **CSV и не больше 1000 строк** — файл скачивается сразу.
- **Excel в любом размере или CSV длиннее 1000 строк** — DoQA готовит файл в фоне. Над таблицей появляется полоса «Подготовка экспорта...», затем «Экспорт...» с процентами; по готовности файл скачается сам. Со страницы в это время лучше не уходить: прогресс отслеживает открытая вкладка.

Что попадёт в файл

Экспорт **игнорирует** тумблер «Только связанные требования»: в файл выгружаются все активные требования пространства, включая те, у которых нет ни одной проверки. Фильтр по состоянию покрытия (чип) при этом учитывается.

Поэтому файл почти всегда длиннее того, что вы видите на экране.

Структура файла: колонки `Requirement key`, `Requirement title`, `Coverage`, дальше по колонке на проверку. Заголовок колонки — `#ID Название` для тест-кейса и `CL#ID Название` для чек-листа (префикс нужен, чтобы кейс и чек-лист с одинаковым ID не слились). В ячейках — результат прогона на латинице (`passed`, `failed`, `broken`, `blocked`, `skipped`) либо пусто, если связи нет или проверка не прогонялась. Имя файла — `coverage-matrix_s<ID пространства>_<метка времени>`.

Экспорт матрицы не связан с **экспортом тестовой документации** — это другая выгрузка и другой формат.

Если матрица пустая или не строится

- «Нет данных для матрицы» и «Свяжите требования с кейсами или чек-листами, чтобы построить покрытие» — связей ещё нет. Как их создать — в разделе **Связывание требований с проверками**.
- «Ничего не найдено. Попробуйте изменить или сбросить фильтры.» — под выбранный чип не попало ни одного требования.
- «Не удалось построить матрицу» и «Превышено время ожидания прогонов» — расчёт не уложился в отведённое время. Нажмите «**Повторить**». Если ошибка повторится, сузьте срез чипом состояния.
- «Матрица отображает кешированные данные и будет пересчитана после восстановления связи с трекером» — трекер недоступен. Числа в матрице верные, но заголовки и статусы требований могли устареть; состояние подключения смотрите в **коннекторе**.

Прогоны

Прогон — набор тест-кейсов и чек-листов, которые проходят в одном цикле тестирования. Прогон может одновременно содержать тест-кейсы и чек-листы.

Создание прогона

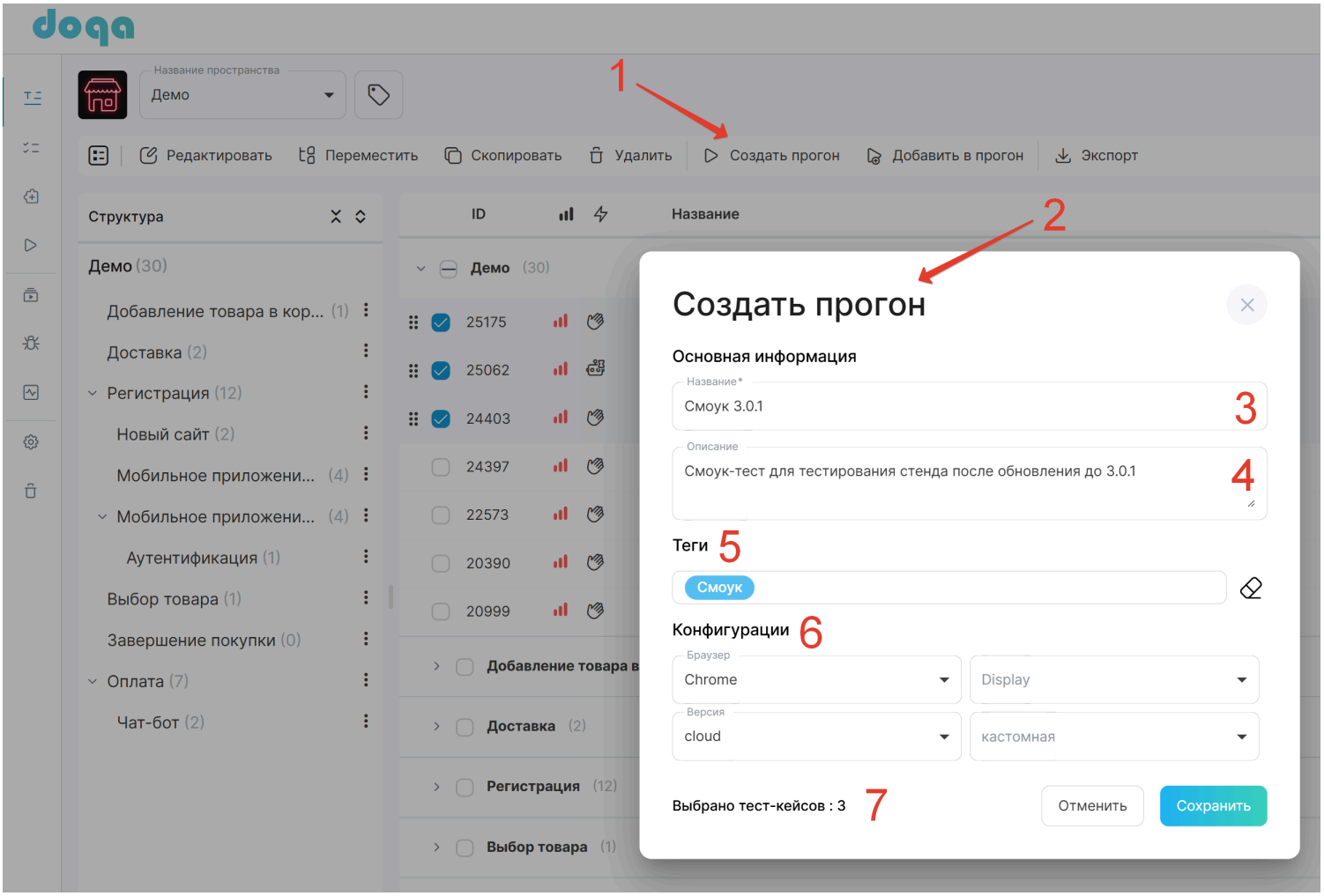
> **Важно**
>
> При создании прогона в нем сохраняются копии исходных тест-кейсов или чек-листов. Любые изменения этих тест-кейсов или чек-листов после создания прогона не повлекут изменения в прогоне.

Прогон можно создать несколькими способами.

Создание прогона во вкладках «Кейсы» или «Чек-листы»

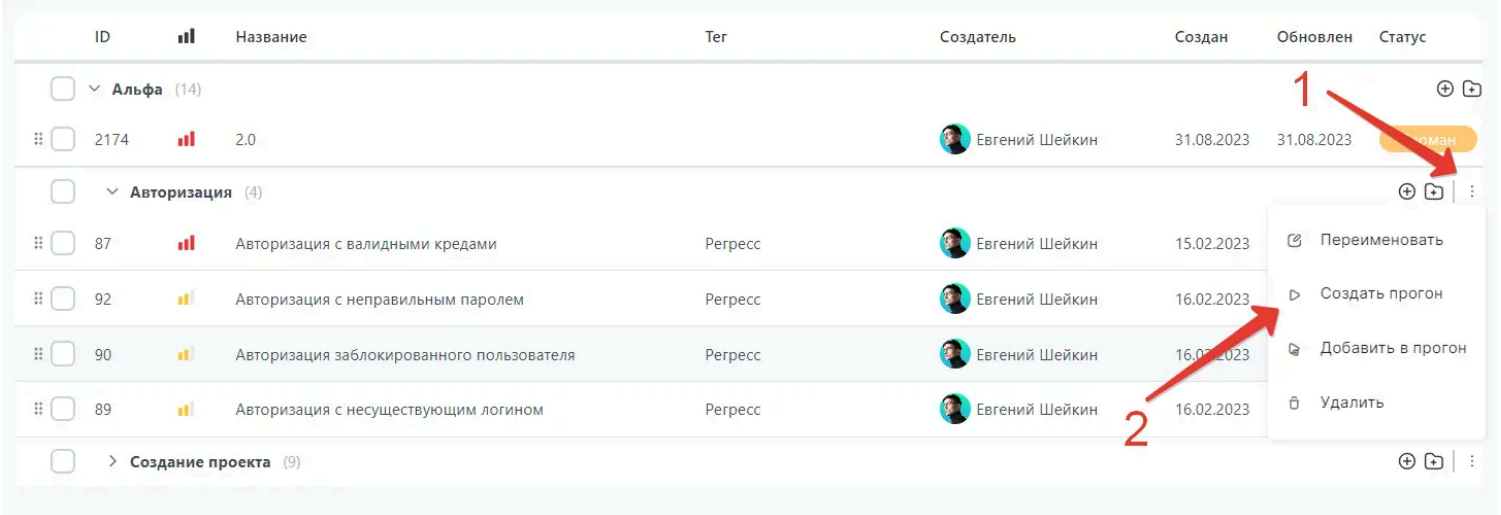
Выберите элементы при помощи чек-боксов и нажмите на кнопку «**Создать прогон**» (цифра 1). Если отметить чек-бокс папки, то будут отмечены все вложенные в нее элементы.

Откроется окно создания нового прогона (цифра 2). Для создания прогона необходимо ввести его название (цифра 3). Также можно заполнить описание (цифра 4), пометить прогон **тегами** (цифра 5), а также выбрать конфигурации для его прохождения (цифра 6). В окне создания прогона отображается число выбранных тест-кейсов (цифра 7).



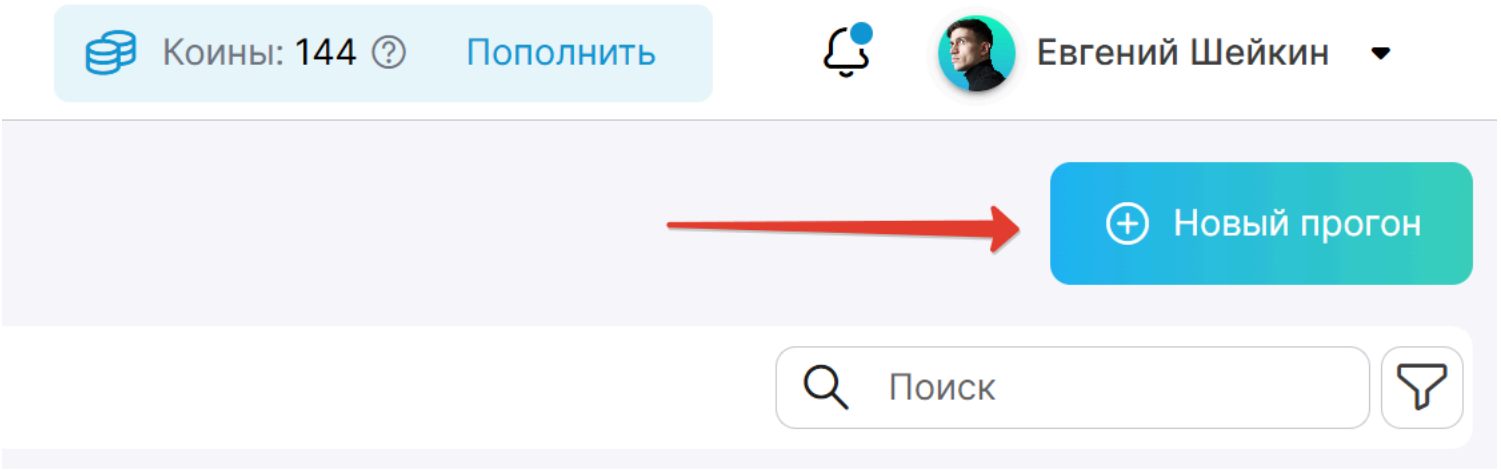
Создание прогона из папки

Прогон можно создать из тест-кейсов или чек-листов, содержащихся в **папке**. Нажмите на иконку «три точки» в шапке папки (цифра 1) и выберите пункт «**Создать прогон**» (цифра 2). Откроется окно создания нового прогона, в который добавлены тестовые документы из папки.



Создание прогона на вкладке «Прогоны»

Находясь на вкладке «Прогоны», нажмите на кнопку «**+ Новый прогон**». Откроется окно создания нового прогона.

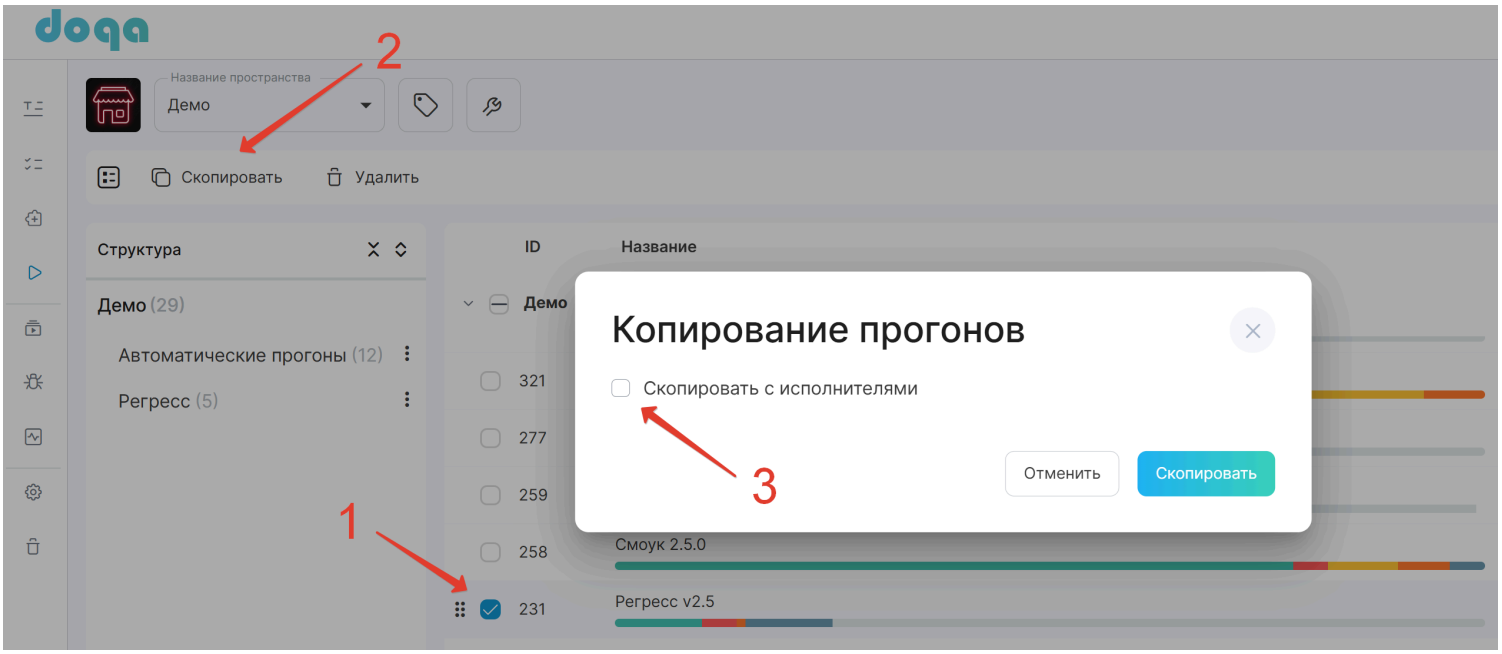


Копирование существующего прогона

Копирование прогона может быть актуально, если у вас есть стабильные наборы тестов для определенных тестовых циклов. Например, полный регрессионный набор.

В копию попадут новые версии всех тест-кейсов и чек-листов из исходного прогона. Если какие-либо кейсы и чек-листы были удалены после создания исходного прогона, они не попадут в копию прогона.

Для копирования прогона отметьте чек-бокс исходного прогона (цифра 1) и нажмите на кнопку «**Скопировать**» (цифра 2).



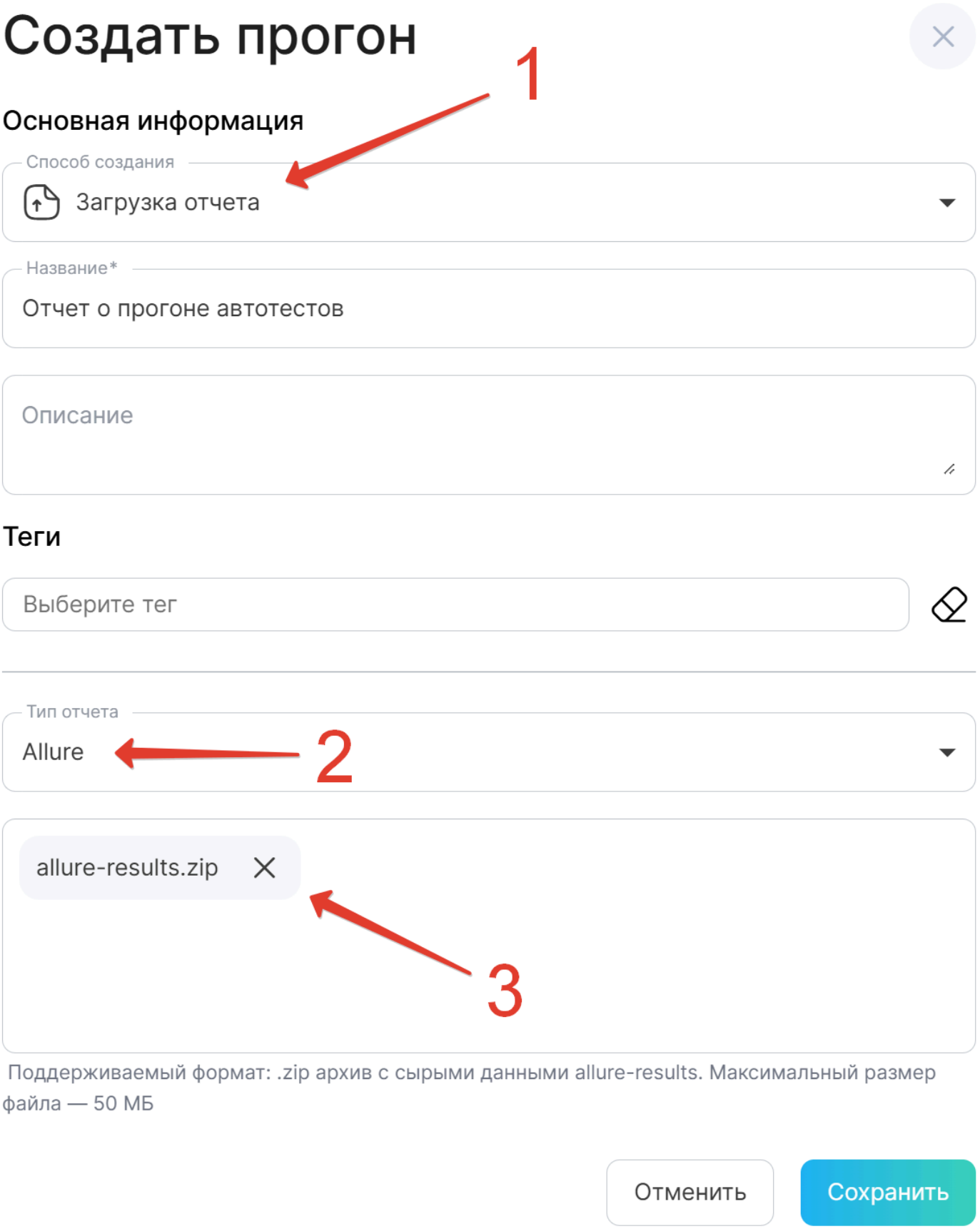
Откроется диалог «Копирование прогонов». Если вам важно сохранить распределение исполнителей как в исходном прогоне, отметьте в нем чек-бокс «Скопировать с исполнителями» и нажмите «**Скопировать**».

Если в исходном прогоне есть автотесты, в диалоге выводится предупреждение: автотесты, не имеющие связи с тест-кейсами, скопированы не будут.

Создание прогона на основе отчета о результатах автотестов

В окне создания прогона выберите способ «Загрузка отчета» (цифра 1). Выберите один из двух форматов отчета: Junit XML или Allure (цифра 2). Затем выберите файл для загрузки (цифра 3). В созданном прогоне будут отображаться результаты прохождения автотестов.

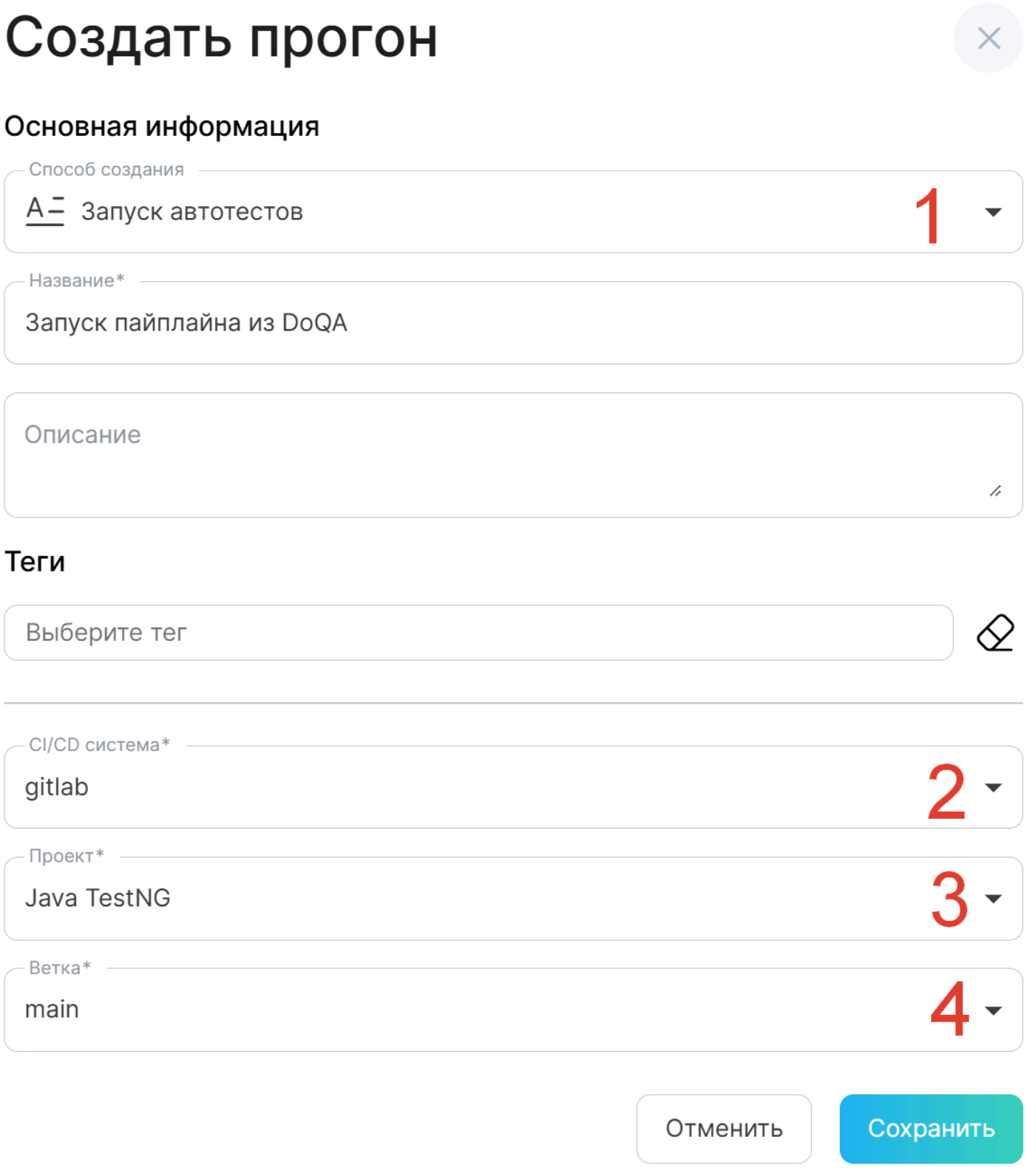
Загрузить можно один файл размером не более 50 МБ. Для JUnit поддерживаются форматы `.xml` или `.zip`-архив с XML-файлами, для Allure — `.zip`-архив с сырыми данными allure-results.



Создание прогона через запуск автотестов

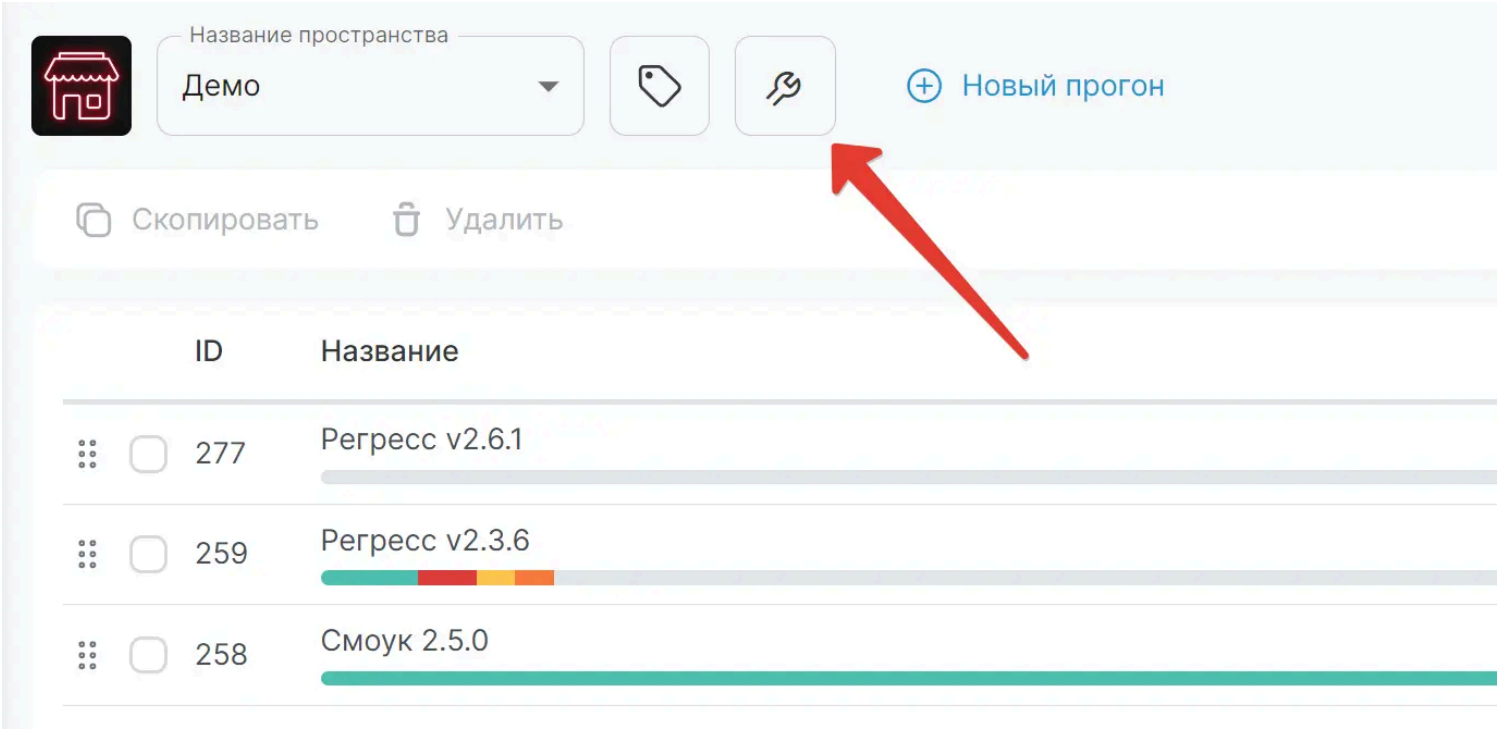
В окне создания прогона выберите способ «Запуск автотестов» (цифра 1). Выберите одну из подключенных **CI/CD систем** (цифра 2), проект из репозитория с автотестами (цифра 3) и ветку (цифра 4).

После создания прогона будет запущен пайплайн в связанной CI/CD системе. По мере выполнения автотестов в прогоне будут появляться их результаты. Подробнее — **Запуск автотестов из DoQA**.



Конфигурации прогонов

Для создания конфигураций нажмите на иконку редактора конфигураций во вкладке «Прогоны».



В редакторе заполните имя конфигурации и ее значения. Например: окружение, браузер, версия сборки.

Конфигурации

Название конфигурации*
Браузер

Chrome
Firefox
Safari

Название конфигурации*
Display

13" laptop
HD
4K

Название конфигурации*
Версия

cloud
box

Добавить конфигурацию

Отменить Сохранить

После сохранения изменений конфигурации будут доступны для выбора в окне создания и редактирования прогона.

Плеер прогонов

Плеер — экран, на котором проходят прогон. Чтобы открыть плеер, нажмите на название прогона в таблице.

Прогоны имеют несколько статусов:

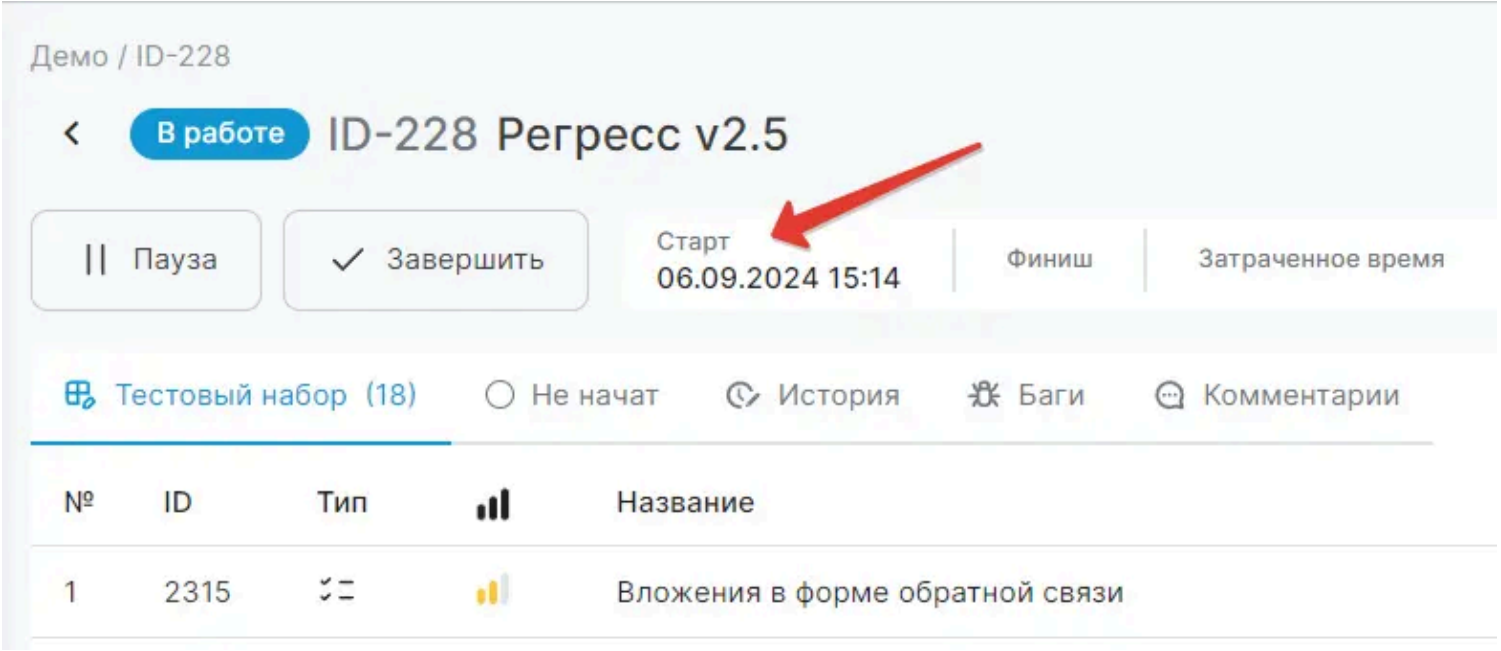
- «Готов» — прогон не начат.
- «В работе» — прогон начат.
- «Пауза» — прохождение прогона приостановлено.
- «Завершен» — прогон завершен.

Статус прогона показан в шапке плеера рядом с его ID и названием. Там же находятся кнопки «**Редактировать**» и «**Отчет**».

По нажатию на кнопку «**Запуск**»:

1. В плеере запускается таймер прохождения прогона;
2. Статус прогона меняется с «Готов» на «В работе»;
3. Вместо кнопки «**Запуск**» отображается кнопка «**Пауза**». Во время паузы останавливается таймер прогона, тесты выполнять нельзя;
4. Появляется возможность проставлять статус кейсам и чек-листам;
5. Становится активной кнопка «**Завершить**», которая завершает прохождение прогона.

Под кнопками выводится время начала и окончания прогона, затраченное время и шкала прогресса.



Если в прогоне есть автотесты, доступные для запуска, рядом появляется кнопка «**Запуск автотестов**». Она активна, только пока прогон в статусе «В работе». По нажатию открывается диалог выбора CI/CD-системы, проекта и ветки: автотесты запускаются в текущем прогоне, новый прогон не создается. Если перед нажатием отметить чек-боксы тестов в таблице, запуск ограничится ими.

Вкладки плеера

В плеере пять вкладок:

- «Тестовый набор» — таблица тестов прогона.
- «Информация» — описание прогона, его теги и конфигурации.
- «История» — история изменений прогона. Подробнее — **История редактирования**.
- «Баги» — баг-репорты, заведенные в этом прогоне. Подробнее — **Баги**.
- «Комментарии» — комментарии к прогону. Подробнее — **Комментарии**.

У вкладок «Тестовый набор», «Баги» и «Комментарии» в заголовке есть счетчик. Он показывается только у открытой вкладки.

Совместная работа

Если прогон изменил другой пользователь, в плеере появляется предупреждение «Прогон обновлен пользователем ...» с кнопками «**Ок**» (скрыть предупреждение) и «**Обновить**»

(перезагрузить данные прогона). Аналогичное предупреждение появляется в окне прохождения, если открытый тест изменил другой пользователь.

Показ предупреждения о прогоне зависит от настроек **уведомлений**.

Состав прогона

Тесты прогона перечислены в таблице на вкладке «Тестовый набор».

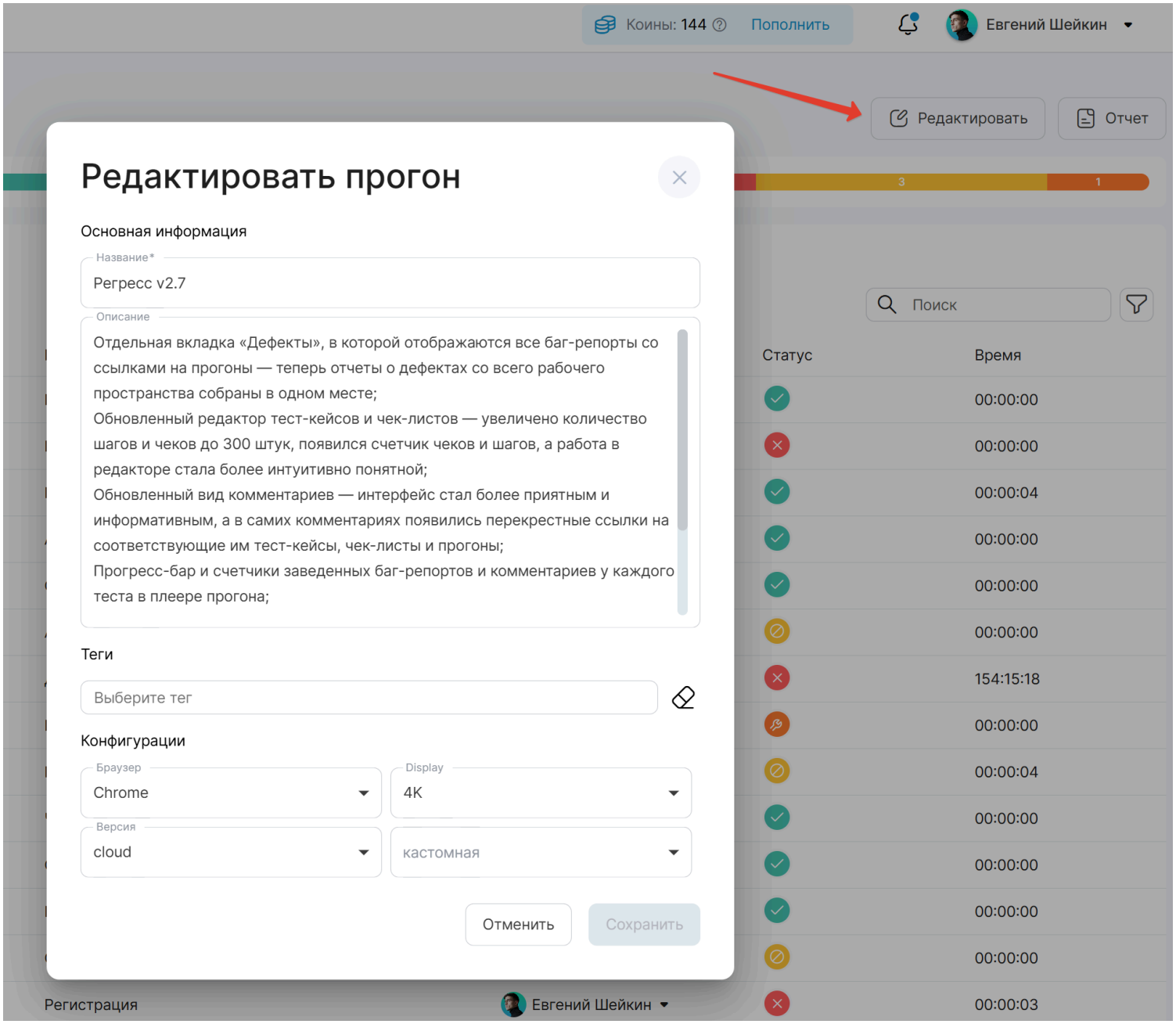
Панель инструментов таблицы

Над таблицей находятся:

- поиск по названию теста;
- фильтр по полям «Путь», «Приоритет», «Статус», «Исполнитель». Подробнее — **Фильтры**;
- кнопка-шестерёнка, которая открывает настройку колонок: показ, скрытие и перестановку.

Редактирование прогона

Для редактирования названия, описания, тегов и конфигураций прогона нажмите на иконку «**Редактировать**» в шапке плеера. Откроется диалоговое окно.



После того, как прогон будет пройден и его статус изменится на «Завершен», его нельзя будет редактировать. Если необходимо отредактировать прогон после его прохождения, нажмите на кнопку «**Запуск**» (для возобновления прогона), внесите изменения и снова нажмите на кнопку «**Завершить**».

Полезная информация

Что делать, если при прохождении тест-кейсов или чек-листов были созданы баг-репорты и комментарии, но потом стало понятно, что для продолжения прохождения нужно обновить документацию?

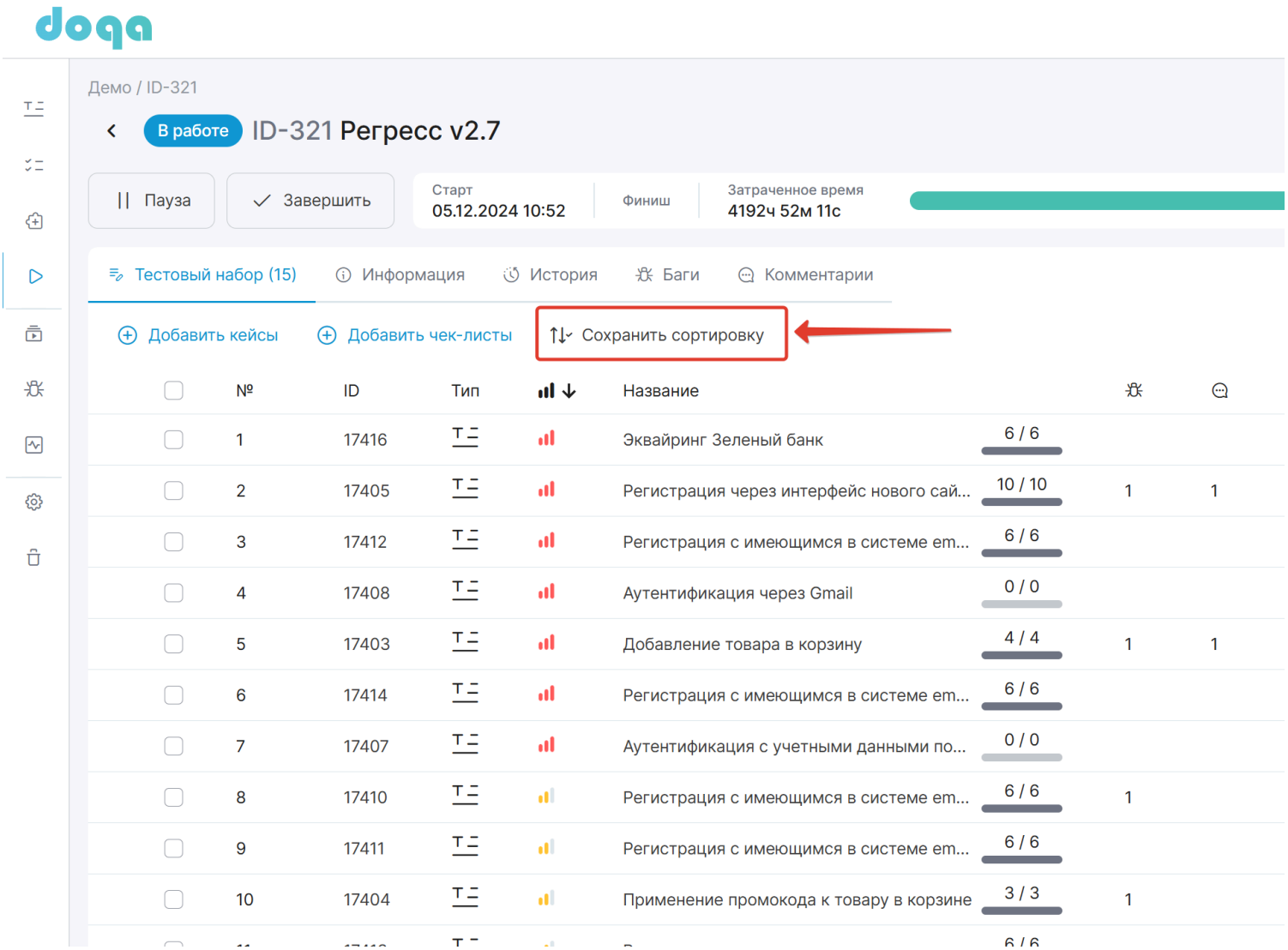
После удаления из прогона кейса или чек-листа, при прохождении которых были созданы баг-репорты или комментарии, эти баг-репорты и комментарии продолжат отображаться на экране прохождения соответствующего элемента, если вы вновь добавите те же тест-кейсы и чек-листы в

прогон (например, после их обновления). Такие баг-репорты и комментарии отображаются после всех шагов и чеков.

Сортировка и порядок тестов

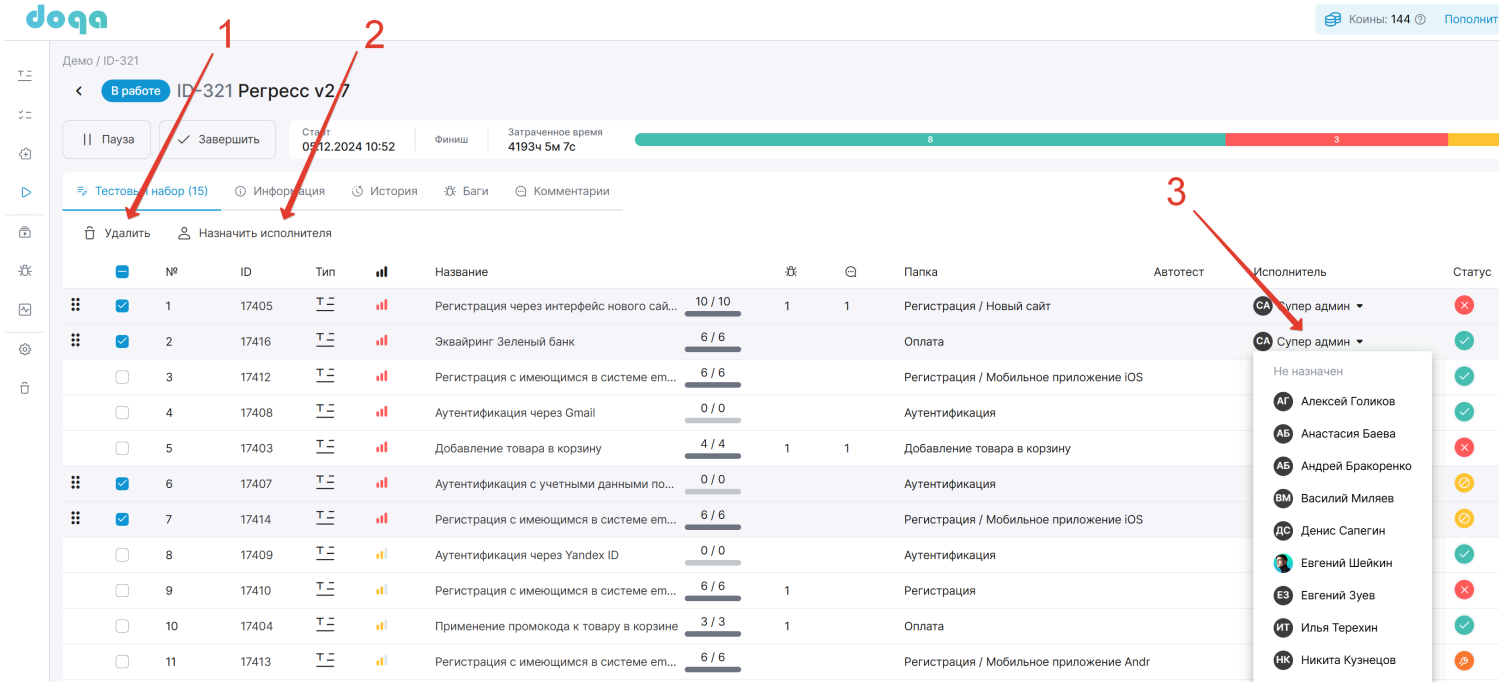
Вы можете отсортировать тесты в прогоне по одному из столбцов таблицы. При этом сортировка индивидуальна и видна только вам. Чтобы зафиксировать текущий порядок для всех, нажмите «**Сохранить сортировку**». Кнопка доступна, только когда не применены поиск и фильтр.

Порядок тестов можно менять и вручную, перетаскивая строки. Перетаскивание отключено, пока активен поиск или фильтр, а также в завершенном прогоне.



Массовые действия

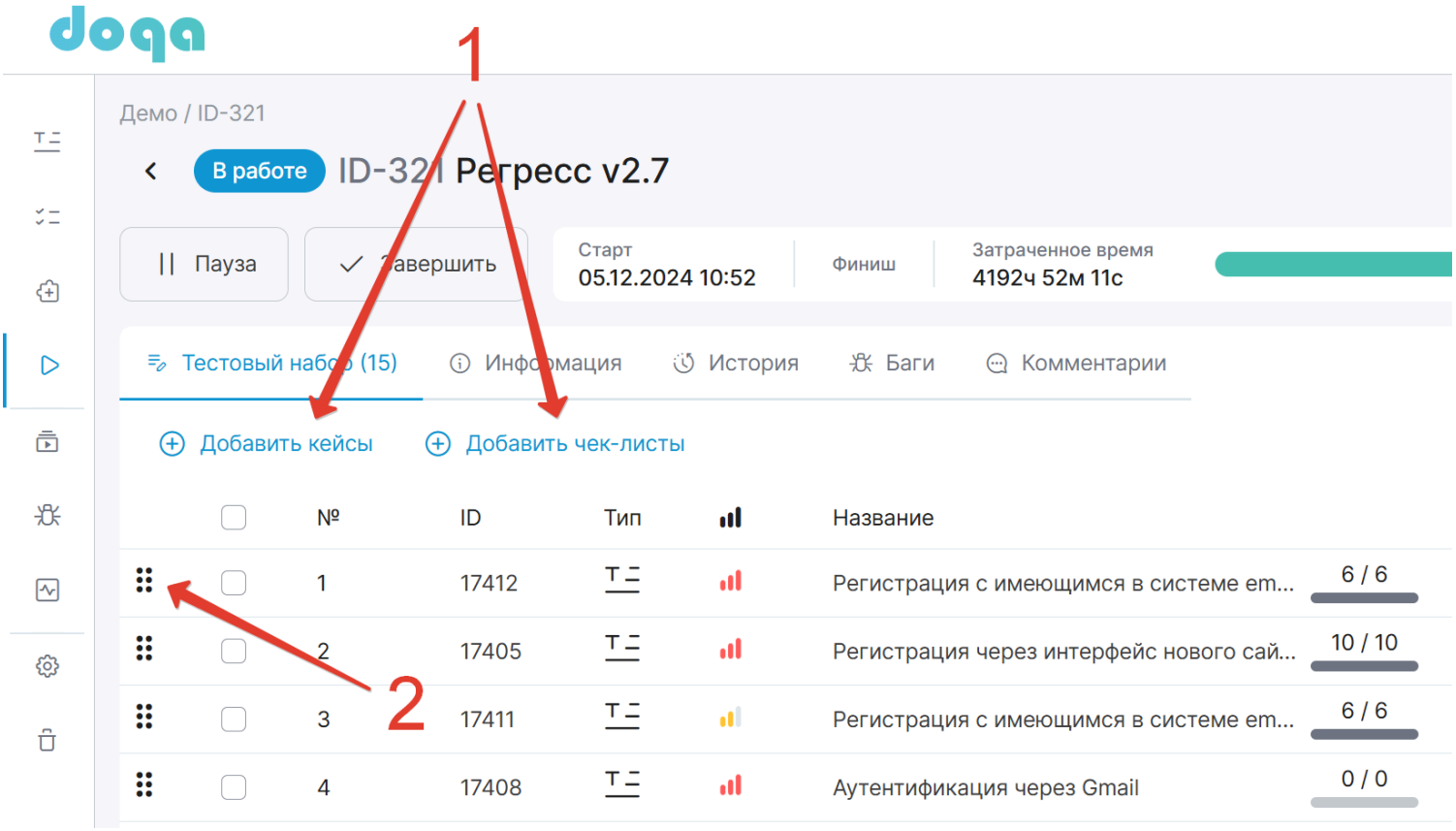
При отметке одного или нескольких чек-боксов тестов в прогоне появляются кнопки с доступными действиями: удаление тестов из прогона (цифра 1) и массовое назначение исполнителей (цифра 2). Исполнителей также можно назначать на каждый тест по-отдельности, нажав на поле выбора исполнителя (цифра 3).



Добавление кейсов или чек-листов через плеер прогона

Добавление в прогон новых кейсов и чек-листов происходит одинаково. Ниже этот процесс показан на примере кейсов.

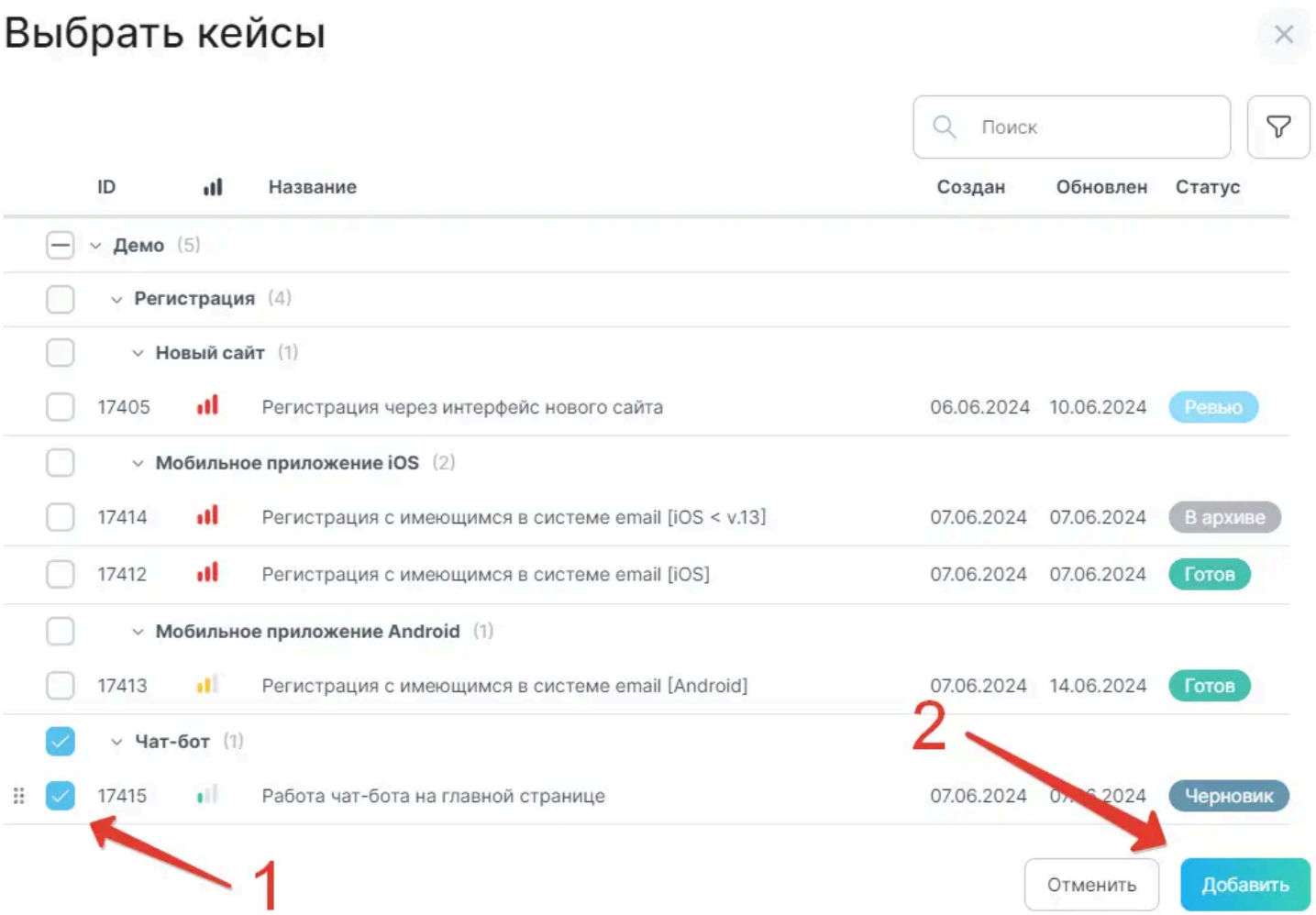
Нажмите на кнопку «**+ Добавить кейсы**» (цифра 1). Порядок добавленных тестов можно затем изменить вручную (цифра 2).



В открывшемся окне будут отображаться все кейсы из текущего рабочего пространства за исключением тех, которые уже содержатся в прогоне.

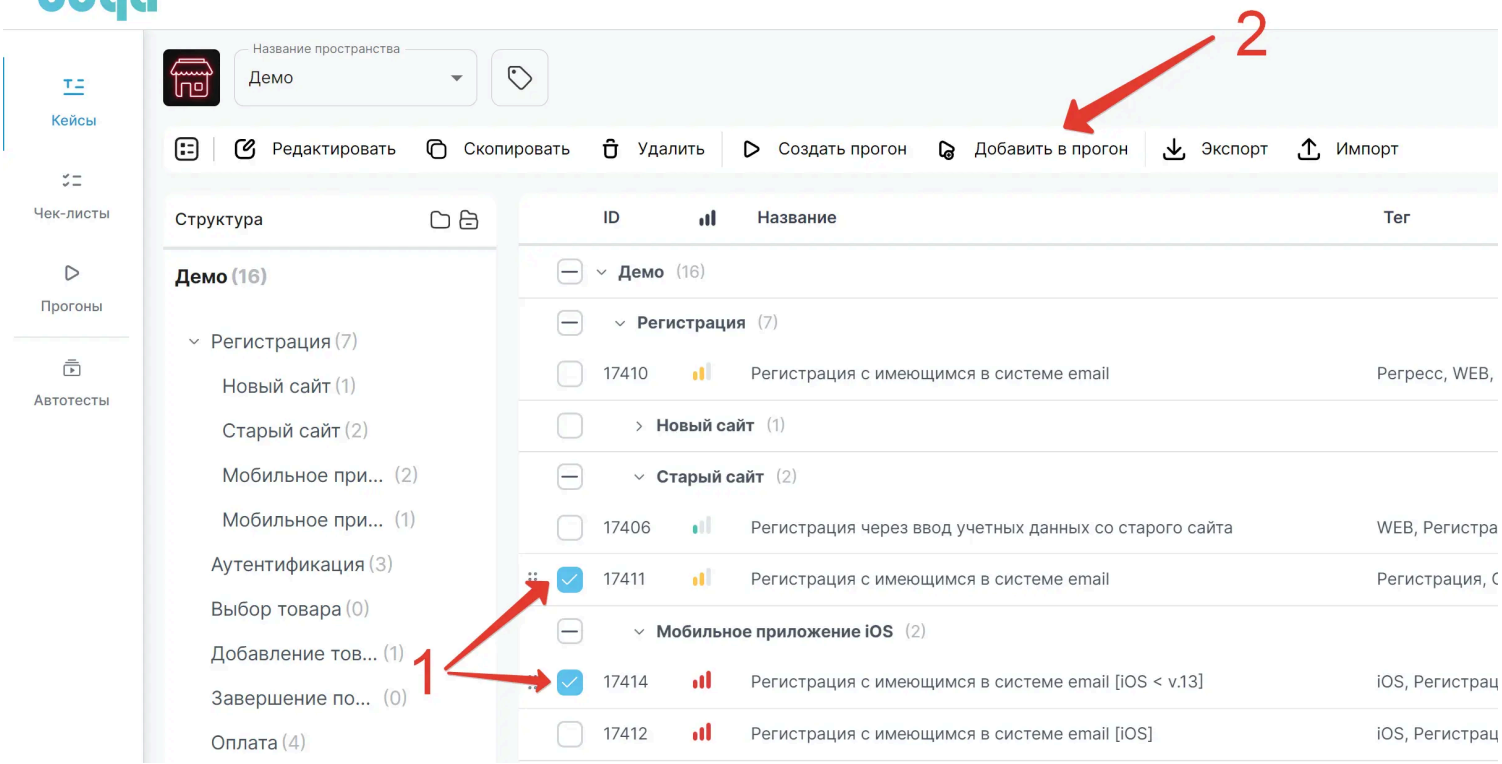
Чтобы найти нужные элементы, можно воспользоваться поиском, фильтром или сортировкой.

Отметьте чек-боксы тех тест-кейсов, которые хотите добавить в прогон (цифра 1). Затем нажмите на кнопку «**Добавить**» (цифра 2). Выбранные тест-кейсы добавятся в прогон.

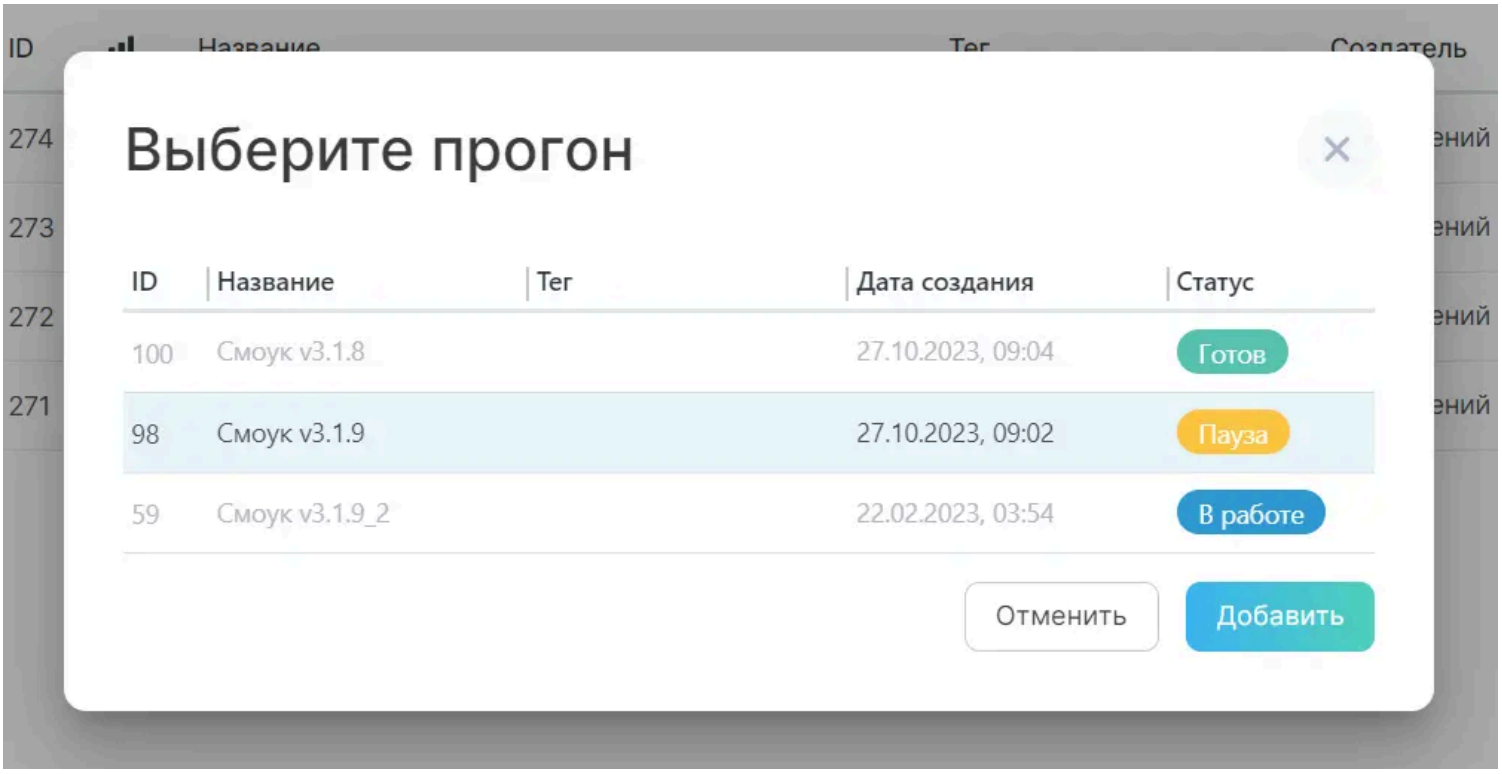


Добавление новых элементов в прогон с вкладок «Кейсы» и «Чек-листы»

Добавление новых тест-кейсов или чек-листов в ранее созданный прогон доступно также напрямую из вкладок «Кейсы» и «Чек-листы». Для этого выберите элементы (цифра 1) и нажмите на кнопку «**Добавить в прогон**» (цифра 2). Если отметить чек-бокс папки, отмечены будут все вложенные элементы.

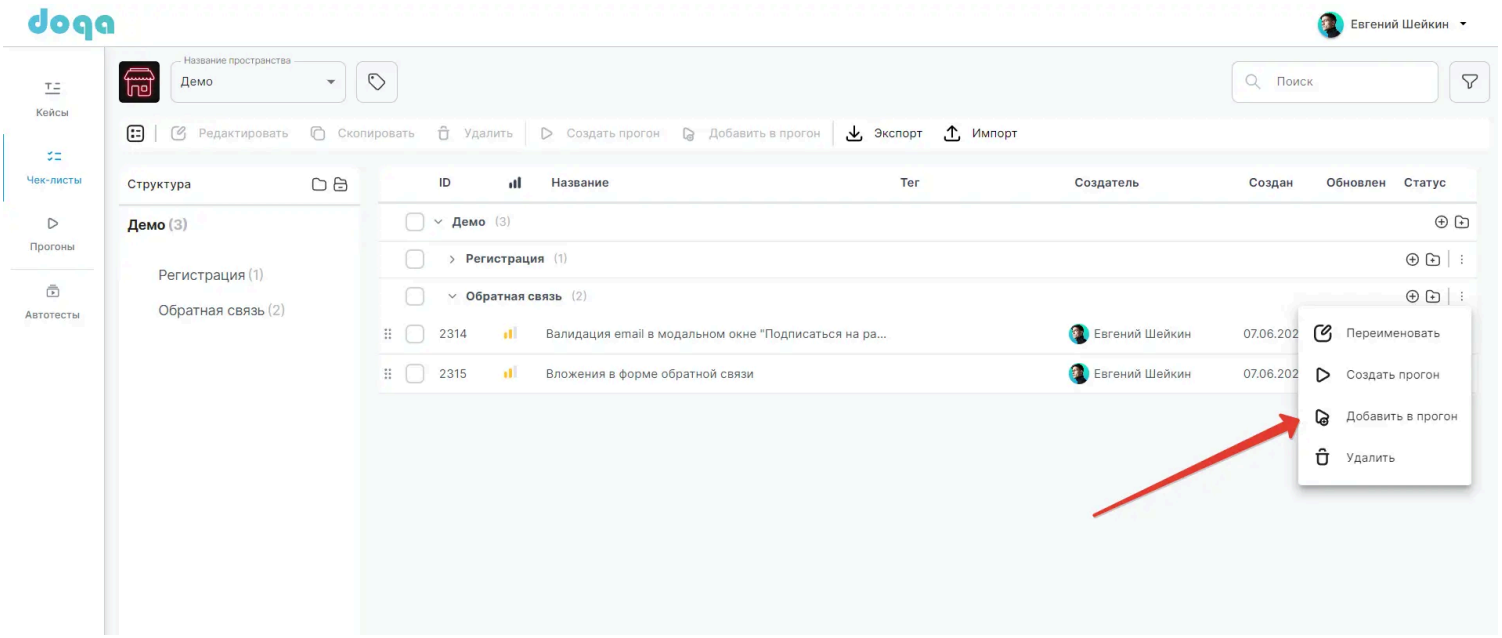


В открывшемся диалоговом окне выберите один из прогонов (цифра 1) и нажмите на кнопку «**Добавить**» (цифра 2). Выбранные тесты добавятся в прогон. Добавлять элементы можно в прогоны со статусами «Готов», «В работе» и «Пауза».



Добавление новых элементов в прогон из папок кейсов или чек-листов

Можно добавлять в существующий прогон тест-кейсы или чек-листы, которые содержатся в определенной папке, нажав на иконку меню папки и выбрав «**Добавить в прогон**».



Исполнители

Если кейсу или чек-листу не назначен **исполнитель**, то их прохождение доступно всем пользователям, у которых есть лицензия и права на выполнение прогонов в этом пространстве (`run.execute`). Подробнее - **Проекты**.



Если кейсу или чек-листу назначен исполнитель, то их прохождение будет доступно только этому пользователю.

В режиме редактирования прогона можно изменить исполнителя кейсов или чек-листов.

Прохождение прогона

Проставление статусов в таблице прогона

Тестам в прогоне можно выставить следующие статусы:

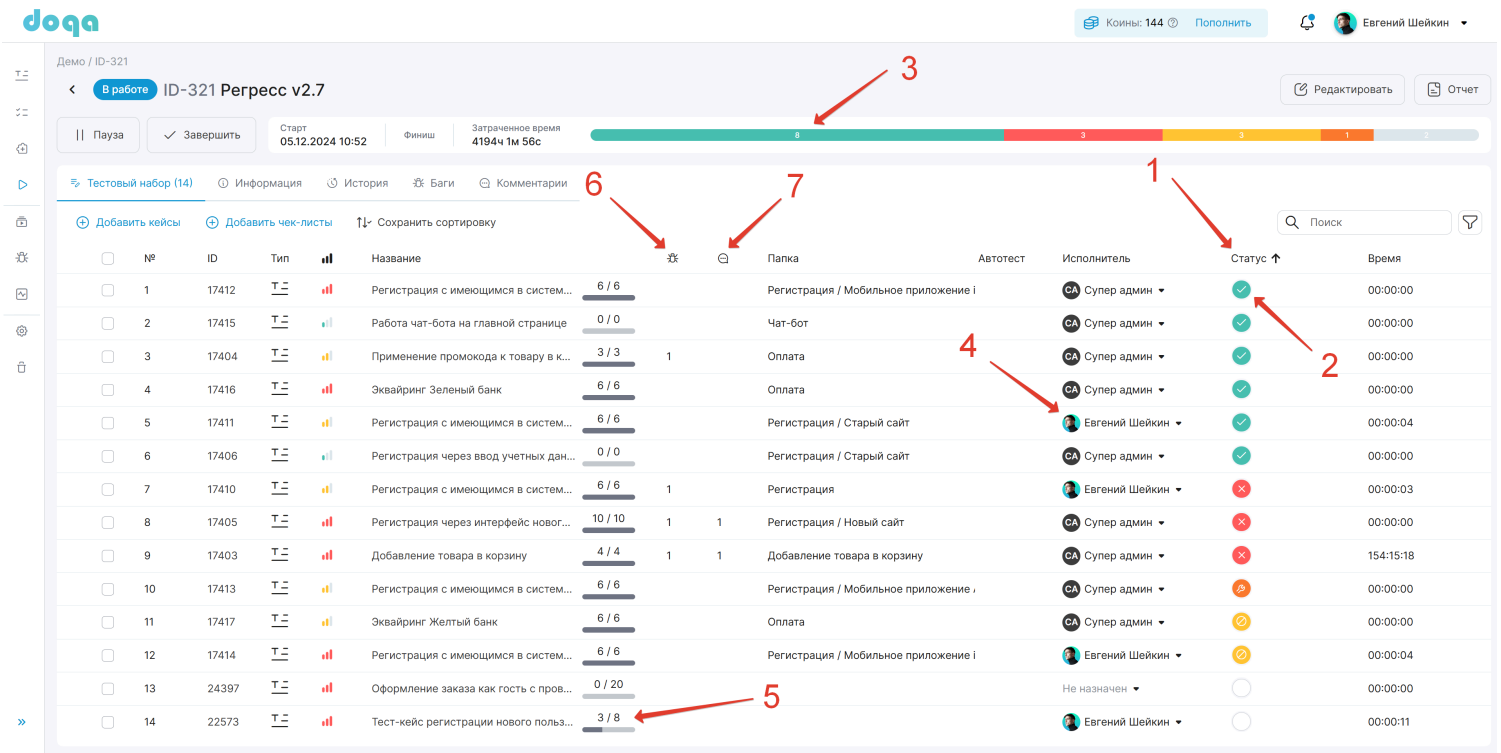
- Пройден
- Провален
- Заблокирован
- Сломан
- Пропущен

Статус элемента может быть проставлен в панели управления статусами в столбце «Статус» (цифра 1) или в окне прохождения, которое будет рассмотрено далее. Выставленный статус отображается в виде иконки соответствующего цвета (цифра 2), а также в шкале прогресса (цифра 3).

Исполнителем записывается пользователь, выставивший статус (цифра 4) — одинаково для тест-кейсов и чек-листов. Автоназначение исполнителя по старту таймера работает только для тест-кейсов.

В таблице отображается прогресс прохождения каждого тест-кейса и чек-листа (цифра 5). В отдельных столбцах указаны счетчики заведенных баг-репортов (цифра 6) и созданных комментариев (цифра 7) по каждому тесту в таблице.

Если статус выставлен всему чек-листу или тест-кейсу, который еще не начат, то такой же статус будет проставлен всем непройденным чекам или шагам внутри него.



Параметризованные тесты помечены в таблице иконкой с номером набора значений. Подробнее — **Параметры**.

Статистика по тесту

По каждому тесту DoQA считает статистику прохождения:

- «Плановое время» — время, заложенное на прохождение теста;
- «Среднее время» — рассчитано на основе 10 последних попыток;
- «Успешно» — процент успешных попыток из 10 последних;
- «Последние прохождения» — 3 последних статуса с датами.

Все четыре показателя выводятся в правой панели окна прохождения. Рядом с «Последними прохождениями» есть стрелка «Перейти в историю»: она открывает в новой вкладке **Историю запуска** тест-кейса или чек-листа. Стрелка не показывается для тестов, пройденных автоматизацией.

Краткая карточка со «Средним временем», «Успешно» и тремя последними прохождениями показывается при наведении на цветной индикатор статуса слева от строки в таблице прогона.

Окно прохождения

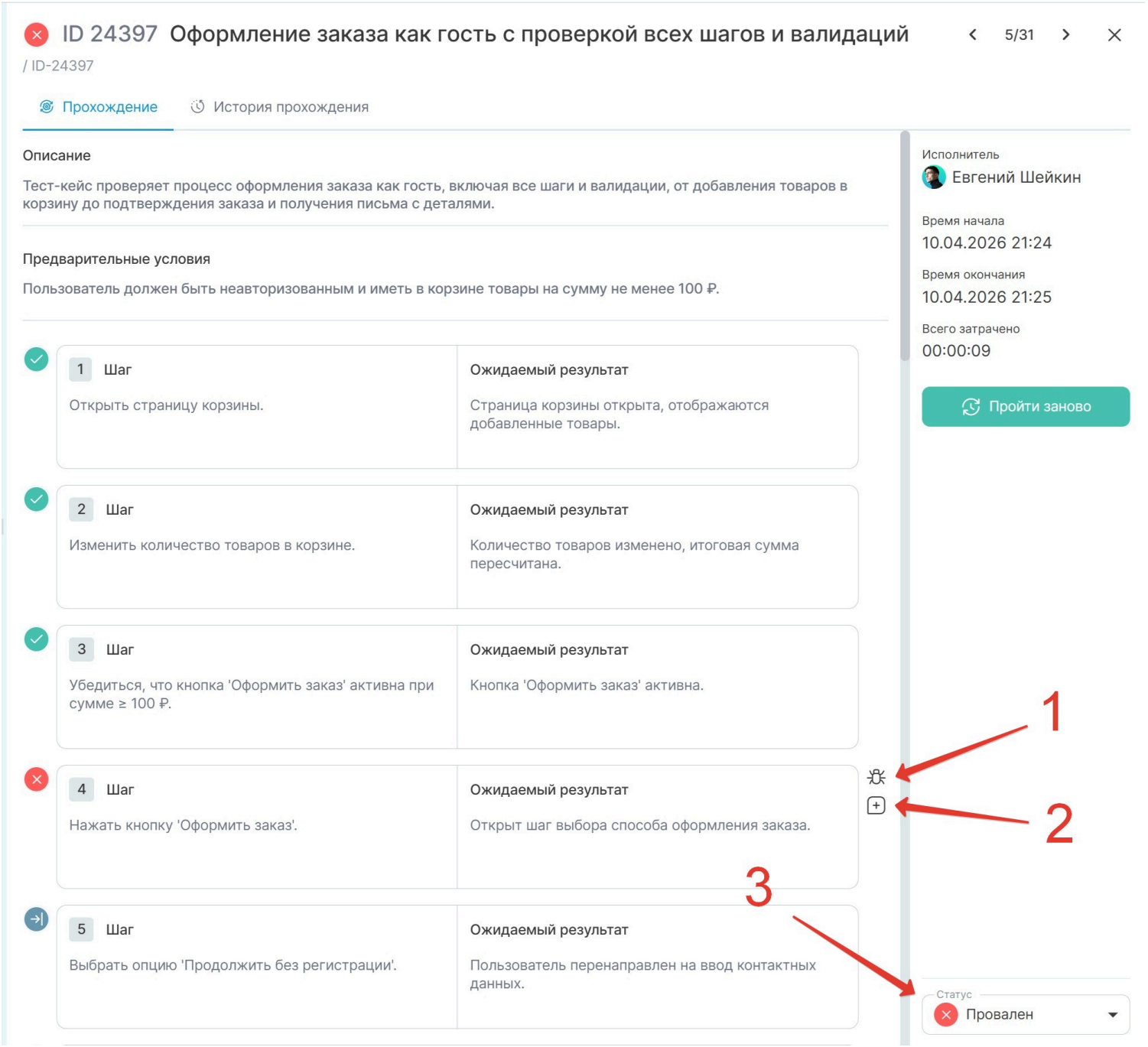
Для прохождения тест-кейса или чек-листа нажмите на его название в таблице прогона. Тест открывается в боковой панели, ширину которой можно менять. Стрелками «‹» и «›» в шапке панели можно переходить к предыдущему и следующему тесту, счетчик рядом показывает позицию теста в общем числе. В панели две вкладки: «Прохождение» и «История прохождения».

Статус тесту выставляется в выпадающем списке «Статус» в правой панели окна прохождения. Для чек-листа рядом с полем есть иконка-подсказка: «Статус чек-листа присваивается непройденным чекам».

Если у теста есть параметры, их значения показаны на вкладке «Прохождение» блоком «Параметры».

Прохождение кейса

К любому шагу кейса можно добавить баг-репорты (цифра 1) и комментарии (цифра 2). Создание баг-репорта не влияет на статус тест-кейса или чек-листа. **Подробнее о создании баг-репортов**

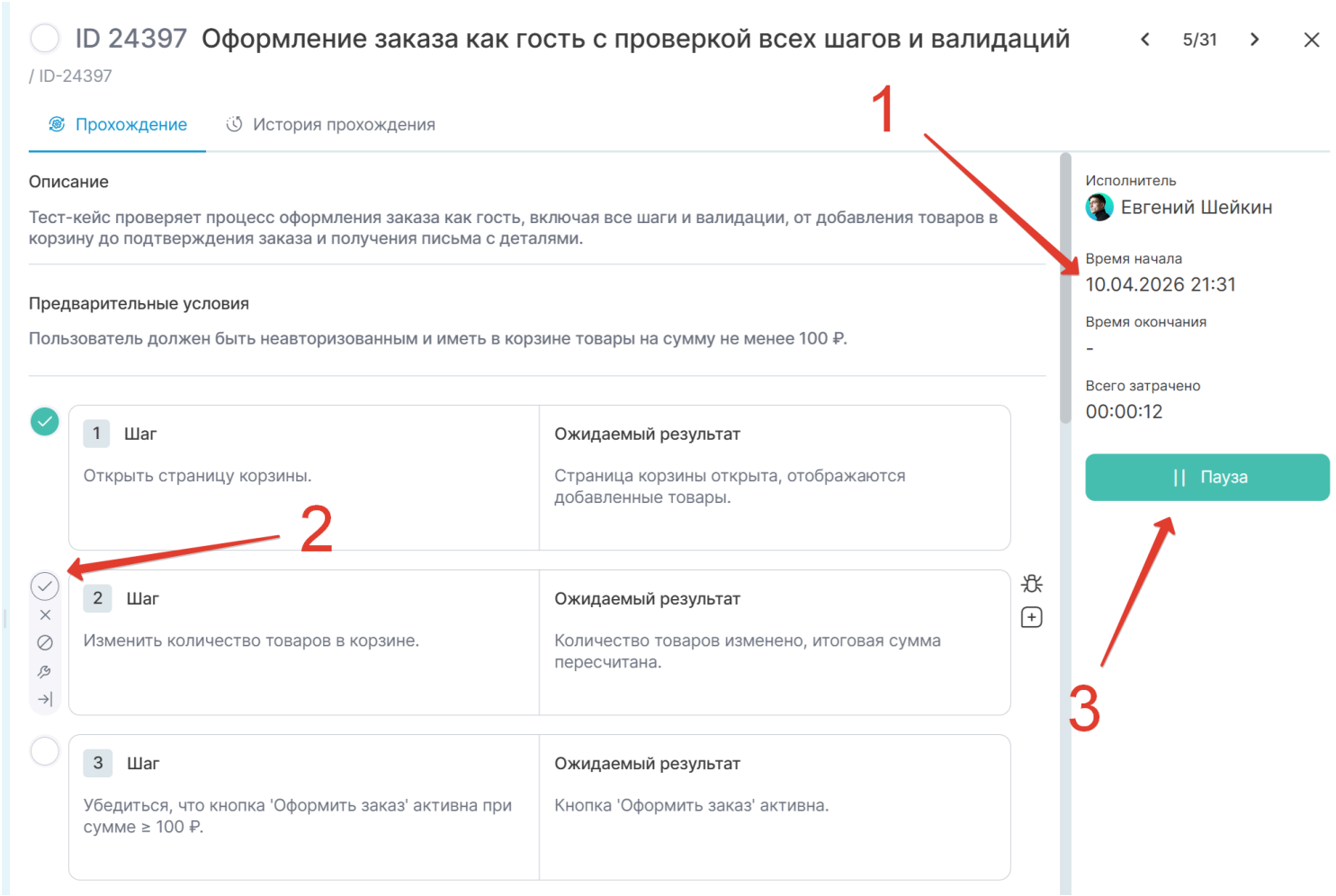


Проходить тест-кейс в прогоне можно также отмечая статус каждого шага (цифра 1). При этом в статистике прохождения тестового прогона учитываются только статусы тест-кейсов, но не статусы отдельных шагов.

После проставления статуса первому шагу, автоматически начинается отсчет времени прохождения тест-кейса (цифра 2). Отсчет времени также можно запустить или

приостановить нажав на кнопку «Старт»/«Пауза»/«Продолжить» (цифра 3). Счетчик будет остановлен, когда будут проставлены статусы всем шагам или самому тест-кейсу.

После завершения прохождения та же кнопка меняется на «**Пройти заново**». По нажатию появится подтверждение «Пройти заново»; после подтверждения статус кейса и шагов, а также таймер прохождения будут сброшены, а связанные баги и комментарии удалятся.



Если всем шагам кейса выставлен одинаковый статус «Пройден», «Заблокирован», «Сломан» или «Пропущен», то всему кейсу автоматически выставляется соответствующий статус. Если хотя бы одному шагу выставлен статус «Провален», то всему кейсу выставляется статус «Провален», а всем оставшимся непройденным шагам — статус «Пропущен».

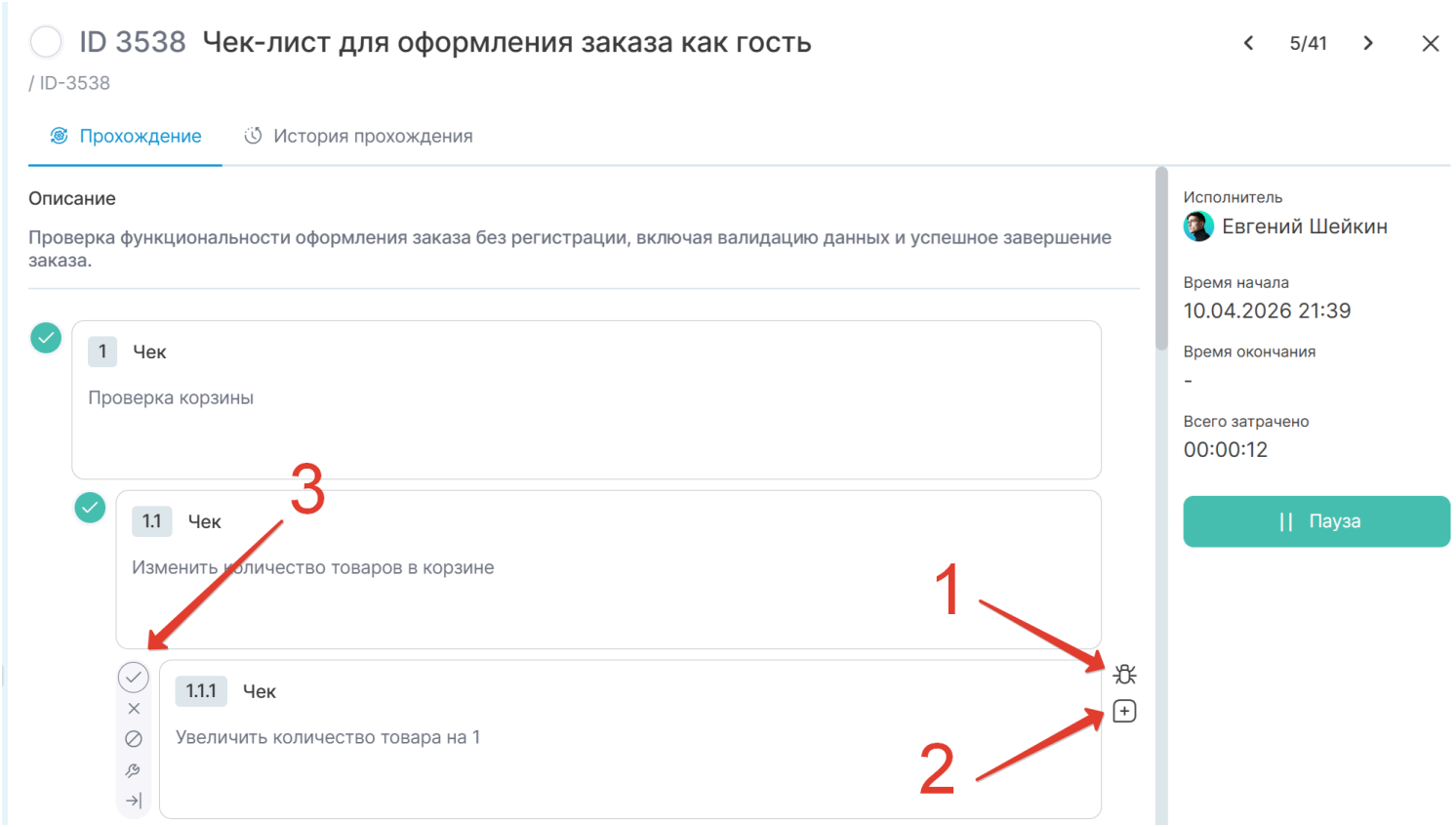
Ранее выбранный статус можно изменить на другой, пока прогон находится в статусе «В работе».

Прохождение чек-листа

К каждому чеку (пункту чек-листа) можно добавить баг-репорты (цифра 1) и комментарии (цифра 2). Создание баг-репорта не влияет на статус тест-кейса или чек-листа. **Подробнее о**

создании баг-репортов

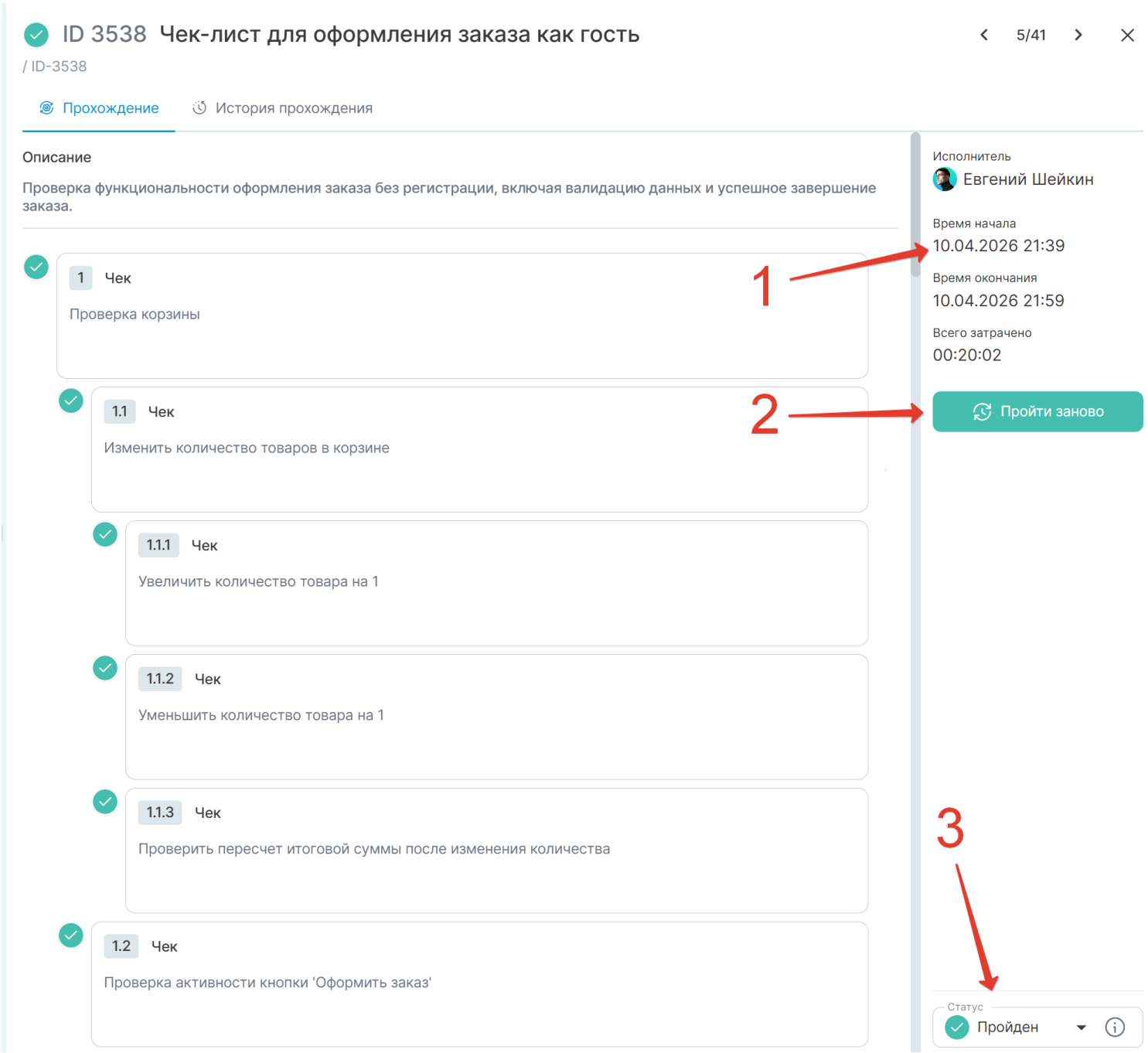
Каждому чеку может быть выставлен один из 5 статусов: Пройден, Провален, Заблокирован, Сломан, Пропущен (цифра 3).



После проставления статуса первому чеку, автоматически начинается отсчет времени прохождения чек-листа (цифра 1). Отсчет времени также можно запустить или приостановить нажав на кнопку «Старт»/«Пауза»/«Продолжить». Счетчик будет остановлен, когда будет проставлен статус чек-листа или всех чеков внутри него.

После завершения прохождения кнопка таймера меняется на «**Пройти заново**» (цифра 2): статус чек-листа и всех чеков внутри него, а также таймер прохождения будут сброшены, а связанные баги и комментарии удалятся.

При проставлении статуса всем чекам в чек-листе статус самого чек-листа не меняется автоматически, его нужно выбрать в списке «Статус» в правой панели окна прохождения (цифра 3).



Работа с автотестами в прогоне

У тестов, пройденных автоматизацией, в качестве исполнителя указана «Автоматизация», а в правой панели окна прохождения выводится ссылка на Job в CI/CD-системе.

Для таких тестов в правой панели доступна кнопка «**Пройти заново**» с двумя вариантами:

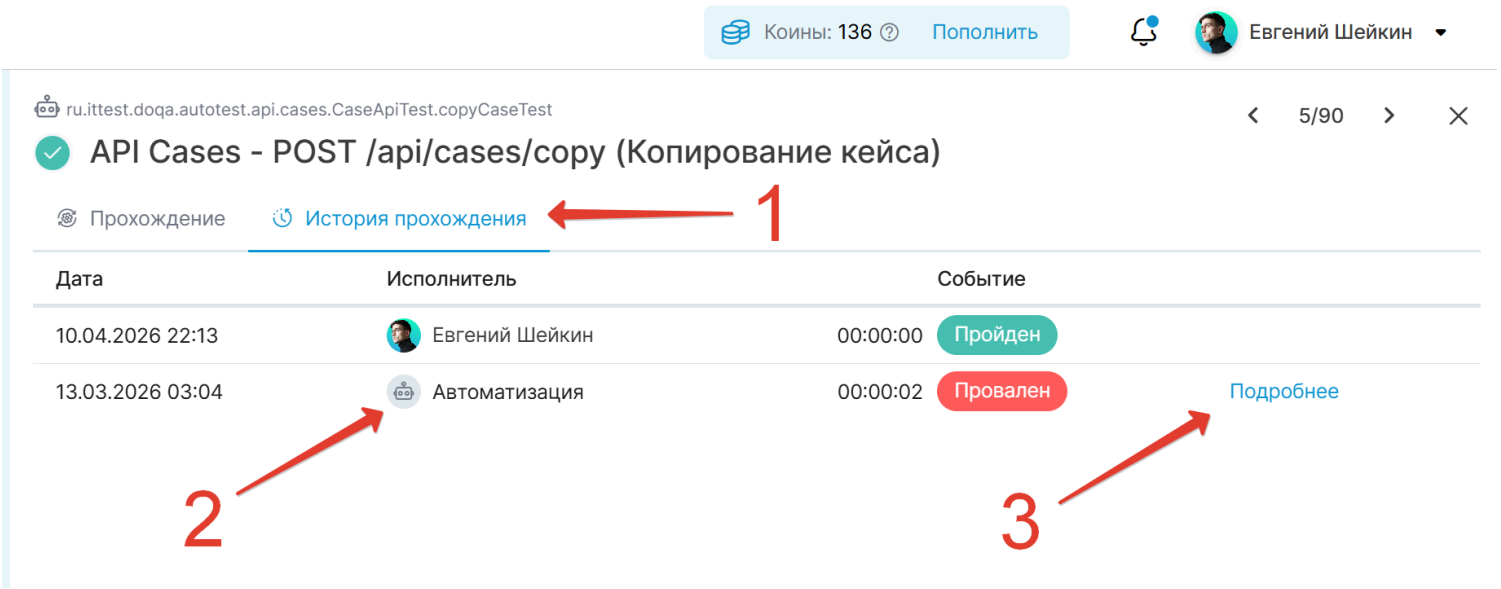
- «Пройти вручную» — после подтверждения DoQA классифицирует тест-кейс как ручной и открывает список шагов;
- «Автоматически» — перезапуск автотеста.

Кнопка недоступна в завершенном прогоне и у тестов, которым еще не выставлен статус. Подробнее — **Перепрохождение одного автотеста**.

История прохождения теста в прогоне

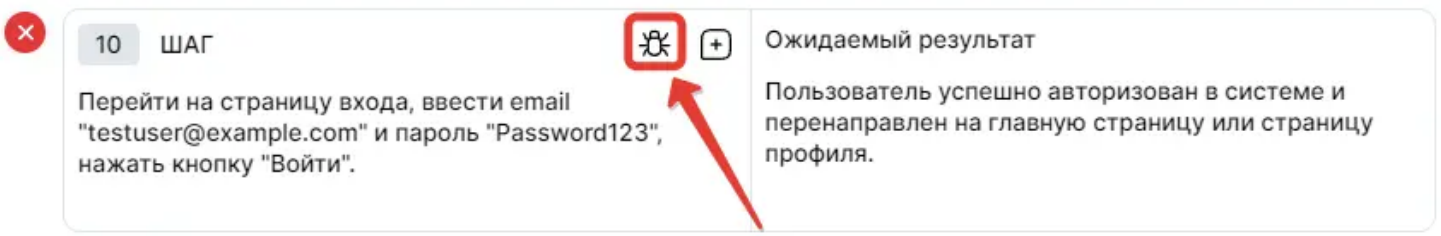
В плеере можно увидеть историю прохождения каждого теста в прогоне. Выберите один из тестов в списке и откройте вкладку «История прохождения» (цифра 1).

У автотестов в качестве исполнителя указана «Автоматизация» (цифра 2). Для получения подробной информации об автоматическом прохождении теста нажмите на кнопку «**Подробнее**» (цифра 3).



Создание баг-репортов

Если при прохождении тест-кейса или чек-листа обнаружен дефект, нажмите на иконку добавления баг-репорта у соответствующего шага или чека.



Открывается окно для создания баг-репорта, в котором ряд полей уже предзаполнены. Ниже описано каждое поле.

В поле «**Имя дефекта**» подставляется название тест-кейса, при прохождении которого обнаружен баг (цифра 1). Поле обязательно для заполнения. Рядом с полем есть кнопка с иконкой AI — «Сгенерировать название баг-репорта». Она недоступна, если AI не настроен; подробнее — **AI-функции**.

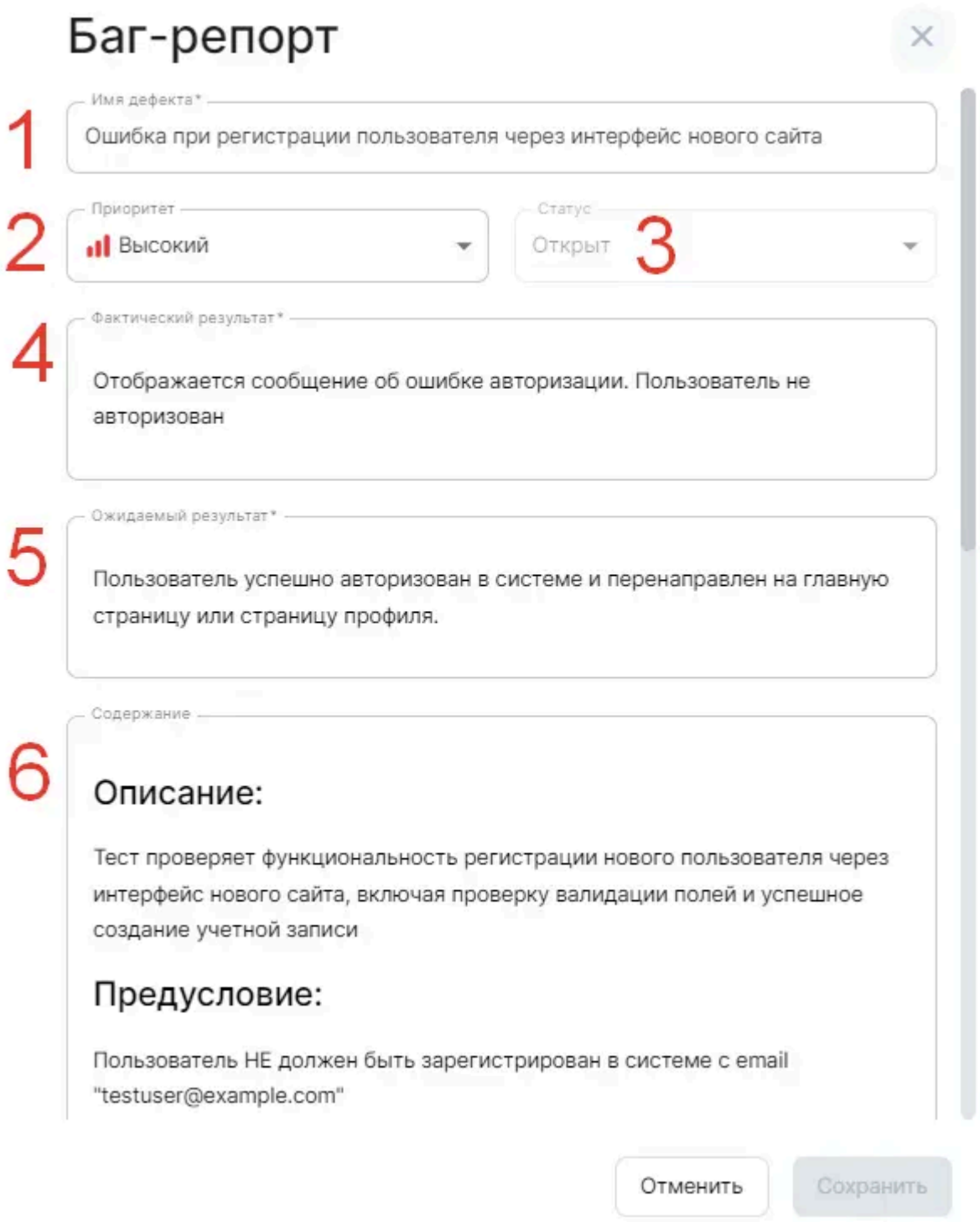
По умолчанию выставлен средний **приоритет** бага. Значение можно изменить на «Низкий» и «Высокий» (цифра 2).

Статус баг-репорта по умолчанию «Открыт» (цифра 3), также можно выбрать значения «В работе», «Тестирование», «Закрыт». Если активен чек-бокс «Связь с трекером» (цифра 8), статус «Открыт» нельзя изменить.

Фактический результат по умолчанию не заполнен (цифра 4). Поле обязательно для заполнения.

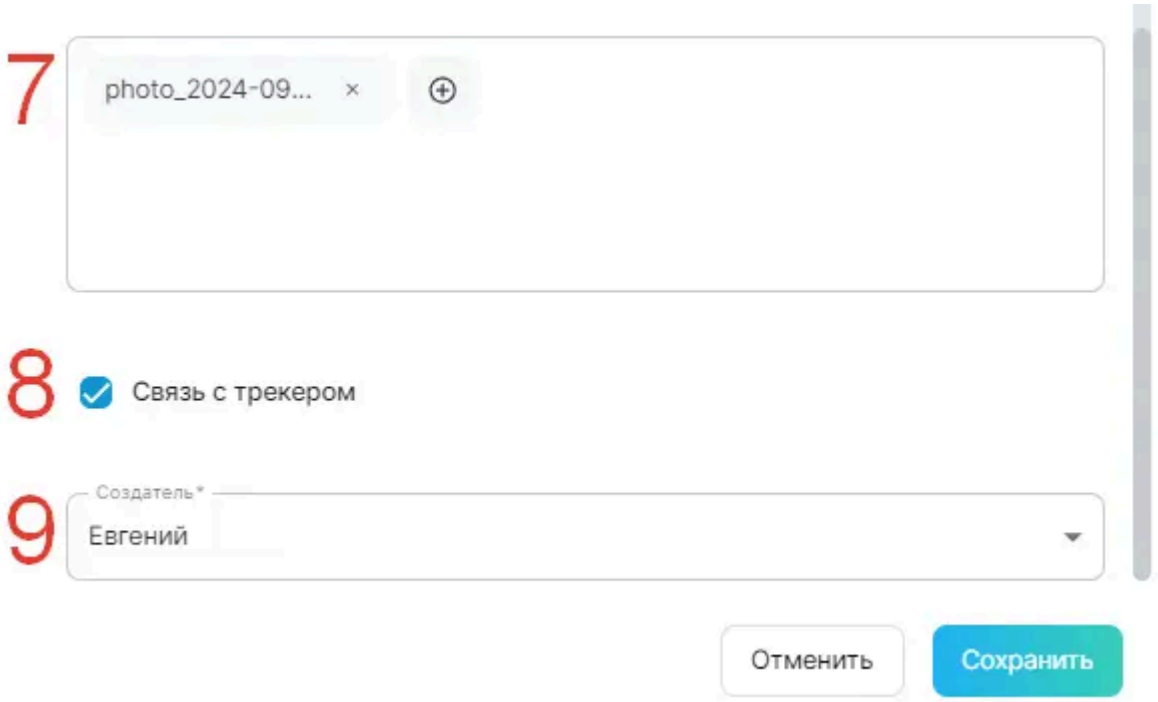
На основе выбранного шага автоматически подставляется соответствующее ему значение **ожидаемого результата** (цифра 5). Поле обязательно для заполнения.

В поле «**Содержание**» предзаполнены данные из описания, предусловий и постусловий тест-кейса, а также описаны шаги воспроизведения дефекта (цифра 6).

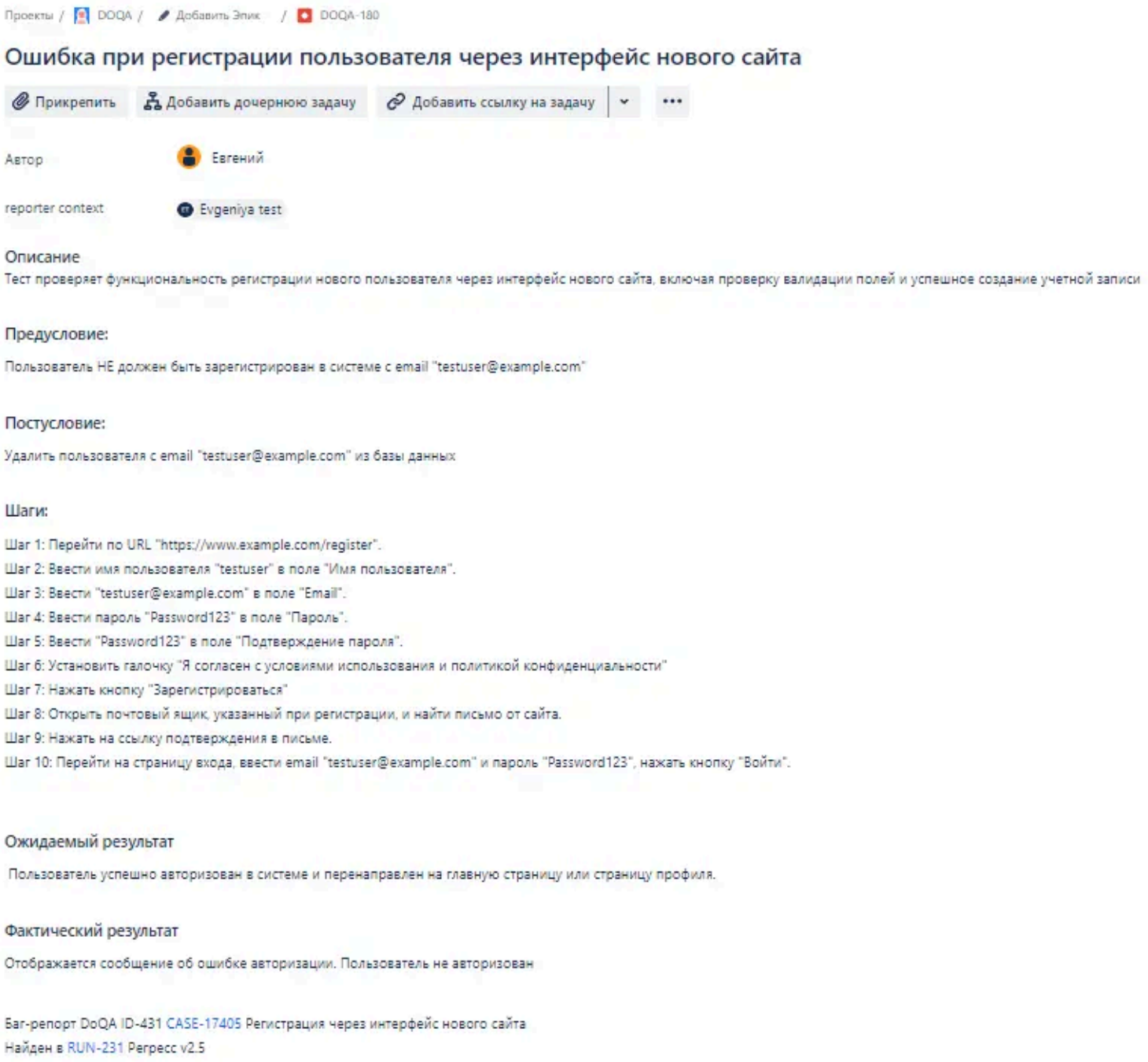


При необходимости к баг-репорту можно прикрепить **вложения** (цифра 7). Допускается любой формат вложения, размер каждого файла не более 10 МБ, не более 50 файлов.

Если настроена **интеграция проекта с трекером задач**, автоматически отмечается чек-бокс «**Связь с трекером**» (цифра 8). Для Jira и Битрикс24 появляется дополнительное обязательное поле «**Создатель**» (цифра 9) — пользователь трекера, от имени которого будет заведена задача. Для YouTrack, Яндекс Трекера, Redmine, GitLab и Kaiten этого поля нет.



Если был выбран чек-бокс «Связь с трекером», баг-репорт создается одновременно в DoQA и в трекере задач. Описание баг-репорта в трекере содержит ссылку на тест-кейс и прогон, в которых был обнаружен дефект.



Завершение прогона

Прогон можно завершить в любое время после его запуска нажатием на кнопку «**Завершить**». Появится диалог подтверждения «Завершение прогона»: непройденным тестам будет выставлен статус «Пропущен», а все незавершенные запуски тестов в CI/CD будут остановлены.

После подтверждения завершения прогона в плеере отображается время завершения тестирования и затраченное на прогон время.



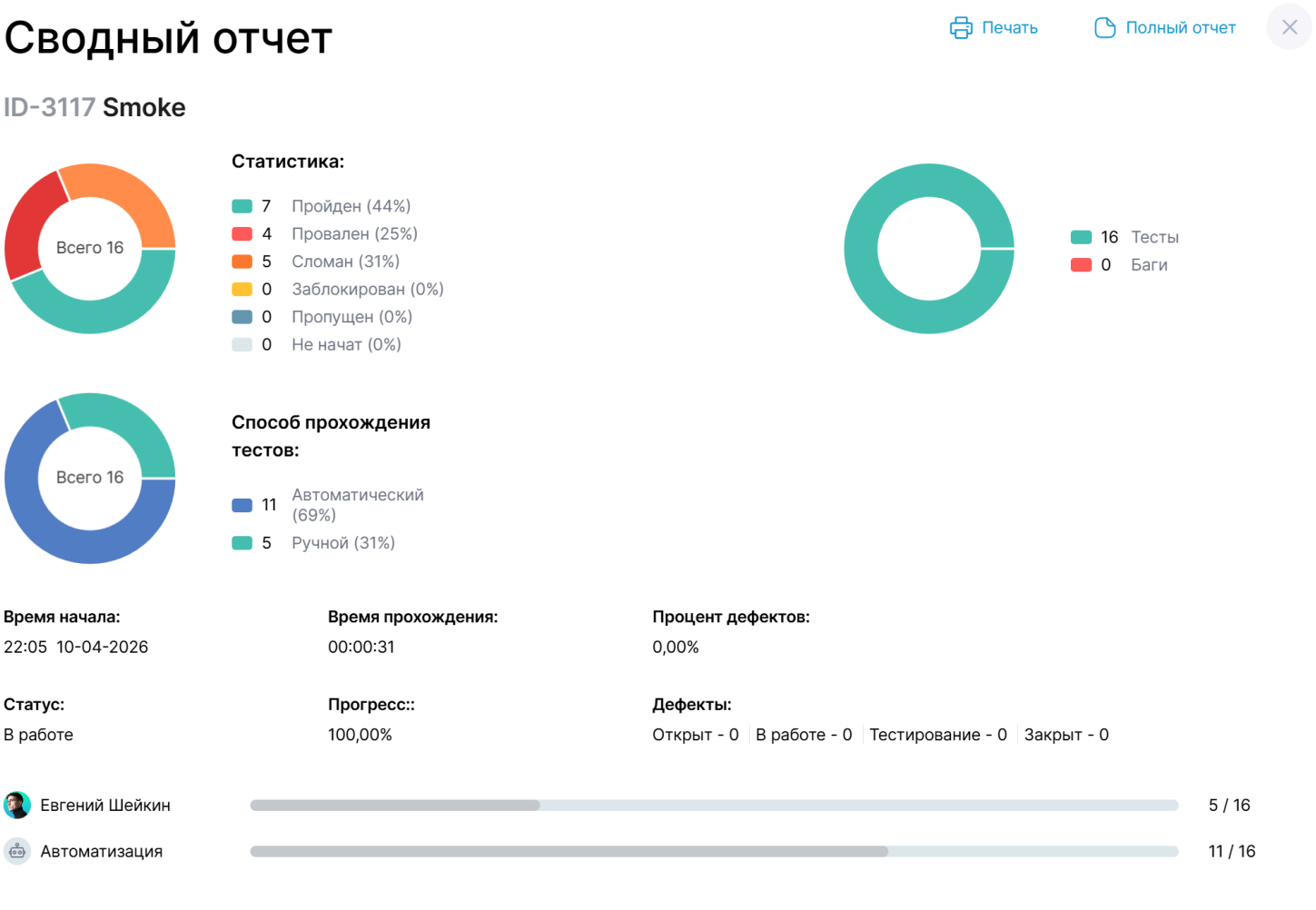
После завершения прохождения прогона его нельзя редактировать или менять статус элементов. Завершенный прогон можно возобновить, повторно нажав на «**Запуск**».

Отчет о тестировании

Для просмотра отчета о тестировании нажмите на кнопку «**Отчет**» в плеере прогона.

В сводном отчете отображается информация:

- время тестирования;
- количество тестов с разбивкой по статусам, в том числе в виде круговой диаграммы;
- процент тестов с обнаруженными дефектами;
- процент ручного и автоматического прохождения тестов;
- разбивка баг-репортов по статусам;
- прогресс прохождения прогона;
- конфигурации прогона;
- статистика прохождения прогона по исполнителям.

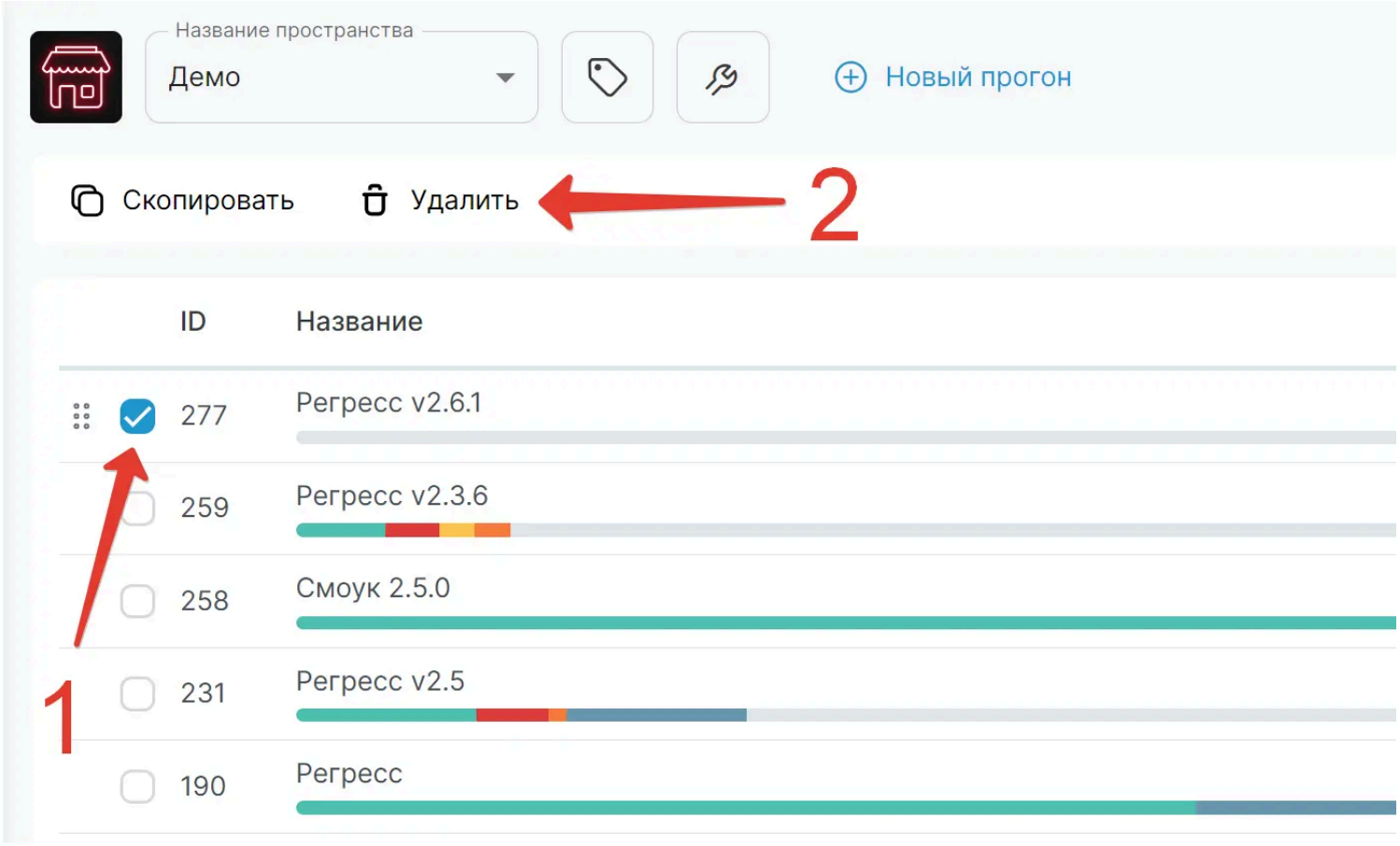


Отчет о тестировании можно распечатать или скачать в формате PDF.

Для детального отчета нажмите кнопку «**Полный отчет**». Помимо данных сводного отчета, в него выводится детальная информация по каждому тест-кейсу и чек-листу. Комментарии в полный отчет не попадают.

Удаление прогона

Для удаления прогона отметьте один или несколько прогонов в таблице (цифра 1) и нажмите на кнопку «**Удалить**» (цифра 2). Откроется диалог подтверждения «Переместить в корзину».



Важно

Прогон не удаляется безвозвратно: он перемещается в **Корзину** и хранится там 30 дней. В течение этого срока прогон можно восстановить вместе со статистикой прохождения, комментариями и баг-репортами. По истечении 30 дней прогон удаляется навсегда.

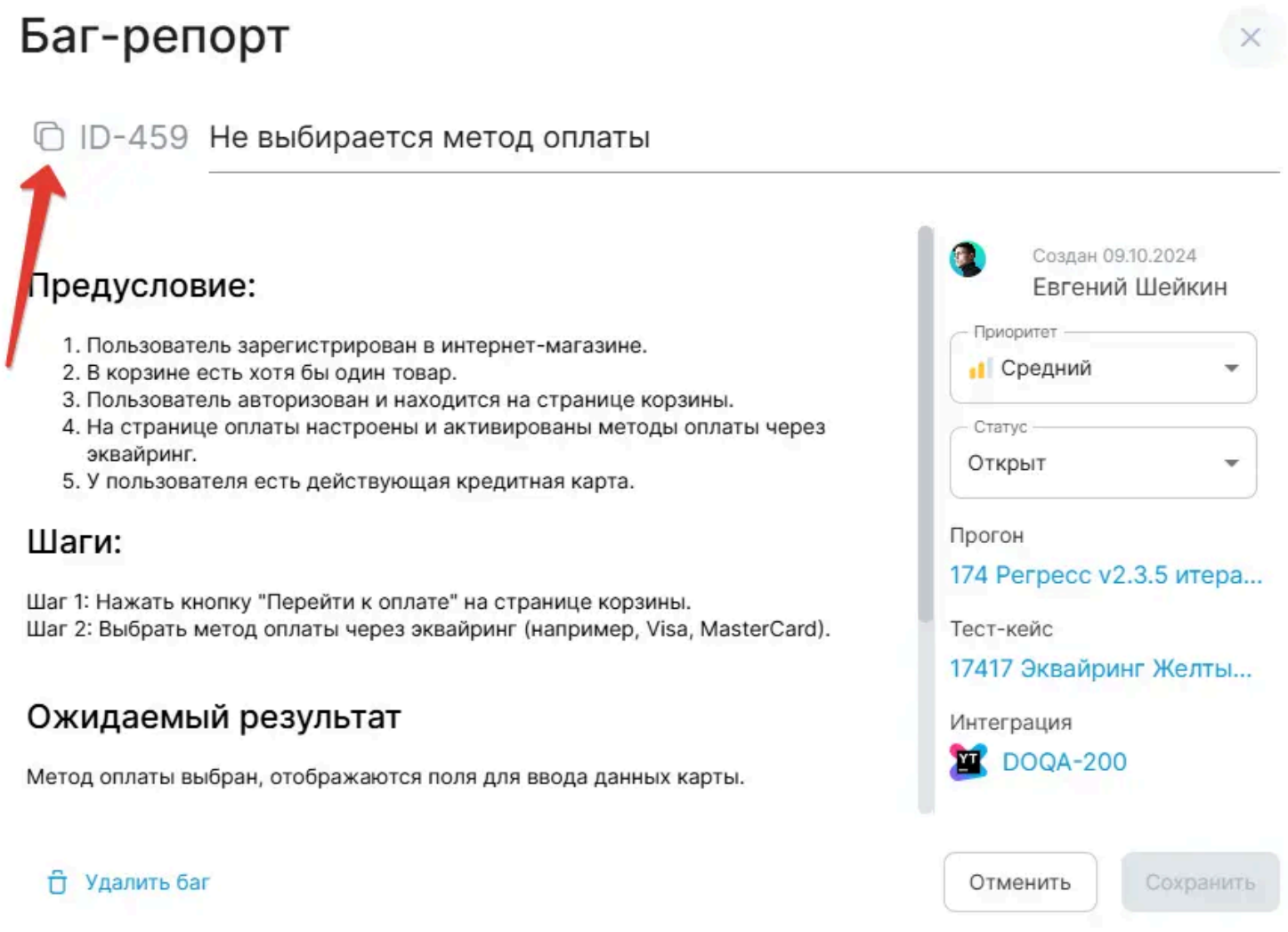
Баг-репорты

Баг-репорт заводят при прохождении теста в **прогоне** — на шаге тест-кейса, на чеке чек-листа или на автотесте. О том, как его завести, читайте в разделе **Создание баг-репортов**. Эта страница — о том, где искать и как читать уже заведенные баг-репорты.

Найти баг-репорты можно в трех местах:

- В разделе «Баги» — все баг-репорты рабочего пространства.
- В плеере прогона на вкладке «Баги» — баг-репорты, созданные при прохождении этого прогона.
- В редакторе тест-кейса или чек-листа на вкладке «Баги» — баг-репорты, найденные при прохождении этого теста.

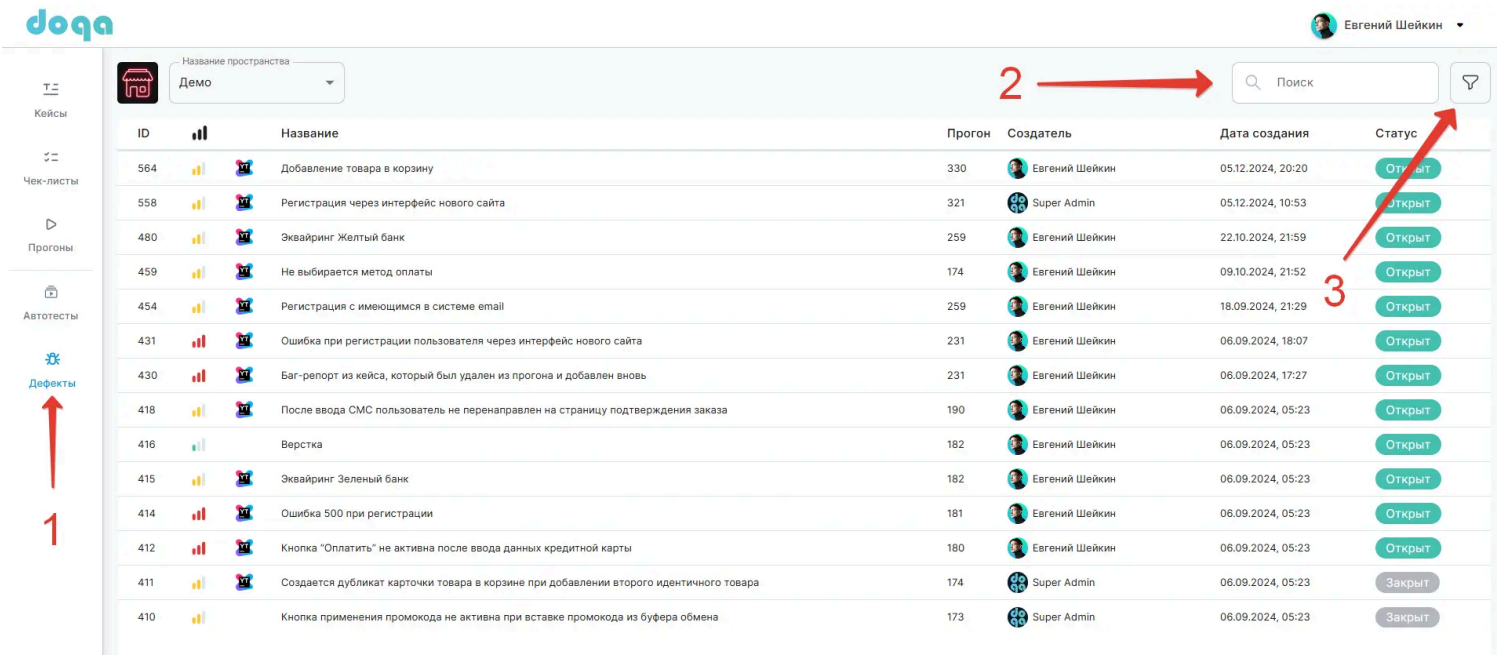
Из списка баг-репорт открывается в модальном окне. Чтобы получить прямую ссылку на него, нажмите на иконку копирования. При переходе по такой ссылке баг-репорт открывается отдельной страницей.



Раздел «Баги»

Чтобы перейти в раздел «Баги», находясь в рабочем пространстве, нажмите «Баги» в левом боковом меню (цифра 1). Поисковое поле находит баг-репорт по точному совпадению ID или частичному совпадению названия (цифра 2). Также доступна фильтрация по приоритету, статусу, создателю баг-репорта и прогону, в котором был обнаружен баг (цифра 3).

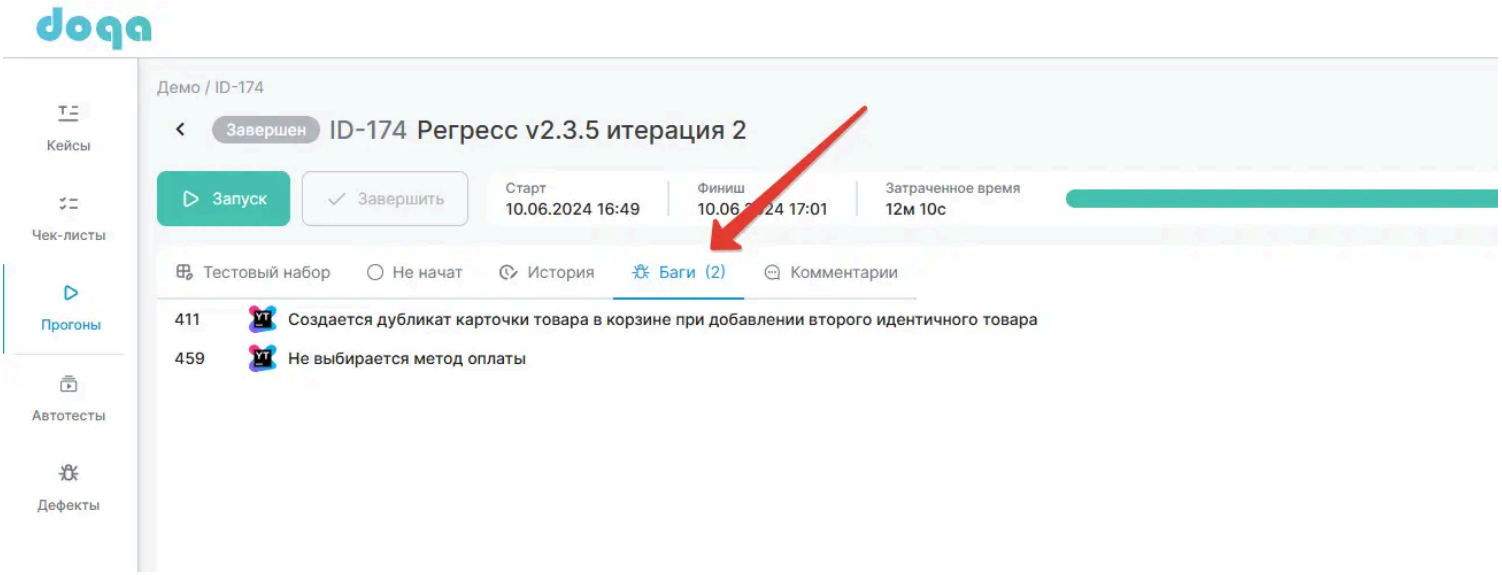
В таблице выводятся колонки: ID, приоритет, иконка трекера, «Название», «Прогон», «Создатель», «Дата создания», «Статус». Если при создании баг-репорта была настроена **интеграция с трекером**, иконка слева от названия указывает на вид трекера.



Для просмотра баг-репорта нажмите на строку в таблице.

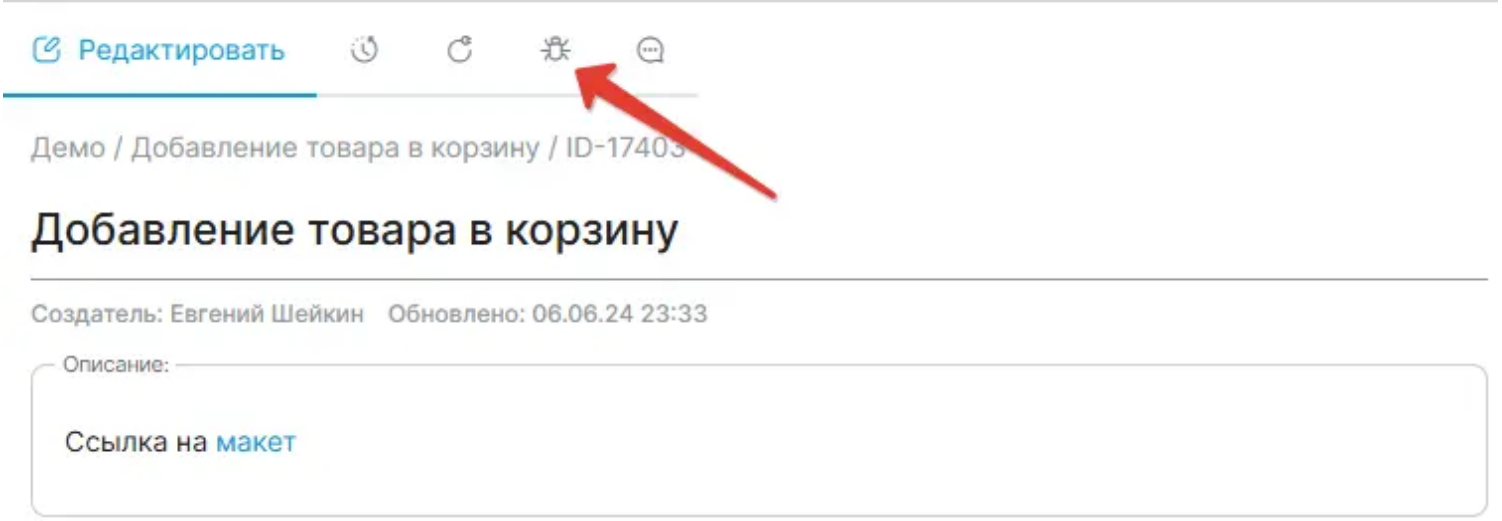
Просмотр баг-репортов в плеере прогона

Для просмотра баг-репортов, созданных при прохождении прогона, нажмите на вкладку «Баги» в плеере прогона.

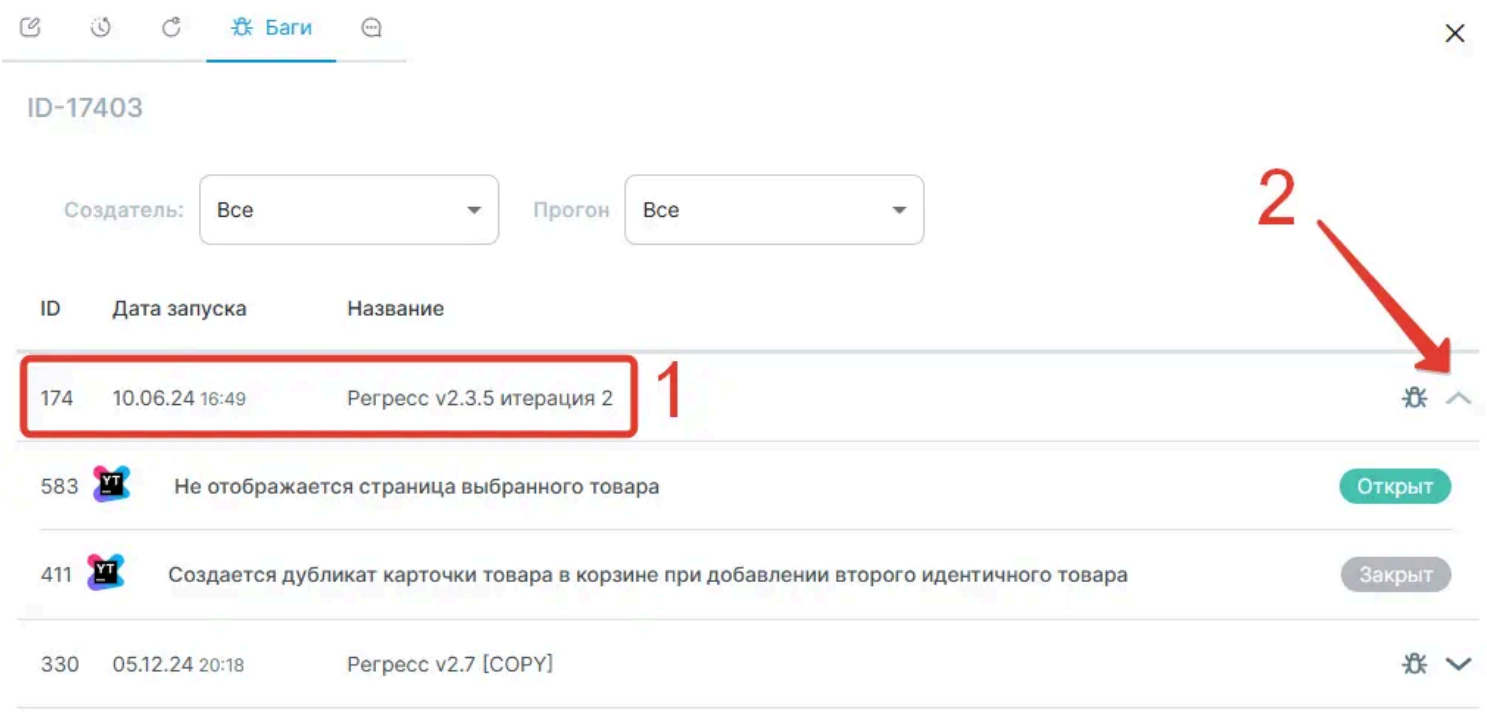


Просмотр баг-репортов в редакторе тест-кейсов и чек-листов

В редакторе тест-кейса или чек-листа нажмите на иконку бага.



Откроется вкладка «Баги» со списком прогонов, в которых были обнаружены баги, с указанием ID, даты запуска и названия прогона (цифра 1). Рядом со стрелкой выводится число багов, найденных в этом прогоне. Чтобы просмотреть баги прогона, нажмите на иконку стрелки (цифра 2).



Над таблицей есть фильтры «Создатель» и «Прогон». Колонки «Дата запуска», «ID» и «Название» сортируются нажатием на заголовок.

По каждому багу указаны:

- ID
- Иконка трекера, с которым установлена интеграция
- Название
- Статус

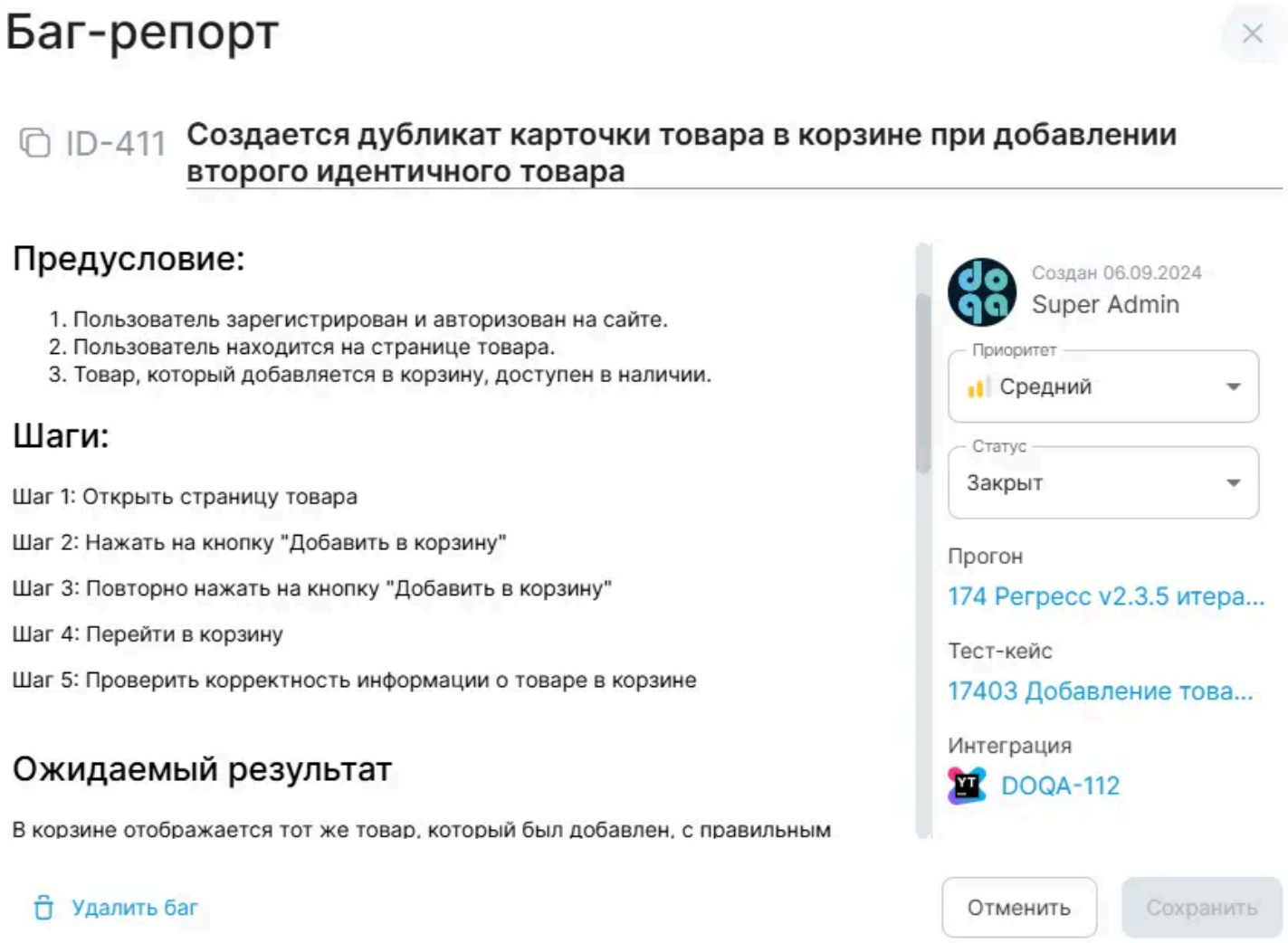
Для чек-листов у каждого бага дополнительно выводится номер чека, на котором он заведен.

Просмотр информации о баге

При нажатии на баг открывается окно с полной информацией:

- ID
- Название (до 255 символов)
- Иконка копирования прямой ссылки на баг-репорт
- Описание с форматированием текста
- Вложения — файлы, прикрепленные к баг-репорту

- Информация о создателе, дата и время создания
- Приоритет
- Статус
- Ссылка на прогон, в котором обнаружен баг
- Ссылка на тест-кейс, чек-лист или **автотест**, при прохождении которого обнаружен баг
- Ссылка на баг в трекере (при наличии интеграции)
- Кнопка «**Сохранить**»
- Кнопка «**Удалить баг**» — для тех, кому доступно удаление



Баг-репорты на автотестах

Баг-репорт заводят на шаге тест-кейса или на чеке, а также на автотесте — кнопкой «**Завести баг**» в карточке автотеста в плеере прогона. У такого бага вместо ссылки на тест-кейс или чек-лист выводится ссылка на автотест.

Если элемент прогона пройден автоматически, баг привязывается к элементу целиком: завести баг на отдельный шаг или чек в нем нельзя. И наоборот, у элемента, пройденного вручную, баг заводится на шаг или чек, а не на элемент целиком.

Статусы и приоритеты

Статус баг-репорта — один из четырех: «Открыт», «В работе», «Тестирование», «Закрыт». Приоритет — «Низкий», «Средний» или «Высокий». Статус и приоритет меняются в окне баг-репорта.

Если для трекера настроена обратная интеграция, статус и приоритет баг-репорта в DoQA обновляются автоматически при изменении связанного баг-репорта в трекере — см. **Интеграции с трекерами**.

Редактирование и удаление баг-репортов

Баг-репорт может быть отредактирован его автором или пользователями с правами на администрирование проектов. После изменений нажмите на кнопку «**Сохранить**».

Кнопка «**Удалить баг**» доступна автору баг-репорта и пользователям с правами на администрирование проекта.

Удаление

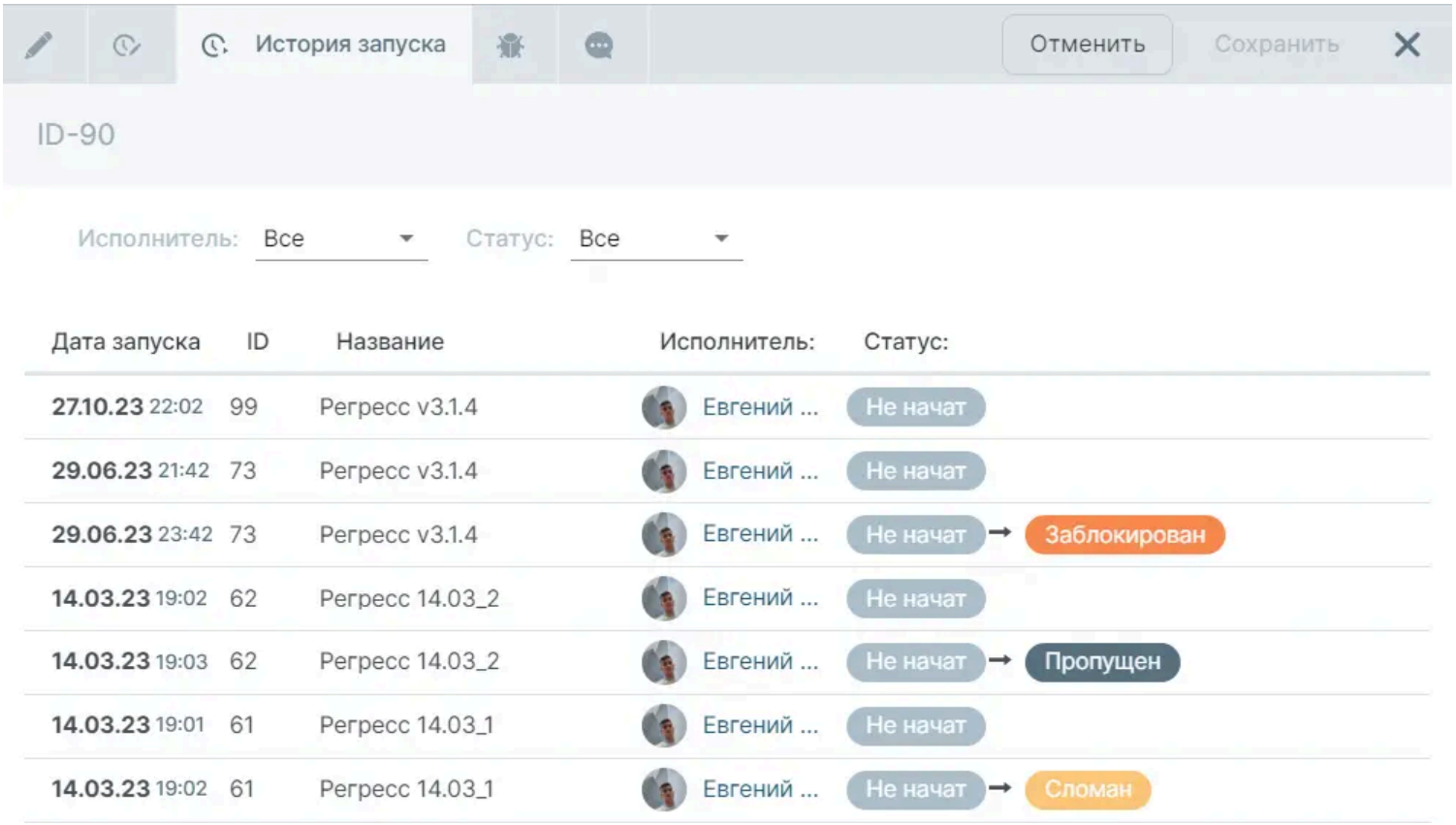
Баг-репорт удаляется в **корзину**, а не безвозвратно. Срок, в течение которого его можно восстановить, по умолчанию — 30 дней; сколько осталось у конкретного баг-репорта, показывает колонка «До удаления» в корзине.

История запуска

История запуска показывает, в каких **прогонах** проходил тест-кейс или чек-лист и с каким результатом.

Как открыть

Откройте редактор нужного тест-кейса или чек-листа и перейдите на вкладку «**История запуска**».



Фильтры и сводка

Над таблицей есть два выпадающих списка — «Исполнитель» и «Статус». По умолчанию в обоих выбрано значение «Все».

Там же выводится сводка по отфильтрованным записям:

- «Среднее время» — среднее время прохождения;

- «Прохождений» — число записей;
- «Успешно» — процент записей со статусом «Пройден».

Показатели пересчитываются под выбранные фильтры. Записи со статусом «Не начат» в сводке не учитываются.

Что в записи

Каждая запись — одно прохождение теста в прогоне. В таблице семь колонок:

- «Дата запуска» — дата и время прохождения.
- «ID» — ID прогона. Ссылка открывает прогон в новой вкладке.
- «Название» — название прогона.
- «Параметр» — набор значений для параметризованных тестов. Подробнее — **Параметры**.
- «Исполнитель» — пользователь, проходивший тест. У **автотестов** указана «Автоматизация».
- «Статус» — итоговый статус теста в этом прогоне.
- «Время» — время прохождения.

Таблица сортируется по любой колонке, кроме «Параметр». По умолчанию записи отсортированы по дате запуска по убыванию.

Историю изменений самого тест-кейса или чек-листа смотрите на соседней вкладке — **История редактирования**.

Автотесты

Автотест появляется в DoQA только тогда, когда в систему приходят результаты его выполнения, — добавить его вручную нельзя. Базовые понятия — на этой странице. Практические руководства — в **карте раздела**.

Что такое автотест

Автотест в DoQA представлен в двух местах:

- **Каталог автотестов** — реестр всех автотестов пространства: раздел «Автотесты» в меню пространства, вкладки «Все автотесты» и «Кластеры ошибок». Здесь у автотеста есть карточка с историей, показателями стабильности и привязками — см. **Каталог автотестов**.
- **Прогоны** — результаты каждого выполнения. Они попадают в обычные **прогоны** рядом с ручными тест-кейсами; разбор результата — **Результаты и разбор падений**.

Запись в каталоге создаётся автоматически при первом результате и дальше накапливает историю: сколько раз тест прошёл и упал, насколько он стабилен, откуда запускался.

Внешний ID

Основной идентификатор автотеста — внешний ID (`externalId`). Он уникален в пределах пространства и задаётся в коде теста:

- в адаптере тестового фреймворка — его разметкой: у каждого адаптера она своя, см. страницу нужного **адаптера**;
- в Allure-отчёте — меткой `doqa_id`;
- в JUnit XML — свойством `<property name="doqa_id" value="..."/>`.

Если явного ID нет, адаптер выводит его сам по строгому приоритету:

1. маркер в имени теста: `[DOQA-123]` или `@DOQA:123` в display name;
2. совместимый ID из Allure-разметки, если она есть в проекте, — **Переезд с Allure**;
3. детерминированный хэш полного имени теста — стабилен, пока тест не переименован и не перенесён в другой класс или модуль.

Если внешний ID не передан вовсе, DoQA опознаёт автотест по полному имени. Это работает, но хрупко: при переименовании теста для DoQA это будет новый автотест без накопленной истории. С внешним ID переименование безопасно: история, стабильность и привязки остаются у той же записи каталога.

Задавайте внешний ID явно

Явный внешний ID — единственный способ сохранить историю автотеста при переименовании теста и переносе его в другой класс или модуль. Для новых проектов задавайте его с первого дня.

Переход на внешний ID

Если автотест раньше отчитывался без внешнего ID, а затем стал присылать его, вторая запись в каталоге не создаётся. DoQA находит существующий автотест по имени и записывает ему присланный внешний ID. Накопленная история переходит к новому ключу. Если у найденного по имени автотеста уже записан другой внешний ID, ключ не перезаписывается: результаты продолжают попадать в существующую запись со старым ключом.

Как результаты попадают в DoQA

Способ	Когда подходит	Подробнее
Адаптер тестового фреймворка	рекомендуемый способ: библиотека в тестовом проекте отправляет результаты в реальном времени, с шагами, вложениями и метками, без файлов отчётов	Адаптеры для тестовых фреймворков
Собственный адаптер	готового адаптера под ваш язык или фреймворк нет — прямое API `/api/autotests/*`	Autotest API
Запуск из DoQA	DoQA сама запускает пайплайн в подключённой CI-системе. Результаты попадают в прогон по ходу или по завершении пайплайна, смотря как пайплайн их отправляет	Запуск автотестов из DoQA
Утилита `doqactl`	нет адаптера под фреймворк или нужно отправлять результаты без правки проекта	Утилита doqactl

Способ	Когда подходит	Подробнее
	тестов	
Загрузка отчёта в интерфейсе	разовая выгрузка готового отчёта JUnit XML или Allure — DoQA создаёт прогон с результатами	Отправка результатов автотестов
`POST /api/autotests/report`	тот же приём отчёта по HTTP, без утилиты	там же

Способы можно совмещать: например, ежедневные запуски идут из DoQA по расписанию через адаптер, а разовые локальные прогоны разработчики отправляют вручную. Результаты в любом случае попадут в один и тот же автотест каталога: идентичность определяет внешний ID, а не способ доставки.

Связь с тест-кейсами

Связь автотеста с ручным **тест-кейсом** показывает, что кейс покрыт автоматизацией. Один автотест может покрывать несколько тест-кейсов; у тест-кейса — не больше одного автотеста.

Способы привязки:

Способ	Откуда задаётся	Поведение
Список кейсов	метка `doqa_cases` в отчёте, разметка адаптера или поле `caseIds` в **Autotest API** — ID нескольких кейсов	аддитивно: прежние связи не снимаются
Allure ID	метка `as_id` в отчёте, её проставляет Allure-разметка — ID одного кейса	связь один-к-одному: прежние связи этого автотеста снимаются
Вручную из каталога	действие «Привязать кейс» в каталоге автотестов: выберите один автотест и отметьте кейсы в диалоге	аддитивно

Кейс во всех способах указывается числовым ID — тем, что показан в колонке «ID» таблицы кейсов. Отдельного «кода» у тест-кейса нет. Подробнее о разметке Allure — **Переезд с Allure**.

Тест-кейсы при привязке не создаются: если кейса с присланным ID в пространстве нет, привязка молча пропускается. Автотест без привязанного кейса помечается в каталоге как «Без связи с кейсами». Для таких есть отдельный чип и фильтр.

Статус автоматизации

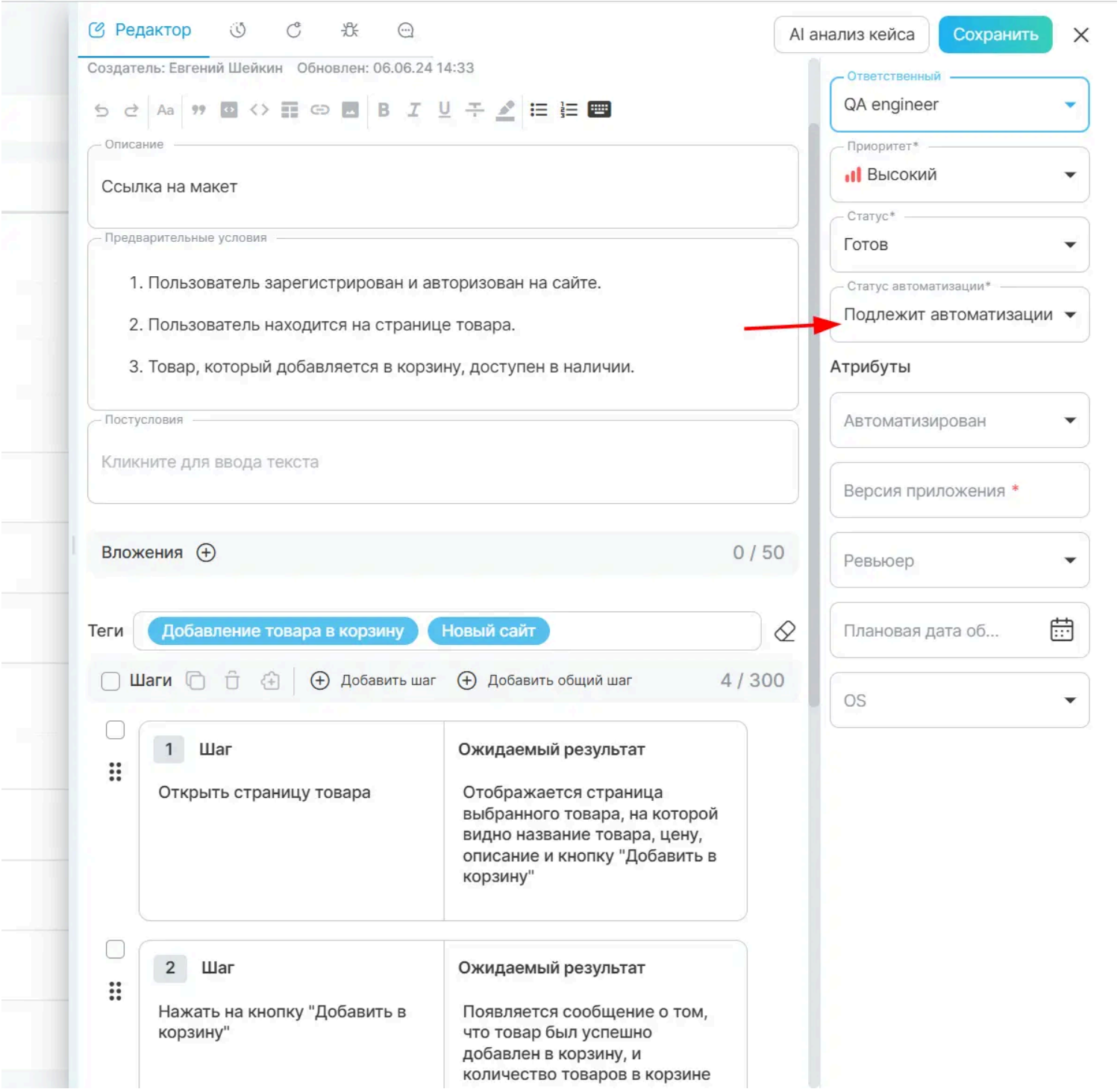
У тест-кейса три значения «Статуса автоматизации»:

- «Ручной»;
- «Подлежит автоматизации» — пометка о том, что тест-кейс планируется автоматизировать;
- «Автоматизирован» — тест-кейс связан с автотестом.

«Автоматизирован» проставляет система при появлении связи, вручную выбрать это значение нельзя. Чтобы разорвать связь, выберите в редакторе тест-кейса «Ручной» или «Подлежит автоматизации».

Что рвёт связь

Смена «Статуса автоматизации» на любое другое значение рвёт связь тест-кейса с автотестом. Копия тест-кейса связь не наследует: у копии связи с автотестом нет.



Связь с требованиями

Автотест можно привязать к **требованию** наравне с тест-кейсом и чек-листом: такая связь равноправно участвует в покрытии требования. Привязка делается из панели «Автотесты» в раскрытой строке требования или из меню строки в каталоге автотестов. При удалении автотеста его связи с требованиями снимаются автоматически.

Страницы раздела

- **Отправка результатов автотестов** — загрузка отчёта в интерфейсе, форматы JUnit XML и Allure, утилита doqactl, `POST /api/autotests/report`.
- **Адаптеры для тестовых фреймворков** — что даёт адаптер по сравнению с отчётами, какие платформы поддержаны и отдельные руководства по каждому адаптеру: установка, конфигурация, разметка.
- **Переезд с Allure** — что подхватывается из Allure-разметки без правок кода и как переехать на разметку DoQA.
- **Плагины для IDE** — статусы автотестов прямо в редакторе IntelliJ IDEA, PyCharm и VS Code, привязка кейсов, закрепление внешнего ID, история прогонов и карантин без перехода в браузер.
- **Autotest API** — прямое API для собственных адаптеров и интеграций.
- **Каталог автотестов** — вкладка «Все автотесты»: сводка над таблицей, фильтры, карточка автотеста, массовые действия.
- **Подключение CI-систем** — CI-подключения на уровне пространства, провайдеры, привязка проектов.
- **Запуск автотестов из DoQA** — все способы запуска, выбор тестов пайплайном, жизненный цикл запуска.
- **Автоматика запусков** — расписания, автоматический перезапуск упавших, критерии приёмки прогона.
- **Результаты и разбор падений** — карточка результата, кластеры ошибок, сравнение прогонов, аналитика.

Смотрите также

- **Первый прогон автотестов** — сценарий: от пустого пространства до первого прогона с результатами
- **Разбор упавших автотестов** — сценарий ежедневного разбора падений
- **Публичный API** — остальные методы API и доступ к справочнику
- **Интеграции с CI/CD** — подключения CI на уровне тенанта (для администратора)

Скачать CLI автотестов

Один бинарь собирается под двумя именами. Wire-имена зависят от продукта:

Продукт	Бинарь	Env	Config
DoQA	`doqactl`	`DOQA_URL`, `DOQA_TOKEN`, `DOQA_SPACE_ID`, …	`~/.doqactl.yaml`
QAscade	`tmsctl`	`TMS_URL`, `TMS_TOKEN`, `TMS_SPACE_ID`, …	`~/.tmsctl.yaml`

CLI **читает оба** префикса env (`DOQA_*` и `TMS_*`), чтобы существующие пайплайны не ломались.

Сборка публикуется GitLab Package Registry и зеркалируется на сайт продукта:

Продукт	Linux	Windows
DoQA	doqactl ↗	doqactl.exe ↗
QAscade	tmsctl ↗	tmsctl.exe ↗

Пока зеркало `qascade.app/downloads` не развёрнуто, используйте артефакт из registry CI: `packages/generic/doqactl/<tag>/doqactl-linux-amd64` или `packages/generic/tmsctl/<tag>/tmsctl-linux-amd64`.

```bash
# DoQA
curl -fsSL https://doqa.app/downloads/doqactl -o doqactl && chmod +x doqactl
./doqactl version

# QAscade
curl -fsSL https://qascade.app/downloads/tmsctl -o tmsctl && chmod +x tmsctl
./tmsctl version
```

Примеры команд — в руководстве по автотестам.

Отправка результатов автотестов

Автотесты выполняются в вашей инфраструктуре (в CI/CD или локально), а в DoQA отправляются их результаты. Рекомендуемый способ — **адаптер тестового фреймворка**: он отправляет результаты сам, без файлов отчётов. Эта страница — про остальные способы, на случай, когда адаптера под ваш фреймворк нет: файл отчёта через интерфейс, утилита **doqactl** и report-API. Из результатов DoQA создаёт **прогон** или дополняет существующий.

Выбор способа

Способ	Когда подходит
Адаптер тестового фреймворка	рекомендуется: результаты уходят прямо из тестового процесса, с шагами, вложениями и привязкой к кейсам, без файлов отчётов
Autotest API	собственный адаптер, когда готового под ваш фреймворк нет
doqactl watch	фреймворк без адаптера: утилита запускает тесты в пайплайне и отправляет результаты по ходу прогона
doqactl upload	отчёт уже собран — отправить одной командой из CI или с локальной машины
report-API	то же, что `upload`, но прямым HTTP-запросом, без утилиты
Загрузка отчёта в интерфейсе	разовая загрузка готового файла отчёта, например после локального прогона

Поддерживаемые форматы отчётов

- **JUnit XML** — файл `.xml` или `.zip`-архив с XML-файлами. Внутри архива обрабатываются только файлы `*.xml`; архив без единого валидного XML отклоняется.
- **Allure** — `.zip`-архив с сырыми данными: файлы `*-result.json`, а рядом с ними `*-container.json` добавляют фикстуры (setup и teardown). Имя каталога внутри архива значения не имеет: DoQA считает архив Allure-отчётом, если в нём есть хотя бы один файл

с именем, которое оканчивается на `result.json`. Поэтому подходят и каталог `results` (так пишут адаптеры DoQA), и `allure-results` (стандартное имя Allure).

Размер одного файла при любом способе загрузки отчёта — не больше 50 МБ.

> **Сырые данные, а не готовый отчёт**
>
> DoQA обрабатывает только сырые данные Allure — папку с `*-result.json` и другими файлами, которую формирует тестовый фреймворк. HTML-отчёт, собранный командой `allure generate`, не принимается. Достаточно упаковать папку результатов в `.zip`-архив, дополнительные действия с данными не требуются.

Форматы не равнозначны. Allure богаче: шаги и фикстуры со статусами и длительностью, параметры, вложения (скриншоты, видео, логи), реальное время старта и окончания. JUnit XML несёт только базовый результат: статус, длительность, ошибку и вывод теста.

Файл вложения больше 100 МБ в хранилище не попадает: результат теста обрабатывается как обычно, а в его карточке появляется предупреждение «Вложения не приняты: N». Лимит настраивает администратор сервера. Это отдельное ограничение от 100 МБ на сумму вложений одного результата в `doqactl watch`. Оба верны одновременно, но это разные лимиты (второй описан в **doqactl**).

Метки DoQA в отчёте

Через метки в отчёте автотест сообщает о себе дополнительные данные. В Allure это метки (labels), в JUnit XML — элементы `<property>` внутри `<properties>`; имена регистронезависимы. Булевы значения принимаются в виде `1` / `true` / `yes` / `on` и `0` / `false` / `no` / `off`.

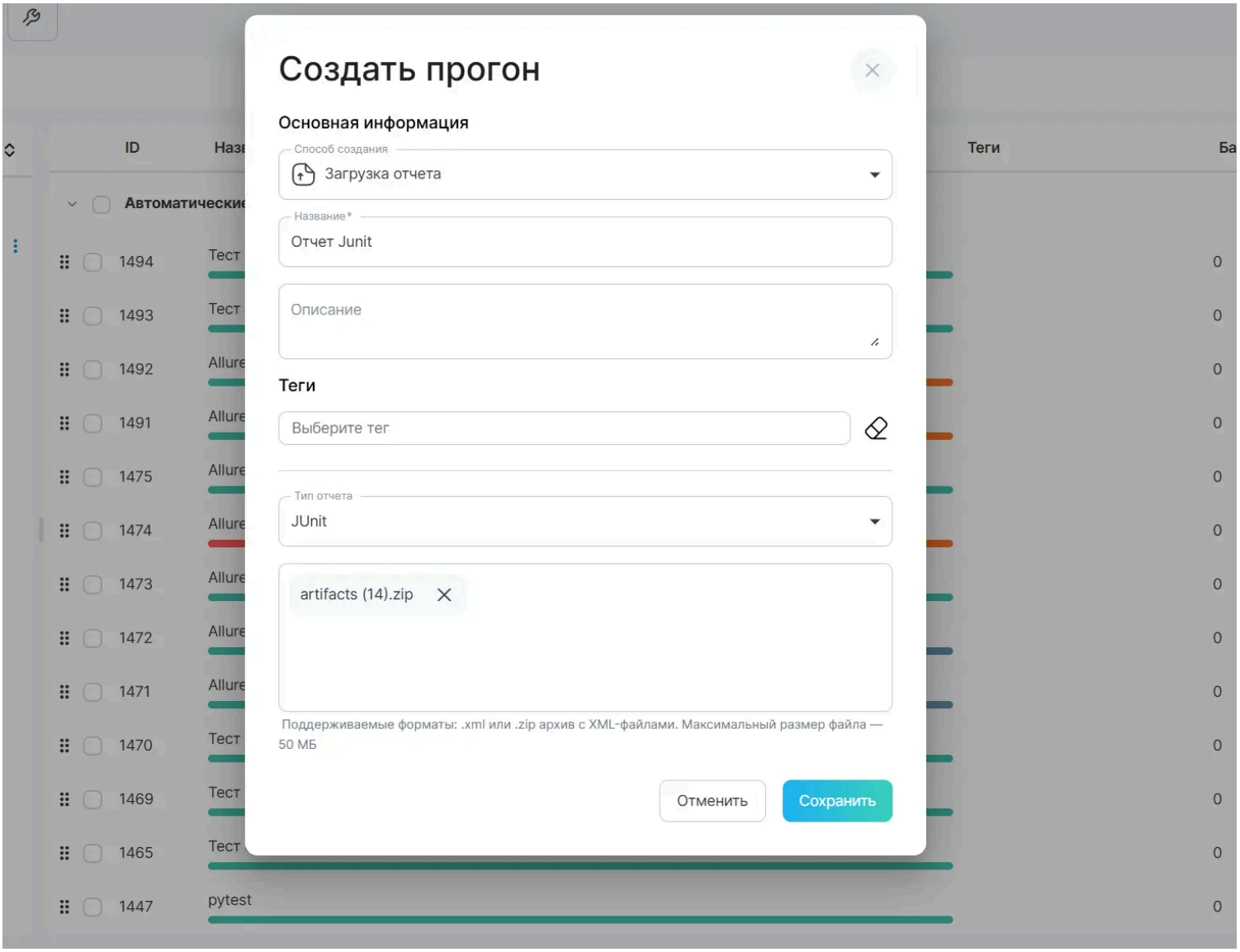
Метка / property	Что задаёт
`doqa_external_id`, `doqa_id`	внешний ID автотеста; если переданы обе, побеждает `doqa_external_id`
`as_id`, `allure_id`	привязка к одному тест-кейсу — способ совместимости, см. **Переезд с Allure**
`doqa_cases`, `doqa_work_items`	привязка к нескольким тест-кейсам — список ID через запятую

Метка / property	Что задаёт
`doqa_title`	заголовок автотеста в каталоге
`doqa_namespace`, `doqa_package`	место автотеста в дереве каталога
`doqa_class`, `doqa_classname`	класс автотеста
`doqa_runner_name`, `doqa_runner_method`	имя и метод автотеста в раннере; по ним DoQA определяет формат отчёта источника
`doqa_create_manual_case`	завести связанный ручной тест-кейс для этого автотеста (только в Allure — в JUnit XML такого поля нет)
`tag`	тег автотеста; несколько меток `tag` собираются в список, а не перетирают друг друга
`package`, `packageName`	запасной путь: место в дереве каталога, если нет `doqa_namespace` / `doqa_package`
`testClass`, `className`, `suite`	запасной путь: класс автотеста, если нет `doqa_class` / `doqa_classname`
`testMethod`, `methodName`	запасной путь: метод в раннере, если нет `doqa_runner_method`

Остальные пары имя-значение попадают в свойства результата; `doqa_create_manual_case` и `tag` туда не дублируются. Что означают внешний ID и привязка к тест-кейсам — **Автотесты**.

Загрузка отчёта через интерфейс

1. На вкладке «Прогоны» нажмите «**Новый прогон**» — откроется диалог «Создать прогон».
2. В поле «Способ создания» выберите «Загрузка отчета». По умолчанию выбран «Стандартный», третий вариант — «Запуск автотестов».
3. Укажите «Тип отчета»: «JUnit» (значение по умолчанию) или «Allure».
4. Приложите файл, ровно один, не больше 50 МБ. Второй файл диалог не примет.
5. Задайте название прогона — оно обязательно. При необходимости добавьте описание, теги и конфигурации.
6. Нажмите «**Сохранить**».



DoQA создаёт прогон с результатами из отчёта. Подробнее о полях диалога — **Создание прогона на основе отчёта о результатах автотестов**.

Отправка отчёта из CI через doqactl

Утилита командной строки **doqactl** (прежнее имя — `doqa-cli`, сохранено как алиас файла) отправляет результаты без открытия браузера. Её берут, когда под фреймворк нет адаптера или проект тестов править нельзя: утилита работает с любым фреймворком, который умеет писать JUnit XML или Allure. Результат тот же, что при загрузке через интерфейс: в системе появляется прогон с результатами. Полный список команд, флагов и переменных, а также разбор ошибок — на странице **doqactl**. Здесь — только то, что нужно для отправки отчёта.

Скачивание doqactl

```bash
curl -fsSL https://doqa.app/downloads/doqactl -o doqactl
chmod +x doqactl
```

Прежняя ссылка `https://doqa.app/downloads/doqa-cli` тоже работает: под этим именем лежит копия того же бинаря, чтобы не менять уже настроенные пайплайны. Установка под другие ОС — **doqactl → Установка**.

Переменные окружения

Создайте переменные в настройках вашего CI/CD (например, Settings > CI/CD > Variables в GitLab или Actions Secrets в GitHub). Для токена используйте режим маскирования (Masked).

- `DOQA_URL` — адрес вашего экземпляра DoQA, например `https://company.doqa.app`.
- `DOQA_TOKEN` — API-токен DoQA, см. **Создание API-токена**.
- `DOQA_SPACE_ID` — ID пространства, см. **Как узнать ID пространства**.

> **Прежние имена продолжают работать**
>
> Раньше адрес задавался переменной `DOQA_ENDPOINT`, а пространство — `DOQA_SPACE`. Утилита читает и их. У каждого параметра есть и более приоритетный синоним (`DOQA_API_BASE` для адреса, `DOQA_CI_TOKEN` для токена). Полный порядок приоритета (флаг > переменная > файл `~/.doqactl.yaml`) — в **doqactl → Общая конфигурация**.

Для GitLab эти три переменные DoQA заводит в проекте сама, кнопкой «**Настроить CI/CD Variables**» в карточке источника, см. **Подключение CI/CD**.

Переменные окружения читают все команды `doqactl`, кроме устаревшей команды `report`: ей адрес, токен и файл передаются позиционными аргументами (пространство может браться и из `--space` / `DOQA_SPACE_ID`/конфига).

Чтобы связать запуск в CI с пайплайном в DoQA, передайте в конфигурации пайплайна:

Переменная	Значение (для GitLab CI)	Описание
DOQA_PIPELINE_ID	$CI_PIPELINE_ID	Уникальный ID текущего запуска. Не обязательна, но без неё результаты будут отправлены в

Переменная	Значение (для GitLab CI)	Описание
		отдельный прогон: по ней DoQA узнаёт прогон, который сама запустила этим пайплайном
CI_PROJECT_ID	$CI_PROJECT_ID	ID проекта в вашей CI-системе
CI_BRANCH	$CI_COMMIT_REF_NAME	Название ветки, в которой запущен тест. Заводить не обязательно: утилита сама берёт ветку из стандартных переменных CI (`CI_COMMIT_REF_NAME` в GitLab, `GITHUB_REF_NAME` в GitHub Actions, `GIT_BRANCH` / `BRANCH_NAME` в Jenkins). Значение `DOQA_BRANCH` старше всех остальных

Для CI-систем без встроенной переменной пайплайна (Jenkins, TeamCity и другие) — полный список того, как `doqactl` определяет пайплайн, в **doqactl → Как определяется CI-система и пайплайн**.

Если пайплайн запускает DoQA

`DOQA_SPACE_ID` заводить вручную нужно только для сценария, когда пайплайн стартует сам, по коммиту или расписанию. Когда пайплайн запускает DoQA, она передаёт `DOQA_SPACE_ID` в переменные пайплайна сама, и значение из DoQA перекрывает то, что задано в настройках CI.

Команда watch

`watch` запускает вашу тестовую команду, а по мере появления результатов отправляет их в DoQA: каждый результат уходит отдельным запросом, и прогон наполняется, не дожидаясь конца пайплайна. Алиас команды — `stream`. По `DOQA_PIPELINE_ID` утилита получает у DoQA прогон для этого пайплайна. Если пайплайн запущен из DoQA, это существующий прогон — тогда `watch` сам запрашивает и применяет тест-план, см. **Запуск автотестов из DoQA**. Если пайплайн стартовал сам, `watch` заводит новый прогон.

```bash
./doqactl watch -- [команда запуска ваших тестов]
```

Всё, что идёт после двойного тире (`--`), воспринимается утилитой как исходная команда для запуска тестов.

Пример этапа тестирования в `.gitlab-ci.yml`. В примере это Maven со стандартным выводом результатов в директорию `allure-results`; для другого фреймворка меняются только команда запуска и каталоги результатов:

```yaml
test_ui:
  stage: test
  before_script:
    # 1. Скачиваем актуальную версию утилиты
    - curl -fsSL https://doqa.app/downloads/doqactl -o doqactl
    # 2. Даем права на выполнение
    - chmod +x doqactl
  script:
    # 3. Запускаем тесты через watch
    - ./doqactl watch -- mvn -B -Dallure.results.directory=allure-results test
  artifacts:
    when: always
    paths:
      - allure-results
      - target/surefire-reports/junitreports/
    reports:
      junit:
        - target/surefire-reports/junitreports/TEST-*.xml
    expire_in: 1 week
```

Что учесть:

- двойное тире (`--`) — обязательный разделитель: он указывает утилите, что настройки самой `doqactl` закончились и дальше начинается ваша команда;
- если тесты упадут, `watch` передаст код выхода обратно в CI/CD, и пайплайн пометится как «failed». Если тестовая команда прошла, а результаты не доставлены — код выхода `2` и список причин в логе;
- перед запуском `watch` чистит каталоги отчётов, чтобы не отправить результаты прошлой сборки; отключается флагом `--no-clean`;
- `watch` не отменяет сохранение артефактов в самом CI/CD (секция `artifacts`) — логи и тяжёлые скриншоты остаются в вашей инфраструктуре;

- по умолчанию результаты ищутся в каталогах `allure-results`, `target/allure-results` (Allure) и `target/surefire-reports`, `surefire-reports` (JUnit XML); другой корень поиска задаётся флагом `--results-dir`, точный каталог — флагом `--result`;
- если фреймворк формирует отчёты сразу в двух форматах, JUnit XML и Allure, `doqactl` использует Allure; принудительно выбрать формат можно флагом `--report-type allure` или `--report-type junit`.

Полный список флагов и разбор крайних случаев (недоставленные результаты, отказ DoQA, вложения, порядок с тест-планом) — **doqactl → Команда watch**.

Команда upload

Отправить уже готовый отчёт одной командой, когда отслеживание в реальном времени не нужно или инфраструктура не позволяет применять `watch` (например, после локального прогона). Команда принимает файл, ZIP-архив или каталог (архивирует на лету — отдельно паковать не нужно), сама определяет формат отчёта и не требует полного URL, только базовый адрес DoQA.

```bash
./doqactl upload [флаги] <файл-или-каталог>
```

- аргумент — ровно один: файл (`*.xml` JUnit или `*-result.json` Allure), ZIP-архив или каталог результатов;
- `--type` — формат отчёта: `auto` (по умолчанию, определяется по имени файла), `allure` или `junit`;
- `--title` — название прогона; без него прогон получит служебное название;
- лимит запроса — **50 МБ**.

Пример для Allure:

```bash
./doqactl upload --pipeline-id "$CI_PIPELINE_ID" allure-results/
```

Без `--pipeline-id` отчёт уйдёт в отдельный прогон

Загрузка отчёта не принимает `DOQA_TEST_RUN_ID`: отчёт адресуется только идентификатором пайплайна. Без `--pipeline-id` (или `DOQA_PIPELINE_ID`) результаты будут отправлены в новый прогон, а прогон, запущенный из DoQA, останется пустым. Команда предупреждает об этом в логе.

Пример шага в `.gitlab-ci.yml`:

```yaml
test_ui:
  stage: test
  script:
    # 1. Запуск тестов (генерация allure-results)
    - mvn test -Dallure.results.directory=allure-results
  after_script:
    # 2. Скачивание doqactl
    - curl -fsSL https://doqa.app/downloads/doqactl -o doqactl
    - chmod +x doqactl

    # 3. Отправка отчёта в DoQA — upload архивирует каталог сам
    - ./doqactl upload --pipeline-id "$CI_PIPELINE_ID" allure-results
  artifacts:
    when: always
    paths:
      - allure-results/
    expire_in: 1 week
```

Устаревшая команда `report` (позиционные аргументы, отправка на явный URL) тоже работает: старые пайплайны менять не обязательно. Она всегда создаёт новый прогон и не умеет адресовать результаты в прогон, запущенный из DoQA.

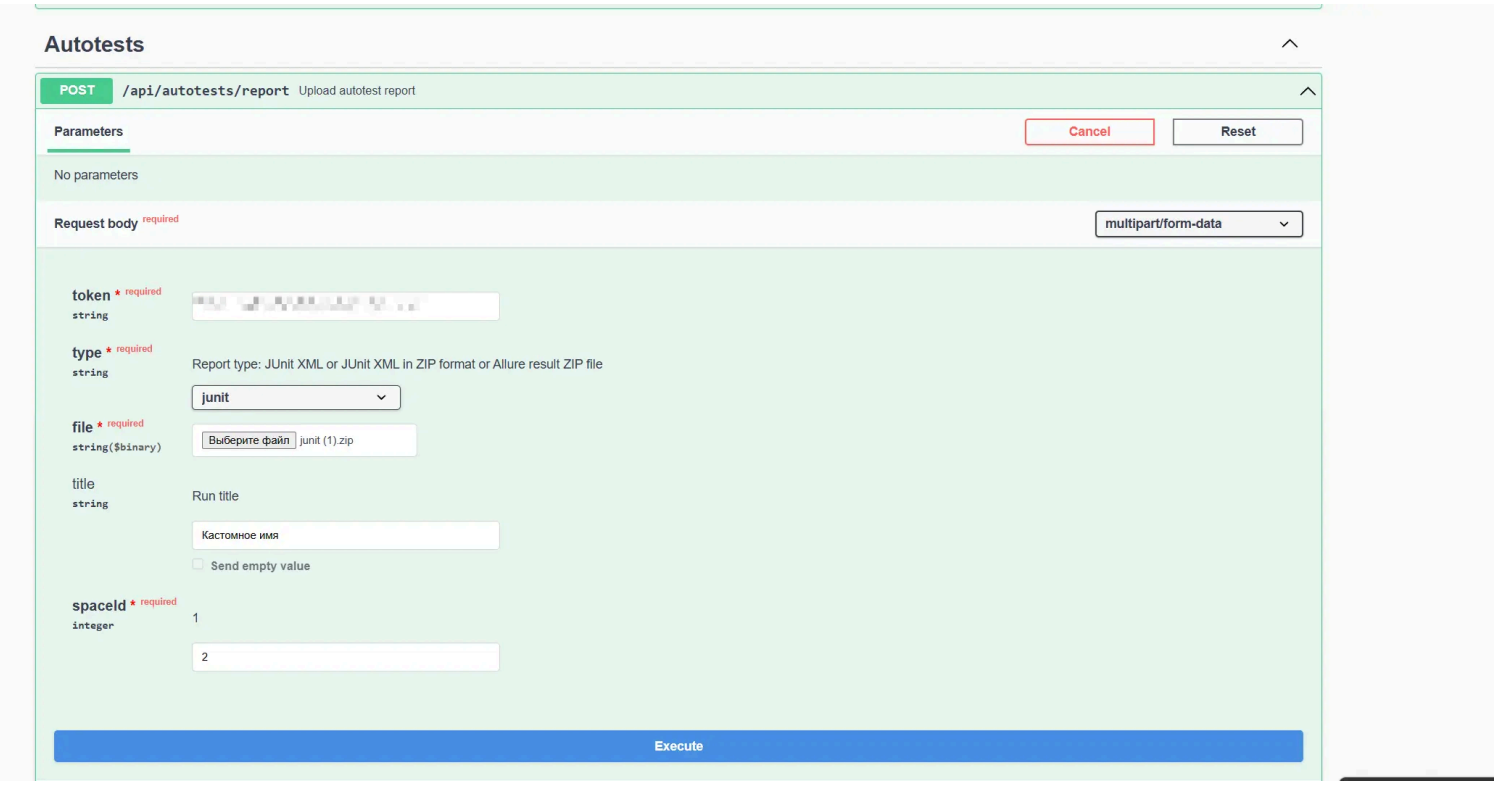
```bash
./doqactl report "$DOQA_URL/api/autotests/report" "$DOQA_SPACE_ID" "$DOQA_TOKEN" "allu
```

Полный список флагов `upload`, восстановление контекста пайплайна из `doqa-reporting.properties` и устаревший синтаксис `report` — doqactl → Команда upload и doqactl → Устаревшая команда report.

Загрузка результатов через API

Воспользуйтесь методом `POST /api/autotests/report`. Логин и пароль для него не нужны: метод авторизуется API-токеном проекта.

1. Укажите API-токен проекта в поле «token» (цифра 1).
2. Выберите формат отчёта — JUnit XML или Allure (цифра 2).
3. Выберите файл (цифра 3).
4. При необходимости введите имя прогона (цифра 4).
5. Укажите ID пространства DoQA (цифра 5).
6. Подтвердите отправку запроса.

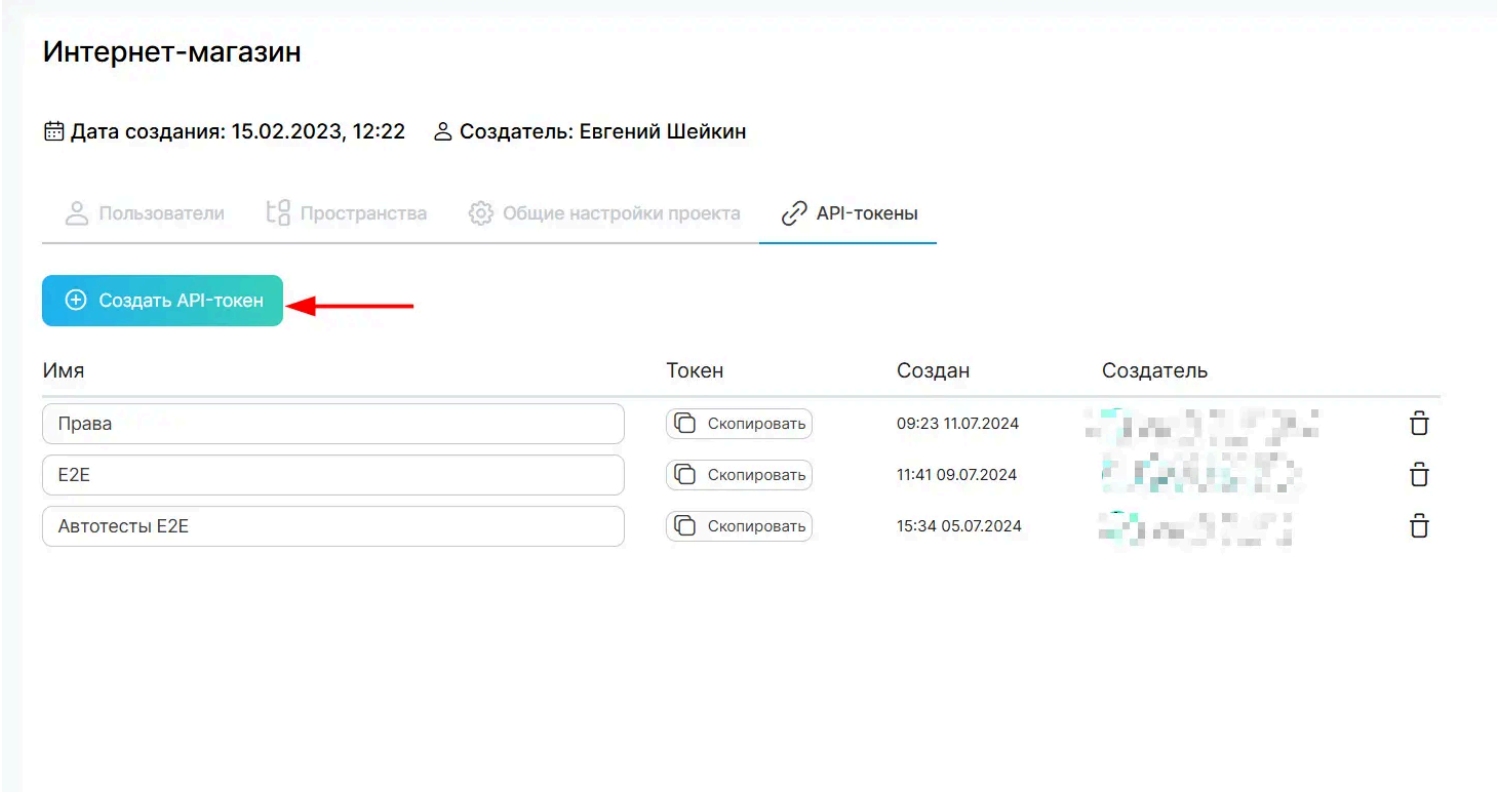


DoQA загружает отчёт и показывает его новым прогоном. Если дополнительно передать поле `pipelineId` с ID пайплайна, запущенного из DoQA, результаты попадут в прогон этого запуска, а не в новый. Форма принимает и остальные поля: `branch`, `system`, `sourceKey`, `correlationId`, `externalId`, `workItemIds`, `schemaVersion`. Если у пространства несколько CI-подключений, `sourceKey` обязателен: без него DoQA не знает, к какому подключению относится отчёт, и отклоняет его. Тексты ошибок — **doqactl → Частые ошибки**. Об остальных методах и о доступе к справочнику API — **Публичный API**.

Не путайте этот метод с `POST /api/autotests/results`: тот принимает не файл отчёта, а отдельные результаты от адаптеров — см. **Autotest API**.

Создание API-токена

Перейдите в настройки администратора, раздел «Проекты». Выберите проект из списка и откройте вкладку «API-токены». Создайте токен: им можно загружать отчёты об автотестах в любое пространство этого проекта. Нужны права администратора.



Результаты автотестов принимаются только по токену проекта, которому принадлежит пространство. Личный токен доступа из профиля пользователя эти методы не принимают, токен другого проекта отклоняется — тексты ошибок в **doqactl → Частые ошибки**.

Как узнать ID пространства

Откройте нужное пространство в браузере. В адресе страницы `.../detail/1/2/cases` второе число (2) — это ID пространства.

Смотрите также

- **Адаптеры для тестовых фреймворков** — отправка результатов прямо из тестового процесса, без файлов отчётов
- **Автотесты** — внешний ID, пути доставки результатов, связь с тест-кейсами

- **Утилита doqactl** — полный список команд, флагов, переменных окружения и текстов ошибок
- **Autotest API** — прямое API для собственного адаптера
- **Прогоны** — куда попадают результаты автотестов

Адаптеры для тестовых фреймворков

Адаптер — библиотека, которую вы подключаете в тестовый проект. Во время прогона она отправляет результаты в DoQA через **Autotest API** прямо из тестового процесса, в реальном времени по ходу выполнения или одной порцией в конце. Результаты приходят структурными: с шагами, фикстурами, вложениями, метками и привязкой к **тест-кейсам**, без файлов отчётов и ручной выгрузки.

Адаптер или отправка отчётов

Адаптер — рекомендуемый способ: глубокая интеграция с наименьшими усилиями на поддержку. Если адаптера под ваш фреймворк нет, DoQA всё равно примет результаты, файлами отчётов JUnit XML или Allure: через интерфейс, утилитой **doqactl** или запросом к report-API. Эти способы описаны на странице **Отправка результатов автотестов**. Чем это отличается от адаптера:

Возможность	Файлы отчётов (интерфейс, doqactl, report-API)	Адаптер
Фреймворки	любой, умеющий писать JUnit XML или Allure	только поддерживаемые платформы (см. ниже)
Шаги и фикстуры	только из Allure-результатов; JUnit XML шагов не содержит	да: декларативные и программные шаги, фикстуры вплоть до `@BeforeAll` / `@AfterAll`
Вложения	из Allure-архива	да: файлы и контент из памяти, прямо из кода теста
Внешний ID без правки кода	нет: ID задаётся меткой или свойством в артефакте отчёта	да: выводится автоматически, вплоть до хэша сигнатуры теста
Привязка к тест-кейсам	метки и свойства в артефакте отчёта	аннотации и вызовы в коде теста
Результаты в реальном времени	только через `doqactl watch`	да: отправка в реальном времени по ходу прогона

Возможность	Файлы отчётов (интерфейс, doqactl, report-API)	Адаптер
Запуск выбранных автотестов из DoQA	частично: через **нативный фильтр** или тест-план, фильтрацию выполнения настраивает пайплайн (`doqactl watch` применяет тест-план сам)	да: селективный режим из коробки — исполняются только тесты из плана прогона

Оба пути сосуществуют. Отчёты — универсальный способ, который работает без правок тестового проекта; адаптер добавляет структуру результата, стабильную идентичность автотеста и селективные запуски.

Поддерживаемые платформы

Платформа	Фреймворк	Библиотека
JVM (Java, Kotlin)	JUnit 5 (Jupiter)	`app.doqa:doqa-junit5`, Maven Central
JVM (Java, Kotlin)	JUnit 4 (4.13+)	`app.doqa:doqa-junit4`, Maven Central
JVM (Java, Kotlin)	TestNG (7.4+)	`app.doqa:doqa-testng`, Maven Central
Python	pytest	`doqa-pytest`, PyPI

Руководства по подключению и настройке: **Адаптер JUnit 5**, **Адаптер JUnit 4**, **Адаптер TestNG**, **Адаптер pytest**. Если проект уже размечен Allure-аннотациями, см. **Переезд с Allure**: всё подхватывается без правок кода.

Если адаптера под ваш фреймворк нет:

- отправляйте готовые отчёты JUnit XML или Allure: **все способы** работают с любым фреймворком, который умеет писать эти форматы;
- напишите собственный адаптер поверх **Autotest API**, прямого API, через которое работают и готовые адаптеры;
- напишите в поддержку **support@doqa.app** ↗: запросы на новые адаптеры принимаются.

Общие понятия

Пользовательские интерфейсы адаптеров устроены одинаково; ниже — понятия, общие для всех. Конкретные ключи, аннотации и примеры — на странице адаптера.

Конфигурация подключения

Адаптеру нужны три параметра: адрес DoQA, API-токен и ID пространства. Токен обычно задаётся переменной окружения в CI, а не в файле в репозитории. Как создать токен и узнать ID пространства — в разделах «**Создание API-токена**» и «**Как узнать ID пространства**».

Внешний ID

Каждый автотест идентифицируется в DoQA внешним ID, стабильным ключом, который не зависит от имени теста. Адаптер либо берёт его из явной аннотации в коде (в JVM — `@DoqaId`), либо выводит автоматически. Как внешний ID работает на стороне DoQA — на странице **Автотесты**.

Режим работы и прогон

Адаптер решает, в какой **прогон** записывать результаты:

- **создать новый** — режим по умолчанию: адаптер сам создаёт прогон и наполняет его;
- **писать в существующий** — прогон создан заранее в интерфейсе или через API, адаптеру передаётся его ID;
- **селективный** — адаптер запрашивает у DoQA список автотестов, которые числятся в прогоне, и исполняет только их.

Селективный режим использует сам DoQA: при **запуске автотестов из интерфейса** пайплайну передаётся ID прогона, и адаптер прогоняет ровно выбранные тесты.

Смотрите также

- **Каталог автотестов** — куда попадают присланные адаптером автотесты
- **Результаты и разбор падений** — как выглядят шаги, вложения и ошибки в прогоне
- **Первый прогон автотестов** — сценарий первой интеграции с DoQA

- **Плагины для IDE** — статусы автотестов и привязка кейсов прямо в IDE; плагин пишет ту же разметку, что читает адаптер

Адаптер JUnit 5

`app.doqa:doqa-junit5` отправляет результаты тестов JUnit 5 (Jupiter) в DoQA: автотесты создаются и обновляются сами, результаты приходят с шагами, фикстурами, параметрами, вложениями и ссылками. Адаптер работает в двух режимах: напрямую в **Autotest API** или файлами (Allure-совместимые артефакты, без сети и без токена в тестовом процессе). Если DoQA недоступен или не настроен, тесты проходят как обычно: ошибка отправки никогда не роняет сборку.

Требования и установка

Нужен JDK 11 или новее (адаптер проверяется на JDK 11, 17 и 21) и JUnit 5 (Jupiter). Добавьте зависимость:

Maven Gradle

xml

```xml
<dependency>
  <groupId>app.doqa</groupId>
  <artifactId>doqa-junit5</artifactId>
  <version>0.1.6</version>
  <scope>test</scope>
</dependency>
```

Уже на этом шаге, без какой-либо конфигурации, адаптер пишет результаты в `./results/`, файлы Allure-совместимого формата, которые принимает конвейер загрузки DoQA. Загрузить их в DoQA можно любым способом со страницы **Отправка результатов автотестов**. Аннотации не обязательны: каждый тест получает стабильный идентификатор автоматически.

Конфигурация

Чтобы отправлять результаты сразу в DoQA, создайте `doqa.properties` в рабочей директории запуска тестов (для Maven это директория модуля; путь можно переопределить

через `-Ddoqa.config=…` или `DOQA_CONFIG`) или задайте те же ключи через переменные окружения:

```properties
url=https://demo.doqa.app
token=<project-токен>
spaceId=42
```

С этими тремя ключами адаптер переключается в API-режим: сам создаёт прогон и наполняет его. По умолчанию — одной порцией в конце, в realtime-режиме — по ходу выполнения. Как создать токен и узнать ID пространства — в разделах «Создание API-токена» и «Как узнать ID пространства».

Источники настроек, по возрастанию приоритета: файл `doqa.properties` → переменные окружения `DOQA_*` → JVM-свойства `-Ddoqa.*`.

Внимание

Токен в конфиге означает, что каждый запуск тестов пишет в DoQA, включая локальные. Обычная схема: локально конфига нет (результаты остаются файлами и никуда не отправляются), а в CI ключи приходят из переменных окружения `DOQA_URL` / `DOQA_TOKEN` / `DOQA_SPACE_ID`.

Все ключи

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	Дефолт
`reporting`	`DOQA_REPORTING`	куда слать: `api` / `files` / `auto` / `off`	`auto`
`resultsDir`	`DOQA_RESULTS_DIR`	каталог файлового режима	`results`
`url`	`DOQA_URL`	адрес DoQA	—
`token`	`DOQA_TOKEN`	project-токен	—

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	Дефолт
`spaceId`	`DOQA_SPACE_ID`	ID пространства	—
`configurationId`	`DOQA_CONFIGURATION_ID`	конфигурация прогона (браузер/ОС и т. п.)	—
`testRunId`	`DOQA_TEST_RUN_ID`	существующий прогон (нужен для режимов `0` и `1`)	—
`testRunName`	`DOQA_TEST_RUN_NAME`	имя создаваемого прогона (режим `2`)	—
`adapterMode`	`DOQA_ADAPTER_MODE`	режим выбора прогона: `2` — new, `1` — existing, `0` — selective (см. ниже)	`2`
`importRealtime`	`DOQA_IMPORT_REALTIME`	`true` — отправка результатов по мере прогона (порция на каждый завершённый класс, вместе с его `@AfterAll`)	`false` (порция в конце)
`certValidation`	`DOQA_CERT_VALIDATION`	`false` — доверять самоподписанным TLS-сертификатам (отключает и проверку hostname)	`true`

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	Дефолт
`proxy`	`DOQA_PROXY`	HTTP-прокси, `host:port`	—
`pipelineId`	`DOQA_PIPELINE_ID`	привязка прогона к CI-пайплайну	авто: `CI_PIPELINE_ID` / `GITHUB_RUN_ID`
`ciRunId`	`DOQA_CI_RUN_ID`	id CI-запуска, инициированного из DoQA; передаётся в пайплайн автоматически и отправляется обратно с результатами	—
`branch`	`DOQA_BRANCH`	ветка прогона	авто: `CI_COMMIT_REF_NAME` / `GITHUB_REF_NAME`
`batchSize`	`DOQA_BATCH_SIZE`	максимум результатов в одной порции (одном запросе)	`100`
`requestTimeoutMs`	`DOQA_REQUEST_TIMEOUT_MS`	таймаут HTTP-запроса, мс	`30000`
`retries`	`DOQA_RETRIES`	попыток на запрос (повторяются только безопасные запросы)	`3`
`retryBackoffMs`	`DOQA_RETRY_BACKOFF_MS`	базовая пауза между попытками (растёт	`500`

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	Дефолт
		экспоненциально), мс	
`maxTraceLength`	`DOQA_MAX_TRACE_LENGTH`	лимит длины стек-трейса в результате, символов	`100000`
`maxMessageLength`	`DOQA_MAX_MESSAGE_LENGTH`	лимит длины сообщений, символов	`10000`
`maxParameterLength`	`DOQA_MAX_PARAMETER_LENGTH`	лимит длины значений параметров, символов	`2000`

`pipelineId` и `branch` подхватываются автоматически из стандартных переменных GitLab и GitHub Actions. Прогон в DoQA привязывается к пайплайну и ветке без настройки.

Куда уходят результаты: `reporting`

Значение	Что происходит
`auto` (дефолт)	есть `url` + `token` + `spaceId` — API; нет — файлы, и в лог выводится предупреждение с перечислением недостающих настроек
`api`	только API (без конфига — предупреждение в лог с перечислением недостающих настроек)
`files`	только файлы Allure-совместимого формата в `resultsDir`
`off`	адаптер выключен полностью

Файловый режим — путь CI-артефактов: тестовому процессу не нужны ни сеть до DoQA, ни секреты. Файлы забирает следующий шаг пайплайна и загружает их в DoQA (см. **пример ниже**).

Режимы работы с прогоном: `adapterMode`

- `2` / **new** (дефолт) — адаптер сам создаёт прогон и отправляет всё в него. Вариант для CI «просто прогони всё».
- `1` / **existing** — всё отправляется в существующий прогон `testRunId` (его создали заранее, в интерфейсе или через API). Если `testRunId` задан, а `adapterMode` не задан явно, адаптер сам работает в этом режиме: указанный прогон никогда не игнорируется молча.
- `0` / **selective** — селективный прогон: адаптер спрашивает у DoQA, какие автотесты числятся в прогоне `testRunId`, и физически исполняет только их. Остальные тесты исключаются ещё на этапе discovery. Этим режимом DoQA перезапускает выбранные тесты при **запуске из интерфейса**.

При запуске пайплайна из DoQA переменные `DOQA_TEST_RUN_ID` и `DOQA_ADAPTER_MODE` передаются пайплайну автоматически. Адаптер читает их как обычные настройки, отдельная конфигурация не нужна.

Что именно исключает селективный прогон

Невыбранный тест не исполняется: обычный тест исключается на этапе discovery. Инвокация параметризованного теста с плейсхолдером в ID (`@DoqaId("LOGIN-{browser}")`) исключается, когда становятся известны её аргументы; при этом её `@BeforeEach` / `@AfterEach` всё равно отрабатывают. Динамический тест исключается на строке `Doqa.addExternalId(...)`, поэтому вызывайте этот метод первым в теле теста. Метод-фабрика `@TestFactory` целиком не вызывается, если DoQA знает класс и метод выбранных автотестов и ни один из них не приходил из этой фабрики.

Результаты невыбранных тестов не отправляются. Если ID определить не удалось, тест остаётся в прогоне.

Порядок прохождения по плану DoQA

Селективный список приходит от DoQA упорядоченным, в порядке набора запуска. Адаптер умеет исполнять тесты в этом порядке, но включить план-ориентированные orderer'ы можете только вы (JUnit Jupiter не позволяет адаптеру навязать их программно):

```properties
# src/test/resources/junit-platform.properties
junit.jupiter.testclass.order.default=app.doqa.junit5.DoqaPlanClassOrderer
junit.jupiter.testmethod.order.default=app.doqa.junit5.DoqaPlanMethodOrderer
```

Методы внутри класса идут по позиции их внешнего ID в плане; классы — по минимальной позиции их методов (Jupiter исполняет тесты класс-блоками, межклассовое чередование методов невозможно). Тесты вне плана стабильно уходят в хвост; порядок никогда не меняет состав прогона. Без этих properties, без DoQA-сеанса или без плана orderer'ы — строгий no-op.

Ограничения: только Jupiter-engine; порядок гарантируется только при последовательном исполнении (`junit.jupiter.execution.parallel.enabled=false` — дефолт).

Разметка тестов

Вся разметка опциональна:

```java
@DoqaLabels({"regression"})                    // класс-уровень: наследуется всеми те
class LoginTests {

    @Test
    @DoqaId("LOGIN-1")                         // стабильный внешний ID автотеста (ре
    @DoqaTitle("Успешный вход")
    @DoqaDescription("Проверяет happy-path входа по паролю")
    @DoqaDisplayName("Вход по паролю")         // имя автотеста (иначе — display name
    @DoqaLabels({"smoke"})                     // объединится с класс-уровнем
    @DoqaTags({"ui"})
    @DoqaLinks({@DoqaLink(url = "https://tracker/BUG-77", type = "defect", title = "фл
    @DoqaCaseIds({1041})                       // привязка к тест-кейсам DoQA (можно
    @DoqaCreateManualCase                      // без привязки — попросить DoQA завес
    void loginHappyPath() { /* … */ }
}
```

Ещё две аннотации управляют местом теста в дереве каталога: `@DoqaNamespace` (по умолчанию — пакет) и `@DoqaClassName` (по умолчанию — простое имя класса). Обе работают

и на уровне класса. Аннотации находятся в пакете `app.doqa.annotations`, рантайм-фасад — `app.doqa.Doqa`: они общие для всех JVM-адаптеров DoQA, смена фреймворка не потребует править импорты.

Если `@DoqaId` нет, идентификатор ищется в таком порядке: `[DOQA-123]` или `@DOQA:123` в display name → совместимый ID из Allure-разметки (**Переезд с Allure**) → детерминированный хэш сигнатуры метода. История автотеста стабильна в любом случае; явный ID делает её устойчивой ещё и к переименованиям. Что происходит с историей и привязками на стороне DoQA — на странице **Автотесты**.

Параметризованные тесты

Аргументы каждой инвокации уходят как именованные параметры результата:

```java
@ParameterizedTest
@ValueSource(strings = {"chrome", "firefox"})
void worksIn(String browser) { /* … */ }        // parameters: [{name: "browser", valu
```

Чтобы получить отдельный автотест на каждую инвокацию, используйте плейсхолдер `{имяАргумента}` в любой аннотации:

```java
@ParameterizedTest
@ValueSource(strings = {"chrome", "firefox"})
@DoqaId("LOGIN-IN-{browser}")                   // → LOGIN-IN-chrome, LOGIN-IN-firefox
@DoqaTitle("Вход в {browser}")
void loginIn(String browser) { /* … */ }
```

Аргументы инвокаций захватывает расширение `DoqaExtension`. Включите автодетект расширений (см. **Фикстуры и шаги**), иначе плейсхолдер останется нераскрытым. А чтобы имена аргументов были настоящими (`browser`, а не `arg0`), включите флаг компилятора `-parameters`.

Рантайм-API из тела теста

```java
import app.doqa.client.LinkType;

Doqa.step("открыть страницу", () -> page.open());          // шаг (вложенные — просто в
int sum = Doqa.step("посчитать", () -> a + b);             // шаг со значением
Doqa.step("чекпоинт пройден");                             // мгновенный passed-шаг без
Doqa.step("создать заказ", "POST /orders", () -> order()); // шаг с описанием
Doqa.addParameter("env", "staging");
Doqa.addAttachments("target/screenshot.png");              // файл к тесту или открытом
Doqa.addAttachment("response.json", bytes, "application/json"); // вложение из памяти
Doqa.addAttachment("app.log", logText);                    // текстовое вложение (text/
Doqa.addLink("https://jira/TASK-5", LinkType.REQUIREMENT);
Doqa.addLink(url, type, title, description);               // расширенная форма; есть и
Doqa.addMessage("покупатель создан через фабрику");
Doqa.addCaseIds(1042);                                     // привязка к тест-кейсу из
Doqa.addCreateManualCase();                                // как @DoqaCreateManualCase
Doqa.addExternalId("CART-DYN-1");                          // стабильный ID динамическо
Doqa.addTitle("…"); Doqa.addDescription("…"); Doqa.addDisplayName("…");
Doqa.addLabels("…"); Doqa.addTags("…");
Doqa.addLabel(Labels.SEVERITY, "critical");                // key:value-метки (severity
```

Все вызовы безопасны: вне активного теста или при `reporting=off` они просто ничего не делают. Нативные JUnit `@Tag` автоматически попадают в теги автотеста. Двойная разметка не нужна.

Шаги и вложения из потоков, которые тест порождает сам (async-код, свои executor'ы), нужно явно перенести в контекст теста:

```java
Doqa.Context ctx = Doqa.captureContext();
executor.submit(() -> Doqa.runWith(ctx, () -> Doqa.step("проверка из воркера")));
```

Фикстуры и шаги @Step

Адаптер работает и без этого, но с двумя настройками отчёт полный.

Фикстуры и параметры инвокаций: автодетект расширений

properties

```
# src/test/resources/junit-platform.properties
junit.jupiter.extensions.autodetection.enabled=true
```

(или точечно `@ExtendWith(DoqaExtension.class)` на классе). Это даёт:

- `@BeforeEach` / `@AfterEach` — блоки setup/teardown в каждом результате;
- `@BeforeAll` / `@AfterAll` — класс-фикстуры у всех тестов класса;
- шаги и вложения внутри класс-фикстур попадают в узел фикстуры;
- именованные параметры и плейсхолдеры `{param}`.

Декларативные шаги `@Step`: AspectJ-агент

java

```
@Step("авторизоваться под {user}")  // {param}-плейсхолдеры из аргументов метода; без
void authorize(String user) { /* … */ }
```

Для `@Step` подключите AspectJ-агент к тестовой JVM (адаптер не приносит его транзитивно):

Maven Gradle

xml

```
<dependency>
  <groupId>org.aspectj</groupId>
  <artifactId>aspectjweaver</artifactId>
  <version>1.9.24</version>
  <scope>test</scope>
</dependency>

<!-- maven-dependency-plugin пишет путь к jar-файлу в property ${org.aspectj:aspectjwe
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-dependency-plugin</artifactId>
  <executions><execution><phase>initialize</phase><goals><goal>properties</goal></goal
</plugin>
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
```

```
    <configuration>
      <argLine>-javaagent:${org.aspectj:aspectjweaver:jar} --add-opens java.base/java.la
    </configuration>
  </plugin>
```

Не хотите агент — используйте явный `Doqa.step("…", () -> …)`, он работает всегда. Версия weaver'a определяет максимальную поддерживаемую версию байткода: для новых JDK берите актуальный `aspectjweaver` (1.9.24 покрывает JDK до 24 включительно).

Совместимость с Allure

Проект, размеченный Allure-аннотациями, адаптер подхватывает без правок кода. Что именно поддерживается, чем становится идентификатор и как постепенно переехать на разметку `@Doqa*` — Переезд с Allure.

Маппинг исходов

Что случилось	Исход в DoQA
Тест прошёл	`passed`
Упала проверка (`AssertionError` , AssertJ, opentest4j)	`failed`
Любое другое исключение (инфраструктура, NPE, таймаут)	`broken`
`@Disabled` / assumption	`skipped`

Различие `failed` и `broken` важно: кластеризация ошибок и аналитика нестабильных (flaky) обрабатывают их по-разному.

Запуск в CI

Отправка в API | Файлы артефактом

yaml

```yaml
# .gitlab-ci.yml — DOQA_URL, DOQA_TOKEN, DOQA_SPACE_ID заданы в переменных CI/CD проек
test:
  image: maven:3.9-eclipse-temurin-17
  script:
    - mvn -B test
```

В API-варианте адаптер сам создаёт прогон и привязывает его к пайплайну и ветке (переменные CI подхватываются автоматически). В файловом варианте каталог `results/` загружает в DoQA следующий шаг пайплайна. Способы описаны на странице **Отправка результатов автотестов**. Для GitLab DoQA умеет сама завести `DOQA_URL` / `DOQA_TOKEN` / `DOQA_SPACE_ID` в переменных проекта — см. **Подключение CI-систем**.

Как результаты доставляются

По умолчанию адаптер копит результаты и в конце прогона отправляет их порциями по `batchSize`; при сбое одной порции теряется только она. При `importRealtime=true` каждый завершённый тестовый класс уходит сразу, вместе со своим `@AfterAll`, поэтому teardown не теряется. Повторяются только безопасные запросы. Ошибка отправки никогда не роняет сборку. Адаптер пишет WARNING в лог и продолжает.

Устранение неполадок

▶ **Частые симптомы и решения**

Смотрите также

- **Адаптеры для тестовых фреймворков** — когда адаптер, а когда отчёты
- **Автоматика запусков** — расписания, авто-повтор упавших, критерии приёмки прогона
- **Первый прогон автотестов** — сценарий первой интеграции с DoQA

Адаптер JUnit 4

`app.doqa:doqa-junit4` отправляет результаты тестов JUnit 4 в DoQA: автотесты создаются и обновляются сами, результаты приходят с шагами, фикстурами, параметрами, вложениями и ссылками. Как и у JUnit 5, ошибка отправки никогда не роняет сборку. Тесты проходят как обычно.

Установка и обязательная регистрация

Нужны JDK 11 или новее (адаптер проверяется на JDK 11, 17 и 21) и JUnit **4.13** (проверяется на 4.13.2; на версиях 4.11 и ниже гарантий нет). Добавьте зависимость:

Maven Gradle

xml

```xml
<dependency>
  <groupId>app.doqa</groupId>
  <artifactId>doqa-junit4</artifactId>
  <version>0.1.6</version>
  <scope>test</scope>
</dependency>
```

В отличие от JUnit 5, у JUnit 4 нет ни SPI, ни `ServiceLoader` для `RunListener`. Без явной регистрации адаптер молчит: зависимость подключена, но результаты никуда не уходят, даже в файлы. Выберите один из двух способов.

Maven (surefire): листенер

Одна настройка на весь прогон, тестовый код не трогаете:

xml

```xml
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-surefire-plugin</artifactId>
  <configuration>
    <properties>
      <property>
```

```
        <name>listener</name>
        <value>app.doqa.junit4.DoqaRunListener</value>
      </property>
    </properties>
  </configuration>
</plugin>
```

Gradle и запуск из IDE: `@RunWith`

Для Gradle и запуска из IDE точки регистрации листенера в JUnit 4 нет вовсе. Разметьте класс (или общий базовый класс) рунером. Он сам подцепит листенер:

```java
@RunWith(app.doqa.junit4.DoqaRunner.class)
public class LoginTests { /* … */ }
```

Способы совместимы: если внешний листенер уже зарегистрирован, рунер второй раз его не добавляет. `DoqaRunner` даёт больше, чем просто отправку результатов, — подробности в разделе «Особенности и ограничения».

Внимание

`DoqaRunner` — обычный `BlockJUnit4ClassRunner`, он не совместим с классами, у которых уже есть свой рунер (`SpringRunner`, `MockitoJUnitRunner`, `Suite`, `Enclosed`). Для таких классов остаётся листенер (базовый уровень отправки результатов) и, для селективного запуска, `DoqaSelectRule` — см. «Селективный запуск: три уровня». У `@RunWith(Parameterized.class)` рунер адаптера свой, он подключается фабрикой — см. «Параметризованные тесты».

Сразу после подключения зависимости и регистрации листенера или рунера адаптер пишет результаты в `./results/`, файлы Allure-совместимого формата, которые принимает конвейер загрузки DoQA. Загрузить их можно любым способом со страницы **Отправка результатов автотестов**. Аннотации не обязательны: каждый тест получает стабильный идентификатор автоматически.

Разметка тестов

Аннотации `@Doqa*` — из пакета `app.doqa.annotations`, общего для всех JVM-адаптеров DoQA. Полный список с примерами — на странице **Адаптер JUnit 5**. Смена фреймворка не требует менять импорты.

```java
public class LoginTests {

    @Test
    @DoqaId("LOGIN-1")
    @DoqaTitle("Успешный вход")
    @DoqaLabels({"smoke"})
    public void loginHappyPath() { /* … */ }
}
```

JUnit 4 дополнительно читает нативную разметку, двойная разметка не нужна:

Что в коде	Что получается в DoQA
`@Category({Smoke.class, Ui.class})` на методе и/или классе	теги `Smoke`, `Ui` (простые имена классов-категорий, метод и класс объединяются)
`@Ignore("нужен стенд")`	исход `skipped` с причиной; на классе — по каждому `@Test`-методу

Если `@DoqaId` нет, идентификатор ищется в порядке: маркер `[DOQA-123]` / `@DOQA:123` в имени теста → Allure `@AllureId` → детерминированный хэш сигнатуры метода. Имя теста в JUnit 4 — это имя метода, поэтому маркер доступен только в имени инвокации `@Parameters(name = …)`. Обычный путь — явный `@DoqaId`. Хэш идёт с префиксом фреймворка (`junit4:<хэш>` против `junit5:<хэш>`), да и сама подпись у фреймворков разная: переход на другой JVM-адаптер без явного `@DoqaId` начинает историю автотеста заново.

Конфигурация

Источники настроек, по возрастанию приоритета: файл `doqa.properties` (в рабочей директории запуска тестов, для Maven — директория модуля; путь переопределяется `-`

`-Ddoqa.config=…` или `DOQA_CONFIG`) → переменные окружения `DOQA_*` → JVM-свойства `-Ddoqa.*`. Механизм общий для всех JVM-адаптеров DoQA.

properties

```
url=https://demo.doqa.app
token=<project-токен>
spaceId=42
```

С этими тремя ключами адаптер сам создаёт **прогон** и наполняет его. Как создать токен и узнать ID пространства — в разделах «**Создание API-токена**» и «**Как узнать ID пространства**».

Все ключи

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	Дефолт
`reporting`	`DOQA_REPORTING`	куда слать: `api` / `files` / `auto` / `off`	`auto`
`resultsDir`	`DOQA_RESULTS_DIR`	каталог файлового режима	`results`
`url`	`DOQA_URL`	адрес DoQA	—
`token`	`DOQA_TOKEN` (алиас `DOQA_PRIVATE_TOKEN`)	project- или personal-токен	—
`spaceId`	`DOQA_SPACE_ID` (алиас `DOQA_PROJECT_ID`)	ID пространства	—
`configurationId`	`DOQA_CONFIGURATION_ID`	конфигурация прогона (браузер/ОС и т. п.)	—
`testRunId`	`DOQA_TEST_RUN_ID`	существующий прогон (нужен для режимов `0` и `1`)	—

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	Дефолт
`testRunName`	`DOQA_TEST_RUN_NAME`	имя создаваемого прогона (режим `2`)	—
`adapterMode`	`DOQA_ADAPTER_MODE`	режим выбора прогона: `2` — new, `1` — existing, `0` — selective	`2`
`importRealtime`	`DOQA_IMPORT_REALTIME`	`true` — отправка результатов по ходу прогона (по границе класса)	`false` (порция в конце)
`certValidation`	`DOQA_CERT_VALIDATION`	`false` — доверять самоподписанным TLS-сертификатам (отключает и проверку hostname)	`true`
`proxy`	`DOQA_PROXY`	HTTP-прокси, `host:port`	—
`pipelineId`	`DOQA_PIPELINE_ID`	привязка прогона к CI-пайплайну	авто: `CI_PIPELINE_ID` / `GITHUB_RUN_ID`
`ciRunId`	`DOQA_CI_RUN_ID`	ID CI-запуска, инициированного из DoQA; приезжает в пайплайн сам и отправляется обратно с результатами	—

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	Дефолт
`branch`	`DOQA_BRANCH`	ветка прогона	авто: `CI_COMMIT_REF_NAME` / `GITHUB_REF_NAME`
`batchSize`	`DOQA_BATCH_SIZE`	максимум результатов в одной порции (одном запросе)	`100`
`requestTimeoutMs`	`DOQA_REQUEST_TIMEOUT_MS`	таймаут HTTP-запроса, мс	`30000`
`retries`	`DOQA_RETRIES`	попыток на запрос (повторяются только безопасные)	`3`
`retryBackoffMs`	`DOQA_RETRY_BACKOFF_MS`	базовая пауза между попытками (растёт экспоненциально), мс	`500`
`maxTraceLength`	`DOQA_MAX_TRACE_LENGTH`	лимит длины стек-трейса, символов	`100000`
`maxMessageLength`	`DOQA_MAX_MESSAGE_LENGTH`	лимит длины сообщений, символов	`10000`
`maxParameterLength`	`DOQA_MAX_PARAMETER_LENGTH`	лимит длины значений параметров, символов	`2000`

`pipelineId` и `branch` подхватываются автоматически из переменных GitLab и GitHub Actions; для остальных CI задавайте `DOQA_PIPELINE_ID` вручную.

Что отправляется

Рантайм-API из тела теста — тот же, что у остальных JVM-адаптеров DoQA:

```java
Doqa.step("открыть страницу", () -> page.open());
Doqa.addParameter("env", "staging");
Doqa.addAttachments("target/screenshot.png");
Doqa.addLink("https://jira/TASK-5", LinkType.REQUIREMENT);
Doqa.addCaseIds(1042);
```

Декларативные шаги `@Step` работают так же, как в JUnit 5, и так же требуют AspectJ-агента в тестовой JVM — **как его подключить**.

Полнота шагов и фикстур зависит от способа регистрации:

Что	Листенер (базовый уровень)	`@RunWith(DoqaRunner.class)`
Исход, сообщение, трейс, длительность	да	да
`Doqa.step` и `@Step`, вложения, ссылки, метки, параметры	да, одним блоком шагов	да, разложено по фазам
`@Before` / `@After`	входят в тело результата, отдельных узлов нет	отдельный шаг на каждый метод
`@BeforeClass` / `@AfterClass`	один агрегированный узел на класс (время — между событиями JUnit, включает и работу рунера)	точный узел на каждый метод и точное время

Параметризованные тесты

Параметризованные тесты (`@Parameterized`) публикуют аргументы **только** через свою фабрику (JUnit 4 не отдаёт их ни в `Description`, ни листенеру):

```java
@RunWith(Parameterized.class)
@Parameterized.UseParametersRunnerFactory(DoqaParametersRunnerFactory.class)
public class LoginParamTests { /* … */ }
```

Фабрика подставляет рунер адаптера, поэтому класс с ней получает и остальное, что даёт `DoqaRunner`: отдельные шаги на `@Before`/`@After`, физический деселект в селективном прогоне и порядок по плану. Узлы `@BeforeClass`/`@AfterClass` остаются на базовом уровне: `Parameterized` выполняет их один раз вокруг всех наборов данных, во внешнем рунере.

Имена параметров берутся из полей `@Parameterized.Parameter`, а при инъекции через конструктор — из имён его аргументов, для чего проект компилируется с флагом `-parameters`. Иначе параметры называются `arg0`, `arg1`.

Без фабрики тест всё равно репортится, но параметр восстанавливается из имени инвокации: `@Parameters(name = "{index}: browser={0}")` даст один параметр `arguments` со значением `browser=chrome`, а имя по умолчанию (только индекс) — ни одного. Плейсхолдер `{имяАргумента}` в `@DoqaId`/`@DoqaTitle` при этом не раскрывается; обходной путь — `Doqa.addParameter("browser", browser)` или `Doqa.addExternalId("LOGIN-" + browser)` в теле теста.

Маппинг исходов

Что случилось	Исход в DoQA
Тест прошёл (в том числе `@Test(expected = …)`, отработавший как ожидалось)	`passed`
Упала проверка (`AssertionError`, `ComparisonFailure`, AssertJ, opentest4j)	`failed`
Любое другое исключение	`broken`
`@Ignore` на методе или на классе	`skipped`
Невыполненное `Assume` — в тесте, в `@Before` или в `@BeforeClass`	`skipped`

Упало `@BeforeClass`/`@ClassRule` — результат заводится по одному на каждый тест класса, плюс ошибка на узле `@BeforeClass`. Один тест может упасть дважды, в теле и в `@After`:

адаптер склеивает сообщения и трейсы в один результат, исход `broken`, если хотя бы одна из ошибок не assertion, иначе `failed`.

Особенности и ограничения

Адаптеру в JUnit 4 негде отфильтровать тесты: листенер не может пропустить тест, а `Filter` провайдеры surefire наружу не отдают. Поэтому исключение из селективного прогона устроено тремя уровнями, а не одним `PostDiscoveryFilter`, как в JUnit 5.

Селективный запуск: три уровня

При запуске выбранных автотестов из DoQA (`adapterMode=0`) применяется тот из уровней, что доступен классу:

1. **Базовый слой (только листенер, код менять не нужно)** — гейт на отправке: тесты исполняются все, результаты невыбранных не отправляются. Прогон корректный, но время CI не экономится.
2. `@RunWith(DoqaRunner.class)` — физический деселект: невыбранный тест помечается ignored, JUnit сообщает `skipped`, адаптер по нему молчит. То же включает и порядок по плану (см. ниже).
3. `DoqaSelectRule` — для классов со своим рунером (`SpringRunner`, `MockitoJUnitRunner`, `Suite`, `Enclosed`), где `DoqaRunner` не подходит; у `@Parameterized` физический деселект даёт фабрика. Правило скипает невыбранный тест через assumption — не исполняются ни `@Before`, ни тело:

```java
public class SpringLoginTests {
    @ClassRule
    public static final DoqaSelectRule CLASS_SELECT = new DoqaSelectRule();
    @Rule
    public final DoqaSelectRule select = new DoqaSelectRule();
}
```

`@Rule` скипает конкретный тест, `@ClassRule` — весь класс целиком, если не выбран ни один его тест. Вне режима `0`, а также если DoQA не смог сопоставить тест по внешнему ID,

фильтр — строгий no-op: адаптер не имеет права выкинуть тест из прогона из-за собственной ошибки идентификации.

Порядок прохождения по плану DoQA

Работает автоматически, но только с рунером адаптера — `@RunWith(DoqaRunner.class)` или **фабрикой для** `@Parameterized`: методы внутри класса идут по позиции их внешнего ID в плане. Порядок классов задаёт хост (например, `runOrder` surefire), а для классов с `@FixMethodOrder` план не применяется никогда.

Параллельность

С surefire `parallel` в базовом слое (только листенер) шаги и вложения внутри класс-фикстур не гарантируются. Сами узлы фикстур при этом записываются. С `@RunWith(DoqaRunner.class)` этого ограничения нет.

Доставка результатов в реальном времени

При `importRealtime=true` результаты класса уходят одной порцией, когда класс завершился, уже после его `@AfterClass`, с полными узлами класс-фикстур. На JUnit 4.12 suite-событий нет, и предыдущий класс отправляется только со стартом следующего, а последний — вместе с концом прогона (ещё один довод держать минимум 4.13).

Прочие ограничения

- Минимум **4.13** — на 4.12 не работают suite-события и перехват отдельного фикстур-метода: узлы `@BeforeClass` / `@AfterClass` базовым слоем не собираются, `@Before` / `@After` не становятся отдельными шагами даже с рунером.
- `@Rule` / `@ClassRule` (кроме `DoqaSelectRule`) не становятся отдельными узлами отчёта.
- `@Theory`-тесты репортятся как обычные. Значения датапоинтов в параметры не попадают. Динамических тестов (аналог JUnit 5 `@TestFactory`) в JUnit 4 нет вовсе.
- Несколько форков surefire (`forkCount>1` или `reuseForks=false`): каждый форк создаёт свой прогон в режиме `2`. Решение: задать `testRunId` и `adapterMode=1` (или `0`).
- Не подключайте к одному прогону сразу два JVM-адаптера DoQA: идентификация фреймворка в общем ядре одна на JVM.

JUnit 4 через JUnit Platform (vintage)

Прогон переходит на платформу сам, как только на тестовом classpath появляется `junit-jupiter`: surefire переключается на платформенный провайдер, и без `junit-vintage-engine` классы JUnit 4 перестают находиться. Сборка не падает, а в логе `Tests run: 0`.

Если JUnit 4 гоняется через `junit-platform` с `junit-vintage-engine`, деселект на discovery выполняет **`doqa-junit5`** своим `PostDiscoveryFilter`: тесты JUnit 4 приходят к нему от vintage-движка. `doqa-junit4` в таком прогоне не участвует: специфика этой страницы (узлы `@BeforeClass`, фазы `@Before` / `@After`, три уровня селекции) не появляется, а fallback-ID считается иначе. На этом пути размечайте тесты явным `@DoqaId`.

Совместимость с Allure

Проект, размеченный Allure-аннотациями, адаптер подхватывает без правок кода, как и остальные JVM-адаптеры DoQA. Что именно поддерживается и как переехать на разметку `@Doqa*` — Переезд с Allure.

Смотрите также

- **Адаптер JUnit 5** — общий для всех JVM-адаптеров DoQA справочник по аннотациям, ключам конфигурации и режимам прогона
- **Адаптеры для тестовых фреймворков** — когда адаптер, а когда отчёты
- **Запуск автотестов из интерфейса DoQA** — как формируется прогон для селективного режима
- **Первый прогон автотестов** — сценарий первой интеграции с DoQA

Адаптер TestNG

`app.doqa:doqa-testng` отправляет результаты тестов TestNG в DoQA: автотесты создаются и обновляются сами, результаты приходят с шагами, фикстурами, параметрами, вложениями и ссылками. Если DoQA недоступен или не настроен, тесты проходят как обычно: ошибка отправки никогда не роняет сборку.

Установка

Нужен JDK 11 или новее (адаптер проверяется на JDK 11, 17 и 21) и TestNG 7.4 или новее (собирается и проверяется против 7.12). Добавьте зависимость:

Maven Gradle

xml

```xml
<dependency>
  <groupId>app.doqa</groupId>
  <artifactId>doqa-testng</artifactId>
  <version>0.1.6</version>
  <scope>test</scope>
</dependency>
```

Регистрация: ничего делать не нужно

В отличие от JUnit 4, у TestNG есть SPI: файл `META-INF/services/org.testng.ITestNGListener` внутри `doqa-testng` подхватывается TestNG автоматически и регистрирует сразу два компонента: листенер результатов `DoqaTestNgListener` и интерцептор селективного прогона `DoqaMethodInterceptor`. Дополнительная настройка проекта не нужна.

Есть ещё три необязательных способа зарегистрировать листенер: `<listeners>` в `testng.xml`, `listener` в конфигурации surefire, аннотация `@Listeners(DoqaTestNgListener.class)`. Совмещать их с автоматической SPI-регистрацией безопасно: TestNG дедуплицирует листенеры по классу.

Выключить адаптер, не убирая зависимость: `doqa.reporting=off` (см. **Конфигурацию**) либо родной рубильник TestNG `-spilistenerstoskip app.doqa.testng.DoqaTestNgListener`.

Уже на этом шаге, без какой-либо конфигурации, адаптер пишет результаты в `./results/`, файлы Allure-совместимого формата, которые принимает конвейер загрузки DoQA. Загрузить их в DoQA можно любым способом со страницы **Отправка результатов автотестов**. Аннотации не обязательны: каждый тест получает стабильный идентификатор автоматически.

Разметка тестов

Разметка находится в пакете `app.doqa.annotations`, рантайм-фасад — `app.doqa.Doqa`: это общее ядро для всех JVM-адаптеров DoQA (JUnit 5, JUnit 4, TestNG), переход между фреймворками не требует менять импорты. Вся разметка опциональна:

```java
@DoqaLabels({"regression"})                    // класс-уровень: наследуется всеми те
public class LoginTests {

    @Test(groups = {"smoke", "api"})
    @DoqaId("LOGIN-1")                         // стабильный внешний ID автотеста (ре
    @DoqaTitle("Успешный вход")
    @DoqaDescription("Проверяет happy-path входа по паролю")
    @DoqaDisplayName("Вход по паролю")         // имя автотеста (если инвокация не на
    @DoqaLabels({"smoke"})                     // объединится с класс-уровнем
    @DoqaLinks({@DoqaLink(url = "https://tracker/BUG-77", type = "defect", title = "фл
    @DoqaCaseIds({1041})                       // привязка к тест-кейсам DoQA (можно
    public void loginHappyPath() { /* … */ }
}
```

Аннотация	Уровень	Семантика
`@DoqaId("LOGIN-1")`	метод, класс	стабильный внешний ID автотеста. Поддерживает плейсхолдер `{имяАргумента}`
`@DoqaTitle("…")`	метод, класс	заголовок автотеста в каталоге
`@DoqaDescription("…")`	метод, класс	описание автотеста

Аннотация	Уровень	Семантика
`@DoqaDisplayName("…")`	метод, класс	имя автотеста; сильнее `@Test(description)` и имени метода, но слабее рантайм-имени инвокации (`@Test(testName)` , `ITest#getTestName()`)
`@DoqaLabels({"smoke"})`	метод, класс	метки; класс и метод складываются
`@DoqaTags({"ui"})`	метод, класс	теги автотеста; складываются
`@DoqaLink` / `@DoqaLinks({...})`	метод, класс	ссылки: `url` , `type` (`related` , `defect` , `requirement` , `blocked_by` , `repository`), `title` , `description` ; складываются
`@DoqaCaseIds({1041})`	метод, класс	привязка к ручным тест-кейсам DoQA, можно несколько
`@DoqaCreateManualCase`	метод, класс	попросить DoQA завести ручной кейс для сиротского автотеста (без привязки к кейсам); действует независимо от настройки этого поведения в пространстве
`@DoqaNamespace("…")`	метод, класс	место в дереве каталога; по умолчанию — пакет
`@DoqaClassName("…")`	метод, класс	класс автотеста; по умолчанию — простое имя класса
`@Step("авторизоваться под {user}")`	метод	декларативный шаг; плейсхолдеры `{param}` из аргументов метода; без текста — имя метода. Требует AspectJ-агента

Скалярные значения: метод важнее класса. Метки, теги и ссылки складываются. Все `@Doqa*` наследуются подклассами.

Разметка самого TestNG (двойная разметка не нужна)

Что читается	Что получается в DoQA
`@Test(groups = {"smoke", "api"})` на методе и/или классе	теги `smoke`, `api` (объединяются)
`@Test(description = "…")`	имя автотеста, если нет `@DoqaDisplayName`; маркер `[DOQA-123]` в описании подхватится каскадом ID
`@Test(testName = "…")` / `ITest#getTestName()`	имя конкретной инвокации перебивает и `@DoqaDisplayName`, и `@Test(description)`; в идентичность автотеста **не** входит
`@Test(enabled = false)`	`skipped` с причиной `disabled with @Test(enabled = false)`
`@DataProvider`, `@Parameters` из `testng.xml`, `@Factory`	параметры результата
Allure: `@AllureId`, `@Epic`, `@Feature`, `@Story`, `@Owner`, `@Severity`, `@Link`, URL-значные `@Issue` / `@TmsLink`, `@Description`	ключ `ALLURE-<id>` + метка `as_id`; типизированные ссылки; описание — читается без зависимости от Allure, подробнее в **Переезде с Allure**

Если `@DoqaId` нет, внешний ID ищется в таком порядке: маркер `[DOQA-123]` или `@DOQA:123` в `@Test(description)` (а если описания нет — в имени метода; `@DoqaDisplayName` и `@Test(testName)` для маркера не читаются) → Allure `@AllureId` (`ALLURE-123`) → детерминированный хэш сигнатуры метода с префиксом `testng:`.

Внимание

У TestNG в этот хэш входит ещё и `@Test(description)` (он же имя автотеста). Правка описания у теста без явного `@DoqaId` начинает его историю в DoQA заново. Параметризованных тестов это не касается: их имя в хэш не входит. Переход тестового класса с другого JVM-фреймворка на TestNG без явного `@DoqaId` тоже начнёт историю заново: префикс хэша у фреймворков разный.

Параметризованные тесты

Без плейсхолдера все инвокации одного метода сворачиваются в один автотест, а аргументы уходят в `parameters[]` результата. TestNG отдаёт значения аргументов листенеру сам через `ITestResult.getParameters()`. В отличие от JUnit 5 (нужен автодетект расширений) и JUnit 4 (нужна отдельная фабрика раннера), включать здесь ничего не нужно:

```java
@Test(dataProvider = "browsers")
@DoqaId("LOGIN-IN-{browser}")                    // → LOGIN-IN-chrome, LOGIN-IN-firefox
@DoqaTitle("Вход в {browser}")
public void loginIn(String browser) { /* … */ }

@DataProvider(name = "browsers")
public Object[][] browsers() {
    return new Object[][]{{"chrome"}, {"firefox"}};
}
```

Аргументы, которые TestNG инжектит сам (`ITestContext`, `ITestResult`, `XmlTest`, `Method`), в параметры не попадают. `@Factory`-инстансы и `invocationCount` сворачиваются в один автотест. Параметры фабрики адаптер не читает, задавайте их вручную через `Doqa.addParameter(...)`.

Конфигурация

Источники по возрастанию приоритета: файл `doqa.properties` → переменные окружения `DOQA_*` → JVM-свойства `-Ddoqa.*`. Файл ищется в рабочей директории запуска тестов (для Maven — директория модуля); путь переопределяется `-Ddoqa.config=…` или `DOQA_CONFIG`.

Минимальный конфиг для отправки в API — три ключа:

```properties
url=https://demo.doqa.app
token=<project-токен>
spaceId=42
```

С этими тремя ключами адаптер переключается в API-режим: сам создаёт прогон и наполняет его. Как создать токен и узнать ID пространства — в разделах «Создание API-токена» и «Как узнать ID пространства».

Внимание

Токен в конфиге означает, что каждый запуск тестов пишет в DoQA, включая локальные. Обычная схема: локально конфига нет (результаты остаются файлами и никуда не отправляются), а в CI ключи приходят из переменных окружения `DOQA_URL` / `DOQA_TOKEN` / `DOQA_SPACE_ID`.

Все ключи

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	По умолчанию
`reporting`	`DOQA_REPORTING`	куда слать: `api` / `files` / `auto` / `off`	`auto`
`resultsDir`	`DOQA_RESULTS_DIR`	каталог файлового режима	`results`
`url`	`DOQA_URL`	адрес DoQA	—
`token`	`DOQA_TOKEN` (алиас `DOQA_PRIVATE_TOKEN`)	project- или personal-токен	—
`spaceId`	`DOQA_SPACE_ID` (алиас `DOQA_PROJECT_ID`)	ID пространства	—
`configurationId`	`DOQA_CONFIGURATION_ID`	конфигурация прогона (браузер/ОС и т. п.)	—
`testRunId`	`DOQA_TEST_RUN_ID`	существующий прогон (нужен для режимов `0` и `1`)	—
`testRunName`	`DOQA_TEST_RUN_NAME`	имя создаваемого прогона (режим `2`)	—
`adapterMode`	`DOQA_ADAPTER_MODE`	режим выбора прогона: `2` —	`2`

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	По умолчанию
		new, `1` — existing, `0` — selective (см. ниже)	
`importRealtime`	`DOQA_IMPORT_REALTIME`	`true` — отправка результатов в реальном времени по ходу прогона (гранулярность — см. «Как результаты доставляются»)	`false` (порция в конце)
`certValidation`	`DOQA_CERT_VALIDATION`	`false` — доверять самоподписанным TLS-сертификатам (отключает и проверку hostname)	`true`
`proxy`	`DOQA_PROXY`	HTTP-прокси, `host:port`	—
`pipelineId`	`DOQA_PIPELINE_ID`	привязка прогона к CI-пайплайну	авто: `CI_PIPELINE_ID` / `GITHUB_RUN_ID`
`ciRunId`	`DOQA_CI_RUN_ID`	id CI-запуска, инициированного из DoQA; приходит в пайплайн сам	—
`branch`	`DOQA_BRANCH`	ветка прогона	авто: `CI_COMMIT_REF_NAME` / `GITHUB_REF_NAME`

Ключ (`doqa.properties` / `-Ddoqa.<ключ>`)	Переменная окружения	Что это	По умолчанию
`batchSize`	`DOQA_BATCH_SIZE`	максимум результатов в одной порции (одном запросе)	`100`
`requestTimeoutMs`	`DOQA_REQUEST_TIMEOUT_MS`	таймаут HTTP-запроса, мс	`30000`
`retries`	`DOQA_RETRIES`	попыток на запрос (повторяются только безопасные запросы)	`3`
`retryBackoffMs`	`DOQA_RETRY_BACKOFF_MS`	базовая пауза между попытками (растёт экспоненциально), мс	`500`
`maxTraceLength`	`DOQA_MAX_TRACE_LENGTH`	лимит длины стек-трейса, символов	`100000`
`maxMessageLength`	`DOQA_MAX_MESSAGE_LENGTH`	лимит длины сообщений, символов	`10000`
`maxParameterLength`	`DOQA_MAX_PARAMETER_LENGTH`	лимит длины значений параметров, символов	`2000`

`pipelineId` подхватывается автоматически только из `CI_PIPELINE_ID` (GitLab) и `GITHUB_RUN_ID` (GitHub Actions); для Jenkins, TeamCity и остальных CI задавайте `DOQA_PIPELINE_ID` сами. При запуске пайплайна из DoQA переменные `DOQA_TEST_RUN_ID`, `DOQA_ADAPTER_MODE`, `DOQA_CI_RUN_ID` приходят в пайплайн автоматически.

Куда уходят результаты: `reporting`

Значение	Что происходит
`auto` (по умолчанию)	есть `url` + `token` + `spaceId` — API; нет — файлы, и в лог выводится предупреждение с перечислением недостающих настроек
`api`	только API (без конфига — то же предупреждение)
`files`	только файлы Allure-совместимого формата в `resultsDir`
`off`	адаптер выключен полностью

Что отправляется

Режимы работы с прогоном: `adapterMode`

- `2` / **new** (по умолчанию) — адаптер сам создаёт прогон и отправляет всё в него.
- `1` / **existing** — всё уходит в существующий прогон `testRunId`. Если `testRunId` задан, а `adapterMode` не задан явно, адаптер сам работает в этом режиме.
- `0` / **selective** — адаптер спрашивает у DoQA, какие автотесты числятся в прогоне `testRunId`, и исполняет только их. Этим режимом DoQA перезапускает выбранные тесты при запуске из интерфейса — подробнее в разделе «Селективный прогон» ниже.

Рантайм-API из тела теста

```java
import app.doqa.client.LinkType;

Doqa.step("открыть страницу", () -> page.open());          // шаг (вложенные — просто
int sum = Doqa.step("посчитать", () -> a + b);              // шаг со значением
Doqa.step("чекпоинт пройден");                               // мгновенный passed-шаг
Doqa.step("создать заказ", "POST /orders", () -> order());   // шаг с описанием
Doqa.addParameter("env", "staging");
Doqa.addAttachments("target/screenshot.png");                // файл к тесту или откры
Doqa.addAttachment("response.json", bytes, "application/json"); // вложение из памяти
Doqa.addAttachment("app.log", logText);                      // текстовое вложение (t
Doqa.addLink("https://jira/TASK-5", LinkType.REQUIREMENT);
```

```
Doqa.addLink(url, type, title, description);                    // расширенная форма; ec
Doqa.addMessage("покупатель создан через фабрику");
Doqa.addCaseIds(1042);
Doqa.addCreateManualCase();                                      // как @DoqaCreateManua
Doqa.addExternalId("CART-DYN-1");
Doqa.addTitle("…"); Doqa.addDescription("…"); Doqa.addDisplayName("…");
Doqa.addLabels("…"); Doqa.addTags("…");
Doqa.addLabel(Labels.SEVERITY, "critical");                      // key:value-метки (sev
```

Все вызовы безопасны: вне активного теста или при `reporting=off` они просто ничего не делают. Шаги и вложения из потоков, которые тест порождает сам, нужно явно перенести в контекст теста:

java

```
Doqa.Context ctx = Doqa.captureContext();
executor.submit(() -> Doqa.runWith(ctx, () -> Doqa.step("проверка из воркера")));
```

Для `@Step` подключите AspectJ-агент к тестовой JVM (адаптер не приносит его транзитивно). Способ подключения через surefire/Gradle такой же, как у JUnit 5. Не хотите агент — используйте явный `Doqa.step("…", () -> …)`, он работает всегда.

Фикстуры

Работают без дополнительных флагов. TestNG сам сообщает листенеру о каждом вызове:

Что	Как попадает в отчёт
`@BeforeMethod` / `@AfterMethod`	отдельный узел на каждый вызов
`@BeforeClass` / `@AfterClass`	узел на каждый метод, время замеряет TestNG
`@BeforeGroups` / `@AfterGroups`	блоки setup/teardown того теста, при котором фикстура запустилась. Так как TestNG вызывает её один раз на группу, это первый тест группы, а не все её тесты
`@BeforeSuite` / `@AfterSuite`, `@BeforeTest` / `@AfterTest`	**не прикрепляются к результатам**: в модели DoQA нет уровня suite/test

Маппинг исходов

Что случилось	Исход в DoQA
Тест прошёл	`passed`
Упала проверка (`AssertionError` , AssertJ, opentest4j)	`failed`
Любое другое исключение (инфраструктура, NPE, таймаут)	`broken`
`@Test(enabled = false)` , `SkipException` , невыполненный `dependsOnMethods` / `dependsOnGroups`	`skipped`

Особенности TestNG:

- исход выводится **только** из статуса `ITestResult` , никогда из «есть ли throwable»: у успешного `@Test(expectedExceptions)` исключение лежит на результате, но исход всё равно `passed` ;
- несовпавшее `@Test(expectedExceptions)` бросает `org.testng.TestException` — это не `AssertionError` , поэтому исход `broken` , а не `failed` ;
- упавшая попытка ретрая (`SKIP` + `wasRetried()`) даёт `failed` / `broken` , а **не** `skipped` ; то же для `SUCCESS_PERCENTAGE_FAILURE` ; ретраем `@Test(retryAnalyzer)` отчитываются все попытки;
- скип по `dependsOnMethods` / `dependsOnGroups` даёт `skipped` с причиной и списком методов-виновников из `getSkipCausedBy()` .

Различие `failed` и `broken` важно: **кластеризация ошибок и аналитика нестабильных (flaky)** обрабатывают их по-разному.

Особенности и ограничения

TestNG даёт адаптеру больше точек подключения, чем остальные JVM-фреймворки: SPI-регистрацию листенера и интерцептора, физический деселект на селективном прогоне и порядок по плану без единой настройки. Взамен у него своя механика отправки: результат отправляется не там, где заканчивается тест.

Селективный прогон: физический деселект интерцептором

В режиме `adapterMode=0` невыбранный тест не исполняется вовсе, а не просто не отправляется:

1. Адаптер запрашивает у DoQA список автотестов, числящихся в прогоне `testRunId`.
2. Один раз на каждый `<test>`-блок TestNG отдаёт **все** его методы интерцептору `DoqaMethodInterceptor`.
3. Интерцептор возвращает TestNG только выбранные методы. Невыбранные не исполняются и не порождают никаких событий (в отличие от JUnit 4 без рунера, где невыбранный тест всё равно выполняется, но его результат просто не отправляется).

Отсечь тест из-за собственной поломки адаптер не может: вне режима `0` и всегда, когда список выбранных получить не удалось, интерцептор отдаёт тот же список методов, что и получил. Исполняется всё, что нашёл TestNG.

> **Ловушка: зависимости деселектнутых тестов возвращаются в прогон**
>
> Если выбранный тест объявляет `dependsOnMethods` / `dependsOnGroups` на тест, которого в прогоне нет, выбросить эту зависимость нельзя: TestNG уронит весь `<test>`-блок. Адаптер добирает транзитивное замыкание зависимостей обратно: эти методы **исполняются**, но их результаты **не отправляются**. Время CI на них тратится.

Если прогон выбрал автотесты, но ни один из них не совпал с тестами, которые нашёл TestNG, не исполнится ничего. В конце прогона в лог идёт `WARNING` со списком выбранных `externalId` — обычно это переименованный метод или переехавший класс. Отправьте в DoQA результаты обычного, неселективного прогона. Каталог подхватит актуальные идентификаторы.

Порядок прохождения по плану DoQA

Включается автоматически тем же интерцептором, в отличие от JUnit 5, где план-ориентированный orderer нужно прописать вручную. Границы:

- порядок применяется только в режиме `adapterMode=0` и только внутри одного `<test>`-блока;
- граф `dependsOn*` сильнее плана: TestNG пересчитывает его **после** интерцептора, поэтому зависимость всегда идёт перед своим зависимым;

- а вот `@Test(priority)` в этом режиме не действует: TestNG исполняет тот список, который вернул интерцептор, то есть порядок плана побеждает приоритеты.

Отложенная отправка результата (неочевидное)

TestNG не даёт `ITestListener` события после `@AfterMethod`: финальный колбэк листенера приходит **до** того, как отработают teardown-фикстуры теста. Отправить результат сразу на этом колбэке значило бы потерять шаги и вложения, записанные в `@AfterMethod`. Поэтому адаптер придерживает готовый результат и отправляет его на первом событии, которое доказывает, что teardown уже отработал:

1. следующая инвокация на том же потоке;
2. следующая before-фикстура на том же потоке;
3. конец класса;
4. конец `<test>`-блока;
5. конец прогона.

Из-за этого механизма отправка в реальном времени (`importRealtime=true`) у TestNG идёт не по классам, как у JUnit 5, а `<test>`-блоками целиком: `IClassListener.onAfterClass` приходит раньше `@AfterClass`, класс-граница не годится для доказательства завершения teardown. Под surefire без `testng.xml` весь прогон — один синтетический `<test>`, поэтому отправка в реальном времени фактически будет одной порцией в конце; чтобы отправлять по частям, разбейте прогон на несколько `<test>` в `testng.xml`.

Параллельность

Поддержана как есть, без ограничений: `parallel="methods"`, `parallel="classes"`, `parallel="tests"`, `threadPoolSize` у `@Test`, `@DataProvider(parallel = true)`. TestNG исполняет инвокацию вместе с её фикстурами на одном потоке, поэтому отложенная отправка и деселект работают корректно и в параллельном прогоне.

Прочие ограничения

- Фикстуры уровня `<suite>` и `<test>` (`@BeforeSuite` / `@AfterSuite` / `@BeforeTest` / `@AfterTest`) не прикрепляются к результатам: в модели DoQA нет такого уровня.

- Класс-фикстуры пишутся один раз на класс за прогон: класс, встречающийся в нескольких `<test>`-блоках, и `@Factory` с несколькими инстансами оставят в отчёте только первый набор узлов.
- Параметры `testng.xml`, не переданные в аргументы метода, в результат не попадают; `@CustomAttribute` не читается.
- Несколько форков — несколько прогонов: при `forkCount>1` или `reuseForks=false` каждый форк — своя JVM, и в режиме `2` каждая создаст свой прогон. Решение — `testRunId` + `adapterMode=1` (или `0`).
- Не подключайте два адаптера DoQA к одному прогону: идентификация фреймворка в общем ядре одна на JVM, адаптеры перезапишут её друг другу.

Как результаты доставляются

По умолчанию адаптер копит результаты и в конце прогона отправляет их порциями по `batchSize` (100); при сбое одной порции теряется только она. При `importRealtime=true` гранулярность — `<test>`-блок целиком (см. **«Отложенная отправка результата»** выше). Повторяются только безопасные запросы. Ошибка отправки никогда не роняет сборку. Адаптер пишет `WARNING` в лог и продолжает.

Результаты при этом не пропадают. Если DoQA недоступен или отказал на старте (истёкший токен, нет активной CI-привязки, нет сети), адаптер не выключается, а переходит в файловый режим на весь прогон и пишет всё в `resultsDir`. Если связь оборвалась посреди прогона, в файлы уходит только незавершённая порция, остальное продолжает идти по API. Рядом с результатами адаптер кладёт `doqa-reporting.properties`: по нему шаг загрузки в CI отличает «файлов нет, потому что всё ушло по API» от «адаптер не отработал» и догружает остаток.

Смотрите также

- **Адаптеры для тестовых фреймворков** — когда адаптер, а когда отчёты
- **Адаптер JUnit 5** — общее ядро JVM-адаптеров, декларативные шаги `@Step` и AspectJ-агент
- **Переезд с Allure** — что подхватывается без правок кода
- **Автоматика запусков** — расписания, авто-повтор упавших, критерии приёмки прогона
- **Первый прогон автотестов** — сценарий первой интеграции с DoQA

Адаптер pytest

`doqa-pytest` отправляет результаты тестов pytest в DoQA: автотесты создаются и обновляются сами, результаты приходят с шагами, фикстурами, параметрами, вложениями и ссылками. Адаптер работает в двух режимах: напрямую в **Autotest API** или файлами (Allure-совместимые артефакты, без сети и без токена в тестовом процессе); режим выбирается конфигурацией, а не состоянием DoQA. Ошибка отправки только пишется в лог предупреждением. Прогон она не роняет.

Требования и установка

Нужен Python 3.9 или новее и pytest 7.0 или новее. Установите пакет:

```bash
pip install doqa-pytest
```

Последняя опубликованная версия — **0.1.1** (тянет за собой `doqa-client==0.1.1`). Адаптер регистрируется автоматически через entry point `pytest11`. Прописывать плагин не нужно. Разово выключить его на конкретный запуск: `pytest -p no:doqa`.

Уже на этом шаге, без какой-либо конфигурации, адаптер пишет результаты в `./results/`, файлы Allure-совместимого формата. Загрузить их в DoQA можно любым способом со страницы **Отправка результатов автотестов**. Обычно это делает следующий шаг пайплайна. Разметка тестов не обязательна: каждый тест получает стабильный идентификатор автоматически.

Конфигурация

Источники по возрастанию приоритета: файл `doqa.properties` → переменные окружения `DOQA_*` → опции pytest `--doqa-*` (у каждой есть ini-двойник `doqa_*` в `pytest.ini` / `[tool.pytest.ini_options]`; опция командной строки сильнее ini-ключа). Путь к файлу конфигурации переопределяется `--doqa-config=…` или `DOQA_CONFIG`; файл читается в UTF-8.

properties

```
url=https://demo.doqa.app
token=<project-токен>
spaceId=42
```

С этими тремя ключами адаптер переключается в API-режим: сам создаёт прогон и наполняет его. По умолчанию — одной порцией в конце, в realtime-режиме — по ходу выполнения. Как создать токен и узнать ID пространства — в разделах «Создание API-токена» и «Как узнать ID пространства».

Внимание

Токен в конфиге означает, что каждый запуск тестов пишет в DoQA, включая локальные. Обычная схема: локально конфига нет (результаты остаются файлами и никуда не отправляются), а в CI ключи приходят из переменных окружения `DOQA_URL` / `DOQA_TOKEN` / `DOQA_SPACE_ID`.

Все ключи

Ключ (`doqa.properties`)	Переменная окружения	Опция pytest	Дефолт
`reporting`	`DOQA_REPORTING`	`--doqa-reporting`	`auto`
`resultsDir`	`DOQA_RESULTS_DIR`	`--doqa-results-dir`	`results`
`url`	`DOQA_URL`	`--doqa-url`	—
`token` (алиас `privateToken`)	`DOQA_TOKEN` (алиас `DOQA_PRIVATE_TOKEN`)	`--doqa-token`	—
`spaceId` (алиас `projectId`)	`DOQA_SPACE_ID` (алиас `DOQA_PROJECT_ID`)	`--doqa-space-id`	—
`configurationId`	`DOQA_CONFIGURATION_ID`	`--doqa-configuration-id`	—

Ключ (`doqa.properties`)	Переменная окружения	Опция pytest	Дефолт
`testRunId`	`DOQA_TEST_RUN_ID`	`--doqa-test-run-id`	—
`testRunName`	`DOQA_TEST_RUN_NAME`	`--doqa-test-run-name`	—
`adapterMode`	`DOQA_ADAPTER_MODE`	`--doqa-adapter-mode`	`2`
`importRealtime`	`DOQA_IMPORT_REALTIME`	`--doqa-import-realtime`	`false`
`certValidation`	`DOQA_CERT_VALIDATION`	`--doqa-cert-validation`	`true`
`proxy`	`DOQA_PROXY`	`--doqa-proxy`	—
`pipelineId`	`DOQA_PIPELINE_ID`	`--doqa-pipeline-id`	авто: `CI_PIPELINE_ID` / `GITHUB_RUN_ID`
`ciRunId`	`DOQA_CI_RUN_ID`	`--doqa-ci-run-id`	—
`branch`	`DOQA_BRANCH`	`--doqa-branch`	авто: `CI_COMMIT_REF_NAME` / `GITHUB_REF_NAME`
`batchSize`	`DOQA_BATCH_SIZE`	`--doqa-batch-size`	`100`
`requestTimeoutMs`	`DOQA_REQUEST_TIMEOUT_MS`	`--doqa-request-timeout-ms`	`30000`
`retries`	`DOQA_RETRIES`	`--doqa-retries`	`3`

Ключ (`doqa.properties`)	Переменная окружения	Опция pytest	Дефолт
`retryBackoffMs`	`DOQA_RETRY_BACKOFF_MS`	`--doqa-retry-backoff-ms`	`500`
`maxTraceLength`	`DOQA_MAX_TRACE_LENGTH`	`--doqa-max-trace-length`	`100000`
`maxMessageLength`	`DOQA_MAX_MESSAGE_LENGTH`	`--doqa-max-message-length`	`10000`
`maxParameterLength`	`DOQA_MAX_PARAMETER_LENGTH`	`--doqa-max-parameter-length`	`2000`

Явно пустое значение опции (`--doqa-pipeline-id=`) очищает поле из нижнего слоя; пустые переменные окружения игнорируются; значение-нераскрытая ссылка (например `$DOQA_TOKEN`) считается незаданным и подсвечивается предупреждением «unexpanded variable reference».

`pipelineId` и `branch` подхватываются автоматически из стандартных переменных GitLab и GitHub Actions. Прогон в DoQA привязывается к пайплайну и ветке без настройки.

Куда уходят результаты: `reporting`

Значение	Что происходит
`auto` (дефолт)	есть `url` + `token` + `spaceId` — API; нет — файлы, и в лог выводится предупреждение с перечислением недостающих настроек
`api`	только API (без конфига — предупреждение в лог с перечислением недостающих настроек)
`files`	только файлы Allure-совместимого формата в `resultsDir`
`off`	адаптер выключен полностью

Режимы работы с прогоном: `adapterMode`

- `2` / **new** (дефолт) — адаптер сам создаёт прогон и отправляет всё в него.
- `1` / **existing** — всё отправляется в существующий прогон `testRunId`. Если `testRunId` задан, а `adapterMode` не задан явно, адаптер сам работает в этом режиме: указанный прогон никогда не игнорируется молча.
- `0` / **selective** — адаптер запрашивает у DoQA, какие автотесты числятся в прогоне `testRunId`, и деселектит невыбранные ещё на сборе коллекции (честный `pytest_deselected`). Этим режимом DoQA перезапускает выбранные тесты при запуске из интерфейса.

При запуске пайплайна из DoQA переменные `DOQA_TEST_RUN_ID`, `DOQA_ADAPTER_MODE`, `DOQA_CI_RUN_ID` передаются пайплайну автоматически. Адаптер читает их как обычные настройки.

В режиме 0 тесты исполняются в порядке плана автоматически. Включать orderer'ы, как у JUnit 5, не нужно; тесты вне плана уходят в хвост. Шаблонные ID с плейсхолдером (`login_{browser}`) матчатся по wildcard на сборе: в план попадают все инвокации, чей раскрытый id в нём есть, точный отбор — при отправке результата.

Разметка тестов

Вся разметка — декораторы модуля `doqa`, опциональна:

```python
import doqa


@doqa.label("regression")                       # класс-уровень: наследуется всеми тес
class TestLogin:

    @doqa.id("DOQA-42")                         # стабильный внешний ID автотеста (рек
    @doqa.title("Успешный вход")
    @doqa.description("Проверяет happy-path входа по паролю")
    @doqa.display_name("Вход по паролю")        # имя автотеста (иначе — имя функции)
    @doqa.label("smoke")                        # объединится с класс-уровнем
    @doqa.label("owner", "qa-team")             # key:value-метка
```

```
    @doqa.tag("ui")
    @doqa.link.defect("https://tracker/BUG-77", title="флак на CI")
    @doqa.case_ids(1041)                          # привязка к тест-кейсам DoQA (можно н
    @doqa.create_manual_case                      # завести связанный ручной кейс
    def test_login_happy_path(self):
        ...
```

Ссылки: универсальный `@doqa.link(url, type=…, title=…, description=…)` и шорткаты `doqa.link.related` / `.defect` / `.requirement` / `.blocked_by` / `.repository`. Место теста в дереве каталога: `@doqa.namespace` (по умолчанию — dotted-путь модуля) и `@doqa.class_name` (по умолчанию — класс или модуль).

`@doqa.id` **ставится только на функции**: на классе он схлопнул бы все его тесты в один автотест, такое применение отклоняется исключением `TypeError`. Остальные декораторы работают и на классе.

Обычные no-arg марки pytest (`@pytest.mark.smoke`) автоматически попадают в теги автотеста. Двойная разметка не нужна. Структурные марки (`skipif`, `parametrize`) в счёт не идут.

Если `@doqa.id` нет, идентификатор ищется в таком порядке: `[DOQA-123]` или `@DOQA:123` в display name → Allure `allure.id` (читается без зависимости от Allure; ключ `ALLURE-<id>`) → детерминированный хэш полного имени теста (модуль + класс + функция, без параметров инвокации). История автотеста стабильна в любом случае; явный ID делает её устойчивой ещё и к переименованиям.

Параметризованные тесты

По умолчанию все инвокации одного теста сворачиваются в **один** автотест, а аргументы попадают в `parameters[]` результата. Чтобы получить отдельный автотест на каждую инвокацию, используйте плейсхолдер `{имя_аргумента}` в любом декораторе:

```python
@pytest.mark.parametrize("browser", ["chrome", "firefox"])
@doqa.id("LOGIN-IN-{browser}")                    # → LOGIN-IN-chrome, LOGIN-IN-firefox
@doqa.title("Вход в {browser}")
```

```
def test_login_in(browser):
    ...
```

Нераспознанный плейсхолдер остаётся в тексте как есть. Значения параметров усекаются по `maxParameterLength` с маркером `… truncated (N chars)`.

Шаги

`doqa.step` — одновременно контекст-менеджер и декоратор, вложенность без ограничений:

```python
with doqa.step("открыть страницу логина"):
    page.open()
    with doqa.step("ввести пароль", "форма из конфига"):   # второй аргумент — описани
        page.type_password(password)


@doqa.step("авторизоваться под {user}")          # {param}-плейсхолдеры из аргументов ф
def authorize(user):
    ...


@doqa.step                                       # без текста — имя функции
def prepare_cart():
    ...
```

Упавший шаг закрывается с исходом ошибки (`failed` для assert и `pytest.fail`, `broken` для прочих исключений); само исключение летит дальше по тесту.

> **Внимание**
>
> Работают только формы `with doqa.step(...)` и `@doqa.step`. Голый вызов `doqa.step("...")` без `with` шаг не создаёт.

Рантайм-API из тела теста

```python
doqa.add_parameter("env", "staging")
doqa.attach_file("artifacts/screenshot.png")    # файл к тесту или открытому шагу
doqa.attach(response_bytes, name="response.json", content_type="application/json")
doqa.attach(log_text, name="app.log")           # строка → text/plain
doqa.add_link("https://tracker/TASK-5", type="requirement")
doqa.add_message("покупатель создан через фабрику")
doqa.add_case_ids(1042)
doqa.add_create_manual_case()
doqa.add_external_id("CART-DYN-1")              # пин id текущей инвокации (сильнее де
doqa.add_title("…"); doqa.add_description("…"); doqa.add_display_name("…")
doqa.add_labels("…"); doqa.add_tags("…")
doqa.add_label("severity", "critical")          # key:value-метка
```

Все вызовы безопасны вне активного теста и при `reporting=off`: просто ничего не делают. Работают и внутри фикстур. Для своих потоков (async-код, свои executor'ы) контекст переносится явно:

```python
ctx = doqa.capture_context()
executor.submit(lambda: doqa.run_with(ctx, lambda: worker_check()))
```

Фикстуры

Без дополнительных флагов, в отличие от JUnit 5, которому для этого нужен автодетект расширений:

- **function-scoped** — блоки setup/teardown в каждом результате;
- **class- и module-scoped** — общие фикстуры у всех тестов своего класса/модуля;
- **session- и package-scoped** — у всех тестов прогона;
- финализаторы (`yield`-часть и `addfinalizer`) — блоки teardown;
- `doqa.step` / `doqa.attach*` внутри фикстуры вкладываются в её узел;
- имя узла — имя фикстуры; `@allure.title` на фикстуре подхватывается.

Служебные фикстуры самого pytest (`tmp_path`, `capsys` и подобные) в отчёт не попадают.

Маппинг исходов

Что случилось	Исход в DoQA
Тест прошёл	`passed`
Упала проверка (`AssertionError` , `pytest.fail`)	`failed`
Любое другое исключение (инфраструктура, таймаут)	`broken`
`pytest.skip` / `skipif`	`skipped`
`xfail` сработал (ожидаемое падение)	`skipped` , в сообщении `XFAIL <reason>` и реальная ошибка
`xfail` не сработал (XPASS)	`passed` , в сообщении `XPASS <reason>` ; при `strict` — `failed`

Падение teardown-фикстуры вливается в результат теста: сообщения складываются, инфраструктурный `broken` перевешивает `failed` тела.

Доставка результатов и pytest-xdist

По умолчанию результаты копятся и отправляются порциями по `batchSize` (100 по умолчанию) в конце прогона; при сбое одной порции теряется только она. При `importRealtime=true` порция уходит на каждый завершённый **модуль**, вместе с его teardown-фикстурами. Остаток буфера отправляется в конце, в том числе обработчиком завершения процесса (shutdown hook).

pytest-xdist поддерживается из коробки: в режиме 2 прогон создаёт контроллер (один раз) и передаёт его воркерам через handshake xdist; создание прогона несёт идемпотентный ключ, повтор запроса при сетевом сбое не оставит два прогона. В режимах 0 и 1 прогон зафиксирован конфигом заранее. Порядок плана в параллельном прогоне не гарантируется.

`pytest --collect-only` сервер не трогает и прогон не создаёт.

Особенности и ограничения

Чем pytest отличается от **адаптера JUnit 5**:

- **Фикстуры без флагов.** JUnit 5 требует включить автодетект расширений, чтобы получить setup/teardown-блоки; pytest пишет все уровни фикстур сразу после установки, без дополнительной настройки.
- **Шаги без агента.** JVM-адаптерам для декларативных шагов нужен AspectJ-агент в тестовой JVM; у pytest `@doqa.step` и `with doqa.step(...)` работают сами по себе: байткод-инструментирование не требуется.
- **Порядок в селективном режиме включается сам.** JUnit 5 требует вручную прописать `DoqaPlanClassOrderer` / `DoqaPlanMethodOrderer` в `junit-platform.properties`; pytest в режиме 0 исполняет тесты по плану автоматически.
- **`@doqa.id` жёстко ограничен уровнем функции**: на классе декоратор выбрасывает `TypeError`; у JVM-адаптеров `@DoqaId` можно ставить и на класс.
- **В опубликованной версии 0.1.1 нет деградации в файлы при отказе DoQA.** JVM-адаптеры версии 0.1.6 при недоступности DoQA на старте прогона переключаются в файловый режим (`resultsDir`) на весь прогон. Адаптер pytest 0.1.1 так не умеет: сборку он не роняет, но результаты, которые не удалось отправить, теряются: в файлы они не переливаются, в логе остаётся предупреждение («could not establish run» при отказе на старте, «results chunk failed» посреди прогона). Деградация в файлы появилась в версии 0.1.2, которая пока не опубликована.

Отдельно — ограничение, общее с JVM-адаптерами: ключа CI-подключения в настройках адаптера нет. Переменную `DOQA_SOURCE_KEY`, которую DoQA передаёт в пайплайн (см. **переменные пайплайна**), адаптер не читает и не отправляет. Пока у пространства одно CI-подключение, это ни на что не влияет; если подключений несколько, а токен не привязан к конкретному из них, приём отклоняет результаты с кодом 409 и перечисляет известные ключи подключений.

Смотрите также

- Адаптеры для тестовых фреймворков — когда адаптер, а когда отчёты
- Запуск автотестов из DoQA — как формируется прогон для селективного режима
- Первый прогон автотестов — сценарий первой интеграции с DoQA

Переезд с Allure

Если тесты уже размечены Allure-аннотациями, а связи с тест-кейсами держатся на Allure ID, **адаптер** подхватывает эту разметку как есть, без Allure-зависимости в classpath и без правок кода. Разметка `@Doqa*` — основной путь; Allure-мост существует, чтобы переехать на неё постепенно, тест за тестом, ничего не сломав по дороге.

Что работает без правок кода

Что у вас есть	Что делает DoQA
Архивы `allure-results`	принимаются как раньше — **Отправка результатов**
Метка `as_id` (`allure_id`) в отчёте	привязка автотеста к тест-кейсу один-к-одному; новая привязка этого автотеста к другому кейсу снимает прежнюю — **Связь с тест-кейсами**
`@AllureId` на тесте или классе	всегда уходит меткой `as_id` (привязка к кейсу) и участвует в выводе внешнего ID — см. ниже
`@Epic` / `@Feature` / `@Story` / `@Owner` / `@Severity`	становятся метками `epic:…` , `feature:…` , `story:…` , `owner:…` , `severity:…`
`@Description`	описание автотеста — если нет `@DoqaDescription`
`@Link` , `@Links` , а также `@Issue` и `@TmsLink` со значением-URL	типизированные ссылки: issue — дефект, tms — требование
Пайплайн, собирающий Allure Report	продолжает работать: файловый режим адаптера пишет Allure-совместимые артефакты

Приоритет у моста односторонний: родные `@Doqa*` -аннотации всегда выигрывают, Allure только заполняет пробелы.

Чем станет внешний ID

Без явного `@DoqaId` адаптер выводит **внешний ID** по строгому приоритету: маркер в имени → `@AllureId` → хэш сигнатуры. Из `@AllureId` внешний ID получается с префиксом: `@AllureId("123")` даёт `ALLURE-123`. Сам Allure ID при этом не расходуется: он отдельно уходит меткой `as_id` и делает привязку к тест-кейсу.

Ключ `ALLURE-…` стабилен: он не зависит ни от имени теста, ни от класса. Жить на таких ключах можно сколько угодно долго.

Как DoQA сопоставляет результат с автотестом

1. Точное совпадение внешнего ID в пространстве: результат попадает в эту запись каталога.
2. Иначе — совпадение по полному имени: если у найденной записи ещё нет своего ключа, присланный ключ закрепляется за ней и история сохраняется; если ключ уже есть и он другой, запись остаётся со старым ключом, а результат всё равно попадает в неё.
3. Иначе создаётся новая запись.

Практические следствия:

- **Отчитывались отчётами без ID и добавили** `@DoqaId`. История переходит к новому ключу. Не меняйте имя теста и ID в одном коммите: чтобы найти нужную запись, DoQA ищет её по прежнему имени.
- **Уже работали с адаптером на ключах** `ALLURE-…` **и добавили** `@DoqaId`. Ключ записи не переписывается: результаты продолжают попадать в неё по имени, а внешний ID остаётся `ALLURE-…`. Свои ключи задавайте новым тестам до их первого прогона; за старыми надёжнее закрепить фактический ключ явно: `@DoqaId("ALLURE-123")`. После этого тест можно свободно переименовывать.
- **Ничего не менять** — тоже рабочий вариант: идентичность на `ALLURE-…`, привязка через `as_id`.

Переезд на разметку @Doqa*

Эти замены безопасны даже массовой автозаменой по проекту. На идентичность автотеста они не влияют:

Было (Allure)	Стало (DoQA)
`@Description("…")`	`@DoqaDescription("…")`
`@Epic("Оплата")`	`@DoqaLabels({"epic:Оплата"})`
`@Feature("…")`, `@Story("…")`	`@DoqaLabels({"feature:…", "story:…"})`
`@Owner("…")`, `@Severity(CRITICAL)`	`@DoqaLabels({"owner:…", "severity:critical"})`
`@Link`, `@Issue`, `@TmsLink`	`@DoqaLink` / `@DoqaLinks`

Полный справочник разметки — Адаптер JUnit 5.

Две замены требуют внимания:

Идентичность и привязка — не автозаменой

- `@AllureId` → `@DoqaId` — смена источника идентичности; последствия описаны в разделе выше. Для тестов с накопленной историей закрепляйте фактический ключ (`@DoqaId("ALLURE-123")`), а не выдумывайте новый.
- Привязка через Allure ID сохраняется: адаптер передаёт `@AllureId` в DoQA и в файловом режиме, и при отправке напрямую в API, связь 1:1 с тест-кейсом с этим id продолжает создаваться. Для привязки к нескольким кейсам используйте `@DoqaCaseIds({…})`. Если вы снимаете `@AllureId`, уже созданные связи не рвутся, но новые этим способом перестанут создаваться.

Смотрите также

- Адаптер JUnit 5 — установка, конфигурация, полный справочник аннотаций
- Автотесты — внешний ID и способы привязки к тест-кейсам
- Отправка результатов автотестов — приём готовых allure-results

Плагины для IDE

Плагин DoQA показывает прямо в редакторе, что DoQA знает о каждом тесте: результат последнего прогона, стабильность, карантин и привязанные **тест-кейсы**. Из IDE можно привязать кейс к тесту, закрепить внешний ID, разобрать падение по стеку и поставить автотест в карантин, не переключаясь в браузер.

Плагин работает поверх адаптеров DoQA: он пишет в код ту же разметку, которую читает **адаптер**, и вычисляет тот же внешний ID. Поэтому статус в редакторе совпадает с тем, что DoQA получает из CI.

Что дают плагины

Возможность	Что видно или что происходит
Статус теста в gutter	значок у каждого тестового метода: последний прогон, карантин, нестабильность, привязанный кейс; в подсказке: стабильность, полоса последних прогонов и карточка кейса
История прогонов	последние прогоны автотеста; у проваленного или сломанного прогона есть сообщение и стек, строки стека ведут в код
Кейсы в IDE	карточка кейса по Ctrl+клику, поиск и привязка кейса с подсказкой похожих, отвязка, черновик теста по ручному кейсу
Внешний ID	закрепление `@DoqaId` (в pytest это `@doqa.id`, в Jest ключ `id` в метаданных `doqa.test`) у одного теста или во всём файле, чтобы автотест не терял историю при переименовании
Карантин	поставить автотест на **карантин** или снять с него
Каталог автотестов	список автотестов пространства с быстрым поиском, как **каталог** в веб-интерфейсе
Помощь в редакторе	дополнение тегов и меток из каталога, предупреждение о повторяющемся внешнем ID, проверка `doqa.properties`

Интерфейс плагина на русском языке.

Поддерживаемые IDE и тесты

IDE. IntelliJ IDEA, PyCharm, WebStorm и другие IDE на платформе IntelliJ версии 2023.3 и новее, а также VS Code 1.85 и новее. Функции у плагинов одни и те же. Плагин для IntelliJ описан на страницах **Работа с тестами в IntelliJ** и **Помощь в редакторе и черновики тестов**, плагин для VS Code на странице **Плагин для VS Code**.

Тесты. Те же, для которых есть адаптеры DoQA:

Язык	Фреймворк	Адаптер
Java, Kotlin	JUnit 5	**Адаптер JUnit 5**
Java, Kotlin	JUnit 4	**Адаптер JUnit 4**
Java, Kotlin	TestNG	**Адаптер TestNG**
Python	pytest	**Адаптер pytest**
JavaScript, TypeScript	Jest	`@doqa-tms/jest`, см. **README адаптера Jest** ↗

В файлах Java и Kotlin фреймворк плагин определяет по импорту аннотации `Test`. Файлы `.py` он читает как тесты pytest, а файлы `.js`, `.cjs`, `.mjs`, `.jsx`, `.ts`, `.cts`, `.mts` и `.tsx` как тесты Jest. Как плагин находит проект Jest и какие файлы считает тестовыми, описано в разделе **Тесты Jest**.

Значки в gutter у тестов на Java и Kotlin показывает IntelliJ IDEA, у тестов pytest PyCharm и IDE с плагином Python, у тестов Jest IntelliJ IDEA Ultimate, WebStorm и другие IDE с плагином JavaScript. Действия меню работают в любой IDE.

Установка

Скачайте плагин для своей IDE. Текущая версия: 0.3.1.

IDE	Файл
IntelliJ IDEA, PyCharm, WebStorm и другие IDE на платформе IntelliJ	**doqa-ide-intellij.zip**

IDE	Файл
VS Code	doqa-ide-vscode.vsix

IntelliJ IDEA и PyCharm

1. Откройте **Settings** → **Plugins**.
2. Нажмите на значок над списком плагинов и выберите «**Install Plugin from Disk...**».
3. Укажите скачанный ZIP-архив, распаковывать его не нужно. Перезапустите IDE, если она попросит.

После установки появляются кнопка DoQA в главном тулбаре и пункт **DoQA** в меню **Tools**.

Новая версия ставится так же, поверх прежней.

VS Code

1. Откройте панель **Extensions** (Ctrl+Shift+X).
2. Нажмите «...» в заголовке панели и выберите «**Install from VSIX...**».
3. Укажите скачанный файл `.vsix`.

То же из терминала: `code --install-extension doqa-ide-vscode.vsix`.

Настройка

Плагину нужны адрес DoQA, персональный API-токен и пространство (в интерфейсе плагина оно называется space), в котором лежат автотесты и кейсы.

Токен

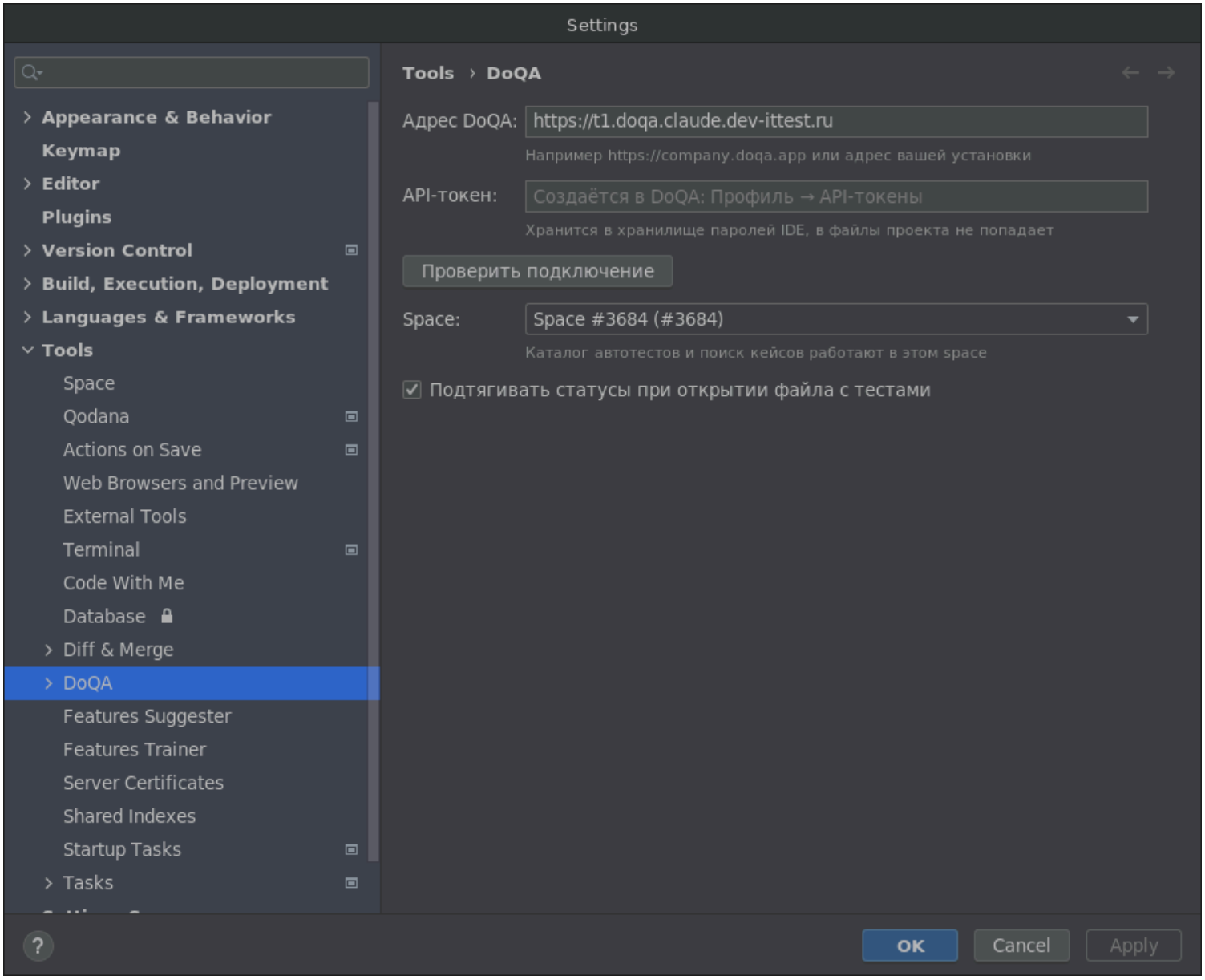
Плагин работает с персональным API-токеном из профиля: имя в правом верхнем углу → «**Профиль**» → раздел «API-токены» → «**Создать API-токен**». Токен начинается с `doqa_` и показывается один раз, скопируйте его сразу. Подробно: **API-токены**.

Внимание

Токен проекта для загрузки результатов из вкладки «API-токены» в настройках администратора здесь не подойдёт: плагин входит в DoQA от вашего имени и видит те пространства, к которым есть доступ у вас.

Подключение к DoQA

Откройте **Settings → Tools → DoQA**.



Поле	Что указать
«Адрес DoQA»	адрес облака вашей компании или своей установки DoQA, начинается с `https://`
«API-токен»	персональный токен из профиля

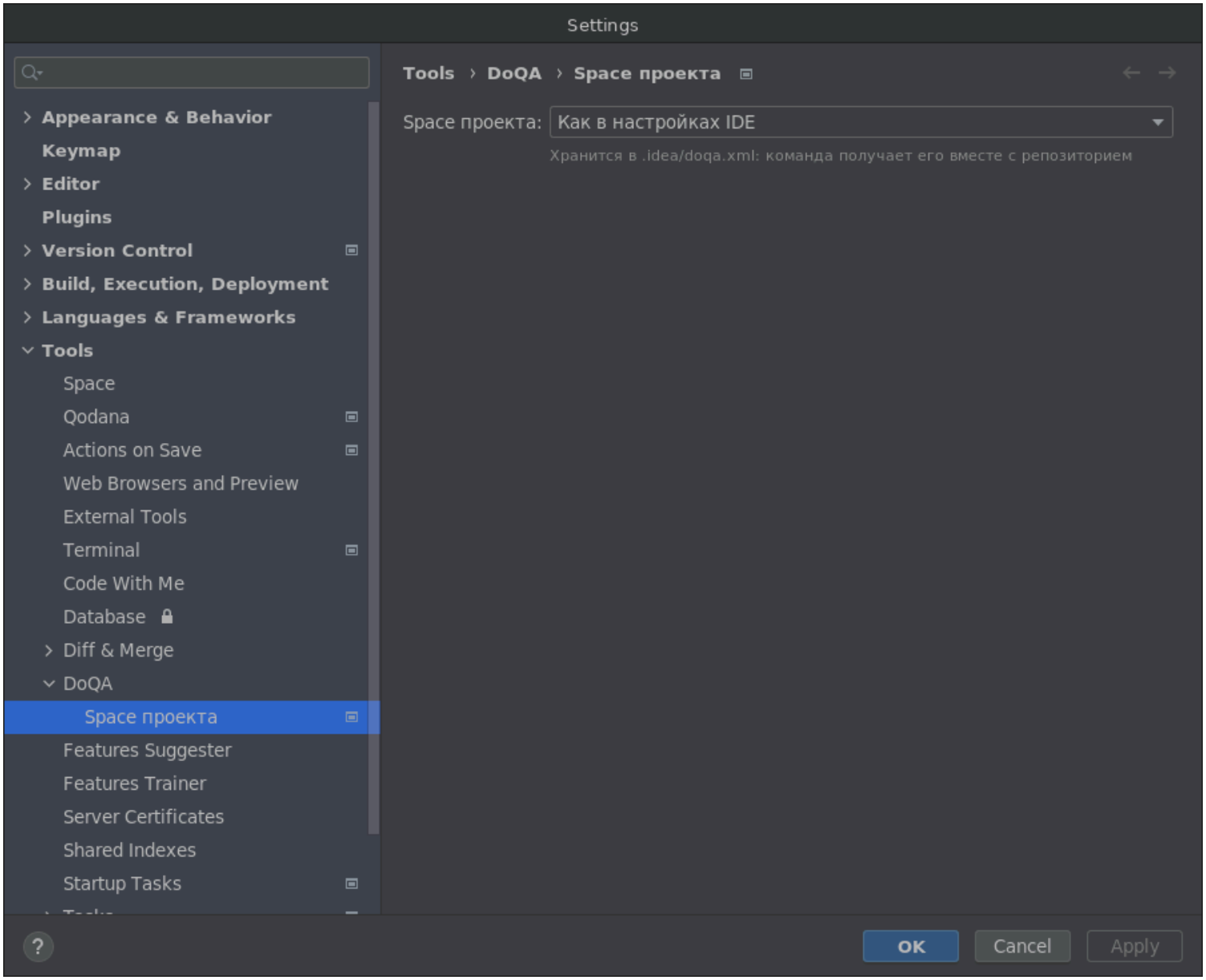
Поле	Что указать
«Проверить подключение»	входит в DoQA с введёнными значениями ещё до сохранения. При успехе рядом появляется «Вход выполнен: <почта>», а список «Space» заполняется пространствами, доступными вам
«Space»	пространство DoQA по умолчанию: в нём работают каталог автотестов и поиск кейсов
«Подтягивать статусы при открытии файла с тестами»	включено по умолчанию: при открытии файла плагин в фоне загружает статусы его тестов из DoQA

Если доступных пространств нет, после проверки появится «Вход выполнен (<почта>), но доступных space нет». Попросите администратора добавить вас в нужное пространство.

Сохраните настройки кнопкой **OK** или **Apply**. После смены адреса или пространства плагин сбрасывает статусы, загруженные раньше, и загружает их заново.

Space проекта и модулей

Кроме пространства по умолчанию для всей IDE, пространство задаётся для проекта, а в проекте из нескольких модулей ещё и для каждого модуля: **Settings → Tools → DoQA → Space проекта**.



- «Space проекта» задаёт пространство для всего проекта. Значение «Как в настройках IDE» берёт пространство со страницы **Tools → DoQA**.
- Таблица «Модули, которые отчитываются в другой space» появляется, если в проекте больше одного модуля. Значение «Как у проекта» оставляет модулю пространство проекта.

Файл с тестами работает с пространством своего модуля, если оно задано, иначе с пространством проекта, иначе с пространством из настроек IDE.

Выбор хранится в файле `.idea/doqa.xml`. Добавьте его в репозиторий, и вся команда получит те же пространства вместе с кодом. Секретов в этом файле нет: только ID пространства проекта и пространств модулей. Адрес DoQA и токен в него не попадают, их каждый указывает в своей IDE.

Баннер «DoQA не настроен»

Пока адрес, токен или пространство не заданы, над файлами с тестами висит баннер «DoQA не настроен: не видно, какие тесты привязаны к ручным кейсам». Ссылка «Настроить» открывает страницу настроек, «Не показывать» скрывает баннер.

Действия, которым нужен DoQA, без настройки не выполняются и показывают уведомление «DoQA не настроен: укажите адрес, токен и space.» со ссылкой на настройки.

Где хранится токен

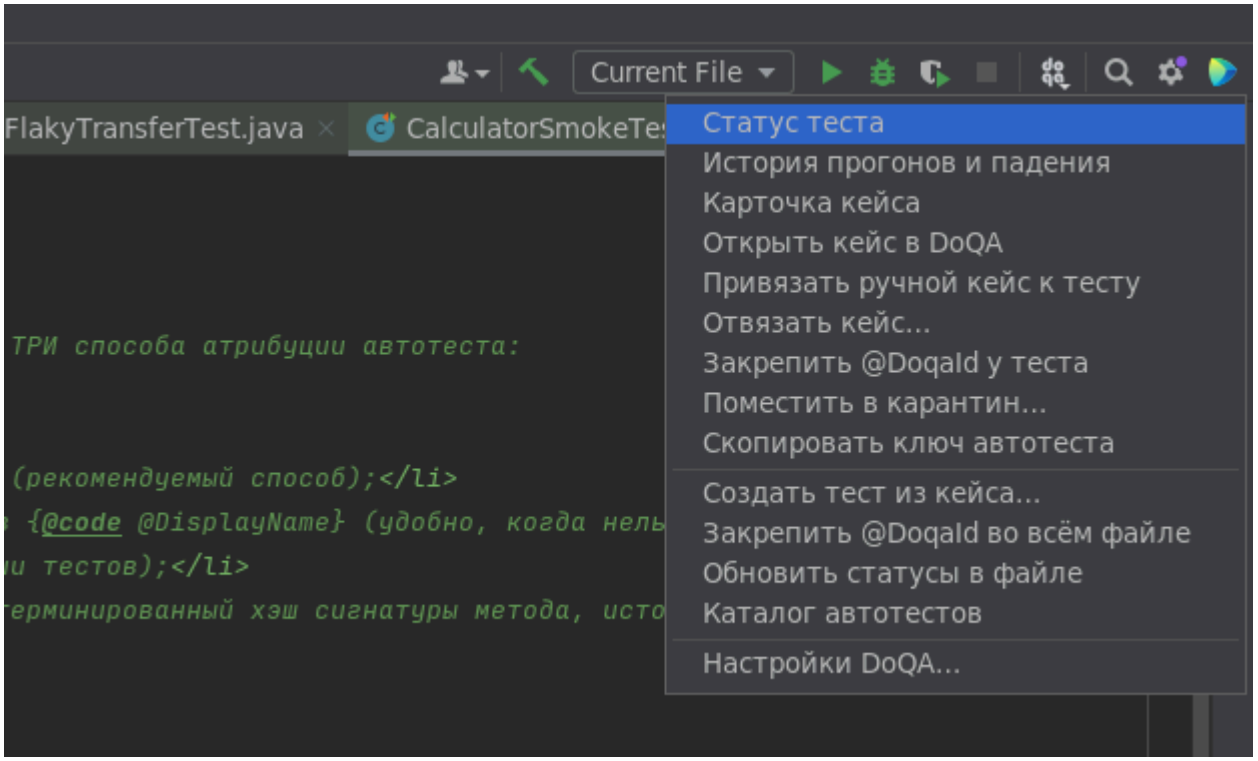
Токен хранится в хранилище паролей IDE, то есть в ключнице операционной системы, и привязан к адресу DoQA. В файлы проекта он не попадает. После сохранения поле токена остаётся пустым с подсказкой «Токен сохранён. Введите новый, чтобы заменить».

Если хранилище недоступно (например, Linux без Secret Service), плагин покажет ошибку «Не удалось сохранить токен DoQA в хранилище паролей IDE. Выберите другое хранилище в Settings → Appearance & Behavior → System Settings → Passwords и введите токен снова.»

Где найти действия

Все действия собраны в меню DoQA. Открыть его можно четырьмя способами:

- кнопкой DoQA в главном тулбаре: в новом интерфейсе она справа, перед кнопкой поиска, в классическом она на панели навигации;
- сочетанием **Ctrl+Alt+Shift+D**: меню откроется у курсора, в нём работает поиск по названию действия;
- через **Tools → DoQA**;
- из контекстного меню редактора, пункт **DoQA**.



Действия над одним тестом также есть в меню по клику на значок в gutter и в меню **Alt+Enter** на строке теста. Действия, недоступные для теста под курсором, в меню кнопки неактивны, а в контекстном меню редактора не показываются. Например, «Открыть кейс в DoQA» доступно, только если к тесту привязан кейс.

Что дальше

- **Работа с тестами в IntelliJ**: значки в gutter, история прогонов, привязка, карточка и отвязка кейсов, внешний ID, карантин, каталог автотестов, тесты Jest.
- **Помощь в редакторе и черновики тестов**: черновик теста из кейса, дополнение тегов, проверка `doqa.properties`, повторяющийся внешний ID, ограничения плагина.
- **Плагин для VS Code**: настройка в VS Code, CodeLens над тестами, меню действий, space для папки и отличия от плагина для IntelliJ.

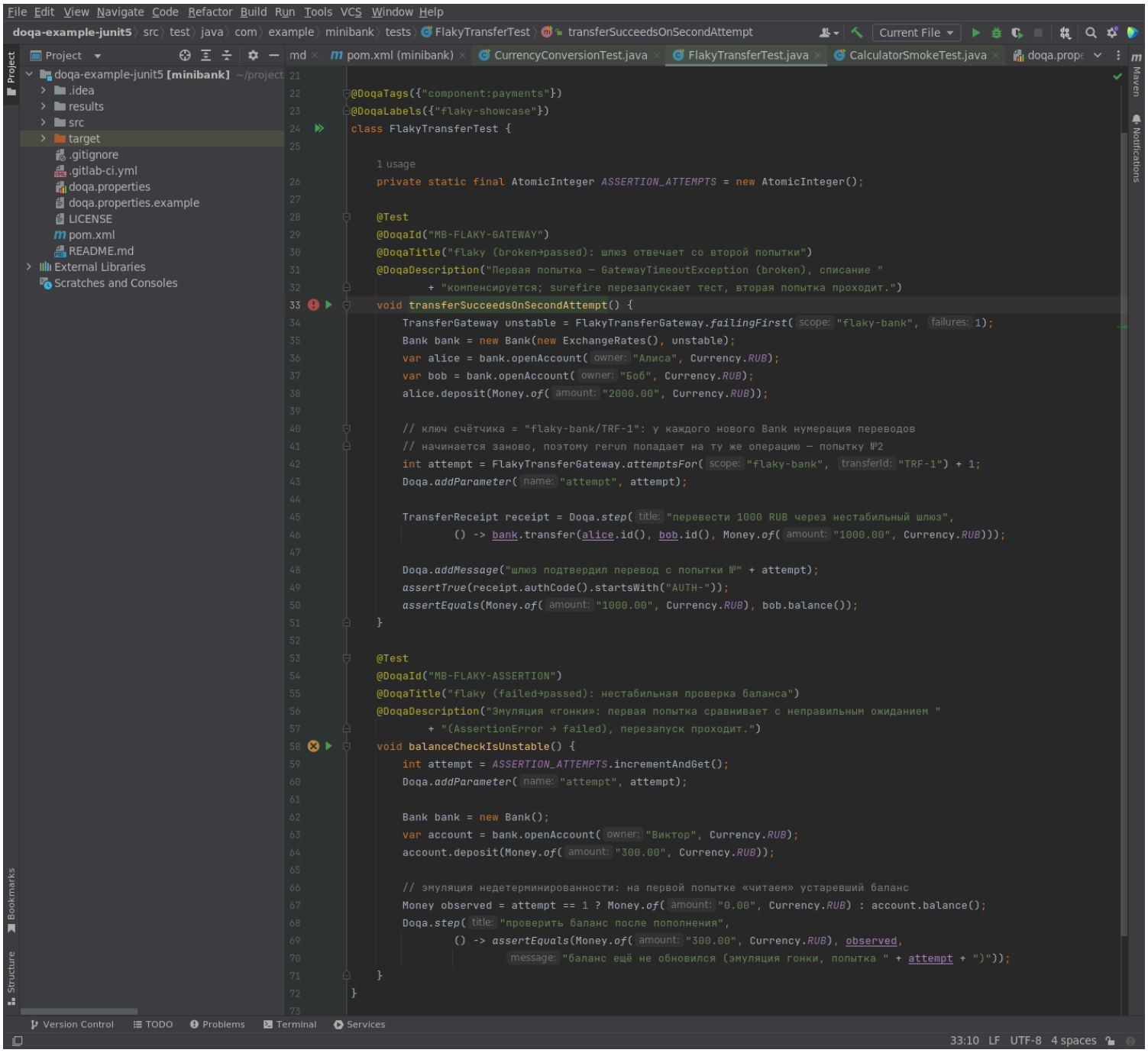
Смотрите также

- **Автотесты**: внешний ID и связь автотестов с тест-кейсами
- **Адаптеры для тестовых фреймворков**: как результаты попадают в DoQA из тестового процесса
- **Каталог автотестов**: тот же каталог в веб-интерфейсе

- **Нестабильные тесты и карантин**: что значит карантин и нестабильность

Работа с тестами в IntelliJ

Плагин DoQA показывает прямо в редакторе, как автотест прошёл в последнем прогоне, насколько он стабилен и к какому **тест-кейсу** привязан. Из IDE можно привязать и отвязать кейс, закрепить внешний ID, посмотреть историю прогонов со стеком падения и поставить автотест в карантин. Эта страница о работе с тестами; установка и настройка подключения описаны на странице **Плагины для IDE**, а подсказки при наборе кода описаны на странице **Помощь в редакторе и черновики тестов**.



Значки в gutter

У каждого тестового метода слева от объявления стоит значок DoQA. Он показывает исход последнего прогона автотеста в DoQA:

Значок	Что значит
зелёная галочка теста	последний прогон пройден
оранжевый кружок с крестиком	последний прогон провален
красный кружок с «!»	последний прогон сломан или заблокирован
значок пропуска	последний прогон пропущен
пауза	автотест в карантине, исход последнего прогона значок не показывает
значок неизвестного результата	DoQA вернул исход, которого плагин не знает

К основному значку добавляются пометки:

- **предупреждающий знак**: DoQA считает автотест нестабильным (у автотеста в карантине пометки нет);
- **маленькая стрелка**: к тесту привязан кейс.

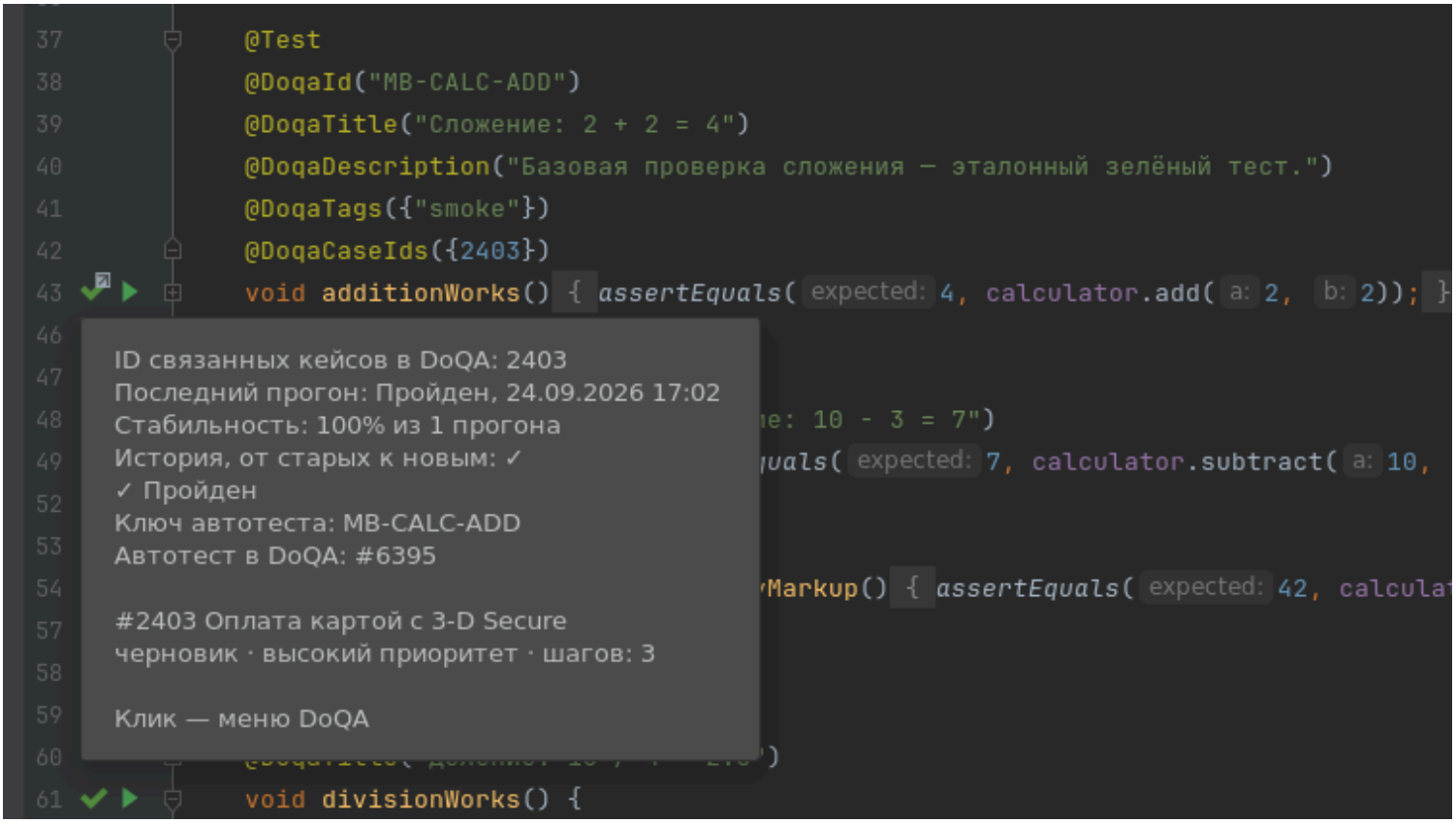
Тест, который ещё ни разу не запускался или которого нет в каталоге, показывает только привязку: галочку, если кейс привязан, и значок , если не привязан.

Состояния стабильности и карантин считает DoQA, а не плагин: как они определяются, описано на странице **Нестабильные тесты и карантин**.

Подсказка значка

При наведении на значок появляется подсказка:

Строка	Пример и смысл
первые строки	связанные кейсы, см. **Связанные кейсы в подсказке**
«Последний прогон:»	исход, дата и время, ветка; «новый сбой», если до этого тест не падал. У теста без прогонов: «Ещё не запускался»
«Стабильность:»	доля пройденных прогонов, например «100% из 21 прогона». У нестабильного и других особых автотестов через «·» добавляется состояние, см. **Статусы как в DoQA**
«История, от старых к новым:»	полоса последних прогонов, по значку на прогон; строкой ниже расшифровка значков, см. **Статусы как в DoQA**
«В карантине: падения не влияют на гейт»	только у автотеста в карантине
«Ключ автотеста:»	внешний ID теста; если плагин не может его вычислить, вместо него «вычислит адаптер» и причина в скобках
«Автотест в DoQA: #...»	номер автотеста в каталоге
карточка кейса	номер и название первого привязанного кейса, его статус, приоритет и число шагов; «Ещё кейсов: N», если привязано несколько
«Клик — меню DoQA»	по клику на значок открывается **меню действий** над этим тестом



Связанные кейсы в подсказке

Первая строка подсказки говорит, какие кейсы связаны с автотестом в DoQA. Если связи в DoQA и ID в `@DoqaCaseIds` (`doqa.case_ids` в pytest) расходятся, плагин показывает и то и другое:

Ситуация	Что написано
в DoQA есть связи	«ID связанных кейсов в DoQA: 15, 16»
связей в DoQA нет, но ID кейсов стоят в `@DoqaCaseIds`	«ID кейсов в @DoqaCaseIds: 15 (в DoQA ещё не привязаны, привяжутся при следующем прогоне)»
связи есть и там и там, но ID различаются	первая строка про DoQA, ниже отдельной строкой «В @DoqaCaseIds: …»
связей нет нигде	«Связанных кейсов в DoQA нет»

Кейсы из `@DoqaCaseIds` DoQA привяжет сам при следующем прогоне теста: адаптер передаёт их вместе с результатом. Кейсы, которых нет в пространстве, не привязываются и не создаются.

В тестах Jest вместо `@DoqaCaseIds` в этих строках написано, где стоят ID: `caseIds` (метаданные `doqa.test`), `doqa.metadata` (вызов в теле теста) или `caseIds и doqa.metadata`, например «ID кейсов в doqa.metadata: 2401». Подробнее в разделе **Кейс в doqa.metadata**.

Если плагин не может сопоставить тест с автотестом в DoQA, ниже идёт отдельная строка с причиной:

- «Автотеста ещё нет в каталоге DoQA: он появится после первого прогона»;
- «Каталог DoQA ещё не проверен: связи в DoQA неизвестны», например пока каталог грузится или не выбрано пространство;
- «Ключ автотеста вычислит адаптер при запуске, поэтому связи в DoQA не проверить»;
- «Ключ с плейсхолдером: у каждого набора параметров свой автотест, связи в DoQA не проверить», если в ключе есть плейсхолдер (`@DoqaId("LOGIN-{browser}")`).

Пока каталог не проверен, первая строка говорит только о коде: «ID кейсов в @DoqaCaseIds: 15» или «В @DoqaCaseIds кейсов нет». Про связи в DoQA плагин в этом случае ничего не утверждает.

Статусы как в DoQA

Исходы прогонов и состояния автотеста плагин называет так же, как веб-интерфейс DoQA.

Исход прогона в строке «Последний прогон:» и в **истории прогонов**: «Пройден», «Провален», «Сломан», «Заблокирован», «Пропущен».

Полоса «История, от старых к новым:» повторяет колонку «История» в **каталоге автотестов**: один значок на прогон, слева старые, справа новые.

Значок	Прогон
▯	Пройден
▯	Провален
!	Сломан или Заблокирован
·	Пропущен; сюда же попадают прогоны, которые не начинались

Под полосой плагин расшифровывает только те значки, которые в ней есть, например « Пройден, Провален». DoQA отдаёт полосу одной отметкой на сломанные и заблокированные прогоны, поэтому точный статус `!` плагин называет, только если такая отметка одна и стоит у последнего прогона: «Сломан» или «Заблокирован». Для остальных смотрите **историю прогонов**.

Состояние автотеста после «·» в строке «Стабильность:»: «Нестабильный», «Сломан», «В карантине», «Отключён», «Исправлен». У стабильного автотеста состояние не пишется.

Откуда берутся данные

Значки читают данные только из кэша плагина и сами в сеть не обращаются. Кэш наполняется:

- при открытии файла с тестами;
- по действию «**Обновить статусы в файле**»;
- в фоне, когда в файле появляется тест, которого кэш ещё не знает: новый тест или только что закреплённый ключ. Запрос уходит после короткой паузы в наборе.

Автоматическое обновление (при открытии файла и в фоне) работает, пока в настройках включено «Подтягивать статусы при открытии файла с тестами». Действие «Обновить статусы в файле» работает всегда.

Кэш хранится на диске, отдельно для каждого сервера и пространства, и переживает правку файла и перезапуск IDE. Если DoQA недоступен, значки показывают последние известные данные, а в подсказке появляется строка «DoQA недоступен: данные от ДД.ММ.ГГГГ ЧЧ:ММ».

Статус не обновился

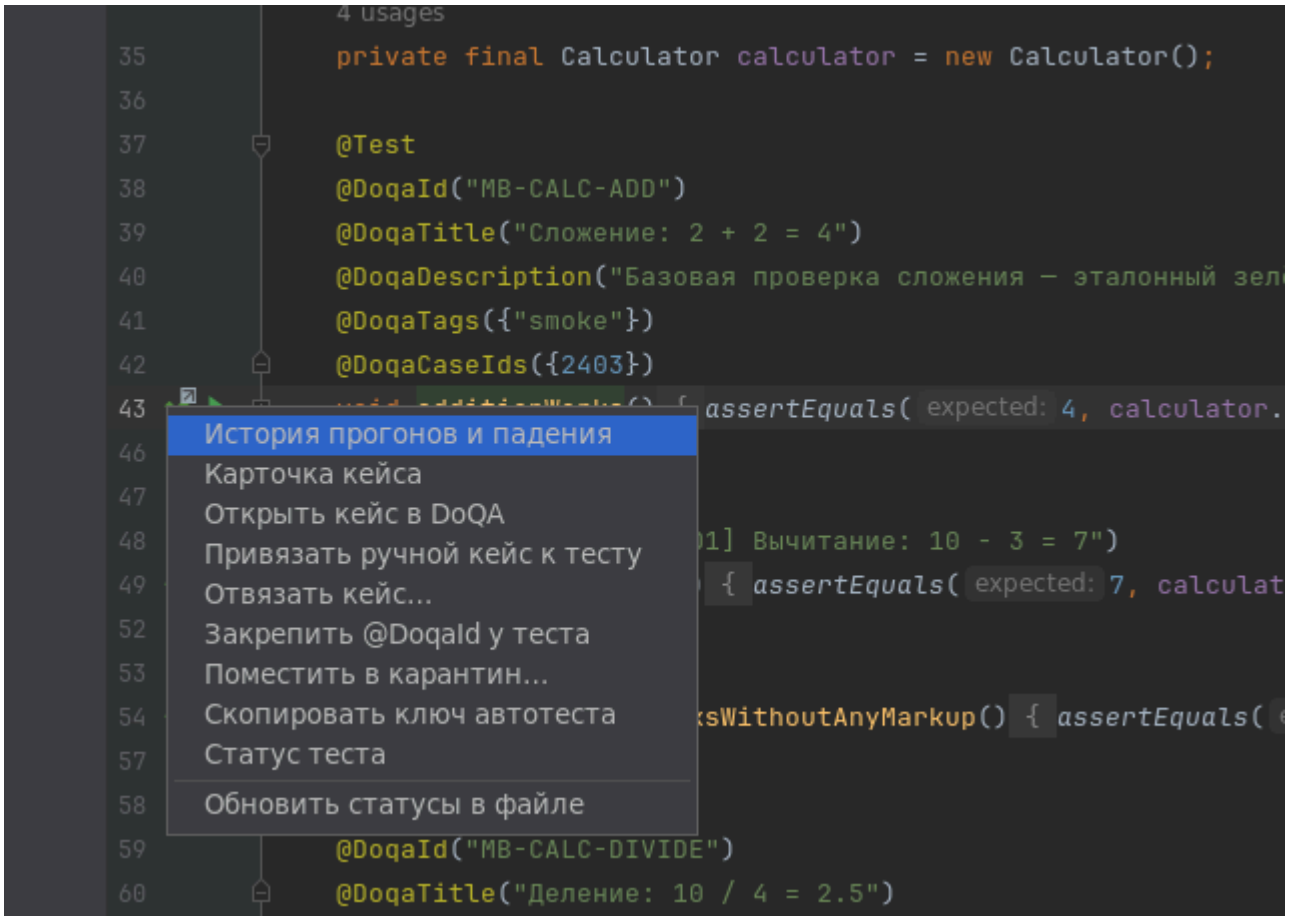
Если результат нового прогона уже виден в DoQA, а значок старый, выполните «**Обновить статусы в файле**». Плагин перечитает каталог для всех тестов файла и покажет подсказку «Статусы обновлены: тестов в файле — N».

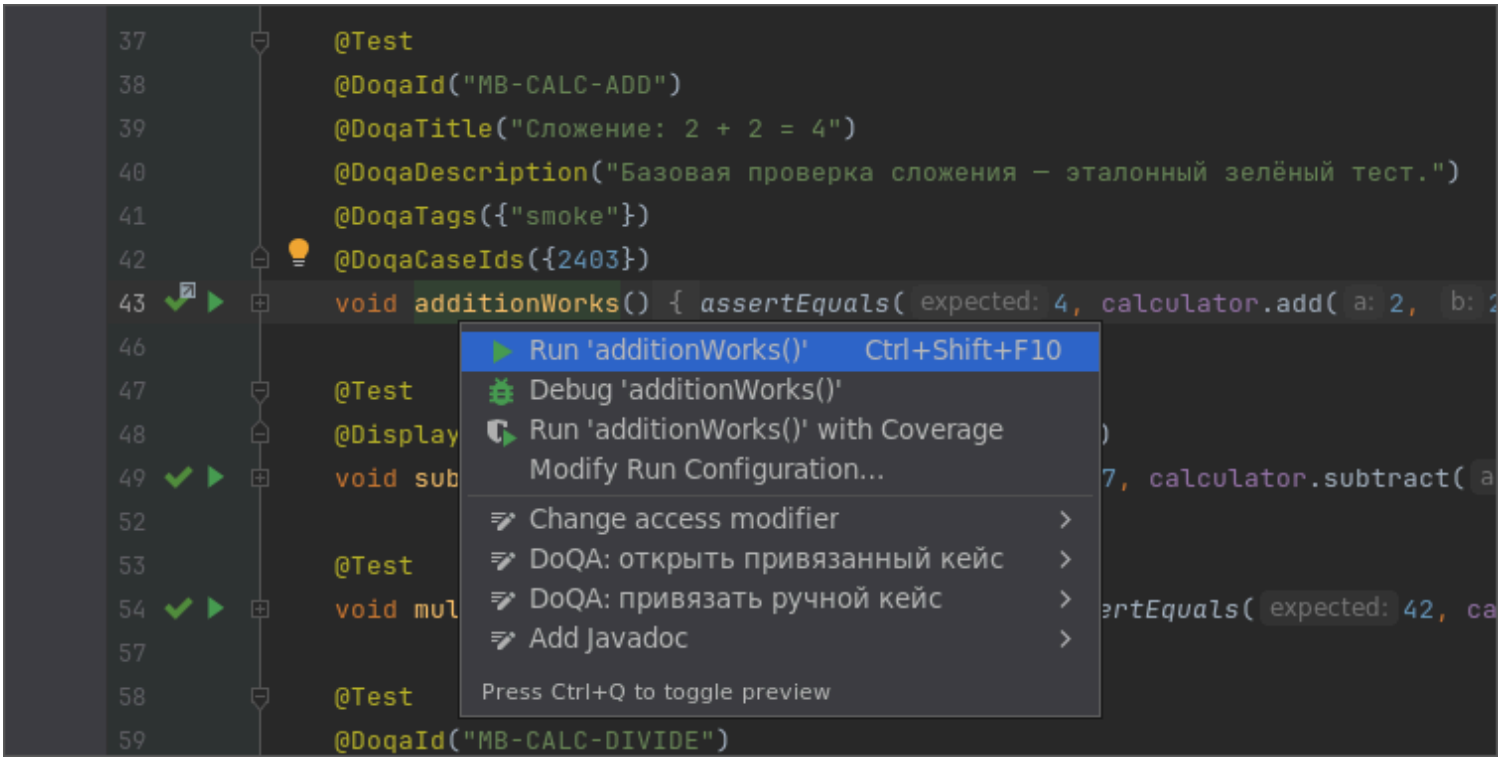
Где открыть меню DoQA

Все действия собраны в меню DoQA: кнопка в главном тулбаре, **Ctrl+Alt+Shift+D**, «Tools» → «DoQA» и контекстное меню редактора, подробнее в разделе **Где найти действия**. Действия над одним тестом есть ещё в двух местах:

Где	Что открывается
клик по значку в gutter	действия над этим тестом, курсор переходит на тест
Alt+Enter на аннотациях или объявлении теста	до трёх действий: «DoQA: привязать ручной кейс»; «DoQA: закрепить ключ автотеста», если у теста ещё нет `@DoqaId`; «DoQA: открыть привязанный кейс», если кейс привязан

Действия над одним тестом работают с тестом под курсором: курсор может стоять на аннотациях, объявлении или в теле метода. Если курсор вне теста, плагин подсказывает «Поставьте курсор в тестовый метод.».





Действия

Действие	Что делает	Когда доступно
«Статус теста»	подсказка у курсора: связанные кейсы, фреймворк, ключ автотеста, автотест в каталоге, последний прогон, стабильность	открыт файл с тестами
«История прогонов и падения»	окно с последними прогонами и стеком падения, см. **История прогонов**	курсор в тесте
«Карточка кейса»	шаги и ожидаемый результат привязанного кейса во всплывающем окне	к тесту привязан кейс
«Открыть кейс в DoQA»	привязанный кейс в браузере	к тесту привязан кейс
«Привязать ручной кейс к тесту»	поиск кейса, аннотация `@DoqaCaseIds` в коде и связь в DoQA	открыт файл с тестами

Действие	Что делает	Когда доступно
«Отвязать кейс...»	снимает связь в DoQA и убирает ID кейса из аннотации	к тесту привязан кейс
«Закрепить @DoqaId у теста»	записывает внешний ID в код, см. **Закрепление @DoqaId**	открыт файл с тестами
«Поместить в карантин...» / «Снять с карантина»	ставит автотест в карантин или снимает его; подпись зависит от текущего состояния	курсор в тесте
«Скопировать ключ автотеста»	копирует внешний ID теста в буфер обмена	открыт файл с тестами
«Создать тест из кейса...»	черновик теста по ручному кейсу, см. **Тест из кейса**	открыт файл с тестами
«Закрепить @DoqaId во всём файле»	закрепляет ключи у всех тестов файла без `@DoqaId`	в файле есть тесты
«Обновить статусы в файле»	перечитывает каталог для тестов открытого файла	открыт файл с тестами
«Каталог автотестов»	список автотестов пространства, см. **Каталог автотестов**	всегда
«Настройки DoQA...»	страница «Settings» → «Tools» → «DoQA»	всегда

В файлах pytest вместо `@DoqaId` и `@DoqaCaseIds` плагин пишет декораторы `@doqa.id` и `@doqa.case_ids`, и названия действий меняются: «Закрепить @doqa.id у теста», «Закрепить @doqa.id во всём файле». В тестах Jest плагин пишет ключи `id` и `caseIds` в метаданные `doqa.test` (кейс иногда вызовом `doqa.metadata` в теле теста), а действия называются «Закрепить id у теста» и «Закрепить id во всём файле», см. **Тесты Jest**.

Результаты и ошибки приходят уведомлениями IDE, а ответы о конкретном месте в коде показываются подсказкой у курсора. Модальный диалог появляется только там, где нужен выбор или подтверждение. Правки кода, которые делает плагин, отменяются одним **Ctrl+Z**.

Плагин не настроен

Если адрес DoQA или пространство не заданы, действия, которым нужен сервер, показывают уведомление «DoQA не настроен: укажите адрес, токен и space.» с кнопкой «Открыть настройки». Настройка описана на странице **Плагины для IDE**.

Статус теста

«**Статус теста**» сначала перечитывает каталог, а затем показывает у курсора подсказку:

- первые строки: связанные кейсы, как в **подсказке значка**, с теми же пояснениями, если автотест не найден в каталоге или ключ не вычисляется;
- «Фреймворк:» со значением JUnit 5, JUnit 4, TestNG, pytest или Jest;
- «Ключ автотеста:» с внешним ID; если он не вычисляется, ниже «Ключ не вычислен:» с причиной;
- «Автотест в DoQA:» с номером и названием автотеста, если он есть в каталоге; если его ещё нет, об этом говорит строка «Автотеста ещё нет в каталоге DoQA: он появится после первого прогона» среди первых строк;
- «Последний прогон:» и «Стабильность:», если прогоны были.

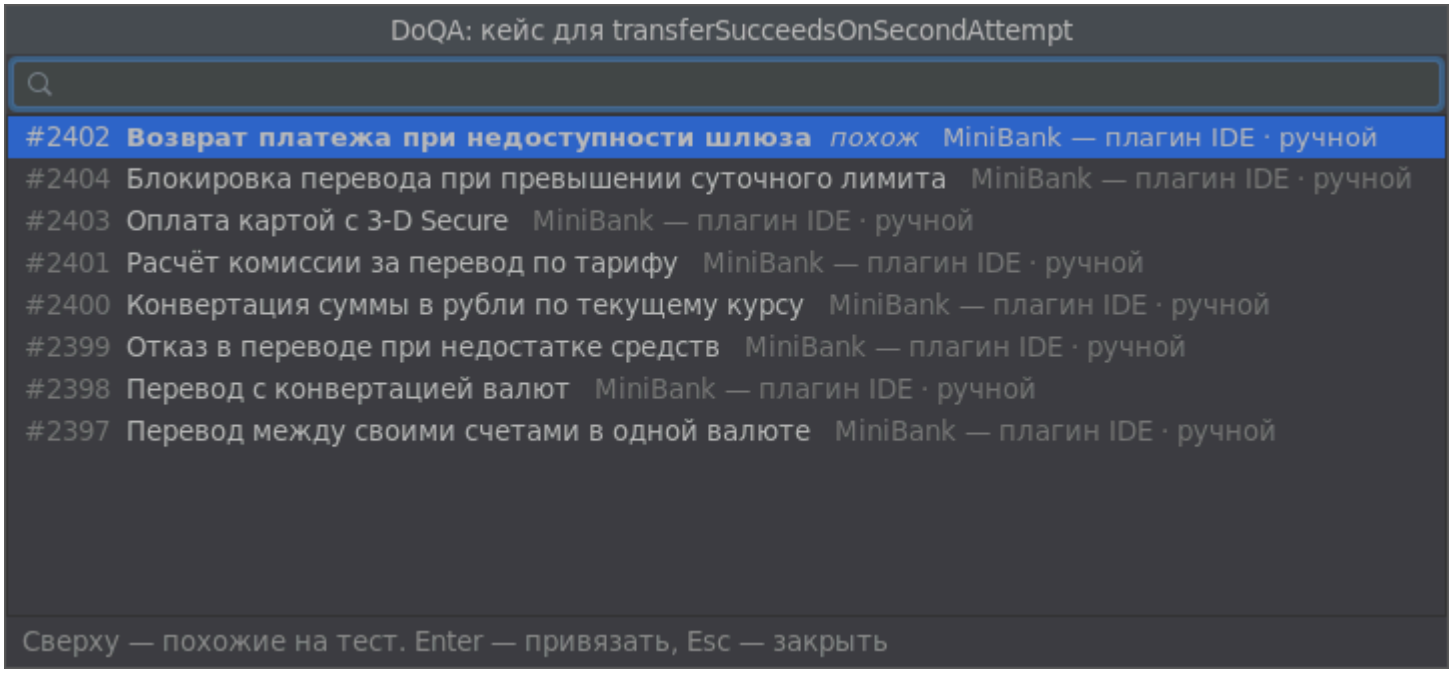
Если для файла не выбрано пространство, вместо строки об автотесте стоит «Space DoQA для этого файла не выбран в настройках: каталог не проверен».

Привязка кейса

1. Поставьте курсор в тест и выберите «**Привязать ручной кейс к тесту**» (в Alt+Enter это «DoQA: привязать ручной кейс»).
2. Откроется попап «DoQA: кейс для <имя метода>». Сразу, без ввода, в нём появляются кейсы из верхних папок дерева. Сверху стоят кейсы, похожие на тест по названию: они выделены жирным и помечены «похож», внизу подсказка «Сверху — похожие на тест. Enter — привязать, Esc — закрыть».
3. Начните вводить название: список обновляется по мере ввода. Стрелки двигают выбор.
4. Нажмите **Enter** или дважды щёлкните по кейсу.

Плагин добавляет ID кейса в `@DoqaCaseIds` над тестом (импорт дописывается сам, существующие ID сохраняются) и создаёт связь в DoQA. Появляется уведомление «Кейс #N

«название» привязан к тесту …» с кнопкой «Открыть кейс». Привязка только добавляет связь: прежние связи теста не снимаются.



Как подбираются похожие кейсы: плагин сравнивает слова из `@DisplayName`, `@DoqaTitle`, `description` у TestNG и имени метода (у Jest из названия теста и `title` в метаданных `doqa.test`) со словами в названиях кейсов, с учётом окончаний. Смысла он не понимает: тест `addsItemToCart` не найдёт кейс «Добавление товара в корзину».

Поиск останавливается на 50 кейсах: если совпадений больше, внизу попапа появляется «Показаны не все совпадения — уточните запрос».

Что происходит с внешним ID при привязке:

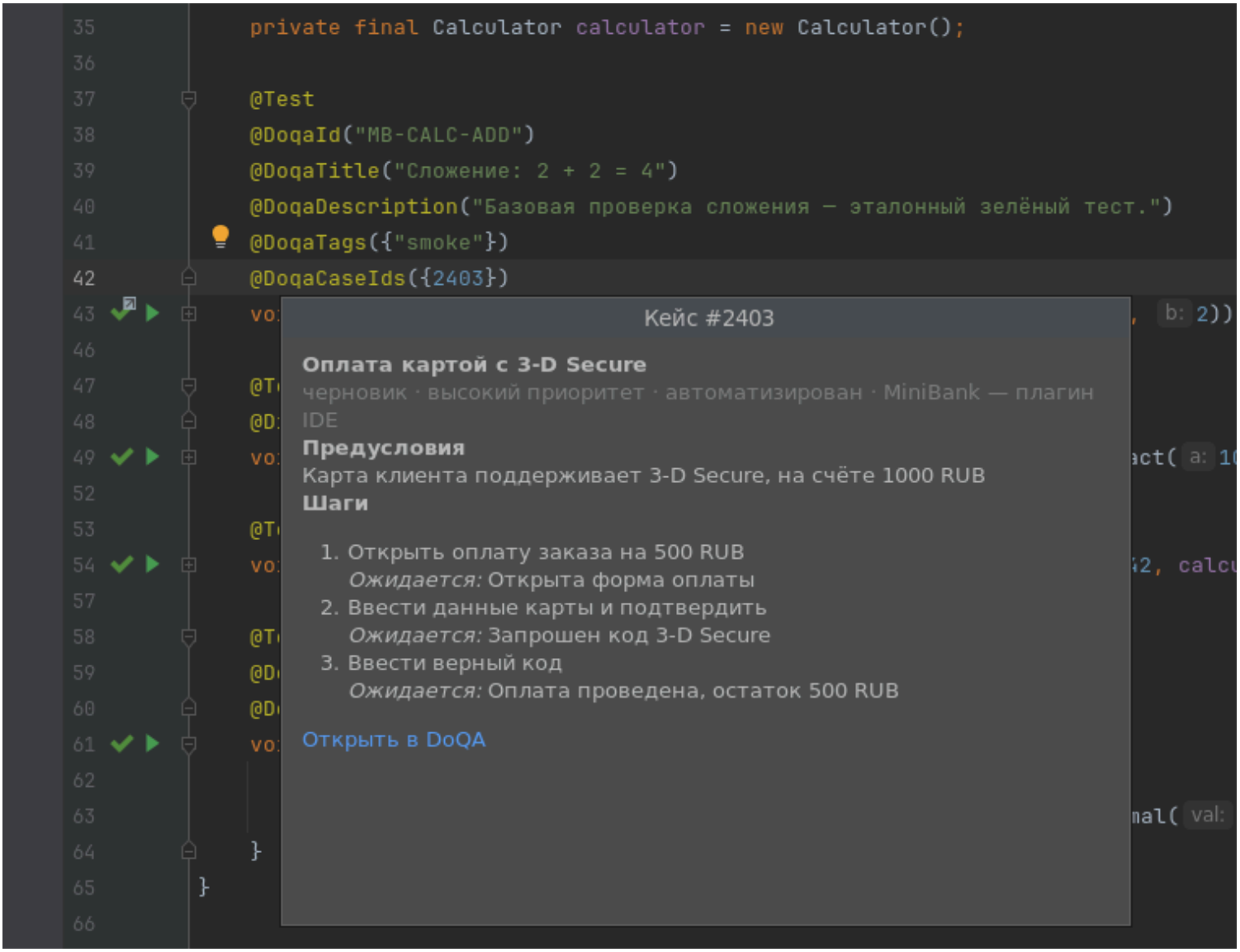
- тест, которого в DoQA ещё нет, привязывается под **читаемым ключом**, и плагин сразу закрепляет его в `@DoqaId`. В каталоге не появится автотест с хэшем вместо имени;
- тест, который уже есть в каталоге, сохраняет свой ключ. Если это хэш, в уведомлении будет «Ключ автотеста — хэш: переименование теста его сменит.» и кнопка «Закрепить @DoqaId»;
- если ключ по тексту файла не вычислить, плагин попросит ввести его в диалоге «DoQA: ключ автотеста» и тоже закрепит в коде.
- у строк `each` в Jest ключ не спрашивается: кейс записывается только в код, и адаптер привяжет его к каждой строке при следующем прогоне. Так же с тестом Jest, у которого ключ не вычислен и не может быть закреплён, см. **Кейс в doqa.metadata**.

Карточка кейса

Карточку привязанного кейса открывает действие «**Карточка кейса**» или **Ctrl+клик** по ID в `@DoqaCaseIds` (`doqa.case_ids` в pytest). Во всплывающем окне «Кейс #N»:

- название кейса;
- статус, приоритет, статус автоматизации и путь по папкам;
- «Предусловия»;
- «Шаги» с ожидаемым результатом каждого шага («Ожидается:»);
- «Ожидаемый результат» кейса;
- ссылка «Открыть в DoQA».

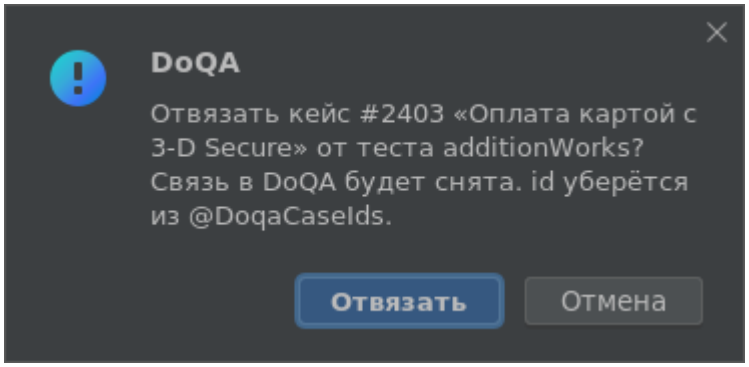
Если к тесту привязано несколько кейсов, действие сначала предлагает выбрать один. Когда DoQA недоступен, показывается карточка из кэша, если плагин уже загружал её раньше.



Отвязка кейса

1. Выберите «**Отвязать кейс...**». Если кейсов несколько, выберите нужный в списке.
2. Подтвердите в диалоге: «Отвязать кейс #N «название» от теста ...?». Под вопросом сказано, что будет сделано: «Связь в DoQA будет снята.» и «id уберётся из @DoqaCaseIds.».
3. Нажмите «**Отвязать**».

ID убирается и из аннотации: иначе следующий прогон привязал бы кейс снова. В тесте Jest ID убирается из `caseIds` и из вызовов `doqa.metadata`, см. **Кейс в doqa.metadata**.



Кейсы на классе

Если ID кейса записан в `@DoqaCaseIds` на классе, для одного теста его не отвязать. Плагин подскажет: «Кейс #N указан в @DoqaCaseIds на классе: для одного теста его не отвязать. Перенесите @DoqaCaseIds на методы или уберите id с класса.»

Отвязать нельзя и последний ID из `@DoqaCaseIds` метода, если свои ID есть у класса: без аннотации на методе адаптер возьмёт ID класса и привяжет их к тесту. Подсказка в этом случае начинается со слов «#N — последний id в @DoqaCaseIds метода».

Читаемый ключ и закрепление @DoqaId

Если в коде нет явного внешнего ID, адаптер вычисляет его сам, чаще всего как хэш сигнатуры теста (`junit5:3c9ce9…`). Такой ключ меняется при переименовании теста, и DoQA заводит новый автотест без истории, см. **Внешний ID**. Закрепление записывает ключ в код аннотацией `@DoqaId`, и дальше тест можно переименовывать и переносить.

Плагин предлагает **читаемый ключ**, то есть полное имя теста, как в Allure: `com.acme.CheckoutTest.addsItemToCart`, для pytest `tests.unit.test_cart.TestCart.test_adds`. Он уникален там же, где уникально полное имя класса или модуля.

Сменить ключ автотеста, который DoQA уже знает, нельзя без потери истории: новый ключ означает новый автотест. Поэтому «**Закрепить @DoqaId у теста**» (в Alt+Enter это «DoQA: закрепить ключ автотеста») выбирает ключ так:

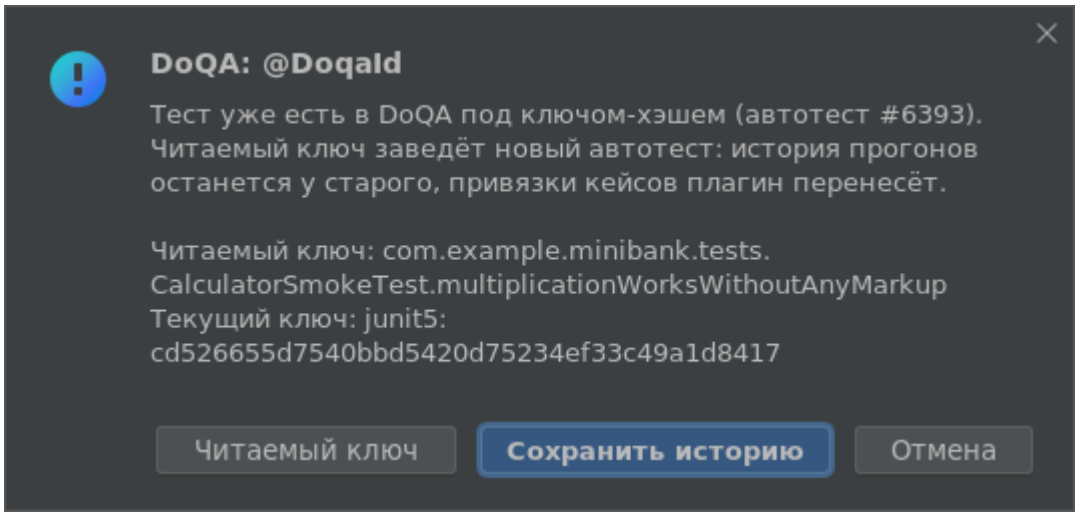
Ситуация	Что делает плагин
ключ уже читаемый: `DOQA-123` из названия теста, `ALLURE-42` из Allure	закрепляет его как есть
теста в DoQA ещё нет	закрепляет читаемый ключ и подсказывает «Новый тест: в DoQA его ещё нет, поэтому ключ читаемый.»
тест уже есть в DoQA под хэшем	спрашивает, что закрепить (см. ниже)
читаемый ключ уже занят другим автотестом	просит ввести другой ключ, по умолчанию предлагает `<ключ>-2`
ключ по тексту файла не вычислить	просит ввести ключ, по умолчанию предлагает читаемый
у теста уже есть `@DoqaId`	ничего не пишет, подсказка «@DoqaId уже стоит: ...»
`@DoqaId` стоит на классе	ничего не пишет: под ключом класса отчитываются все его тесты. Подсказка «Ключ задан на классе (...), под ним отчитываются все тесты класса. Чтобы закрепить ключ у отдельного теста, уберите @DoqaId с класса.»

Когда тест уже есть в DoQA под хэшем, диалог «DoQA: @DoqaId» объясняет: «Тест уже есть в DoQA под ключом-хэшем (автотест #N). Читаемый ключ заведёт новый автотест: история прогонов останется у старого, привязки кейсов плагин перенесёт.» Ниже показаны оба ключа: «Читаемый ключ:» и «Текущий ключ:».

- «**Сохранить историю**» (выбрано по умолчанию): закрепить текущий ключ-хэш. История остаётся, но ключ нечитаемый.

- «**Читаемый ключ**»: закрепить читаемый ключ. Привязки кейсов плагин переносит на новый ключ и сообщает об этом уведомлением «Кейсы #… привязаны к новому ключу ….. История прогонов осталась у автотеста #N. Ctrl+Z вернёт ключ в коде, но не привязки в DoQA.». Кейс в DoQA привязан к одному автотесту, поэтому у старого автотеста привязки пропадают.
- «**Отмена**»: ничего не менять.

Если пространство не выбрано, плагин не может проверить каталог. Тогда диалог тот же, но вопрос другой: «Каталог DoQA не проверен: плагин не настроен…», и по умолчанию выбран «Читаемый ключ».



После записи у курсора появляется подсказка «Закреплено: @DoqaId("...")» и «Отменить — Ctrl+Z».

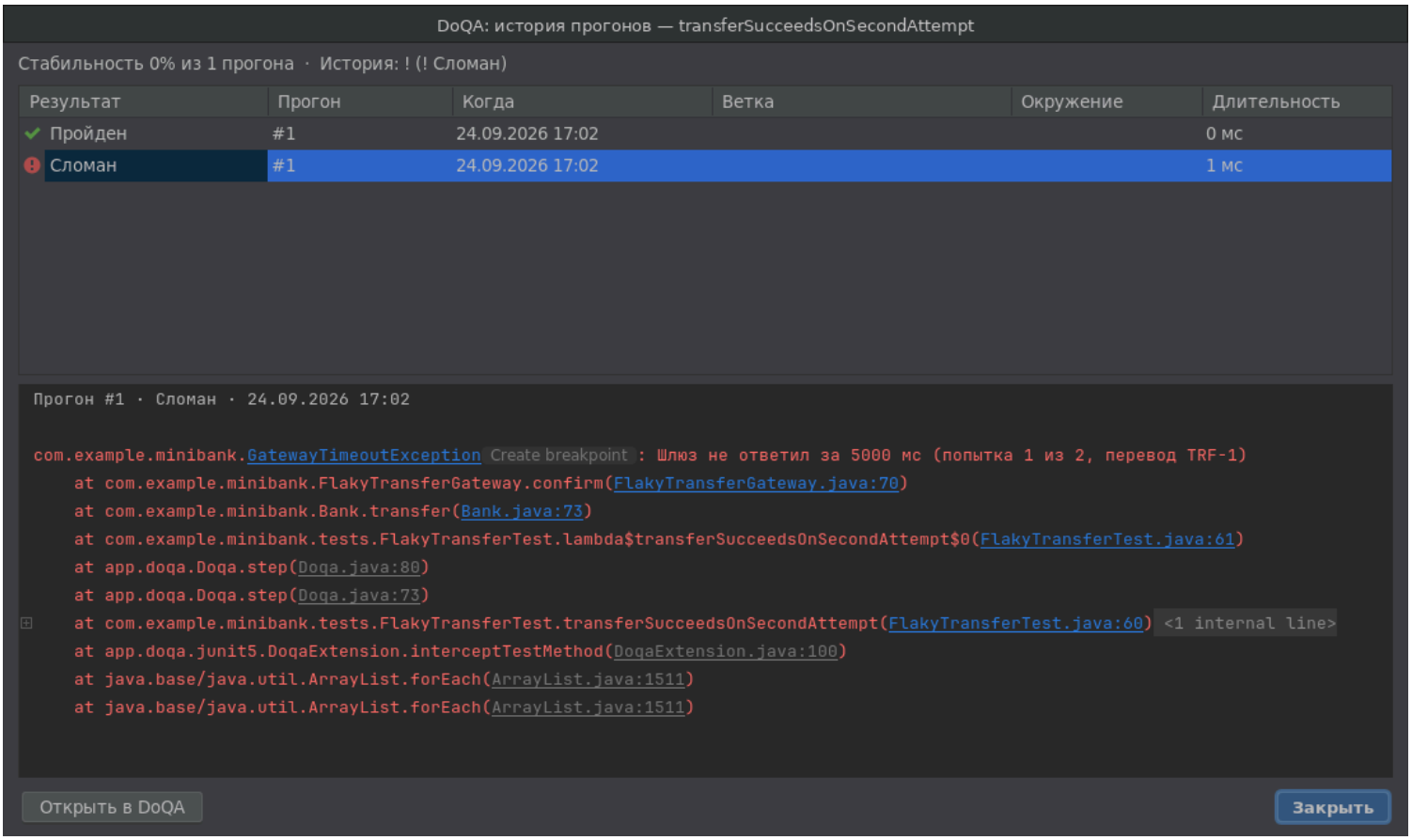
Весь файл

«**Закрепить @DoqaId во всём файле**» проставляет ключи всем тестам файла без `@DoqaId`, у которых ключ вычисляется. Тесты класса с `@DoqaId` на самом классе пропускаются. Правило то же, но без вопроса по каждому тесту: тесты, которых в DoQA нет, получают читаемый ключ, а тесты с историей сохраняют текущий. Перед записью плагин один раз спрашивает подтверждение, например: «Проставить @DoqaId у 5 тест(ов): читаемый ключ — у 3 (в DoQA их ещё нет); текущий ключ — у 2 (сохраняется история прогонов).». Если каталог проверить не удалось, везде записываются текущие ключи, о чём сказано в том же диалоге.

История прогонов и падения

«**История прогонов и падения**» открывает окно «DoQA: история прогонов — <имя метода>» с последними 30 прогонами автотеста. Окно не модальное: его можно держать открытым рядом с кодом, пока чините тест.

- Сверху показана сводка: стабильность, состояние («Нестабильный», «Сломан» и другие, см. **Статусы как в DoQA**), «В карантине» и полоса последних прогонов с расшифровкой: «История: ! (Пройден, Провален, ! Сломан)».
- Таблица: «Результат», «Прогон», «Когда», «Ветка», «Окружение», «Длительность». В колонке «Результат» статус прогона: «Пройден», «Провален», «Сломан», «Заблокирован» или «Пропущен». Прогон, который DoQA отметил как нестабильный, помечен «Нестабильный».
- Внизу показан выбранный прогон: заголовок с номером, исходом, временем, веткой, коммитом и окружением, затем сообщение об ошибке и стек. Строки стека ведут в код проекта. У тестов Jest ссылками становятся строки стека Node вида `tests/cart.test.js:12:5`, кроме строк из `node_modules`.
- При открытии выбран последний проваленный или сломанный прогон.
- Кнопка «**Открыть в DoQA**» ведёт на автотест в веб-интерфейсе.



Если автотеста ещё нет в каталоге, плагин подскажет «Автотеста ... ещё нет в DoQA: он появится после первого прогона.». Если у теста ключ с плейсхолдером (`@DoqaId("LOGIN-`

`{browser}")`), у каждого набора параметров свой автотест и история для теста целиком недоступна.

Версия DoQA

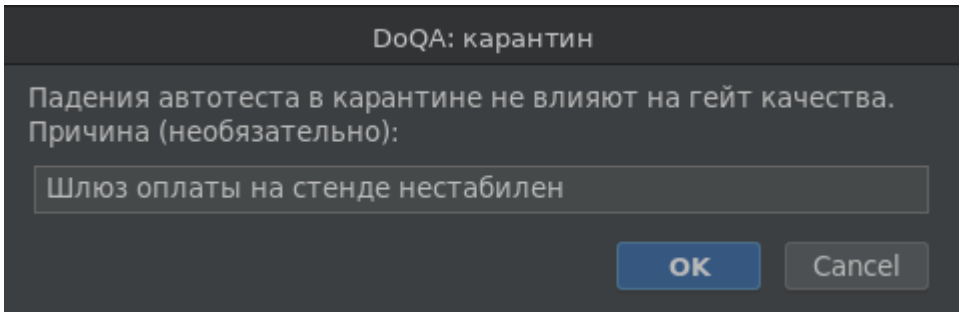
История, отвязка кейса и карантин работают, только если сервер DoQA принимает для этих запросов персональный API-токен. На сервере, где это ещё не так, плагин напишет, что сервер DoQA пока не принимает API-токен для истории прогонов (отвязки, карантина) и эта возможность появится после обновления DoQA. Токен при этом в порядке, менять его не нужно. Подробнее в разделе Ограничения.

Карантин

1. Выберите «**Поместить в карантин...**».
2. В диалоге «DoQA: карантин» при желании укажите причину: «Падения автотеста в карантине не влияют на гейт качества. Причина (необязательно):».
3. Нажмите «**OK**». У курсора появится «<имя метода> в карантине.», значок в gutter сменится на паузу.

Если автотест уже в карантине, то же действие называется «**Снять с карантина**» и спрашивает: «Снять ... с карантина? Его падения снова будут влиять на гейт.».

Плагин не передаёт ветку: карантин ставится и снимается для автотеста целиком. Что карантин меняет и как DoQA снимает его сам, описано на странице **Нестабильные тесты и карантин**.

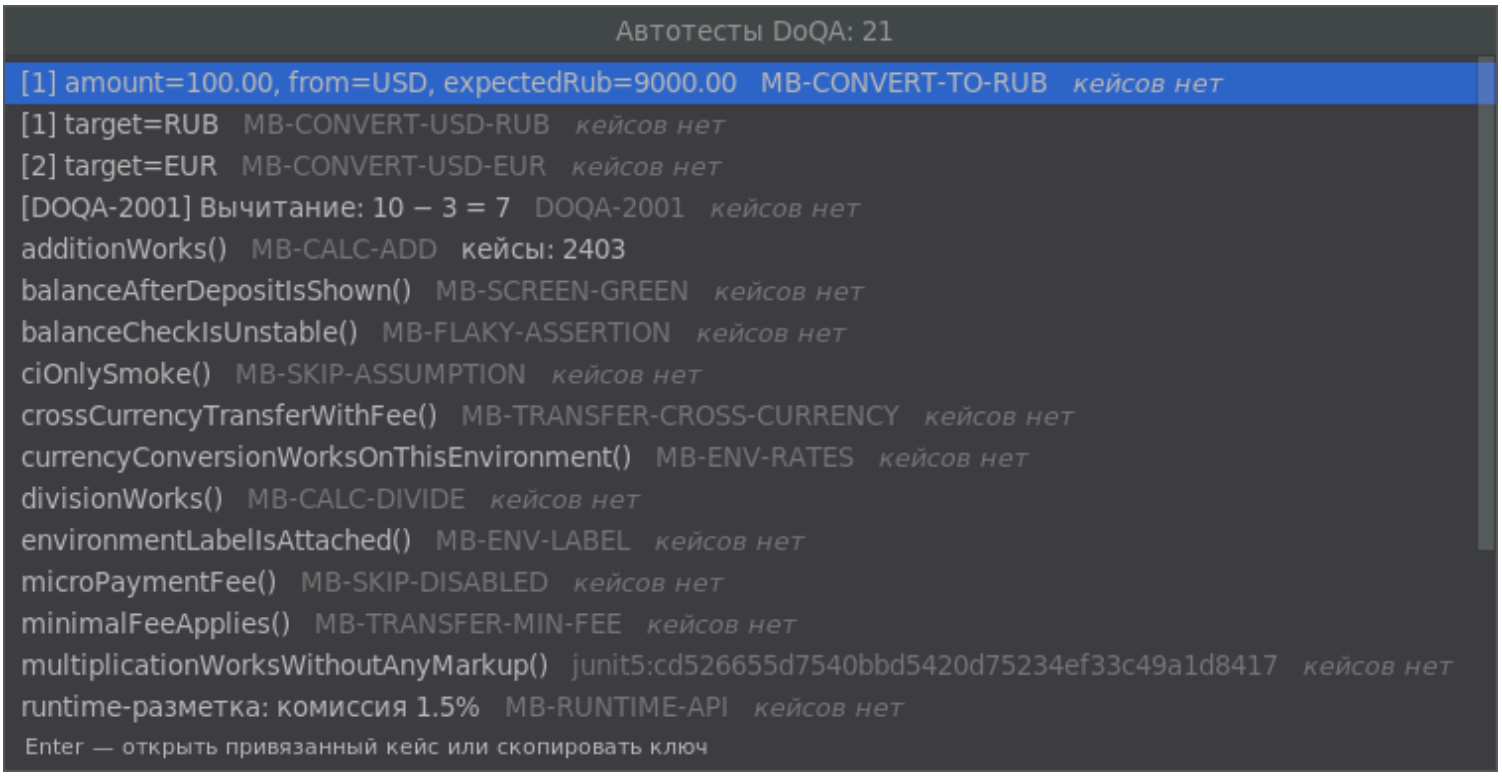


Каталог автотестов

«**Каталог автотестов**» показывает автотесты пространства в попапе, первые 200. В заголовке стоит «Автотесты DoQA: N», а если автотестов больше, «Автотесты DoQA: 200 из N — наберите часть имени».

- В строке показаны название автотеста, внешний ID серым (если он отличается от названия) и привязанные кейсы: «кейсы: 1041, 7» или «кейсов нет».
- Ввод текста фильтрует список по названию и внешнему ID.
- **Enter** открывает привязанный кейс в браузере. Если кейсов нет, копирует внешний ID и сообщает «Кейсы к автотесту не привязаны. Ключ скопирован: ...».

Полный каталог с фильтрами, стабильностью и массовыми действиями есть в веб-интерфейсе, см. Каталог автотестов.



Тесты Jest

Плагин работает с тестами Jest, которые отчитываются в DoQA через адаптер `@doqa-tms/jest`. Отдельной страницы об адаптере в этом руководстве пока нет, всё о нём написано в README адаптера Jest ↗.

В Jest нет аннотаций, поэтому разметка DoQA хранится в метаданных `doqa.test`, во втором аргументе вызова. Внешний ID это ключ `id`, привязка к кейсам это `caseIds`, теги и метки

лежат в `tags` и `labels`:

```js
doqa.test('добавляет товар', { id: 'CART-ADD', caseIds: [1041] }, () => {
  // …
});
```

Кейсы можно передать и вызовом `doqa.metadata({ caseIds: [...] })` в теле теста: так плагин привязывает кейс к тестам, которые нельзя превратить в `doqa.test`, см. **Кейс в doqa.metadata**.

Значки в gutter у тестов Jest показывают IDE с плагином JavaScript: IntelliJ IDEA Ultimate и WebStorm. Действия меню работают в любой IDE на платформе IntelliJ.

Какие тесты видит плагин

Файлы. Плагин читает файлы `.js`, `.cjs`, `.mjs`, `.jsx`, `.ts`, `.cts`, `.mts` и `.tsx`.

Проект Jest. От каталога файла плагин поднимается вверх и берёт ближайший каталог, где лежит `jest.config.js` (или `.ts`, `.mjs`, `.cjs`, `.mts`, `.cts`, `.json`) либо `package.json` с ключом `"jest"`. Так же конфиг выбирает Jest, если запустить его из этого каталога. Конфиг плагин читает как текст: `module.exports = …` или `export default …`, обёртку `withDoqa(config, options)` и константы модуля.

Тестовые файлы. Тестом считается файл, который Jest запустил бы по шаблонам по умолчанию: всё в каталоге `__tests__`, а также `*.test.*` и `*.spec.*`. Если в конфиге задан свой `testMatch` или `testRegex`, тестом может быть любой файл JavaScript или TypeScript. Файлы из `node_modules` не читаются. Если конфига нет, плагин читает только файл, который импортирует `@doqa-tms/jest` или `@jest/globals`.

Тесты. Это вызовы `test` и `it` с модификаторами `.only`, `.skip`, `.concurrent`, `.todo`, `.failing` и `.each`, а также `xit`, `xtest`, `fit` и `doqa.test`. Имя, под которым импортирован `doqa`, не важно: `import { doqa as dq }`, `const { doqa } = require(…)`, `import * as adapter`. Блоки `describe`, `xdescribe` и `fdescribe` задают цепочку названий. Если `test` импортирован из другого модуля, например `import { test } from 'vitest'`, это не тест Jest, и плагин его пропускает.

Значок в gutter стоит у строки с вызовом теста.

Внешний ID теста Jest

Плагин вычисляет внешний ID по тем же правилам, что и адаптер:

1. `id` из метаданных `doqa.test`. Значение берётся как есть, пробелы по краям не обрезаются. Если в теле теста отдельной инструкцией стоит `doqa.metadata({ id: '…' })`, адаптер берёт этот `id`, и плагин тоже.
2. Номер из названия теста: `[DOQA-123]` или `@DOQA:123` дают ключ `DOQA-123`.
3. Иначе хэш: `jest:` и SHA-1 от строк, соединённых переводом строки: название проекта, путь файла, названия блоков `describe` по порядку и название теста.

Из чего собирается хэш:

- **путь файла** считается от `rootDir` проекта Jest и пишется через `/`. `rootDir` это каталог конфига или значение `rootDir` из конфига;
- **название проекта** берётся из `projectName` во втором аргументе `withDoqa` (адаптер не различает в ключе регистр, `-` и `_`, поэтому подходят и `project-name`, и `project_name`), затем из `projectName` в файле `doqa.properties` рядом с конфигом, затем из `displayName` проекта. Если ничего из этого не задано, строка пустая;
- **название теста** у `doqa.test.each` это шаблон, например `роль %s`. Поэтому все строки данных `doqa.test.each` относятся к одному автотесту. У обычного `test.each` каждая строка становится отдельным автотестом с названием, которое Jest собирает из шаблона и данных.

Внешний ID длиннее 255 символов адаптер обрезает до 255, плагин тоже.

Пример. В проекте с `jest.config.js` в корне и без названия проекта тест `добавляет товар` из блока `describe('Корзина')` в файле `tests/cart.test.js` получит ключ `jest:e6b5a5d6d483742d113a6dbd93151ed704c70f2e`. После переименования теста или переноса в другой блок ключ станет другим, и DoQA заведёт новый автотест. Чтобы этого не случилось, закрепите `id`.

Когда плагин не вычисляет ID

Если по тексту файла и конфига нельзя получить тот же ID, что вычислит адаптер при запуске, плагин ключ не показывает. В подсказке у значка вместо ключа написано «вычислит

адаптер» и причина:

Ситуация	Причина в подсказке
конфига Jest нет ни в каталоге файла, ни выше	«не найден конфиг Jest (jest.config.* или ключ «jest» в package.json), а от него зависят путь файла и название проекта в id»
в одном каталоге два конфига	«в одном каталоге несколько конфигов Jest: Jest не запустится, пока не останется один»
конфиг собирается кодом: функция, импорт, spread; конфиг меняют после объявления (`config.displayName = …` , `module.exports.rootDir = …`)	«конфиг Jest собирается кодом (функция, импорт, spread): rootDir и название проекта из текста не прочитать»
`rootDir` задан выражением	«rootDir в конфиге Jest задан не строковым литералом, а путь файла в id считается от него»
`projectName` или `displayName` задан выражением, объект опций `withDoqa` меняют после объявления	«projectName или displayName в конфиге Jest задан не литералом, а название проекта входит в id»
в конфиге есть список `projects`	«в конфиге Jest задан список projects: к какому проекту относится файл и как он называется, из текста не понять»
обычный `test.each` или `describe.each`	«test.each или describe.each: у каждой строки данных свой автотест, название строки формирует Jest»
номер `DOQA` в данных `each` или в шаблоне (`[DOQA-%d]`); у `doqa.test.each` с подстановкой в названии данные заданы не литералом (переменная, модуль), и номер может прийти из них	«номер DOQA в параметрах теста: у разных наборов параметров разные id»
название теста не строка: шаблон с `${…}` , склейка, переменная	«название теста задано не строковым литералом»

Ситуация	Причина в подсказке
название `describe` не строка	«название describe задано не строковым литералом»
тест объявлен внутри функции: хелпер, цикл `forEach`	«тест объявлен внутри функции: в какой describe он попадёт и сколько раз, по тексту не понять»
метаданные не объект в вызове и не константа модуля, константу меняют после объявления (`META.id = …` , повторное присваивание), `id` не строка	«метаданные doqa.test заданы не литералом объекта или id в них не строка»
`id: ''`	«id в метаданных doqa.test пустой: DoQA не примет автотест без id»
`doqa.metadata({ id })` в хуке, во вспомогательной функции или под условием; `doqa.metadata` взят под другим именем (`const { metadata } = doqa`)	«id задаётся вызовом doqa.metadata в хуке, функции или условии: какой id окажется у теста, по тексту не понять»

Для таких тестов закрепите `id` в метаданных `doqa.test` : явный ключ плагин и адаптер читают одинаково. Явный `id` плагин читает, даже если не понял конфиг Jest: конфиг нужен только для хэша.

Как проверяли

Правила вычисления сверены с настоящим Jest 30 и адаптером `@doqa-tms/jest` : на 29 проектах с разными конфигурациями ключи плагина совпали с ключами адаптера, а в остальных случаях плагин назвал причину отказа.

Что плагин пишет в код

Привязка кейса и закрепление ключа записывают `caseIds` и `id` в метаданные `doqa.test` . Обычный `test` или `it` плагин превращает в `doqa.test` . Было:

```js
const { Cart } = require('../src/cart');

describe('Корзина', () => {
  test('добавляет товар', () => {
```

```
    // …
  });
});
```

Стало после привязки кейса #2401 к тесту, которого в DoQA ещё нет:

```js
const { Cart } = require('../src/cart');
const { doqa } = require('@doqa-tms/jest');

describe('Корзина', () => {
  doqa.test('добавляет товар', { id: 'tests.cart > Корзина > добавляет товар', caseIds
    // …
  });
});
```

Как устроена правка:

- модификаторы сохраняются: `xit` и `xtest` становятся `doqa.test.skip`, `fit` становится `doqa.test.only`, `.concurrent` остаётся на месте;
- импорт `doqa` плагин пишет в стиле файла: `import { doqa } from '@doqa-tms/jest';` или `const { doqa } = require('@doqa-tms/jest');`. Он встаёт после последнего импорта верхнего уровня, а без импортов после начальных комментариев, чтобы docblock Jest остался первым. Если `doqa` уже импортирован под другим именем, плагин пишет через это имя. Если имя `doqa` в файле занято чем-то другим, адаптер импортируется как `doqaTms`: `const { doqa: doqaTms } = require('@doqa-tms/jest');`;
- у существующего `doqa.test` плагин меняет только нужный ключ, остальные ключи и форматирование объекта не трогает. В однострочный объект ключ добавляется в конец строки, в многострочный новой строкой с тем же отступом;
- в `caseIds` новые ID добавляются к старым;
- отвязка убирает ID из `caseIds`. Если ID был последним, из объекта уходит весь ключ `caseIds`, а пустой объект убирается из вызова.

Некоторые тесты нельзя превратить в `doqa.test`, не изменив их поведение. Для них кейс пишется вызовом `doqa.metadata` в теле теста, см. Кейс в doqa.metadata.

Кейс в doqa.metadata

Адаптер `@doqa-tms/jest` собирает ID кейсов из двух мест: из `caseIds` в метаданных `doqa.test` и из вызова `doqa.metadata({ caseIds: [...] })` в теле теста. Второй способ подходит любому тесту, у которого есть тело. Поэтому, если `doqa.test` изменил бы тест, плагин записывает кейс вызовом `doqa.metadata` первой строкой тела:

Тест	Почему не `doqa.test`
обычный `test.each` или `it.each`	`doqa.test.each` свёл бы строки данных в один автотест, и строки потеряли бы историю
таблица `each` задана шаблонной строкой (`test.each`…`)	`doqa.test.each` принимает только массив строк
тест принимает колбэк `done`	`doqa.test` не передаёт колбэк `done`
`test.failing`	у `doqa.test` нет режима `failing`
метаданные `doqa.test` заданы не объектом в самом вызове или `caseIds` в них не массив чисел	плагин не может поправить их, не сломав

Если в теле уже есть такой вызов с `caseIds`, новый ID дописывается в него, второй вызов не появляется. Иначе плагин вставляет новый вызов первой строкой тела, с отступом тела, и добавляет импорт `doqa` так же, как для `doqa.test` (при занятом имени через `doqaTms`). Если у стрелочной функции тело это выражение (`(n) => expect(n).toBe(n)`), плагин делает из него блок: сначала вызов, затем `return` прежнего выражения. Обычный `test`, которому ничто не мешает, по-прежнему становится `doqa.test`.

Пример с `test.each`. Было:

```js
describe('Корзина', () => {
  test.each([[1, 100], [3, 300]])('сумма для %i шт.', (qty, expected) => {
    const cart = new Cart();
    cart.add('A-1', 100, qty);
    expect(cart.total()).toBe(expected);
  });
});
```

Стало после привязки кейса #2401:

```js
const { doqa } = require('@doqa-tms/jest');

describe('Корзина', () => {
  test.each([[1, 100], [3, 300]])('сумма для %i шт.', (qty, expected) => {
    doqa.metadata({ caseIds: [2401] });
    const cart = new Cart();
    cart.add('A-1', 100, qty);
    expect(cart.total()).toBe(expected);
  });
});
```

Каждая строка данных остаётся отдельным автотестом со своим ключом, и кейс #2401 получают обе строки, как у параметризованного теста JUnit с `@DoqaCaseIds`.

Пример с колбэком `done`. Было:

```js
test('пустая корзина', (done) => {
  loadCart((cart) => {
    expect(cart.items).toHaveLength(0);
    done();
  });
});
```

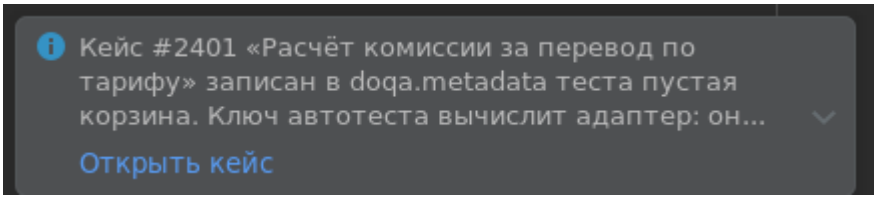
Стало:

```js
test('пустая корзина', (done) => {
  doqa.metadata({ caseIds: [2401] });
  loadCart((cart) => {
    expect(cart.items).toHaveLength(0);
    done();
  });
});
```

После такой привязки уведомление добавляет «ID записан в doqa.metadata в теле теста.».

Связь в DoQA. У каждой строки `each` свой автотест, а ключи строк Jest собирает только при запуске. Поэтому для строк `each` плагин не спрашивает ключ и не создаёт связь в DoQA: кейс записывается только в код, и адаптер привяжет его к каждой строке при следующем прогоне. Уведомление: «Кейс #N «...» записан в doqa.metadata теста .... У каждой строки each свой автотест: адаптер привяжет кейс к каждой из них при следующем прогоне.».

То же с тестом, у которого плагин не вычислил ключ и не может его закрепить (например, тест с `done` без конфига Jest): кейс пишется только в код, а в уведомлении «Ключ автотеста вычислит адаптер: он привяжет кейс при следующем прогоне.». Если ключ такого теста вычислен, связь в DoQA создаётся сразу, под тем ключом, который пришлёт адаптер.

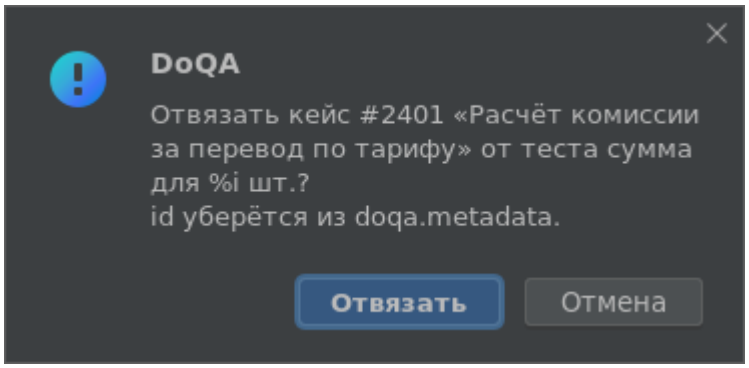


Какие вызовы плагин читает. Вызов `doqa.metadata({ caseIds: [2401, 2402] })` засчитывается, если он стоит отдельной инструкцией на верхнем уровне тела теста (`await` перед ним допустим), а `caseIds` это массив чисел. Имя, под которым импортирован `doqa`, не важно. Кейсы из вызова плагин складывает с `caseIds` из метаданных `doqa.test` и показывает вместе: в подсказке значка и в «Статусе теста» это строка «ID кейсов в doqa.metadata: 2401», а если кейсы есть и там и там, «ID кейсов в caseIds и doqa.metadata: ...». **Ctrl+клик** по ID в вызове открывает карточку кейса.

Если `caseIds` в вызове не литерал, вызов стоит в условии, хуке или вспомогательной функции либо `doqa.metadata` взят под другим именем (`const { metadata } = doqa`), плагин не гадает, какие кейсы получит тест. Он показывает те, что видит, и добавляет строку «Не все ID кейсов видны в исходнике: часть задана не литералом или вызовом doqa.metadata в условии, хуке или функции». Если вызов с `caseIds` стоит в хуке, такая строка появляется у всех тестов файла.



Отвязка. «Отвязать кейс…» убирает ID везде, где он записан: и из `caseIds` в метаданных `doqa.test`, и из вызовов `doqa.metadata`. Если ID был последним в вызове, уходит ключ `caseIds`, а если в вызове больше ничего нет, удаляется весь вызов. В диалоге подтверждения сказано «id уберётся из doqa.metadata.». Если кейсы теста задаются ещё и там, где плагин их не прочитает или не поправит, диалог предупреждает: «Внимание: кейсы теста задаются и там, где плагин их не прочитает или не поправит (…) — проверьте их вручную.».



У строк `each` отвязка снимает кейс только в коде: связи строк в DoQA плагин найти не может, потому что не знает их ключей. Блок, сделанный из тела-выражения, после отвязки остаётся блоком, а добавленный импорт `doqa` остаётся в файле.

Когда плагин не пишет в код

Плагин не пишет в код и объясняет почему, если правка изменила бы поведение теста или сломала бы его.

Привязка кейса:

Что в коде	Сообщение
`test.todo`	«у test.todo нет тела: привяжите кейс, когда тест будет написан»
пропущенный тест (`xit` , `.skip`), который нельзя сделать `doqa.test` , например `test.skip.each`	«тест пропускается (skip), а doqa.metadata в его теле не выполнится — кейс не привязался бы, пока тест пропущен»
функция теста передана ссылкой, а не написана в самом вызове	«функция теста задана не в самом вызове — добавьте doqa.metadata({ caseIds: [...] }) в её тело вручную»
имя `doqa` внутри теста занято (параметр, локальное объявление)	«имя doqa внутри теста занято параметром или локальным объявлением — переименуйте его, и плагин сможет привязать кейс»

Закрепление `id` (ключ пишется только в метаданные `doqa.test` , поэтому здесь отказов больше):

Что в коде	Сообщение
`test.todo`	«у test.todo нет тела, а doqa.test требует функцию теста»
`test.failing`	«у doqa.test нет режима failing»
тест принимает колбэк `done`	«тест принимает колбэк done, а doqa.test его не передаёт — перепишите тест на async»
обычный `test.each` (каждая строка данных сейчас отдельный автотест)	«у test.each каждая строка данных — отдельный автотест, а doqa.test.each свёл бы их в один: строки потеряли бы историю»
таблица `each` задана шаблонной строкой (`test.each`…`)	«таблица each задана шаблонной строкой, а doqa.test.each принимает только массив строк»

Что в коде	Сообщение
метаданные заданы не объектом в самом вызове	«метаданные doqa.test заданы не объектом в самом вызове — поправьте их вручную»
`id` задан через `doqa.metadata` в теле теста	«id задаётся вызовом doqa.metadata в теле теста — поправьте его там»
`id` может заменить `doqa.metadata` в хуке, функции или условии	«id может заменить вызов doqa.metadata во время прогона — закреплённый ключ не сработал бы»

Отвязка отказывает, только если ID некуда убрать: ни `caseIds` в метаданных, ни вызова `doqa.metadata`, который плагин может поправить. Например, «caseIds в метаданных задан не списком чисел в самом вызове — поправьте его вручную».

При привязке кейса к тесту, у которого нельзя закрепить `id`, кейс записывается, а `id` нет.

Читаемый ключ Jest

Читаемый ключ теста Jest собирается из пути файла и названий, разделитель `>`:

```text
tests.cart > Корзина > добавляет товар
```

Первая часть это путь файла от `rootDir` через точки, без расширения и без `.test` или `.spec`: `tests/cart.test.js` даёт `tests.cart`. Точка внутри имени каталога или файла заменяется на `_`. Дальше идут названия блоков `describe` и название теста. Если у проекта есть название, оно стоит первым: `web > tests.cart > Корзина > добавляет товар`.

Читаемого ключа нет, если название теста или `describe` не строка, тест объявлен внутри функции, проект Jest не определён или ключ длиннее 255 символов. Тогда при закреплении плагин просит ввести ключ, как для теста, у которого ключ не вычисляется.

Действия с ключом в тестах Jest называются «Закрепить id у теста» и «Закрепить id во всём файле», а после записи плагин показывает «Закреплено: id: '...'». Правила выбора ключа те же, что в разделе **Читаемый ключ и закрепление @DoqaId**. **Ctrl+клик** по ID в `caseIds` и в вызове `doqa.metadata` открывает карточку кейса.

Ограничения

Если по тексту файла нельзя получить тот же внешний ID, что вычислит адаптер (например, у метода есть параметры или имя задано константой), плагин ключ не показывает и называет причину. Для таких тестов закрепите `@DoqaId`, а в Jest `id` в метаданных `doqa.test`: явный ключ плагин и адаптер читают одинаково. Причины для Jest собраны в разделе **Когда плагин не вычисляет ID**. Полный список есть в разделе **Ограничения** страницы «Помощь в редакторе и черновики тестов».

Смотрите также

- **Плагины для IDE**: установка плагина, токен и выбор пространства
- **Помощь в редакторе и черновики тестов**: черновик теста из кейса, дополнение тегов, проверка `doqa.properties`, повторяющийся `@DoqaId`
- **Каталог автотестов**: тот же каталог в веб-интерфейсе
- **Нестабильные тесты и карантин**: как DoQA определяет стабильность и что делает карантин
- **Адаптеры для тестовых фреймворков**: как адаптер вычисляет внешний ID

Помощь в редакторе и черновики тестов

Кроме значков в gutter и меню DoQA, плагин помогает прямо при наборе кода. Он пишет черновик теста по ручному кейсу, дополняет теги и метки из каталога, проверяет `doqa.properties` и предупреждает, если у двух тестов одинаковый внешний ID. Здесь же собраны ограничения плагина.

Внешний ID автотеста плагин называет «ключом автотеста». Это одно и то же: значение `@DoqaId` в Java и Kotlin, `@doqa.id` в pytest, ключ `id` в метаданных `doqa.test` в Jest. Что такое внешний ID и откуда он берётся, описано на странице **Автотесты**.

Тест из кейса

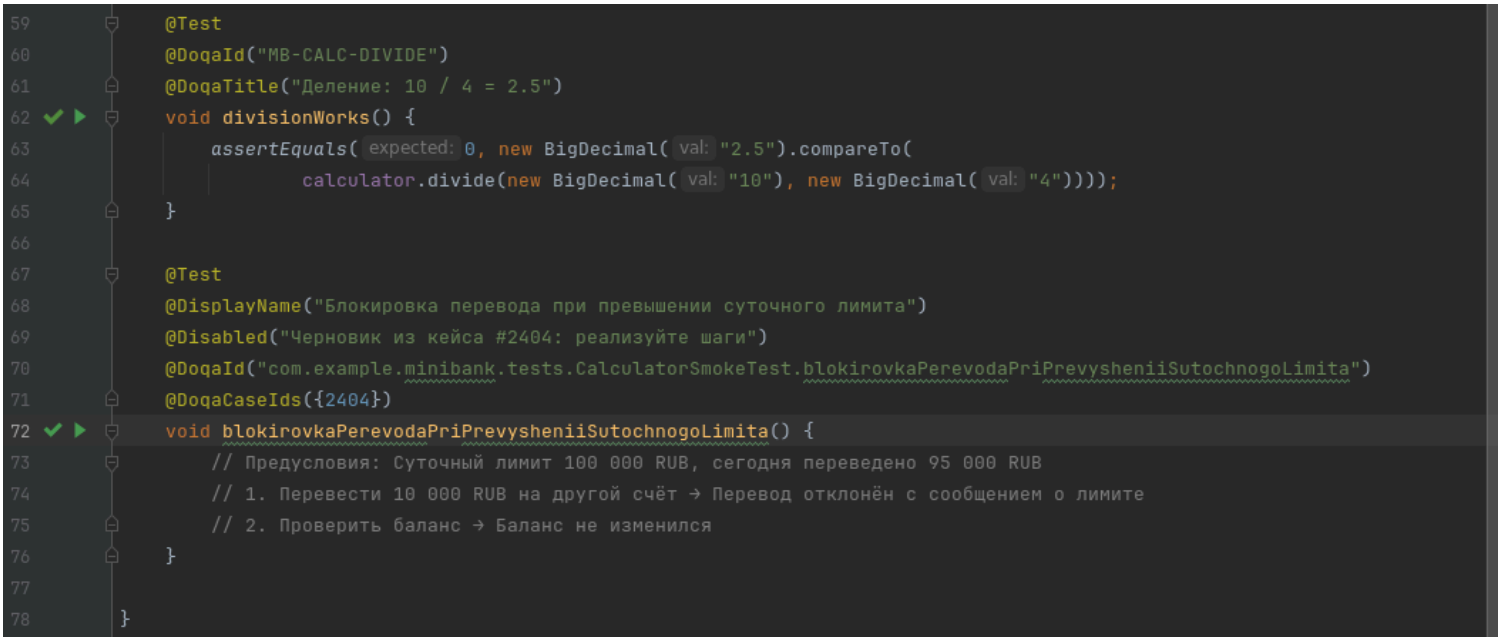
Действие «**Создать тест из кейса...**» пишет в открытый файл заготовку автотеста по ручному **тест-кейсу**: название кейса, привязку к нему, внешний ID, предусловия и шаги комментариями. Остаётся написать сам тест.

Как создать черновик

1. Откройте файл с тестами (`.java` , `.kt` , `.kts` , `.py` или файл JavaScript и TypeScript с тестами Jest) и поставьте курсор на пустую строку между тестами, туда, где нужен новый тест.
2. Откройте меню DoQA и выберите «**Создать тест из кейса...**». Где найти меню, описано на странице **Плагины для IDE**.
3. В окне «DoQA: тест из кейса» начните вводить название кейса: список обновляется по мере ввода. Стрелками выберите кейс и нажмите **Enter**.

Плагин вставит черновик, поставит курсор на объявление теста и покажет подсказку «Черновик из кейса #N: реализуйте шаги и включите тест. Отменить — Ctrl+Z». Вся вставка, вместе с импортами, отменяется одним **Ctrl+Z**.

Действие доступно в любом файле Java, Kotlin или Python, даже если тестов в нём пока нет, а также в файлах Jest. Если плагин не настроен, вместо окна поиска появится уведомление «DoQA не настроен: укажите адрес, токен и space.». Настройка описана на странице **Плагины для IDE**.



```
    @Test
    @DoqaId("MB-CALC-DIVIDE")
    @DoqaTitle("Деление: 10 / 4 = 2.5")
    void divisionWorks() {
        assertEquals( expected: 0, new BigDecimal( val: "2.5").compareTo(
                calculator.divide(new BigDecimal( val: "10"), new BigDecimal( val: "4"))));
    }

    @Test
    @DisplayName("Блокировка перевода при превышении суточного лимита")
    @Disabled("Черновик из кейса #2404: реализуйте шаги")
    @DoqaId("com.example.minibank.tests.CalculatorSmokeTest.blokirovkaPerevodaPriPrevysheniiSutochnogoLimita")
    @DoqaCaseIds({2404})
    void blokirovkaPerevodaPriPrevysheniiSutochnogoLimita() {
        // Предусловия: Суточный лимит 100 000 RUB, сегодня переведено 95 000 RUB
        // 1. Перевести 10 000 RUB на другой счёт → Перевод отклонён с сообщением о лимите
        // 2. Проверить баланс → Баланс не изменился
    }

}
```

Что попадает в черновик

Разметка зависит от фреймворка. Фреймворк плагин берёт у теста под курсором, а если курсор не на тесте, то у первого теста в файле. Если тестов нет, он смотрит на импорты: по `org.testng` узнаёт TestNG, по `org.junit.jupiter` JUnit 5, по `org.junit.Test` JUnit 4. Если по импортам не понять, пишется тест JUnit 5. В файлах `.py` всегда пишется тест pytest, в файлах JavaScript и TypeScript тест Jest.

	JUnit 5	JUnit 4	TestNG	pytest
Название кейса	`@DisplayName("…")`	не пишется	`@Test(description = "…")`	`@doqa.title("…`
Как тест выключен	`@Disabled("Черновик из кейса #N: реализуйте шаги")`	`@Ignore("Черновик из кейса #N: реализуйте шаги")`	`enabled = false` в `@Test`	`@pytest.mark.s из кейса #N: pe`
Привязка к кейсу	`@DoqaCaseIds({N})`	`@DoqaCaseIds({N})`	`@DoqaCaseIds({N})`	`@doqa.case_ids`
Внешний ID	`@DoqaId("<класс>.<метод>")`	`@DoqaId("<класс>.<метод>")`	`@DoqaId("<класс>.<метод>")`	`@doqa.id("<мод`

	JUnit 5	JUnit 4	TestNG	pytest
Имя	транслитерация названия	транслитерация названия	транслитерация названия	`test_` + трансл

В Kotlin разметка та же, что в Java, с двумя отличиями: привязка пишется без фигурных скобок, `@DoqaCaseIds(N)`, а именем функции становится само название кейса в обратных кавычках, например `fun `Оплата картой с 3D Secure`()`. Символы, которые JVM не допускает в именах (точка, двоеточие, косая черта, угловые и квадратные скобки и т. п.), заменяются пробелами.

Внешний ID в Java и Kotlin пишется всегда: класс берётся у теста под курсором или у первого теста файла, а если тестов нет, то из пакета и имени файла. В pytest `@doqa.id` появляется, только если плагин нашёл rootdir pytest (`pytest.ini`, `pyproject.toml`, `tox.ini`, `setup.cfg` или `setup.py`): без него путь модуля неизвестен, и плагин лучше не напишет ключ, чем напишет неверный. Если модуль импортирует `doqa` под другим именем (`import doqa as dq`), декораторы пишутся через это имя (`@dq.id`).

В Jest черновик это вызов `doqa.test.skip` с метаданными `id` и `caseIds`. Название теста это название кейса без транслитерации. Внешний ID это **читаемый ключ Jest**: путь файла, цепочка `describe` и название. Если путь файла в проекте Jest или цепочку `describe` определить нельзя (например, не найден конфиг Jest), плагин пишет только `caseIds`, без `id`. Если `doqa` в файле ещё не импортирован, плагин добавит импорт в стиле файла: `import { doqa } from '@doqa-tms/jest';` или `const { doqa } = require('@doqa-tms/jest');`. Если имя `doqa` в файле занято чем-то другим, адаптер импортируется как `doqaTms`.

Если транслитерация названия совпадает с ключевым словом Java (кейс «Default», «Return»), к имени метода добавляется `case`: `caseDefault`.

В тело теста пишутся комментарии:

- `Предусловия: …`, если в кейсе есть предусловия;
- по строке на каждый шаг: номер, действие и через `→` ожидаемый результат. Многострочный шаг склеивается в одну строку;
- `Шагов в кейсе нет: опишите проверку`, если шагов нет.

Недостающие импорты аннотаций, а в pytest `import pytest` и `import doqa`, плагин добавляет сам.

Пример: кейс #1041 «Оплата картой с 3D Secure» в классе `com.acme.PaymentTest` на JUnit 5.

```java
@Test
@DisplayName("Оплата картой с 3D Secure")
@Disabled("Черновик из кейса #1041: реализуйте шаги")
@DoqaId("com.acme.PaymentTest.oplataKartoyS3dSecure")
@DoqaCaseIds({1041})
void oplataKartoyS3dSecure() {
    // Предусловия: пользователь авторизован, в профиле сохранена карта с 3D Secure
    // 1. Открыть корзину и нажать «Оплатить» → открыта форма оплаты
    // 2. Выбрать сохранённую карту → открыта страница подтверждения банка
    // 3. Ввести код из SMS → заказ оплачен
}
```

Тот же кейс в pytest, в модуле `tests/test_payment.py`:

```python
@pytest.mark.skip(reason="Черновик из кейса #1041: реализуйте шаги")
@doqa.title("Оплата картой с 3D Secure")
@doqa.id("tests.test_payment.test_oplata_kartoy_s_3d_secure")
@doqa.case_ids(1041)
def test_oplata_kartoy_s_3d_secure():
    # Предусловия: пользователь авторизован, в профиле сохранена карта с 3D Secure
    # 1. Открыть корзину и нажать «Оплатить» → открыта форма оплаты
    # 2. Выбрать сохранённую карту → открыта страница подтверждения банка
    # 3. Ввести код из SMS → заказ оплачен
    pass
```

Тот же кейс в Jest, в блоке `describe('Оплата')` файла `tests/payment.test.js`:

```js
describe('Оплата', () => {
  doqa.test.skip('Оплата картой с 3D Secure', { id: 'tests.payment > Оплата > Оплата к
    // Черновик из кейса #1041: реализуйте шаги
    // Предусловия: пользователь авторизован, в профиле сохранена карта с 3D Secure
    // 1. Открыть корзину и нажать «Оплатить» → открыта форма оплаты
    // 2. Выбрать сохранённую карту → открыта страница подтверждения банка
```

```
    // 3. Ввести код из SMS → заказ оплачен
  });
});
```

Почему тест выключен

Пустой тест прошёл бы с первого запуска и отчитался бы в DoQA за кейс, который никто ещё не автоматизировал. Поэтому черновик выключен до тех пор, пока вы не напишете проверку. Когда тест готов, удалите `@Disabled`, `@Ignore`, `enabled = false`, `@pytest.mark.skip` или `.skip` у `doqa.test.skip`.

Внешний ID фиксируется при первом прогоне

Черновик сразу получает читаемый внешний ID: полное имя теста. Если вы переименуете метод до первого прогона, поправьте и `@DoqaId` (в Jest `id`): значение в аннотации не обновляется само, а после первого прогона смена ключа создаст в DoQA новый автотест без истории.

Куда вставляется черновик

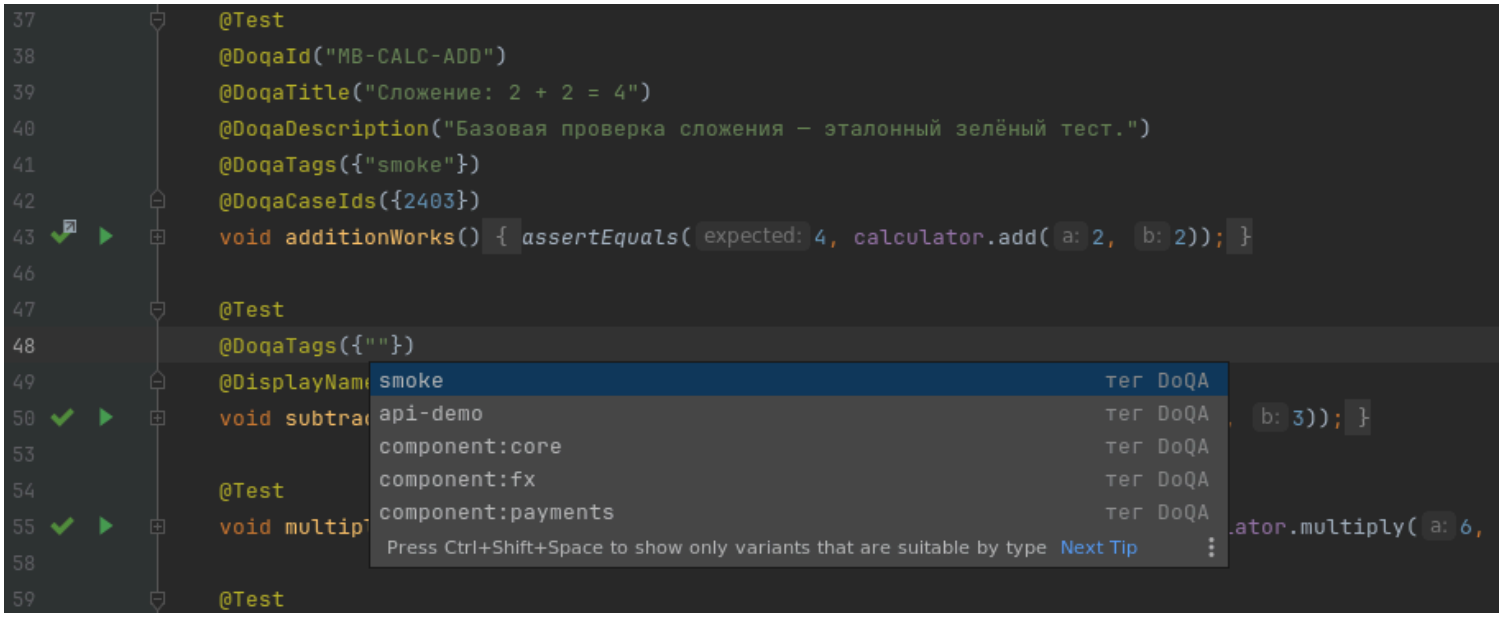
- Если курсор стоит на пустой строке между членами класса, черновик встаёт на неё. Пустая строка внутри метода или между аннотацией и методом не подходит.
- Иначе черновик встаёт перед следующим после курсора тестом того же класса, а если его нет, то в конец этого класса. Черновик остаётся в том же классе, вложенном или внешнем; функции Kotlin вне класса его не притягивают.
- В pytest черновик это функция модуля: он встаёт на пустую строку вне класса и функции ниже импортов, иначе в конец файла.
- В Jest черновик встаёт в блок `describe`, в котором стоит курсор: на пустую строку между тестами самого блока (пустая строка внутри объекта, условия, функции или многострочной строки не подходит), иначе перед следующим тестом этого блока, иначе перед закрывающей скобкой блока. Если курсор вне `describe`, черновик встаёт на пустую строку под курсором ниже импортов, иначе перед следующим тестом верхнего уровня, иначе в конец файла. Внутрь теста черновик не попадает.

Пустые строки вокруг черновика плагин добавляет с учётом тех, что уже есть: одну между членами класса в Java и Kotlin, две вокруг функции в pytest, одну между тестами в Jest.

В Java имя метода это транслитерация названия кейса в camelCase: «Оплата картой с 3D Secure» → `oplataKartoyS3dSecure`. Если название пустое после транслитерации, имя будет `caseN`, в pytest `test_case_N`. Если метод с таким именем в файле уже есть, к имени добавляется номер: `oplataKartoyS3dSecure2`, в pytest `test_…_2`. В Jest название теста это название кейса; если в блоке уже есть тест с таким названием, к нему добавляется номер в скобках: `Оплата картой с 3D Secure (2)`.

Дополнение тегов и меток

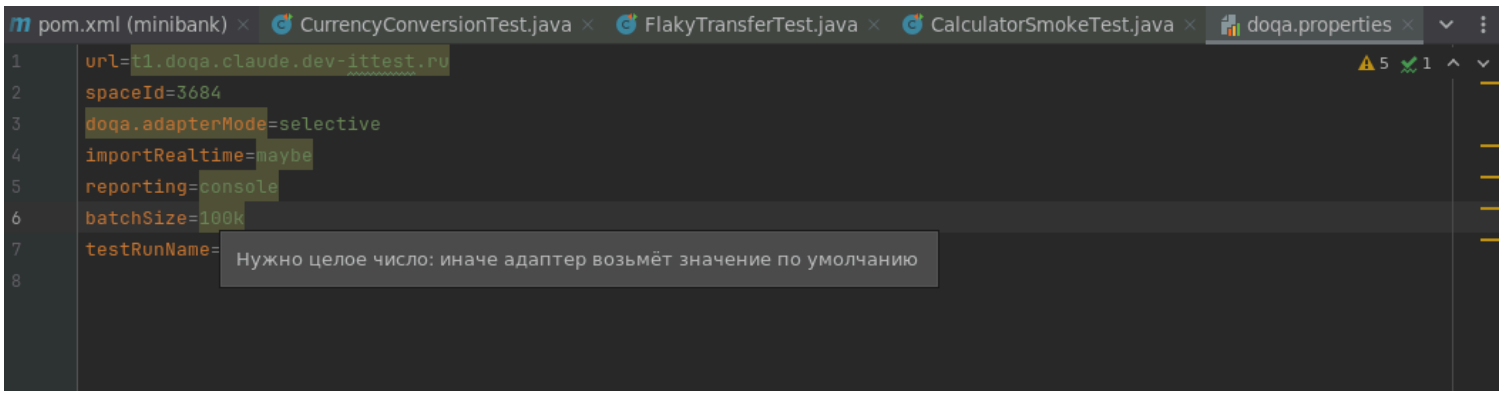
Внутри строки в `@DoqaTags` и `@DoqaLabels` (в pytest в `doqa.tag` и `doqa.label`, под каким бы именем ни был импортирован модуль `doqa`, в Jest в массивах `tags: [...]` и `labels: [...]` метаданных `doqa.test` и `doqa.metadata`) плагин предлагает теги и метки, которые уже есть у автотестов в каталоге пространства. Список открывается сам сразу после открывающей кавычки; в строке списка справа подписано «тег DoQA» или «метка DoQA». В `@DoqaTags` и `tags` предлагаются только теги, в `@DoqaLabels` и `labels` только метки. В Jest дополнение работает только в файле, который импортирует `doqa` из `@doqa-tms/jest`: массив `tags: [` встречается и в коде, не связанном с DoQA. Другие варианты дополнения, например слова из файла, внутри этих аннотаций не показываются.



Список тегов и меток плагин загружает при первом дополнении и обновляет не чаще раза в 10 минут. Теги автотестов из открытых файлов добавляются в список, когда плагин обновляет их статусы. Если DoQA недоступен или в каталоге нет ни одного тега и метки, плагину нечего предложить. Чем теги отличаются от меток, описано на страницах адаптеров, например Адаптер JUnit 5.

Проверка doqa.properties

В файле `doqa.properties` плагин подсвечивает строки, которые адаптер DoQA прочитал бы не так, как вы ожидаете. Плагин знает ключи JVM-адаптеров и адаптера Jest `@doqa-tms/jest`. Наведите курсор на подсвеченное место, чтобы увидеть текст предупреждения.



Что не так	Текст предупреждения
Ключ, которого адаптер не знает	«Адаптер DoQA не знает ключ «...»: он будет пропущен»
Ключ с префиксом `doqa.`	«В файле ключи пишутся без префикса doqa.: адаптер не узнает «...»»
Значение ссылается на переменную: `${…}` или `$NAME`	«Ссылки на переменные в файле не раскрываются: адаптер сочтёт ключ незаданным»
Не номер из цифр в ID (`spaceId` , `configurationId` , `testRunId` , `ciRunId`)	«Нужен номер из цифр: адаптер передаст значение в DoQA как есть»
Не целое число или число вне диапазона в числовой настройке (`batchSize` , `retries` , `requestTimeoutMs` и др.)	«Нужно целое число: иначе адаптер возьмёт значение по умолчанию»

Что не так	Текст предупреждения
Непонятное значение флага (`importRealtime` , `certValidation`)	«Адаптер поймёт «...» как false: пишите true или false»
Неизвестный режим в `adapterMode`	«Режим: 0 или selective, 1 или existing, 2 или new»
Неизвестное значение `reporting`	«Варианты: auto, api, files, off; иначе сработает auto»
Неизвестное значение `executionOrder`	«Варианты: jest или plan; иначе сработает порядок Jest»
`url` без `http://` или `https://`	«Адрес начинается с http:// или https://»
`token` в файле под контролем версий	«Токен в файле под контролем версий: передайте его через DOQA_TOKEN»

В строке показывается одно предупреждение: сначала проверяется ключ, потом значение. Для флагов адаптер понимает `true` / `false` , `yes` / `no` , `on` / `off` , `y` / `n` и `1` / `0` , для `adapterMode` числа 0–2 и названия режимов, для `executionOrder` значения `jest` и `plan` в любом регистре и с пробелами по краям, как их читает адаптер Jest.

Значения плагин читает так же, как адаптер: раскрывает экранирования (`\uXXXX` , `\:`), склеивает строку, продолженную обратной косой чертой, а числа разбирает как Java, поэтому `01` и `+1` в `adapterMode` верны.

Регистр, дефисы и подчёркивания в именах ключей адаптер не различает, плагин тоже: `spaceId` , `space_id` и `SPACE-ID` это один ключ. Старые имена `privateToken` и `projectId` плагин считает синонимами `token` и `spaceId` . Пустое значение не проверяется.

Два ключа нужны только адаптеру Jest:

Ключ	Что задаёт	Что проверяет плагин
`projectName`	название проекта Jest. Оно входит в **ключ автотеста**, и плагин берёт его отсюда, если оно не задано во втором аргументе `withDoqa`	как любой текст: только ссылки на переменные

Ключ	Что задаёт	Что проверяет плагин
`executionOrder`	порядок тестов: `jest` (по умолчанию) или `plan`, в порядке плана DoQA	значение `jest` или `plan`

Эти ключи плагин не подсвечивает и в `doqa.properties` проектов на JUnit, TestNG и pytest: проверка общая для всех адаптеров.



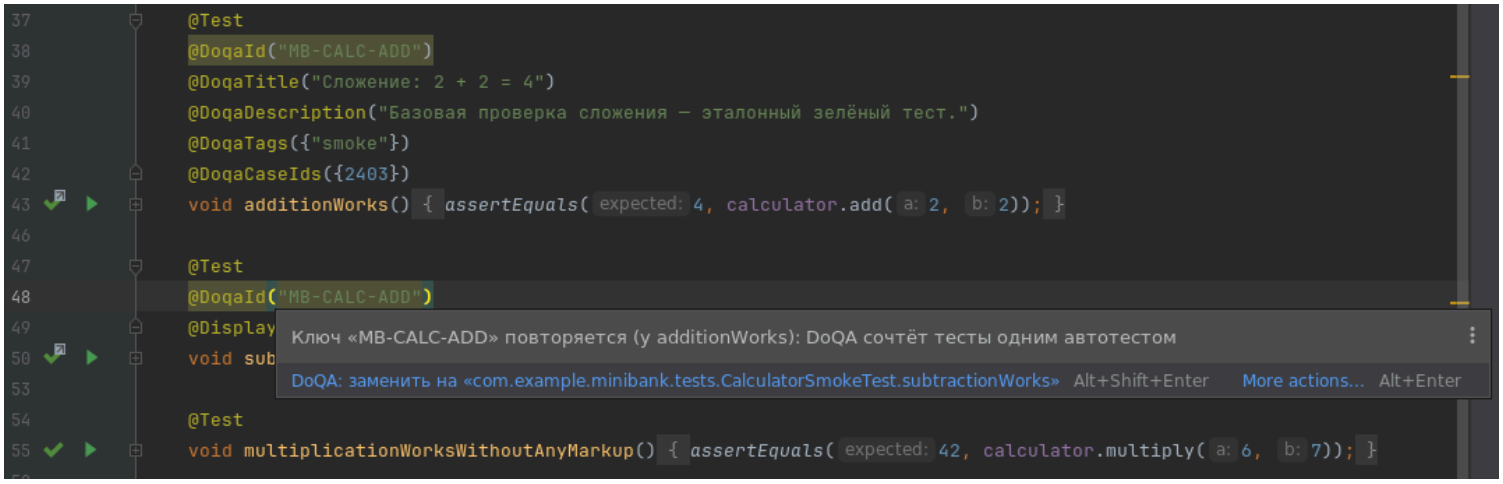
Токен считается закоммиченным, если файл лежит в репозитории системы контроля версий, не игнорируется и уже добавлен в неё. Для файла из `.gitignore` предупреждения нет.

Имена ключей и значения тоже дополняются. На месте ключа плагин предлагает ключи, которых в файле ещё нет, с коротким описанием каждого. После `=` или `:` предлагаются допустимые значения: `true` / `false` для флагов, `selective`, `existing` и `new` для `adapterMode`, `auto`, `api`, `files` и `off` для `reporting`, `jest` и `plan` для `executionOrder`. Полный список ключей и их смысл есть на странице **Адаптер JUnit 5**.

> **Нужен плагин Properties**
>
> Проверка и дополнение работают, только если в IDE есть плагин Properties. В IntelliJ IDEA он встроен, в других IDE его может не быть.

Повторяющийся внешний ID

Если у двух тестов одинаковый `@DoqaId` (в pytest `@doqa.id`, в Jest `id` в метаданных `doqa.test`), DoQA сочтёт их одним автотестом и смешает их прогоны. Инспекция «Повторяющийся ключ автотеста (@DoqaId, doqa.id)» находит такие повторы в открытом файле и в остальных файлах проекта и подсвечивает значение аннотации. Текст предупреждения: «Ключ «…» повторяется (…): DoQA сочтёт тесты одним автотестом». В скобках перечислены другие тесты этого файла с тем же ключом («у имяМетода») и другие файлы, где он встречается («в ИмяФайла.java»).



Чтобы исправить повтор, поставьте курсор на подсвеченное значение, нажмите **Alt+Enter** и выберите «DoQA: заменить на «...»». Плагин запишет вместо повторяющегося ключа читаемый: полное имя теста, например `com.acme.CheckoutTest.addsItemToCart`, в Jest `tests.cart > Корзина > добавляет товар`. Этого варианта нет, если у теста уже стоит его читаемый ключ: тогда поменяйте ключ у другого теста или впишите свой.

Повторы в других файлах плагин находит после того, как в фоне прочитает файлы проекта. Сразу после открытия проекта предупреждение может появиться с задержкой. Из файлов JavaScript и TypeScript заранее читаются только тестовые файлы по шаблонам Jest по умолчанию (`__tests__`, `*.test.*`, `*.spec.*`). Тесты из файлов с другими именами, например по своему `testMatch`, участвуют в проверке, когда такой файл открыт в редакторе. Инспекция включена по умолчанию; её уровень меняется в «Settings» → «Editor» → «Inspections», группа «DoQA».

Смена ключа у существующего автотеста

Если автотест с этим ключом уже есть в DoQA, новый ключ станет новым автотестом, и история прогонов останется у старого. Меняйте ключ у того теста, который появился позже.

Ограничения

Поддерживаются Java, Kotlin, Python, JavaScript и TypeScript. Плагин понимает тесты JUnit 5, JUnit 4, TestNG, pytest и Jest, то есть те же фреймворки, для которых есть **адаптеры** (для Jest это адаптер `@doqa-tms/jest`, см. **README адаптера Jest** ↗). Другие фреймворки

JavaScript, например Vitest или Mocha, плагин не видит. Тесты JUnit 3 (`extends TestCase` без аннотаций) плагин не видит.

Плагин читает текст файла. Он не опирается на компилятор, поэтому работает в любой IDE на платформе IntelliJ. Необычно отформатированное объявление теста плагин пропускает, а не размечает наугад.

Наследование из других файлов не видно. Тесты базового класса не показываются у наследника. Если `@RunWith(Parameterized.class)` или `@Test` на классе достались от базового класса, внешний ID, который вычислил плагин, может не совпасть с тем, что пришлёт адаптер. В таких классах закрепите `@DoqaId`: явный ключ плагин и адаптер читают одинаково.

В pytest плагин знает только сам файл и конфигурацию pytest. Ему не видны `conftest.py`, плагины pytest, `pytest_generate_tests`, `--rootdir` в командной строке, параметры из других модулей и метки `allure.id` базового класса из другого модуля. Не показываются тесты базового класса у наследника, тесты-лямбды (`test_x = lambda: …`) и алиасы (`test_b = test_a`). Имена сопоставляются с масками `python_files`, `python_classes` и `python_functions` с учётом регистра, как pytest на Linux и macOS. Значки в gutter для Python есть в PyCharm и в IDE с плагином Python; действия меню работают в любой IDE.

В Jest плагин знает только сам файл и ближайший конфиг Jest. Ему не видны переменные окружения (`DOQA_PROJECT_NAME`, `DOQA_CONFIG`), флаги командной строки Jest (`--rootDir`, `--config`, `--selectProjects`) и конфиг выше ближайшего. Файл `doqa.properties` плагин ищет в каталоге конфига, как будто Jest запускают оттуда. Вызов `doqa.metadata({ id })` или `doqa.metadata({ caseIds })` из функции другого модуля ему тоже не виден. Если название проекта задаётся переменной `DOQA_PROJECT_NAME`, хэш у плагина и у адаптера разойдётся: задайте `projectName` во втором аргументе `withDoqa` или закрепите `id`. Значки в gutter для JavaScript и TypeScript есть в IntelliJ IDEA Ultimate, WebStorm и других IDE с плагином JavaScript; действия меню работают в любой IDE.

Кейсы из `doqa.metadata` в Jest. Вызов `doqa.metadata({ caseIds })` плагин засчитывает в любом месте верхнего уровня тела теста. Если тест упадёт раньше вызова, этот результат придёт в DoQA без кейсов; сам плагин всегда ставит вызов первой строкой. У пропущенного теста (`xit`, `.skip`) плагин показывает кейсы из `doqa.metadata`, хотя адаптер их не получит,

пока тест пропущен. Вызов `doqa.metadata` без `caseIds` (например, только с `title`) плагин не дополняет, а вставляет рядом новый.

В Jest плагин отказывается там, где ID неоднозначен. Обычный `test.each` без плейсхолдеров в названии, метаданные вида `{ id, ...CONST }`, `id`, заданный выражением, константа метаданных или конфига, которую меняют после объявления (даже если меняют другой ключ), `doqa.test.each` с подстановкой в названии и данными из переменной и `doqa.metadata({ id })` в любом хуке (даже в `beforeAll`, где `id` на деле не меняется) плагин показывает без ключа, хотя адаптер в этих случаях выдал бы один ID. Причины перечислены в разделе **Когда плагин не вычисляет ID**.

pytest без аргументов из каталога выше конфигурации. Если `pytest.ini` лежит в `tests/`, а `pytest` без аргументов запускают из корня репозитория, pytest этот файл не видит, и внешние ID у него получатся другие. Плагин показывает ID для запуска из `tests/` или с аргументом (`pytest tests`).

Значки в gutter показывают сохранённые данные. Каталог плагин загружает в фоне при открытии файла и по действию «Обновить статусы в файле». Пока данных нет, значок показывает только то, что размечено в коде.

История прогонов, отвязка кейса и карантин требуют версии DoQA, которая принимает для них API-токен. Остальные запросы плагина сервер принимает с персональным API-токеном, а эти три на части серверов пока принимает только из веб-интерфейса. Тогда плагин не станет говорить, что токен недействителен, и не предложит открыть настройки, а покажет сообщение:

> Сервер DoQA пока не принимает API-токен для истории прогонов: эта возможность появится после обновления DoQA

Для отвязки кейса и карантина в сообщении вместо истории прогонов названо само действие. Токен менять не нужно: всё остальное в плагине работает. Когда сервер обновят, эти действия заработают без перенастройки плагина. На совсем старых версиях DoQA история может показать падение без стека.

Похожие кейсы подбираются по словам названия, а не по смыслу. Как это работает, описано в разделе **Привязка кейса**.

Дополнение тегов и меток видит первые 200 автотестов пространства. Теги и метки, которые есть только у автотестов дальше в каталоге, предлагаются, лишь если эти автотесты уже встречались плагину в открытых файлах.

Поиск кейсов ограничен. Поиск останавливается на 50 кейсах. Если нашлось больше, внизу окна появится «Показаны не все совпадения — уточните запрос».

Привязка только добавляет связи. Привязка кейса ставит ему статус автоматизации «Автоматизирован» и не снимает связи других кейсов с этим автотестом. Чтобы убрать связь, используйте «Отвязать кейс...».

Пока нет: проверки «кейс устарел» и окна с деревом покрытия.

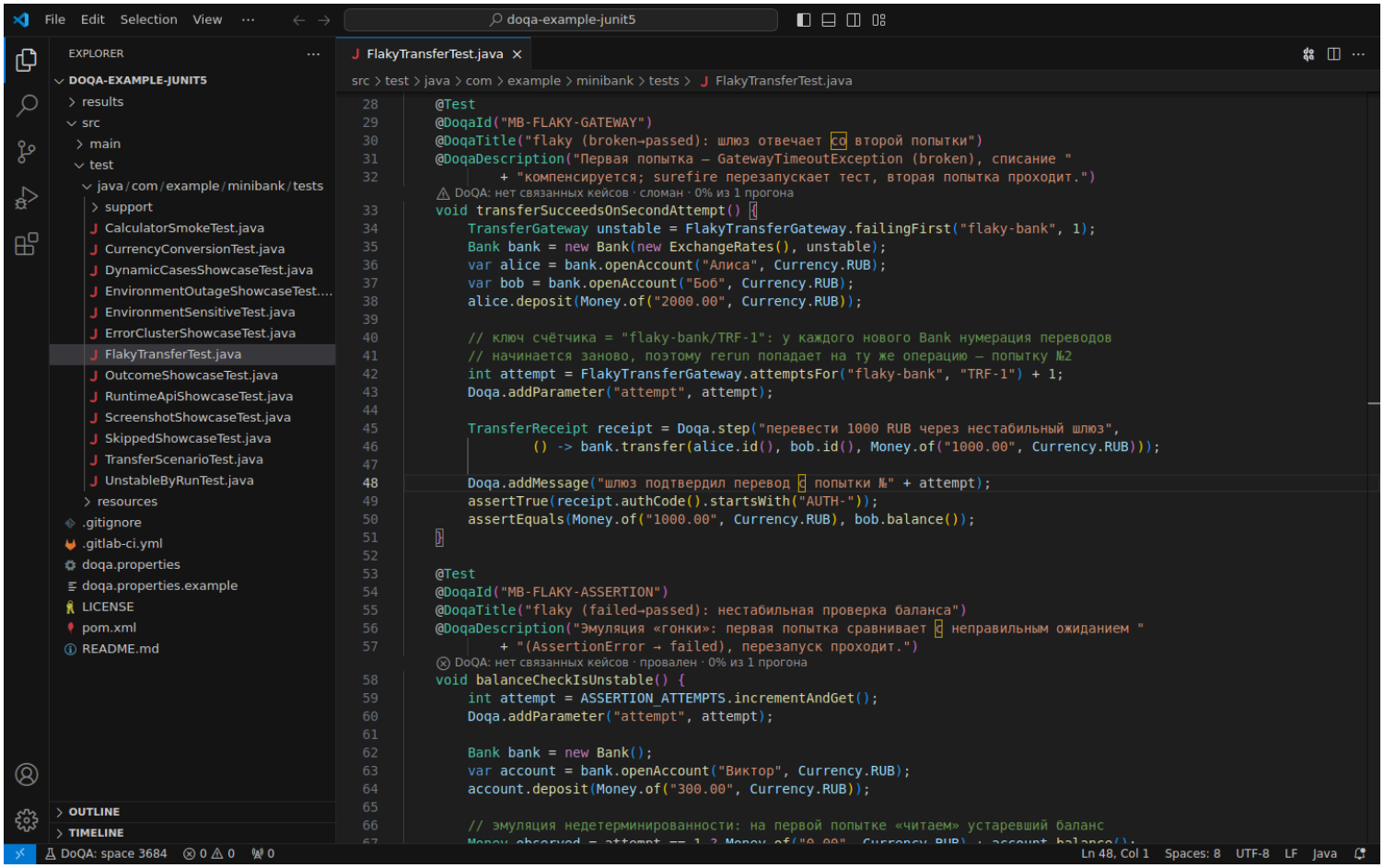
Смотрите также

- **Плагины для IDE**: установка плагина, токен и выбор пространства
- **Работа с тестами в IntelliJ**: значки в gutter, привязка кейса, закрепление `@DoqaId`, история прогонов, **тесты Jest**
- **Автотесты**: что такое внешний ID и как DoQA опознаёт автотест
- **Адаптеры для тестовых фреймворков**: разметка, которую пишет и читает плагин
- **Каталог автотестов**: откуда берутся теги, метки и статусы

Плагин для VS Code

Плагин DoQA для VS Code умеет то же, что плагин для IntelliJ: показывает над каждым тестом последний прогон, стабильность и привязанный **тест-кейс**, привязывает и отвязывает кейсы, закрепляет внешний ID, открывает историю прогонов со стеком падения и ставит автотест в карантин. Отличается интерфейс: вместо значков в gutter над тестом стоит строка CodeLens, а меню действий открывается списком быстрого выбора, как палитра команд.

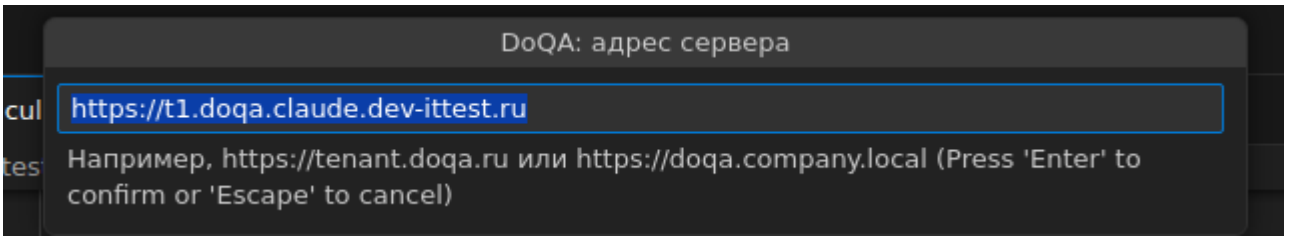
Как установить плагин, описано в разделе **VS Code** страницы «Плагины для IDE», а где взять персональный API-токен, сказано в разделе **Токен**. Эта страница про настройку подключения в VS Code и работу с тестами.



Подключение к DoQA

Выполните команду «**DoQA: Настроить (адрес, токен, space)**» из палитры команд (**Ctrl+Shift+P**) или щёлкните по «DoQA» в строке состояния. Плагин по очереди спросит три вещи в строке ввода вверху окна:

1. «**DoQA: адрес сервера**»: адрес облака вашей компании или своей установки DoQA, например `https://tenant.doqa.ru`. Адрес должен начинаться с `https://` или `http://`, иначе появится «Введите полный адрес http(s)».
2. «**DoQA: персональный токен**»: персональный API-токен из профиля. Токен начинается с `doqa_`, другой плагин не примет: «Токен начинается с doqa_».
3. «**DoQA: space по умолчанию**»: список пространств, доступных вам. Выберите то, в котором лежат автотесты и кейсы.



После выбора пространства появится уведомление «DoQA: вход выполнен как <почта>.», и плагин сразу загрузит статусы тестов открытого файла. Если пространств нет, будет «DoQA: вход выполнен как <почта>, но доступных space нет.»: попросите администратора добавить вас в нужное пространство.

Адрес и токен сохраняются, только когда введены оба: **Esc** на шаге токена оставляет прежние настройки. Если вы сменили адрес, пространство по умолчанию сбрасывается: номера пространств у разных установок DoQA свои.

Где хранится токен

Токен хранится в хранилище секретов VS Code, то есть в ключнице операционной системы, и привязан к адресу DoQA. В `settings.json` он не попадает и через синхронизацию настроек не уходит. Чтобы заменить токен, выполните «DoQA: Настроить» ещё раз.

Строка состояния

Слева в строке состояния плагин показывает пространство открытого файла: «**DoQA: space 3684**». Подсказка при наведении показывает адрес DoQA и номер проекта. Пока адрес или пространство не заданы, там написано «**DoQA: войти**» с подсказкой «Укажите адрес DoQA и токен». Щелчок по строке в обоих случаях запускает «DoQA: Настроить».

Баннера «DoQA не настроен» над файлом, как в IntelliJ, в VS Code нет. Команды, которым нужен сервер, без настройки показывают предупреждение, например «DoQA: сначала

выполните «DoQA: Настроить» (нужны адрес сервера, токен и space).», а если не задан токен, показывают ошибку «DoQA: Не задан токен DoQA» с кнопкой «Открыть настройки».

Space для папки или рабочей области

«DoQA: Настроить» задаёт пространство по умолчанию для всех проектов. Если тесты одного репозитория отчитываются в другое пространство, укажите его в настройках рабочей области или папки, в файле `.vscode/settings.json`:

```json
{
  "doqa.spaceId": 3684,
  "doqa.projectId": 812
}
```

Файл с тестами работает с пространством своей папки рабочей области, если оно задано, иначе с пространством рабочей области, иначе с `doqa.spaceId` из пользовательских настроек, иначе с пространством по умолчанию. `doqa.projectId` берётся с того же уровня, что и `doqa.spaceId`: он нужен только для ссылок в веб-интерфейс. Без него плагин работает, но в подсказке не будет ссылок «Кейс в DoQA» и «Автотест в DoQA», а в карточке кейса и истории прогонов не будет ссылки «Открыть в DoQA». Номер проекта виден в адресе страницы DoQA: `…/home/detail/<проект>/<пространство>/…`.

Добавьте `.vscode/settings.json` в репозиторий, и вся команда получит те же пространства вместе с кодом. Адрес DoQA и токен в этот файл не попадают.

Что видно над тестом

Строка CodeLens

Над строкой с объявлением каждого тестового метода, под его аннотациями, плагин пишет строку CodeLens, например:

```text
DoQA: кейс 2403 · пройден · 100% из 21 прогона
```

```
DoQA: нет связанных кейсов · провален · 67% из 3 прогонов · нестабильный
```

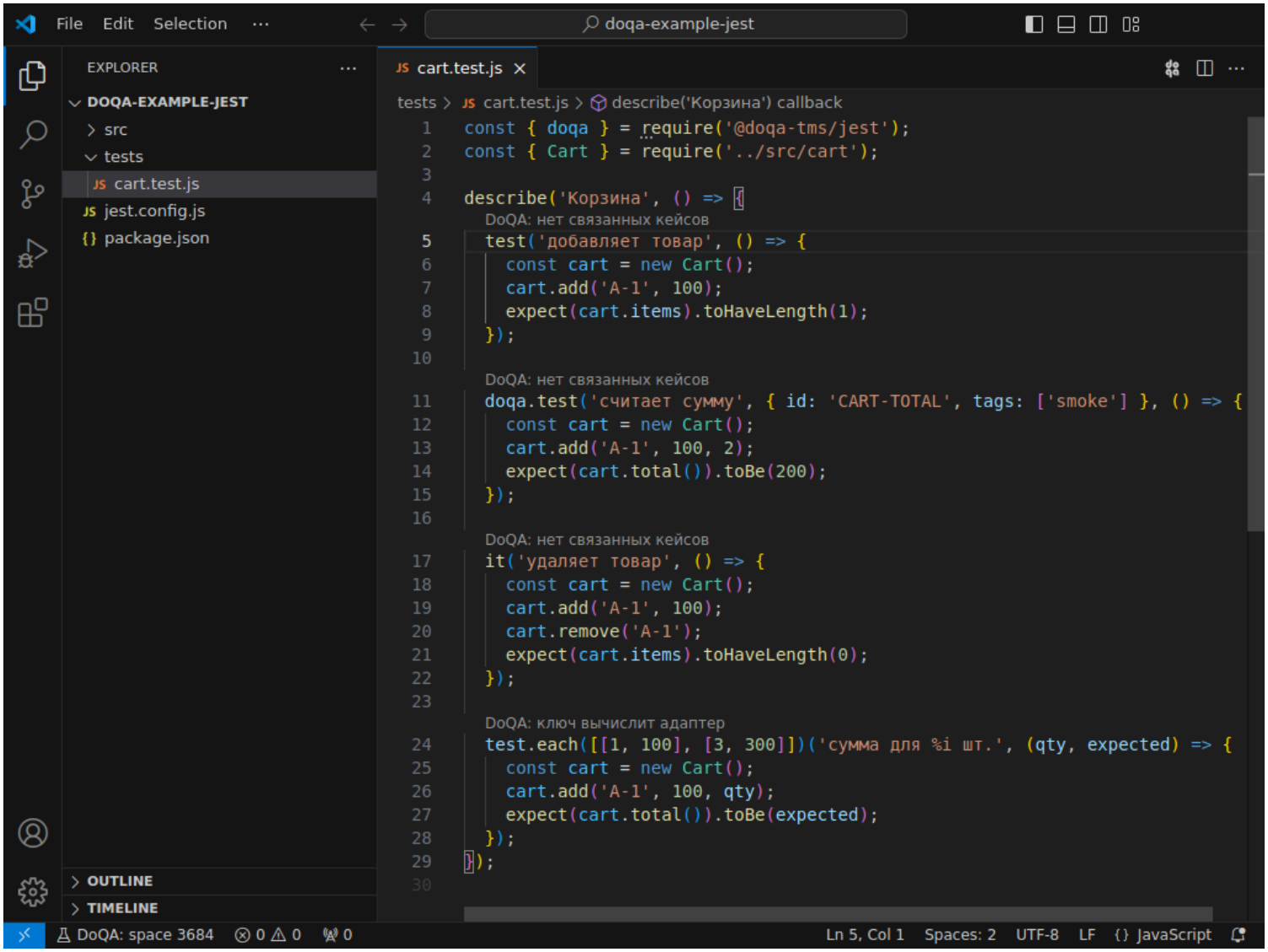
Первая часть коротко называет связанные кейсы: те, что связаны с автотестом в DoQA («DoQA: кейс 2403»), а если связей в DoQA ещё нет, то ID из `@DoqaCaseIds` («DoQA: кейс 15 из @DoqaCaseIds ещё не привязан»). Если каталог по тесту ещё не ответил, там стоит «DoQA: каталог не проверен», «DoQA: ключ вычислит адаптер» или «DoQA: ключ с плейсхолдером». Полная формулировка с пояснениями есть в **подсказке над тестом**. Дальше идут исход последнего прогона, стабильность и оценка DoQA. Исходы и оценки называются так же, как в веб-интерфейсе DoQA, см. **Статусы как в DoQA**. У теста, который есть в каталоге, но ни разу не запускался, стоит «ещё не запускался».

Значок в начале строки показывает исход последнего прогона:

Значок	Что значит
кружок с галочкой	последний прогон пройден
кружок с крестиком	последний прогон провален
треугольник с «!»	последний прогон сломан или заблокирован
перечёркнутый круг	последний прогон пропущен
пауза	автотест в карантине, исход последнего прогона значок не показывает
знак вопроса	DoQA вернул исход, которого плагин не знает

Тест, который ещё не запускался или которого плагин пока не нашёл в каталоге, показывает строку без значка, только с привязкой.

В файлах Jest строка CodeLens стоит над вызовом `test`, `it` или `doqa.test`. Над тестом, ключ которого плагин не вычисляет (например, над обычным `test.each`), написано «DoQA: ключ вычислит адаптер». Какие тесты Jest видит плагин и как он вычисляет их ключи, описано в разделе **Тесты Jest** страницы про IntelliJ.



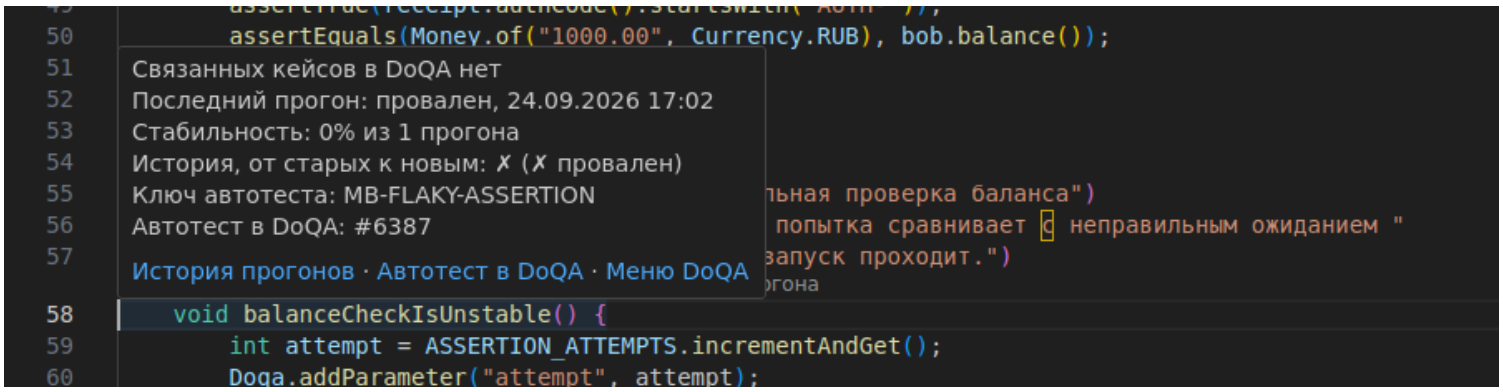
Щелчок по строке CodeLens открывает **меню действий** над этим тестом. Как DoQA считает стабильность и карантин, описано на странице Нестабильные тесты и карантин.

Подсказка над тестом

Наведите курсор на объявление теста или на его аннотации, и появится подсказка DoQA. Над телом метода она не показывается.

Строка	Пример и смысл
первые строки	связанные кейсы, см. **Связанные кейсы в подсказке**
«Последний прогон:»	исход, дата и время, ветка; «новый сбой», если до этого тест не падал. У теста без прогонов написано «Ещё не запускался»
«Стабильность:»	доля пройденных прогонов, например «67% из 3 прогонов», и через «·» оценка DoQA. Если доли нет, а оценка есть, показано «Оценка

Строка	Пример и смысл
	DoQA: ...»
«История, от старых к новым:»	полоса последних прогонов, по значку на прогон, и в скобках расшифровка значков, см. **Статусы как в DoQA**
«В карантине: падения не влияют на гейт»	только у автотеста в карантине
«Ключ автотеста:»	внешний ID теста; если плагин не может его вычислить, написано «вычислит адаптер» и причина в скобках
«Автотест в DoQA: #...»	номер автотеста в каталоге
карточка кейса	номер и название первого привязанного кейса, его статус, приоритет и число шагов; «Ещё кейсов: N», если привязано несколько
ссылки	«История прогонов» и «Меню DoQA» открывают историю и меню действий, а «Кейс в DoQA» и «Автотест в DoQA» ведут на страницы в браузере



Связанные кейсы в подсказке

Первая строка подсказки говорит, какие кейсы связаны с автотестом в DoQA. Если связи в DoQA и ID в `@DoqaCaseIds` (`doqa.case_ids` в pytest) расходятся, плагин показывает и то и другое:

Ситуация	Что написано
в DoQA есть связи	«ID связанных кейсов в DoQA: 15, 16»

Ситуация	Что написано
связей в DoQA нет, но ID кейсов стоят в `@DoqaCaseIds`	«ID кейсов в @DoqaCaseIds: 15 (в DoQA ещё не привязаны, привяжутся при следующем прогоне)»
связи есть и там и там, но ID различаются	первая строка про DoQA, ниже отдельной строкой «В @DoqaCaseIds: …»
связей нет нигде	«Связанных кейсов в DoQA нет»

Кейсы из `@DoqaCaseIds` DoQA привяжет сам при следующем прогоне теста: адаптер передаёт их вместе с результатом. Кейсы, которых нет в пространстве, не привязываются и не создаются.

Если плагин не может сопоставить тест с автотестом в DoQA, ниже идёт отдельная строка с причиной:

- «Автотеста ещё нет в каталоге DoQA: он появится после первого прогона»;
- «Каталог DoQA ещё не проверен: связи в DoQA неизвестны», например пока каталог грузится или не выбрано пространство;
- «Ключ автотеста вычислит адаптер при запуске, поэтому связи в DoQA не проверить»;
- «Ключ с плейсхолдером: у каждого набора параметров свой автотест, связи в DoQA не проверить», если в ключе есть плейсхолдер (`@DoqaId("LOGIN-{browser}")`).

Пока каталог не проверен, первая строка говорит только о коде: «ID кейсов в @DoqaCaseIds: 15» или «В @DoqaCaseIds кейсов нет». Про связи в DoQA плагин в этом случае ничего не утверждает.

Статусы как в DoQA

Исходы прогонов и оценки DoQA плагин называет так же, как веб-интерфейс DoQA.

Исход прогона в строке «Последний прогон:», в CodeLens и в **истории прогонов**: «Пройден», «Провален», «Сломан», «Заблокирован», «Пропущен». Внутри строки VS Code пишет их со строчной буквы: «Последний прогон: провален».

Полоса «История, от старых к новым:» повторяет колонку «История» в **каталоге автотестов**: один значок на прогон, слева старые, справа новые.

Значок	Прогон
	Пройден
	Провален
!	Сломан или Заблокирован
·	Пропущен; сюда же попадают прогоны, которые не начинались

В скобках после полосы плагин расшифровывает только те значки, которые в ней есть, например « пройден, провален». DoQA отдаёт полосу одной отметкой на сломанные и заблокированные прогоны, поэтому точный статус `!` плагин называет, только если такая отметка одна и стоит у последнего прогона: «сломан» или «заблокирован». Для остальных смотрите **историю прогонов**.

Оценка DoQA после «·» в строке «Стабильность:» и в конце CodeLens: «Нестабильный», «Сломан», «В карантине», «Отключён», «Исправлен». У стабильного автотеста оценка не пишется.

Откуда берутся данные

CodeLens и подсказка читают статусы из кэша плагина и сами за ними в сеть не ходят. Подсказка обращается к DoQA только за карточкой привязанного кейса, если её ещё нет в кэше. Кэш статусов наполняется:

- когда файл с тестами становится активным: при открытии и при переключении на его вкладку;
- при сохранении файла;
- по команде «**DoQA: Обновить статусы в файле**»;
- в фоне, когда в файле появляется тест, которого кэш ещё не знает: новый тест или только что закреплённый ключ.

Автоматическое обновление работает, пока включена настройка `doqa.coverage.autoRefresh`. Команда «Обновить статусы в файле» работает всегда.

Кэш хранится на диске, отдельно для каждого сервера и пространства, и переживает правку файла и перезапуск VS Code. Если DoQA недоступен, CodeLens показывает последние

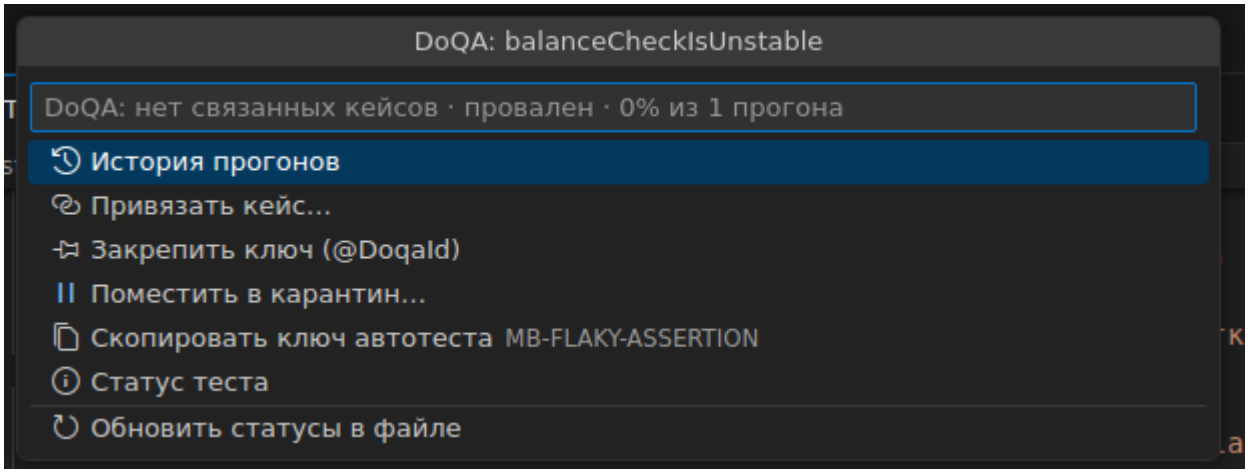
известные данные, а в подсказке появляется «DoQA недоступен: данные от ДД.ММ.ГГГГ ЧЧ:ММ». После смены адреса, пространства или токена кэш сбрасывается.

Меню действий над тестом

Все действия над одним тестом собраны в меню DoQA. Его открывают:

- щелчок по строке CodeLens над тестом;
- **Ctrl+Shift+Alt+D** (в macOS **Cmd+Shift+Alt+D**), когда курсор в тесте;
- ссылка «Меню DoQA» в подсказке над тестом;
- команда «**DoQA: Действия с тестом...**» из палитры команд или из подменю DoQA.

Меню открывается списком быстрого выбора с заголовком «DoQA: <имя метода>». В поле ввода можно набрать часть названия действия, **Enter** выполняет выбранное.



Действие	Что делает	Когда есть в меню
«История прогонов»	вкладка с последними прогонами и стеком падения, см. История прогонов	всегда; если ключ не вычисляется, под пунктом пояснение «ключ автотеста не вычисляется из исходника»
«Карточка кейса»	вкладка с шагами привязанного кейса	к тесту привязан кейс
«Открыть кейс в DoQA»	привязанный кейс в браузере; если кейсов несколько, сначала спросит какой	к тесту привязан кейс

Действие	Что делает	Когда есть в меню
«Привязать кейс...»	поиск кейса, `@DoqaCaseIds` в коде и связь в DoQA	всегда
«Отвязать кейс...»	снимает связь в DoQA и убирает ID кейса из аннотации	к тесту привязан кейс
«Закрепить ключ (@DoqaId)»	записывает внешний ID в код, см. **Закрепление @DoqaId**	всегда
«Поместить в карантин...» / «Снять с карантина»	ставит автотест в карантин или снимает с карантина	ключ теста вычисляется
«Скопировать ключ автотеста»	копирует внешний ID в буфер обмена; сам ключ виден справа от пункта	ключ теста вычисляется
«Статус теста»	сводка по тесту в уведомлении, см. **Статус теста**	всегда
«Обновить статусы в файле»	перечитывает каталог для всех тестов файла	всегда

В файлах pytest пункт закрепления называется «Закрепить ключ (@doqa.id)», а плагин пишет декораторы `@doqa.id` и `@doqa.case_ids` вместо аннотаций. В файлах Jest пункт называется «Закрепить ключ (id)», а плагин пишет ключи `id` и `caseIds` в метаданные `doqa.test`, см. **Что плагин пишет в код**.

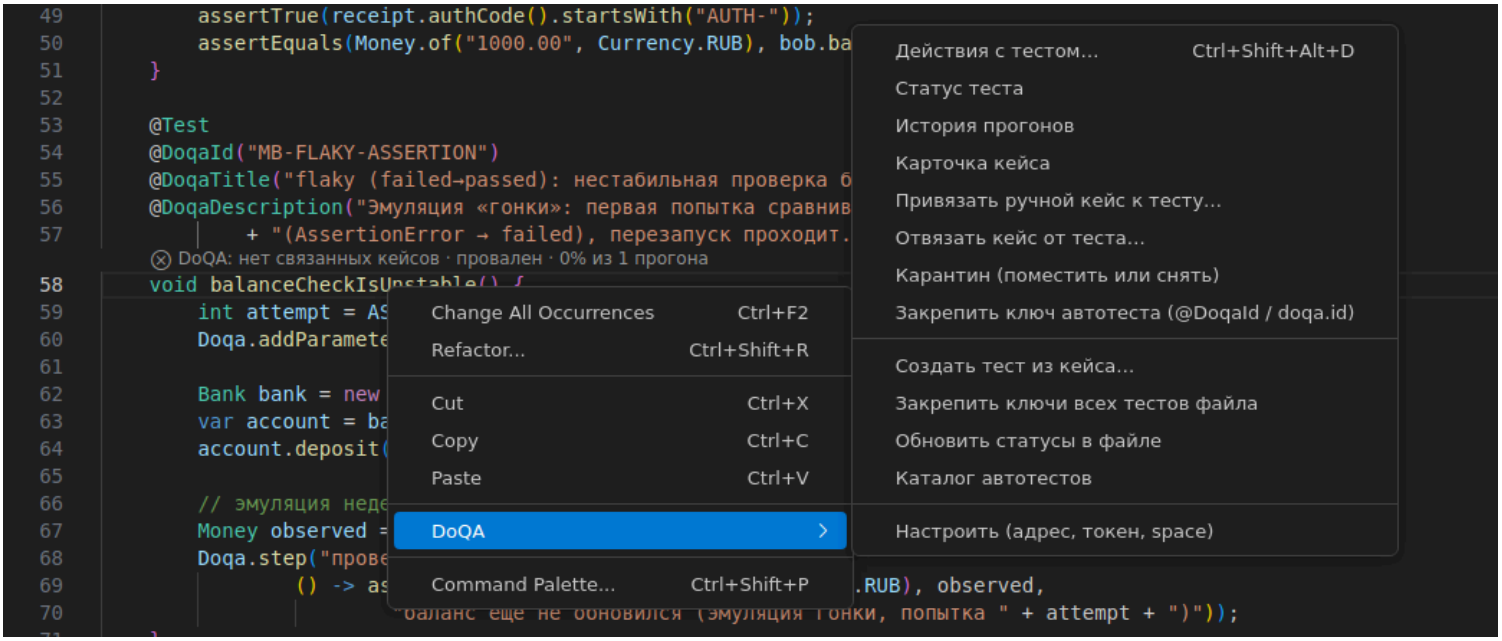
Действия работают с тестом под курсором. Щелчок по CodeLens сам ставит курсор на тест. Если курсор выше первого теста, плагин напишет «DoQA: поставьте курсор внутри тестового метода.», а в файле Jest «DoQA: поставьте курсор на тест Jest (test, it или doqa.test).».

Подменю DoQA и палитра команд

Те же команды есть ещё в трёх местах:

- в контекстном меню редактора, подменю **DoQA**;
- в кнопке **DoQA** в заголовке вкладки редактора;
- в палитре команд: наберите «DoQA».

Подменю и кнопка есть у файлов Java, Kotlin, Python, JavaScript и TypeScript. В подменю и кнопке команды названы без приставки «DoQA:», например «Статус теста», а в палитре с ней. В палитре команды над тестом и файлом видны, только когда открыт такой файл, а «DoQA: Каталог автотестов» и «DoQA: Настроить (адрес, токен, space)» видны всегда.



Команда	Что делает
«DoQA: Действия с тестом...»	меню действий над тестом под курсором
«DoQA: Статус теста»	сводка по тесту в уведомлении
«DoQA: История прогонов»	вкладка с историей прогонов
«DoQA: Карточка кейса»	вкладка с карточкой привязанного кейса
«DoQA: Привязать ручной кейс к тесту...»	поиск и привязка кейса
«DoQA: Отвязать кейс от теста...»	отвязка кейса
«DoQA: Карантин (поместить или снять)»	ставит автотест в карантин или снимает с карантина, смотря по текущему состоянию
«DoQA: Закрепить ключ автотеста (@DoqaId / doqa.id)»	закрепление ключа у теста под курсором
«DoQA: Создать тест из кейса...»	черновик теста по ручному кейсу, см. **Тест из кейса**

Команда	Что делает
«DoQA: Закрепить ключи всех тестов файла»	закрепление ключей у всех тестов файла без `@DoqaId`
«DoQA: Обновить статусы в файле»	перечитывает каталог для тестов открытого файла
«DoQA: Каталог автотестов»	список автотестов пространства, см. **Каталог автотестов**
«DoQA: Настроить (адрес, токен, space)»	адрес, токен и пространство по умолчанию

В отличие от меню действий, подменю показывает все команды всегда. Если команда к тесту не подходит, она объяснит почему: например, «Карточка кейса» у теста без кейсов ответит «DoQA: к тесту не привязан ни один кейс.».

«Закрепить ключи всех тестов файла» есть и в контекстном меню проводника у файлов `.java`, `.kt`, `.py` и файлов JavaScript и TypeScript (`.js`, `.cjs`, `.mjs`, `.jsx`, `.ts`, `.cts`, `.mts`, `.tsx`), в подменю **DoQA**.

Результаты приходят уведомлениями VS Code в правом нижнем углу, а короткие сообщения в строке состояния. Модальный диалог появляется, только когда нужен выбор или подтверждение. Правки кода, которые делает плагин, отменяются одним **Ctrl+Z**.

Статус теста

«**Статус теста**» перечитывает каталог для теста под курсором и показывает уведомление в одну строку, части разделены «·»:

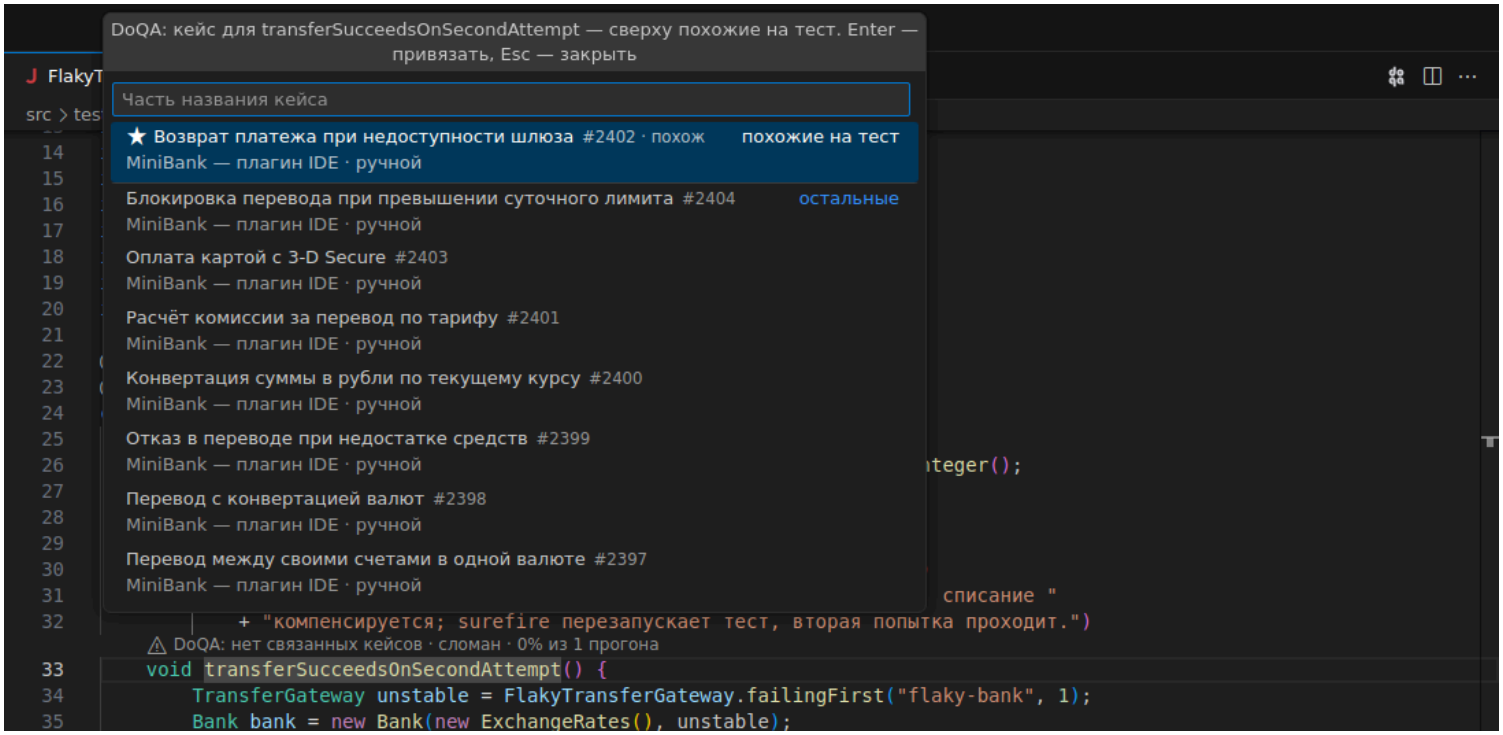
- связанные кейсы, как в **подсказке над тестом**, с теми же пояснениями, если автотест не найден в каталоге или ключ не вычисляется;
- «фреймворк:» JUnit 5, JUnit 4, TestNG, pytest или Jest;
- «ключ:» внешний ID и в скобках, откуда он взят: `explicit` (из `@DoqaId`, в Jest из `id` в метаданных), `title` (номер из названия теста), `allure` (из `@AllureId`), `signature` (хэш сигнатуры), `unresolved` (плагин ключ не вычислил);
- «ключ не вычислен:» и причина, если ключа нет;

- «автотест:» номер автотеста в каталоге, «нет в каталоге» или «не проверен», если каталог по тесту не ответил.

Привязка кейса

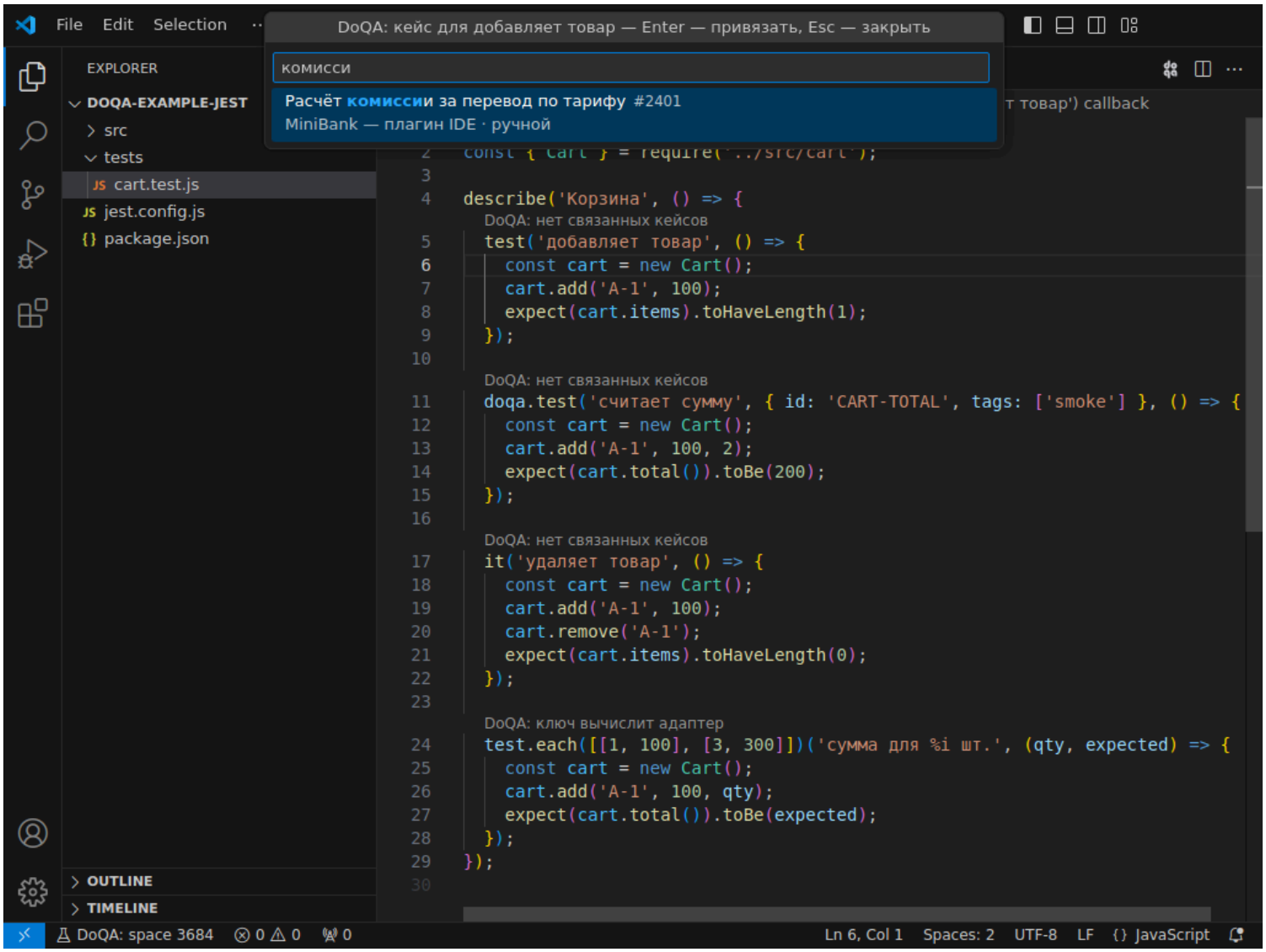
1. Поставьте курсор в тест и выберите в меню «**Привязать кейс...**» (в палитре «DoQA: Привязать ручной кейс к тесту...»).
2. Откроется список «DoQA: кейс для <имя метода>». Сразу, без ввода, в нём появляются кейсы из верхних папок дерева. Сверху стоят кейсы, похожие на тест по названию: они отмечены звёздочкой и пометкой «похож» рядом с номером, а справа у группы подписано «похожие на тест». Ниже идёт группа «остальные».
3. Начните вводить название: список обновляется по мере ввода. Стрелки двигают выбор.
4. Нажмите **Enter**.

Под названием кейса видны папка и статус автоматизации: «ручной», «подлежит автоматизации» или «автоматизирован». Что сейчас происходит, плагин пишет в заголовке списка: «ищу похожие на тест...», «поиск...», «кейсы не найдены», «сверху похожие на тест. Enter — привязать, Esc — закрыть».



Плагин добавляет ID кейса в `@DoqaCaseIds` над тестом (импорт дописывается сам, существующие ID сохраняются) и создаёт связь в DoQA. В тесте Jest ID кейса попадает в `caseIds` метаданных `doqa.test`, а обычный `test` или `it` становится `doqa.test`. Если

`doqa.test` изменил бы тест (`test.each`, колбэк `done`, `test.failing`, таблица шаблонной строкой), ID записывается вызовом `doqa.metadata({ caseIds: [...] })` первой строкой тела, а над тестом появляется «DoQA: кейс 2401 из doqa.metadata», см. **Кейс в doqa.metadata**. Появится уведомление «DoQA: кейс #N «название» привязан к тесту …» с кнопкой «Открыть кейс». Привязка только добавляет связь: прежние связи теста не снимаются.



Похожие кейсы подбираются так же, как в IntelliJ: по словам из `@DisplayName`, `@DoqaTitle`, `description` у TestNG и имени метода (у Jest из названия теста и `title` в метаданных), а не по смыслу. Подробнее об этом в разделе **Привязка кейса** страницы про IntelliJ. Одна особенность VS Code: пока в поле поиска есть текст, VS Code сам сортирует список по совпадению с вводом, и похожие кейсы могут оказаться не сверху. Звёздочка и пометка «похож» у них остаются.

Поиск останавливается на 50 кейсах. Если нашлось больше, в заголовке появится «показаны не все совпадения — уточните запрос».

Что происходит с внешним ID при привязке:

- тест, которого в DoQA ещё нет, привязывается под **читаемым ключом**, и плагин сразу закрепляет его в `@DoqaId`. В уведомлении тогда есть «Ключ автотеста: ...». В Jest привязка и ключ записываются одной правкой: `doqa.test('добавляет товар', { id: 'tests.cart > Корзина > добавляет товар', caseIds: [2401] }, …)`;
- тест, который уже есть в каталоге, сохраняет свой ключ. Если это хэш, в уведомлении будет «Ключ автотеста — хэш: переименование теста его сменит.» и кнопка «Закрепить @DoqaId»;
- если ключ по тексту файла не вычислить, плагин предложит выбрать автотест из каталога в списке «DoQA: ключ автотеста, чтобы привязать кейс» или пункт «Ввести ключ вручную...». Выбранный или введённый ключ тоже закрепляется в коде;
- у строк `each` в Jest ключ не спрашивается и связь в DoQA не создаётся: кейс записывается только в код, и адаптер привяжет его к каждой строке при следующем прогоне. Так же с тестом Jest, у которого ключ не вычислен и не может быть закреплён: в уведомлении «Ключ автотеста вычислит адаптер: он привяжет кейс при следующем прогоне.».

Карточка кейса

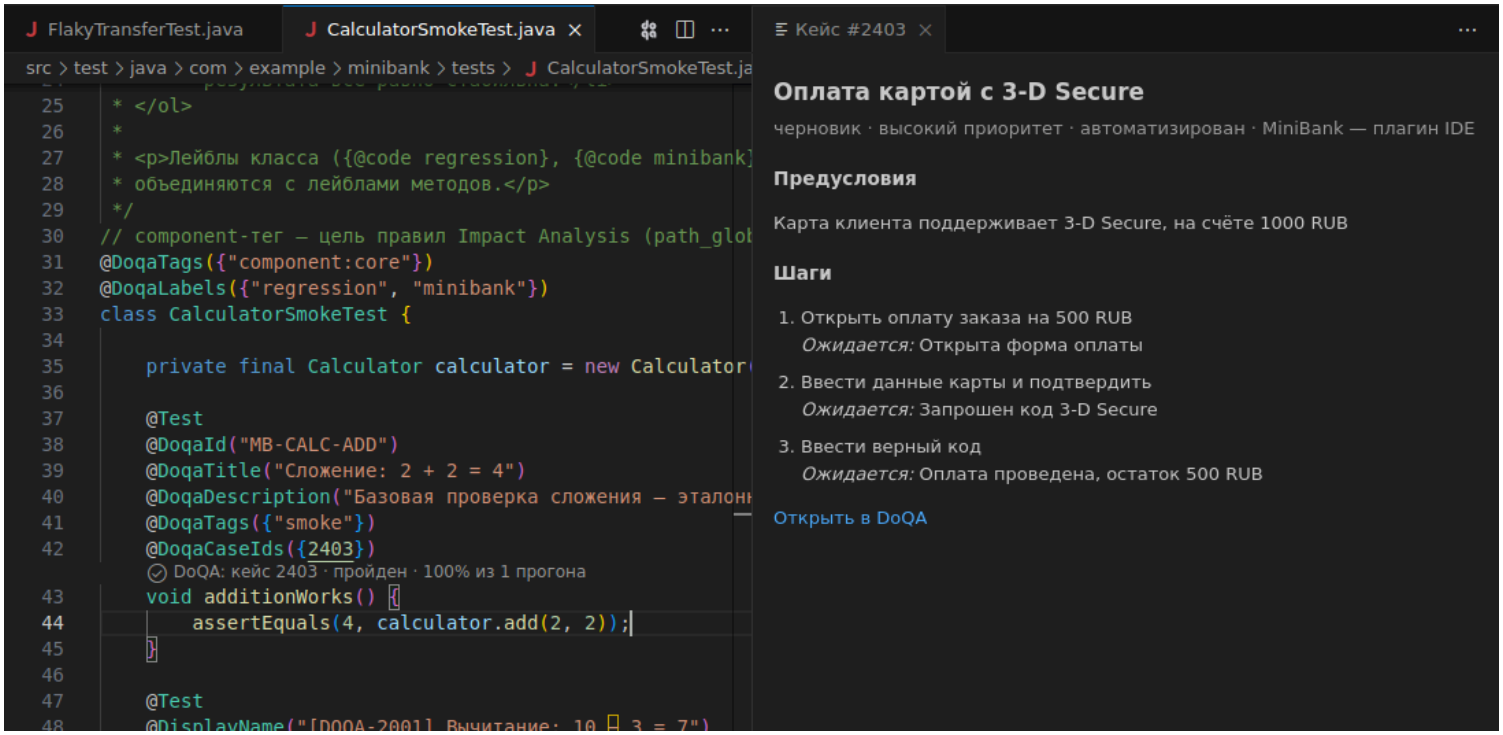
Карточку привязанного кейса открывает «**Карточка кейса**» в меню или **Ctrl+клик** по ID в `@DoqaCaseIds` (`doqa.case_ids` в pytest, `caseIds` и `doqa.metadata` в Jest). При наведении на ID видна подсказка «Карточка кейса DoQA #N».

Карточка открывается вкладкой «Кейс #N» рядом с редактором:

- название кейса;
- статус, приоритет, статус автоматизации и путь по папкам;
- «Предусловия»;
- «Шаги» с ожидаемым результатом каждого шага («Ожидается:»);
- «Ожидаемый результат» кейса;
- ссылка «Открыть в DoQA».

Вкладка карточки одна: следующий кейс открывается в ней же. Если к тесту привязано несколько кейсов, команда сначала предложит выбрать один в списке «DoQA: карточка кейса». Когда DoQA недоступен, показывается карточка из кэша с пометкой «DoQA

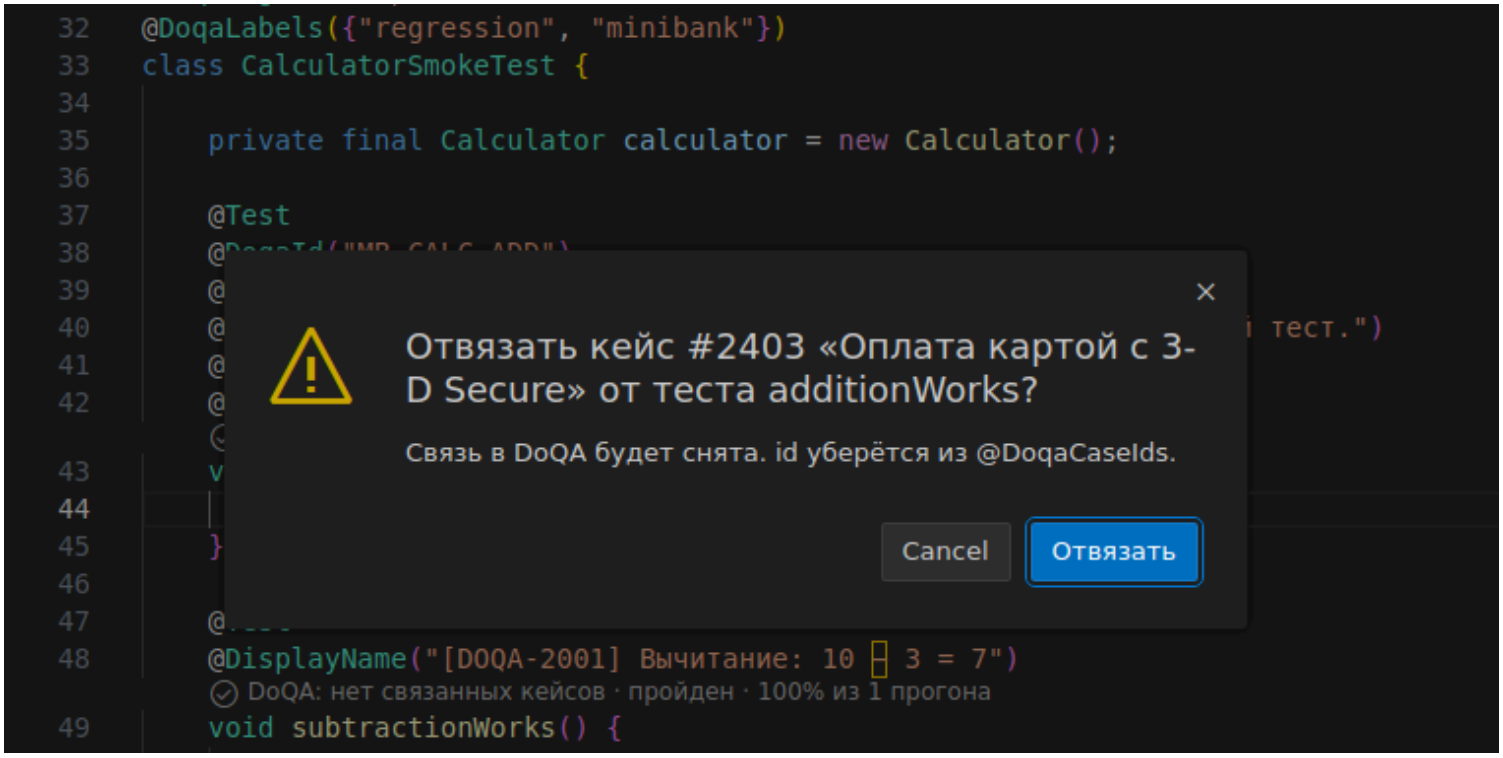
недоступен: показана карточка с последнего обновления.», если плагин уже загружал её раньше.



Отвязка кейса

1. Выберите «**Отвязать кейс...**». Если кейсов несколько, выберите нужный в списке «DoQA: отвязать кейс».
2. Подтвердите в диалоге: «Отвязать кейс #N «название» от теста ...?». Под вопросом сказано, что будет сделано: «Связь в DoQA будет снята.» и «id уберётся из @DoqaCaseIds.».
3. Нажмите «**Отвязать**».

Плагин сообщит «DoQA: кейс #N отвязан от ...». ID убирается и из аннотации: иначе следующий прогон привязал бы кейс снова. В тесте Jest ID убирается из `caseIds` и из вызовов `doqa.metadata`, и в диалоге сказано «id уберётся из doqa.metadata.».



Кейс, записанный в `@DoqaCaseIds` на классе, для одного теста не отвязать, как и последний ID метода, если свои ID есть у класса: без него адаптер возьмёт ID класса. Плагин объяснит это предупреждением и предложит перенести `@DoqaCaseIds` с класса на методы или убрать ID с класса. Подробнее об этом в разделе **Отвязка кейса** страницы про IntelliJ.

Читаемый ключ и закрепление @Doqald

Закрепление записывает внешний ID в код аннотацией `@DoqaId`, и дальше тест можно переименовывать и переносить без потери истории. Плагин предлагает **читаемый ключ**, то есть полное имя теста, например `com.example.minibank.tests.CalculatorSmokeTest.multiplicationWorksWithoutAnyMarkup`, а в Jest путь файла и цепочку названий: `tests.cart > Корзина > добавляет товар`, см. **Читаемый ключ Jest**. Зачем это нужно и чем читаемый ключ лучше хэша, описано в разделе **Читаемый ключ и закрепление @Doqald** страницы про IntelliJ. Правила выбора ключа в VS Code те же:

Ситуация	Что делает плагин
ключ уже читаемый: `DOQA-123` из названия теста, `ALLURE-42` из Allure	закрепляет его как есть

Ситуация	Что делает плагин
теста в DoQA ещё нет	закрепляет читаемый ключ, в уведомлении будет «Новый тест: в DoQA его ещё нет, поэтому ключ читаемый.»
тест уже есть в DoQA под хэшем	спрашивает, что закрепить (см. ниже)
читаемый ключ уже занят другим автотестом	просит ввести другой ключ в строке «DoQA: @DoqaId», по умолчанию предлагает `<ключ>-2`
ключ по тексту файла не вычислить	предлагает выбрать автотест из каталога или ввести ключ, по умолчанию читаемый
у теста уже есть `@DoqaId`	ничего не пишет: «DoQA: @DoqaId уже стоит: ...»
`@DoqaId` стоит на классе	ничего не пишет: «DoQA: ключ задан на классе (...), под ним отчитываются все тесты класса. Чтобы закрепить ключ у отдельного теста, уберите @DoqaId с класса.»

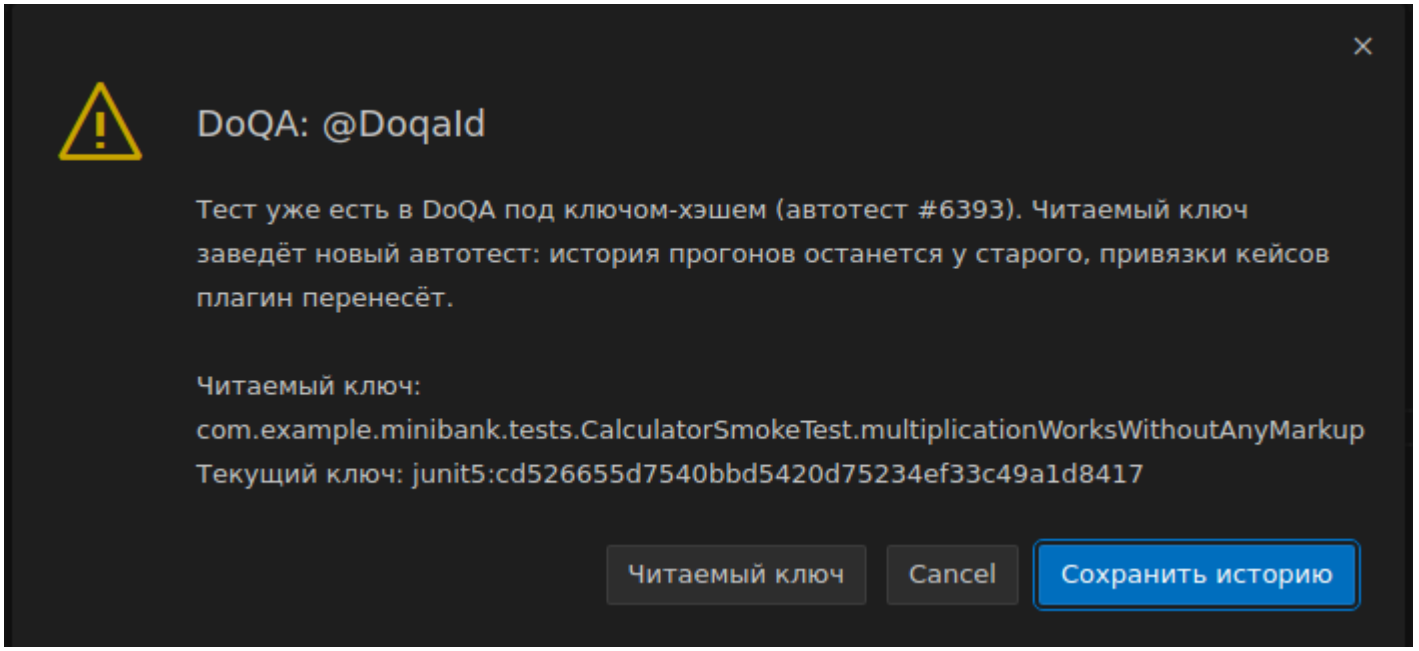
Когда тест уже есть в DoQA под хэшем, появляется модальный диалог «DoQA: @DoqaId»: «Тест уже есть в DoQA под ключом-хэшем (автотест #N). Читаемый ключ заведёт новый автотест: история прогонов останется у старого, привязки кейсов плагин перенесёт.» Ниже показаны оба ключа: «Читаемый ключ:» и «Текущий ключ:».

- «**Сохранить историю**» (основная кнопка, выделена цветом): закрепить текущий ключ-хэш. История остаётся, но ключ нечитаемый.
- «**Читаемый ключ**»: закрепить читаемый ключ. Привязки кейсов плагин переносит на новый ключ и сообщает: «DoQA: кейсы #... привязаны к новому ключу .... История прогонов осталась у автотеста #N. Ctrl+Z вернёт ключ в коде, но не привязки в DoQA.».
- «**Cancel**»: ничего не менять.

Если плагин не настроен, он не может проверить каталог. Тогда в диалоге другой текст: «Каталог DoQA не проверен: плагин не настроен. Если тест уже запускался, читаемый ключ заведёт новый автотест, и история прогонов останется у старого.», и основной становится кнопка «Читаемый ключ».

После записи появляется уведомление «DoQA: закреплено @DoqaId("..."). Отменить — Ctrl+Z.», в Jest «DoQA: закреплено id: '...'. Отменить — Ctrl+Z.». Если метаданные теста Jest записать нельзя (например, у `test.todo` или теста с колбэком `done`), плагин ничего не

пишет и объясняет причину предупреждением, список есть в разделе **Когда плагин не пишет в код**.



Весь файл

«**DoQA: Закрепить ключи всех тестов файла**» проставляет ключи всем тестам файла без `@DoqaId`, у которых ключ вычисляется. Команду можно вызвать и из контекстного меню файла в проводнике. Тесты, которых в DoQA нет, получают читаемый ключ, тесты с историей сохраняют текущий. Перед записью плагин один раз спрашивает подтверждение, например: «Проставить @Doqald у 5 тест(ов): читаемый ключ — у 3 (в DoQA их ещё нет); текущий ключ — у 2 (сохраняется история прогонов).», кнопка «Закрепить». Если плагин не настроен и каталог не проверить, везде записываются текущие ключи, о чём сказано в том же диалоге. Если DoQA ответил ошибкой, плагин покажет её и ничего не запишет.

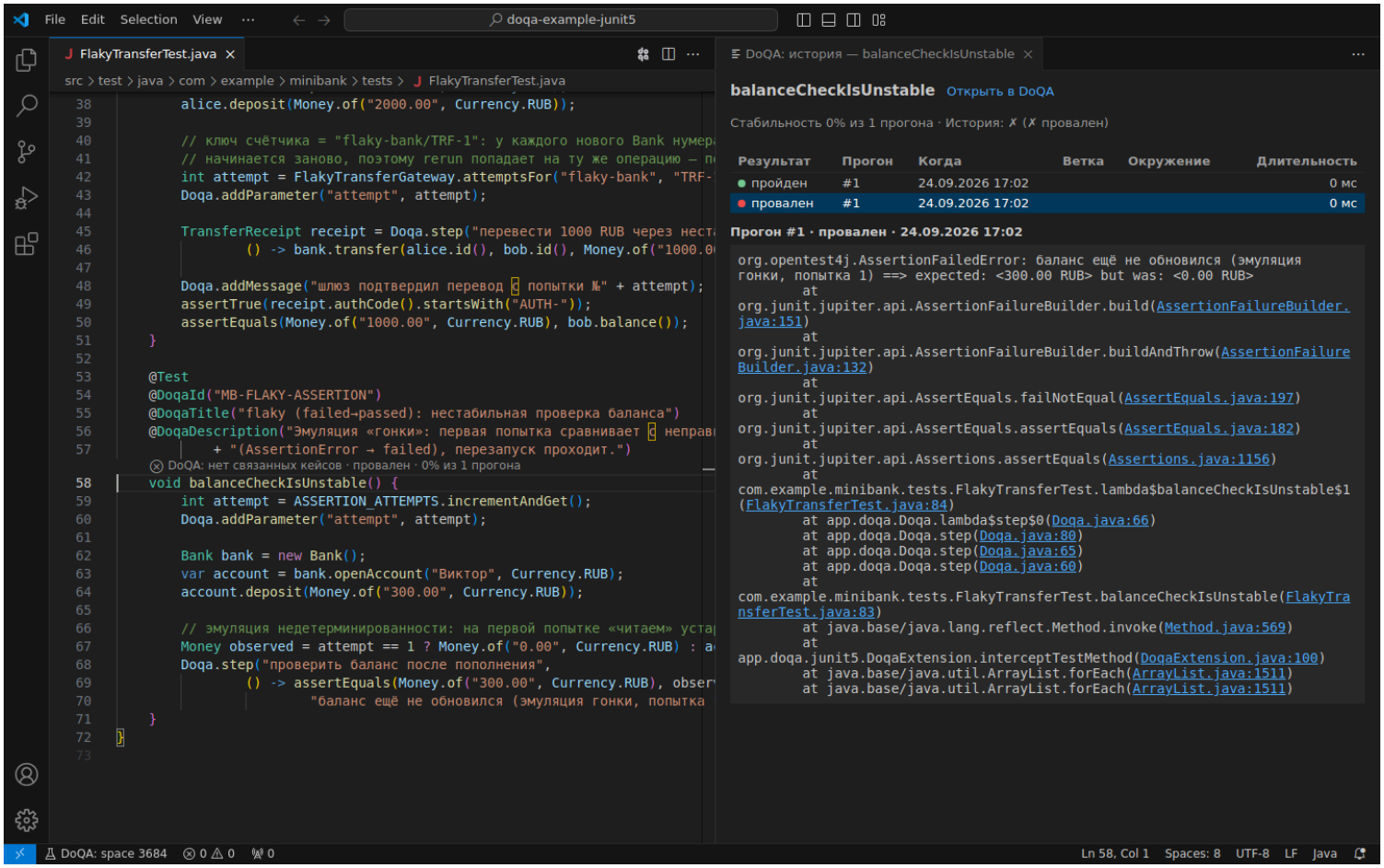
Если закреплять нечего, плагин ответит «DoQA: закреплять нечего — у всех тестов файла уже есть @Doqald или ключ не вычисляется.».

История прогонов

«**История прогонов**» открывает рядом с редактором вкладку «DoQA: история — <имя метода>» с последними 30 прогонами автотеста. Вкладку можно держать открытой рядом с кодом, пока чините тест. Для каждого автотеста вкладка своя: повторная команда выводит её вперёд со свежими данными.

- Сверху показаны имя метода, кнопка «**Открыть в DoQA**» и сводка: стабильность, оценка DoQA, «В карантине» и полоса последних прогонов с расшифровкой: «История: (провален)».
- Таблица: «Результат», «Прогон», «Когда», «Ветка», «Окружение», «Длительность». В колонке «Результат» статус прогона: «Пройден», «Провален», «Сломан», «Заблокирован» или «Пропущен». Прогон, который DoQA отметил как нестабильный, помечен «нестабильный». Если прогонов больше 30, под таблицей написано «Показаны последние 30 из N.».
- Ниже показан выбранный прогон: заголовок с номером, попыткой, исходом, временем, веткой, коммитом и окружением, затем сообщение об ошибке и стек. У пройденного прогона вместо ошибки написано «Прогон пройден.».
- При открытии выбран последний проваленный или сломанный прогон. Другой прогон выбирается щелчком по строке или стрелками.

Строки стека вида `FlakyTransferTest.java:83`, а у тестов Jest строки стека Node вида `tests/cart.test.js:12:5` работают как ссылки: щелчок открывает файл из рабочей области на этой строке. Если файлов с таким именем несколько, плагин выберет тот, чей путь совпадает с пакетом из стека, а если не сможет выбрать, спросит в списке «DoQA: какой ... открыть». Если файла нет, появится «DoQA: файла ... в рабочей области нет.». Строки стека из `node_modules` ссылками не становятся.



Если автотеста ещё нет в каталоге, плагин напишет «DoQA: автотеста … ещё нет в DoQA: он появится после первого прогона.». Если у теста ключ с плейсхолдером (`@DoqaId("LOGIN-{browser}")`), у каждого набора параметров свой автотест, и история для теста целиком недоступна.

Версия DoQA

История, отвязка кейса и карантин работают, только если сервер DoQA принимает для этих запросов персональный API-токен. На сервере, где это ещё не так, плагин напишет, что сервер DoQA пока не принимает API-токен для истории прогонов (отвязки, карантина) и эта возможность появится после обновления DoQA. Токен при этом в порядке, менять его не нужно. Подробнее в разделе Ограничения.

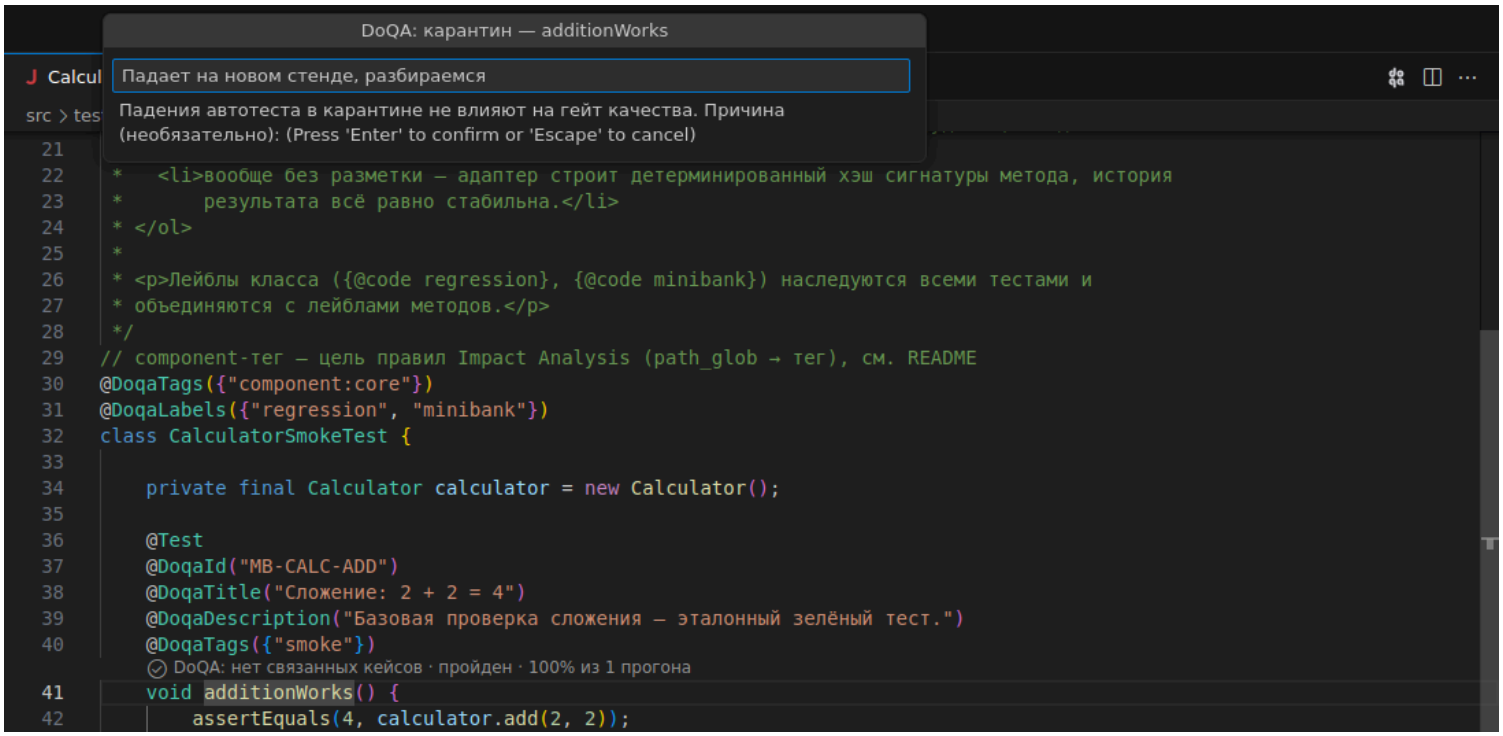
Карантин

1. Выберите в меню «**Поместить в карантин…**».
2. В строке ввода «DoQA: карантин — <имя метода>» при желании укажите причину: «Падения автотеста в карантине не влияют на гейт качества. Причина (необязательно):».

Причину можно не писать.

3. Нажмите **Enter**. В строке состояния появится «DoQA: <имя метода> в карантине.», значок в CodeLens сменится на паузу, а в конце строки появится «В карантине».

Esc отменяет постановку в карантин.



Если автотест уже в карантине, пункт меню называется «**Снять с карантина**» и спрашивает в модальном диалоге: «Снять … с карантина? Его падения снова будут влиять на гейт.», кнопка «Снять». В палитре обе операции собраны в одну команду «DoQA: Карантин (поместить или снять)».

Плагин не передаёт ветку: карантин ставится и снимается для автотеста целиком. Что карантин меняет, описано на странице Нестабильные тесты и карантин.

Каталог автотестов

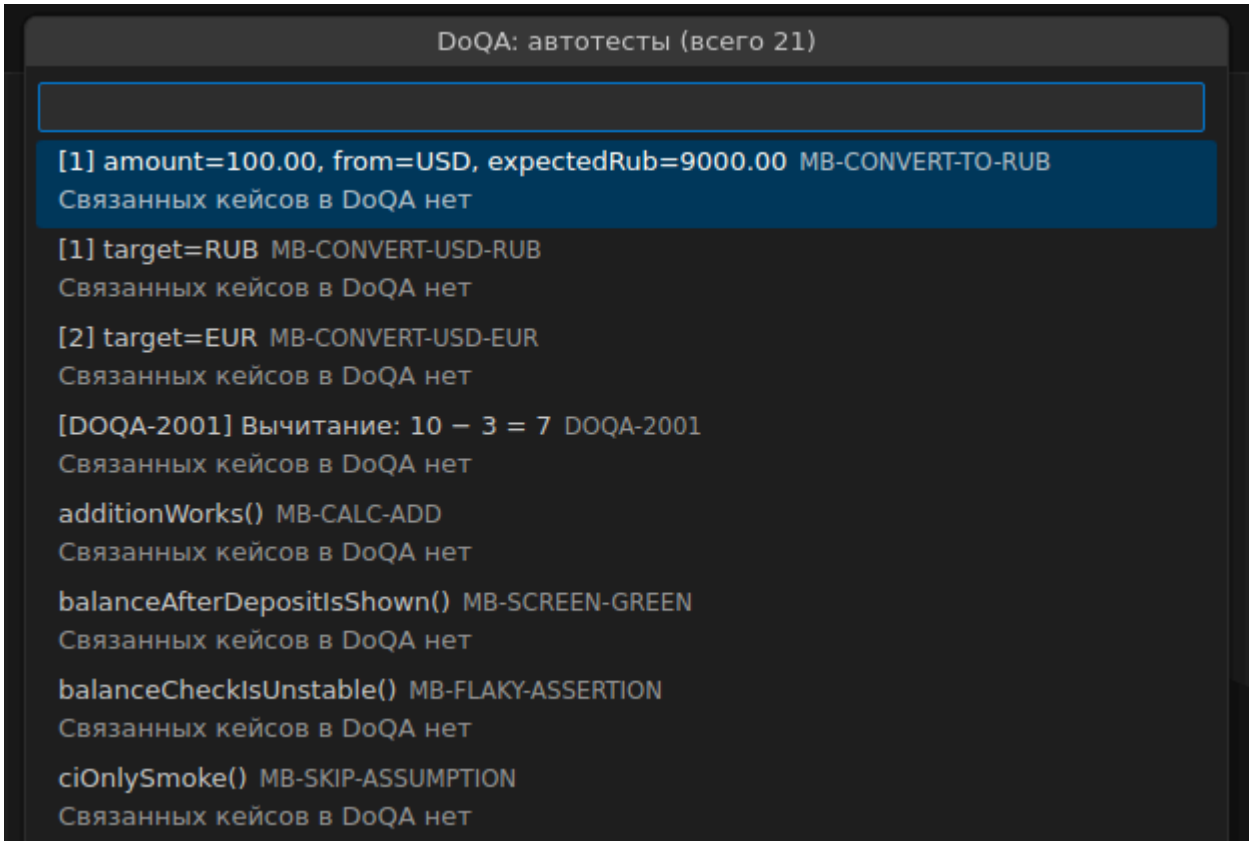
«**DoQA: Каталог автотестов**» ищет автотесты пространства открытого файла:

1. В строке «DoQA: каталог автотестов» введите часть названия или ключа автотеста либо оставьте поле пустым, чтобы увидеть все. Нажмите **Enter**.
2. Откроется список «DoQA: автотесты (всего N)», в нём до 100 автотестов. В строке показано название автотеста, справа ключ («(без ключа)», если его нет), ниже «ID связанных кейсов

в DoQA: ...» или «Связанных кейсов в DoQA нет». Ввод текста дальше фильтрует список по названию, ключу и кейсам.

3. **Enter** на автотесте с кейсом открывает первый привязанный кейс в браузере. У автотеста без кейсов **Enter** просто закрывает список.

Если пространство не выбрано, плагин сначала попросит ввести его номер: «Space не выбран. Введите его номер или выполните «DoQA: Настроить».». Если по запросу ничего не нашлось, плагин напишет «DoQA: автотестов по этому фильтру нет.».



Полный каталог с фильтрами, стабильностью и массовыми действиями есть в веб-интерфейсе, см. Каталог автотестов.

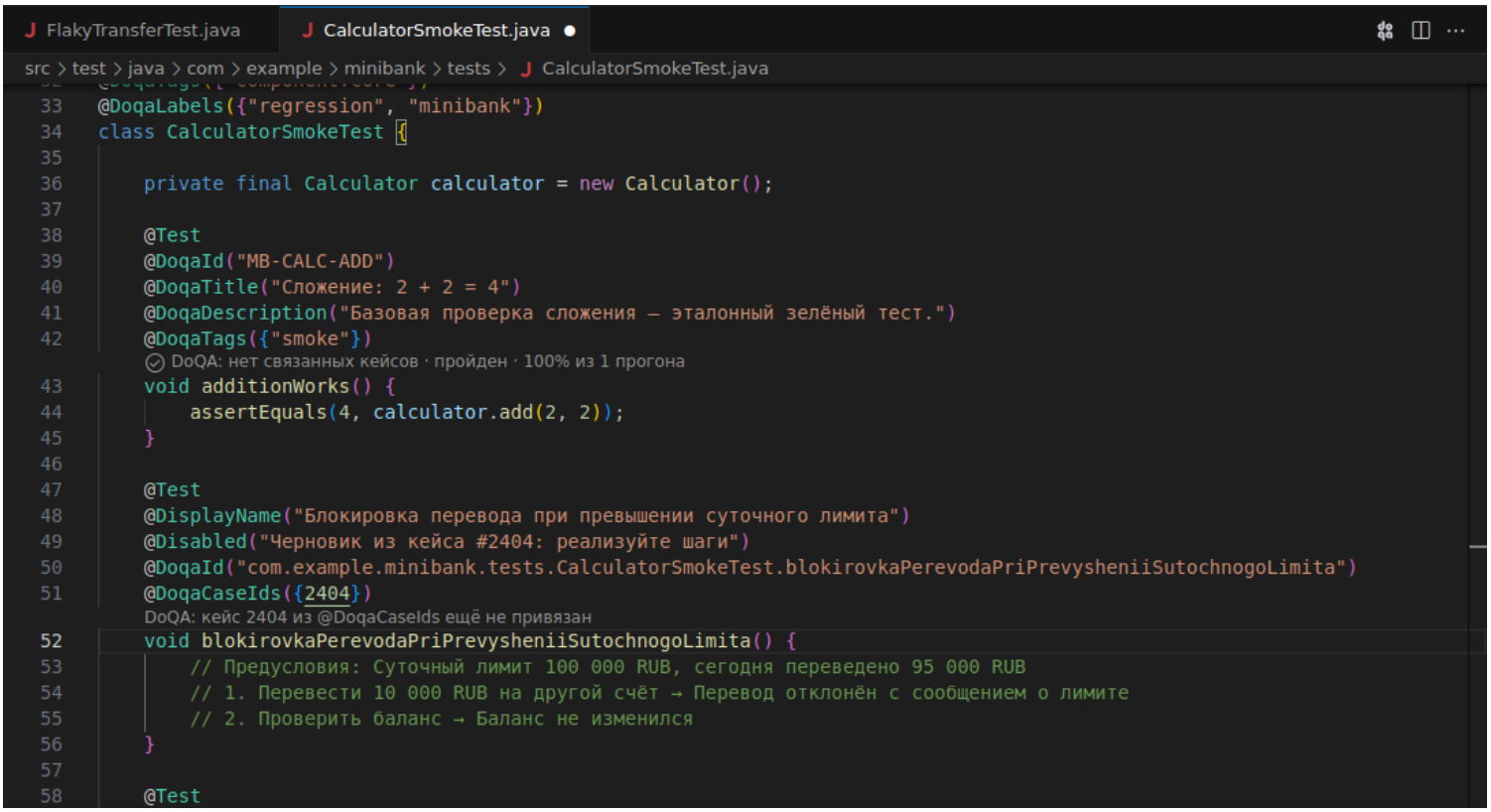
Тест из кейса

«**DoQA: Создать тест из кейса...**» пишет в открытый файл черновик автотеста по ручному тест-кейсу: название кейса, привязку к нему, внешний ID, предусловия и шаги комментариями. Тест выключен, пока вы не напишете проверку.

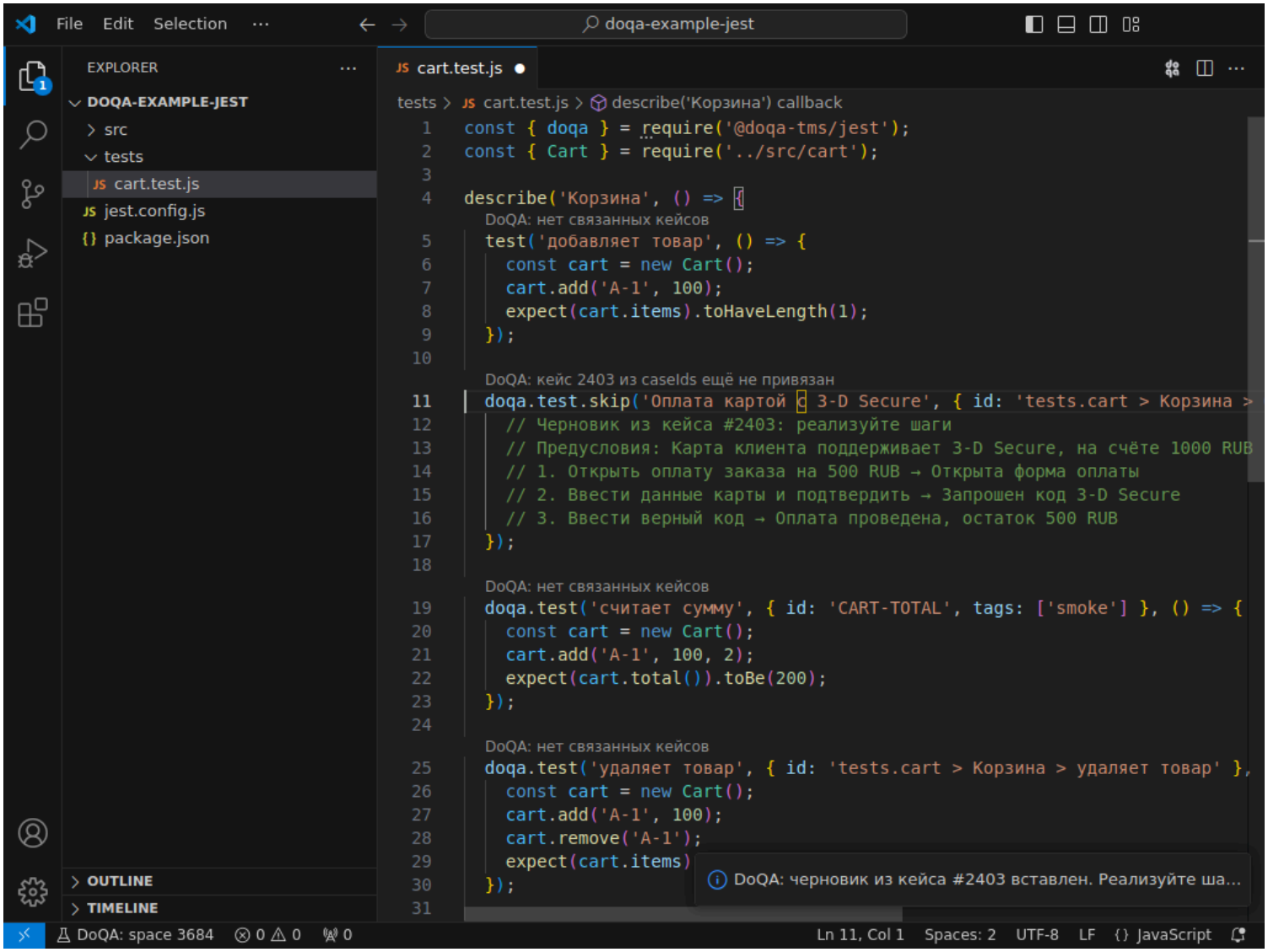
1. Откройте файл с тестами (`.java` , `.kt` , `.py` или файл с тестами Jest) и поставьте курсор на пустую строку между тестами, туда, где нужен новый тест.

2. Выполните «DoQA: Создать тест из кейса...» из палитры команд или подменю DoQA.
3. В списке «DoQA: тест из кейса» начните вводить название кейса, выберите кейс стрелками и нажмите **Enter**.

Плагин вставит черновик, поставит курсор на объявление теста и покажет уведомление «DoQA: черновик из кейса #N вставлен. Реализуйте шаги и включите тест. Отменить — Ctrl+Z.». Вся вставка вместе с импортами отменяется одним **Ctrl+Z**.



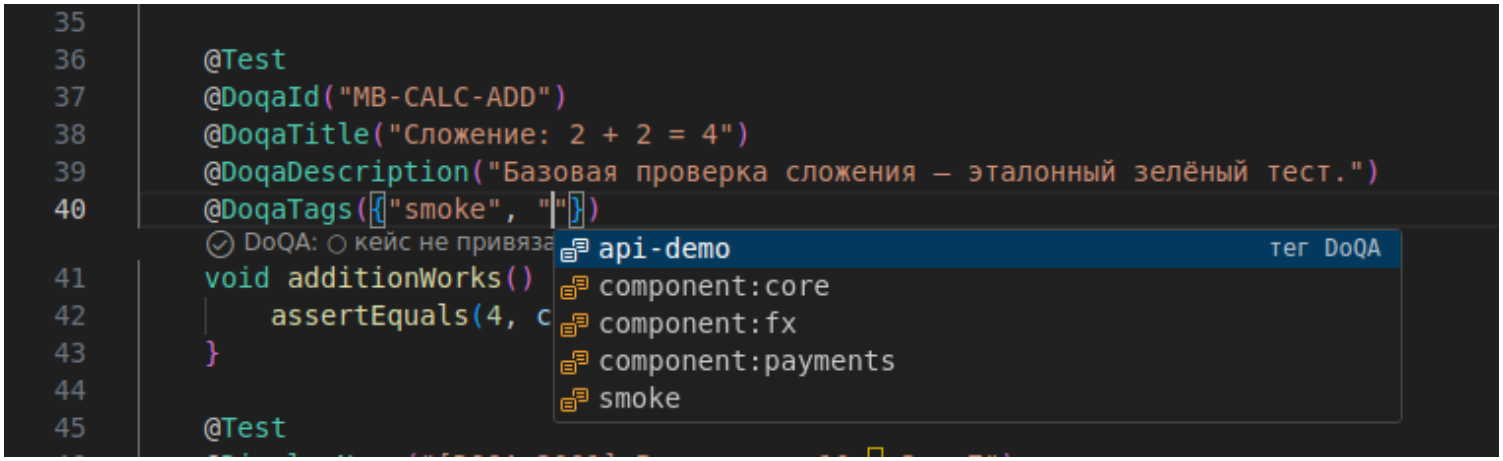
В файле Jest черновик это вызов `doqa.test.skip` в блоке `describe`, где стоит курсор, с `id`, `caseIds`, предусловиями и шагами комментариями:

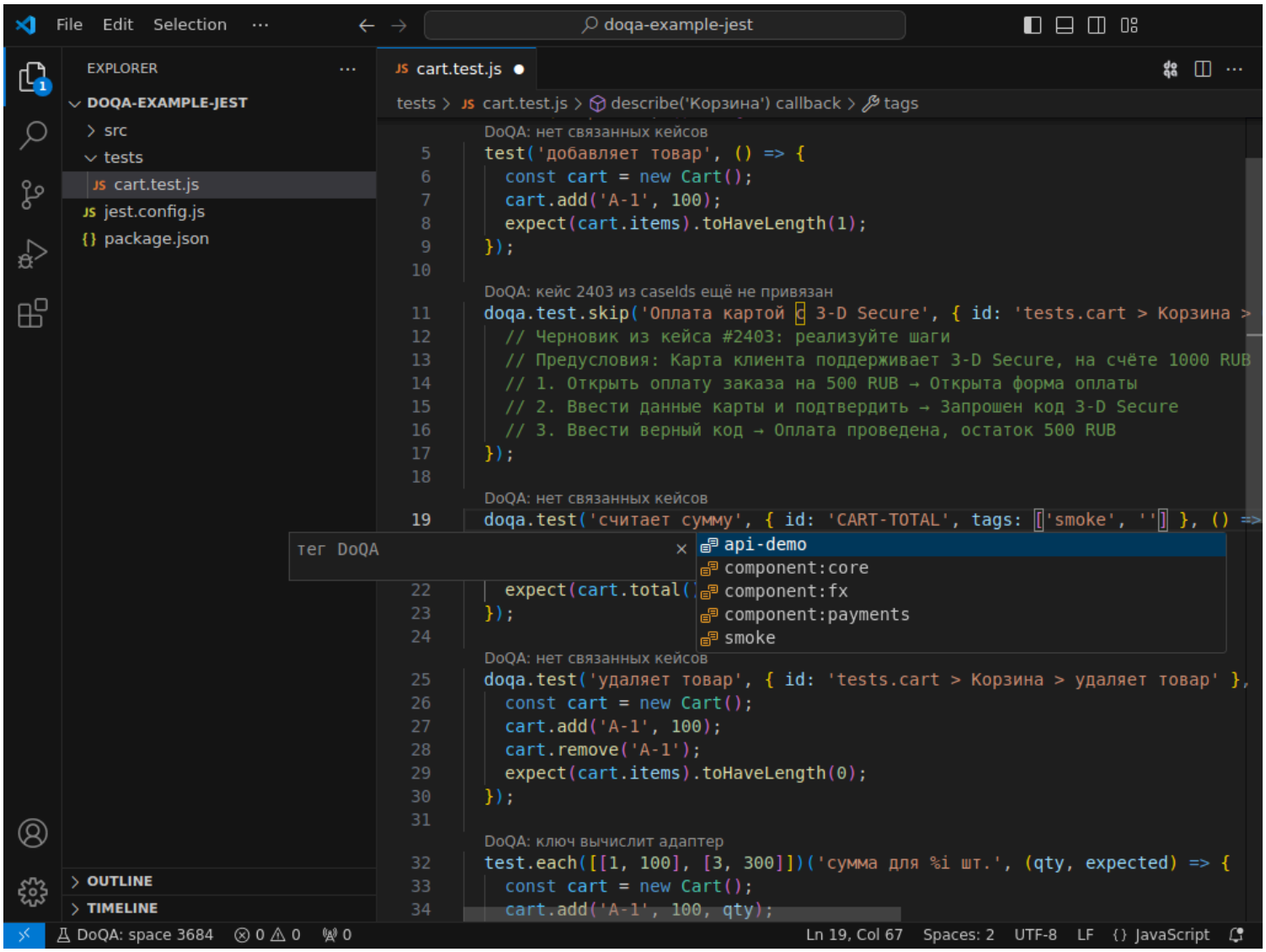


Черновик в VS Code такой же, как в IntelliJ. Какая разметка пишется для JUnit 5, JUnit 4, TestNG, pytest и Jest, как называется метод и куда встаёт черновик, описано в разделах **Что попадает в черновик** и **Куда вставляется черновик**.

Дополнение тегов и меток

Внутри строки в `@DoqaTags` и `@DoqaLabels` (в pytest в `doqa.tag` и `doqa.label`, в Jest в массивах `tags: [...]` и `labels: [...]` метаданных `doqa.test` и `doqa.metadata`) плагин предлагает теги и метки, которые уже есть у автотестов в каталоге пространства. Список открывается сам сразу после открывающей кавычки, вызвать его вручную можно сочетанием **Ctrl+Space**. У каждого варианта подписано «тег DoQA» или «метка DoQA». В `@DoqaTags` и `tags` предлагаются только теги, а в `@DoqaLabels` и `labels` только метки. В Jest дополнение работает в файлах, которые импортируют `doqa` из `@doqa-tms/jest`.

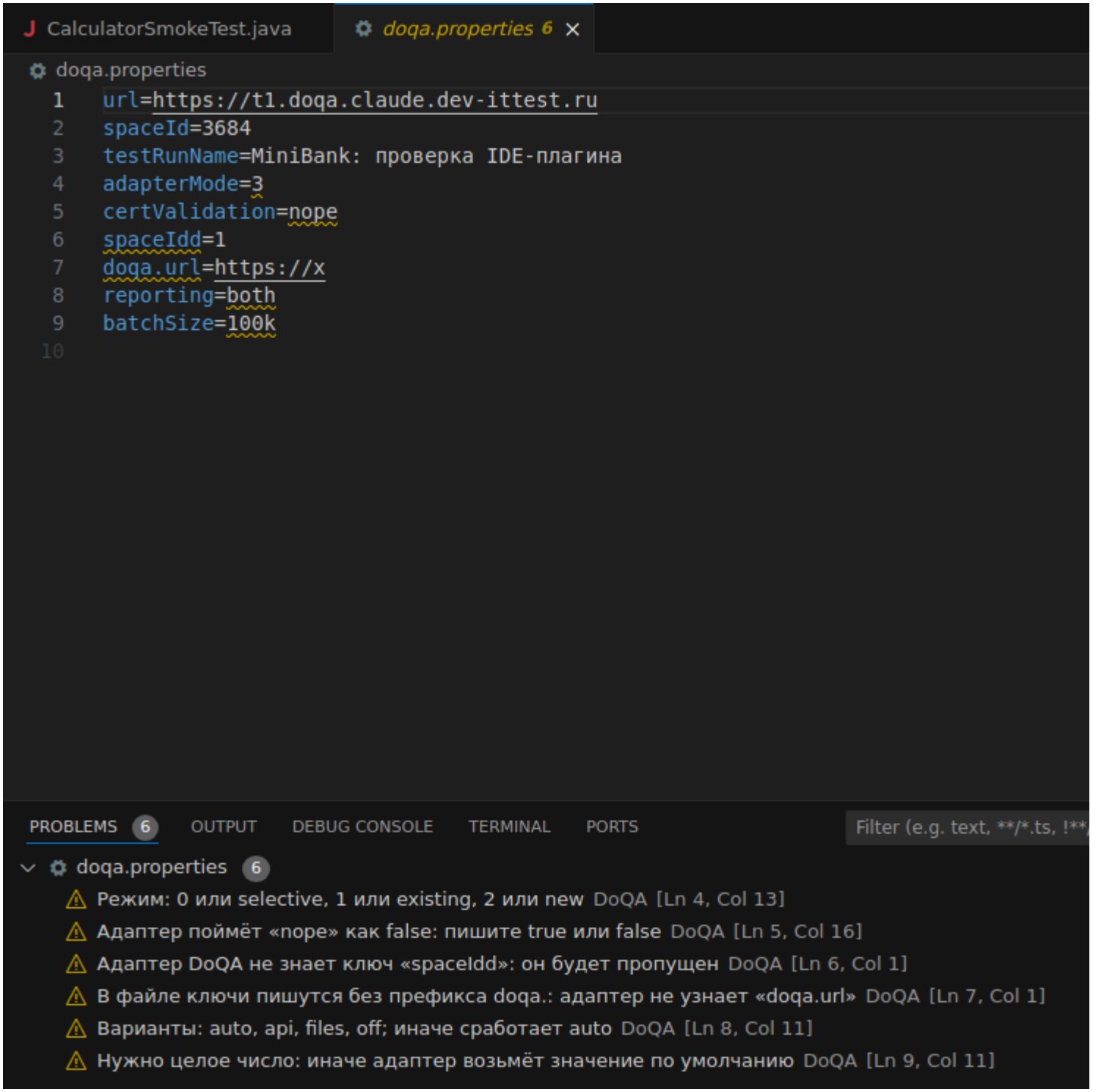




Список тегов и меток плагин загружает при первом дополнении из первых 200 автотестов пространства и обновляет не чаще раза в 10 минут. Если DoQA недоступен или в каталоге нет ни одного тега и метки, плагину нечего предложить. Чем теги отличаются от меток, описано на страницах адаптеров, например **Адаптер JUnit 5**.

Проверка doqa.properties

В файле `doqa.properties` плагин подчёркивает строки, которые адаптер DoQA (JVM или Jest) прочитал бы не так, как вы ожидаете. Текст предупреждения виден при наведении на подчёркнутое место и в панели **Problems** (**Ctrl+Shift+M**) с источником «DoQA».



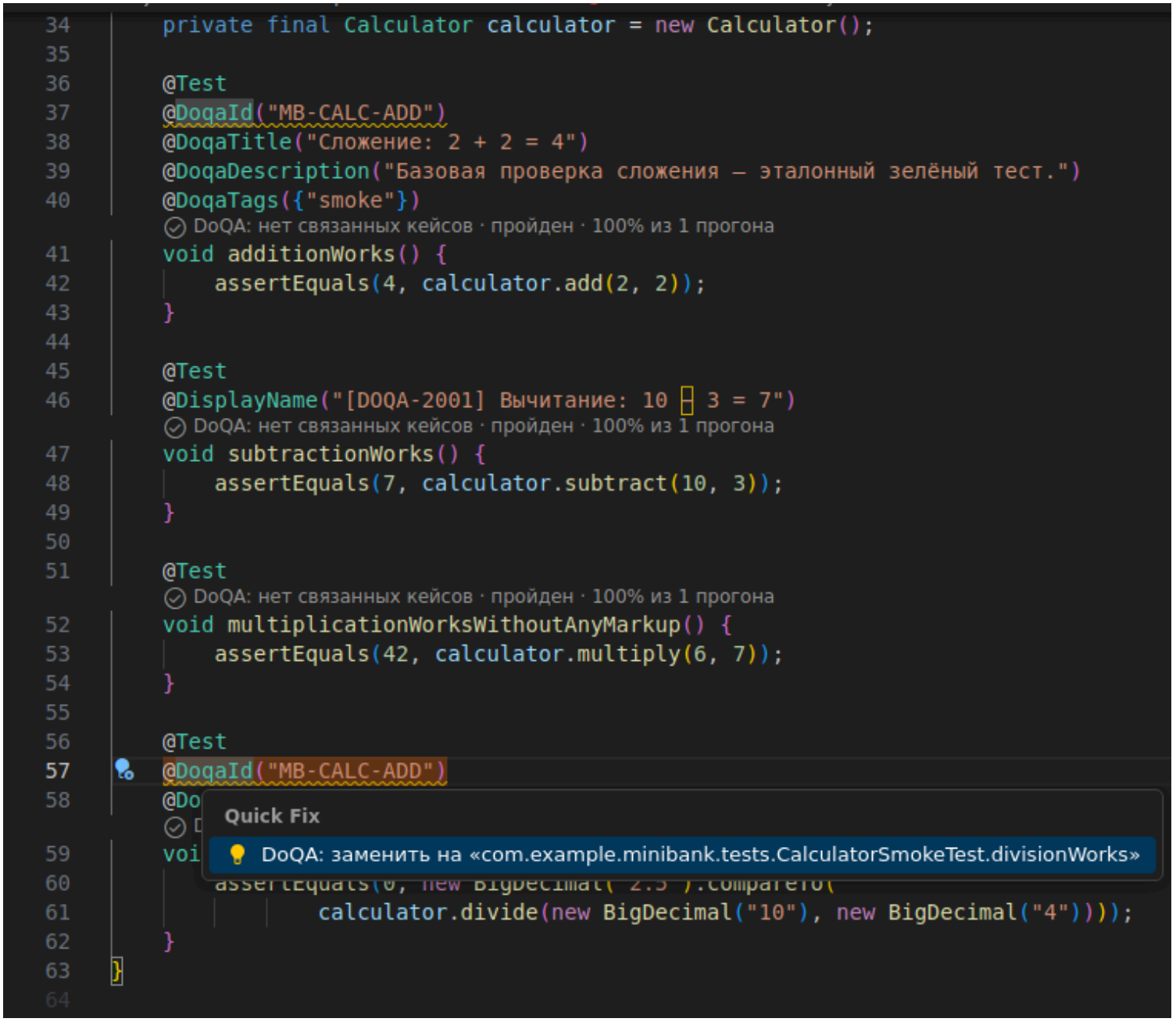
Проверки и тексты предупреждений те же, что в IntelliJ. Полная таблица есть в разделе **Проверка doqa.properties**. Несколько отличий VS Code:

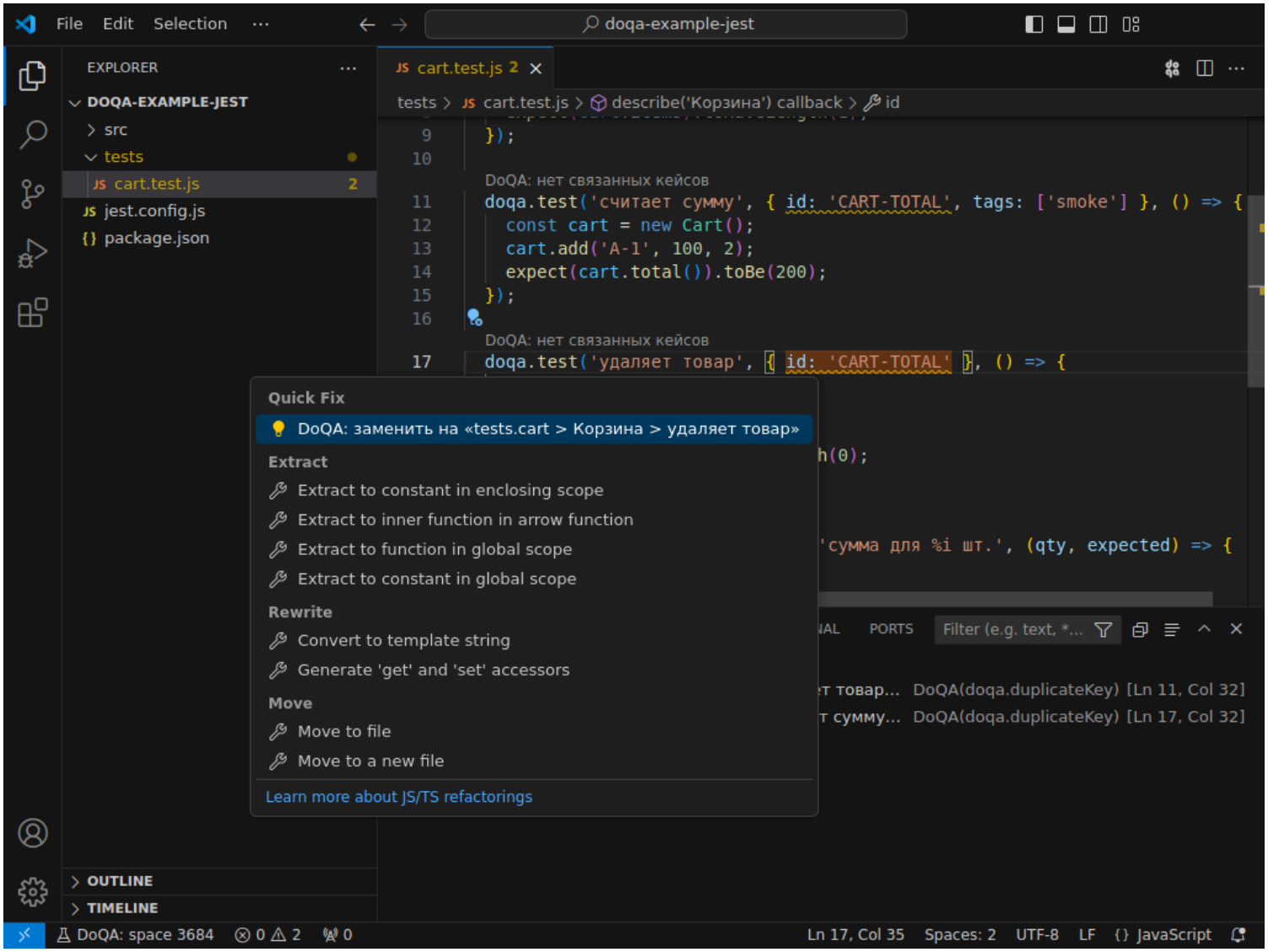
- отдельный плагин для `.properties` не нужен: проверка работает в любом файле с именем `doqa.properties`;
- токен считается закоммиченным, если файл уже добавлен в git (плагин спрашивает `git ls-files`). Для этого в системе должен быть git. Проверка повторяется при сохранении файла;
- на месте ключа плагин предлагает ключи, которых в файле ещё нет, с коротким описанием, а после `=` или `:` подсказывает допустимые значения: `true` / `false` для флагов, `selective`, `existing` и `new` для `adapterMode`, `auto`, `api`, `files` и `off` для `reporting`, `jest` и `plan` для `executionOrder`. Ключи адаптера Jest `projectName` и `executionOrder` описаны в том же разделе.

Повторяющийся внешний ID

Если у двух тестов одинаковый `@DoqaId` (в pytest `@doqa.id`, в Jest `id` в метаданных `doqa.test`), DoQA сочтёт их одним автотестом и смешает их прогоны. Плагин подчёркивает значение аннотации (в Jest запись `id`) у каждого из таких тестов: «Ключ «...» повторяется (...): DoQA сочтёт тесты одним автотестом». В скобках перечислены другие тесты этого файла с тем же ключом («у имяМетода») и другие файлы, где он встречается («в ИмяФайла.java»).

Чтобы исправить повтор, поставьте курсор на подчёркнутое значение, нажмите **Ctrl+.** (или щёлкните по лампочке) и выберите «**DoQA: заменить на «...»**». Плагин запишет вместо повторяющегося ключа читаемый: полное имя теста. Этого варианта нет, если у теста уже стоит его читаемый ключ или ключ задан на классе: тогда поменяйте ключ у другого теста или впишите свой.


private final Calculator calculator = new Calculator();
@Test
@DoqaId("MB-CALC-ADD")
@DoqaTitle("Сложение: 2 + 2 = 4")
@DoqaDescription("Базовая проверка сложения — эталонный зелёный тест.")
@DoqaTags({"smoke"})
DoQA: нет связанных кейсов · пройден · 100% из 1 прогона
void additionWorks() {
assertEquals(4, calculator.add(2, 2));
}
@Test
@DisplayName("[DOQA-2001] Вычитание: 10 − 3 = 7")
DoQA: нет связанных кейсов · пройден · 100% из 1 прогона
void subtractionWorks() {
assertEquals(7, calculator.subtract(10, 3));
}
@Test
DoQA: нет связанных кейсов · пройден · 100% из 1 прогона
void multiplicationWorksWithoutAnyMarkup() {
assertEquals(42, calculator.multiply(6, 7));
}
@Test
@DoqaId("MB-CALC-ADD")
Quick Fix
DoQA: заменить на «com.example.minibank.tests.CalculatorSmokeTest.divisionWorks»
calculator.divide(new BigDecimal("10"), new BigDecimal("4"))));
}
}



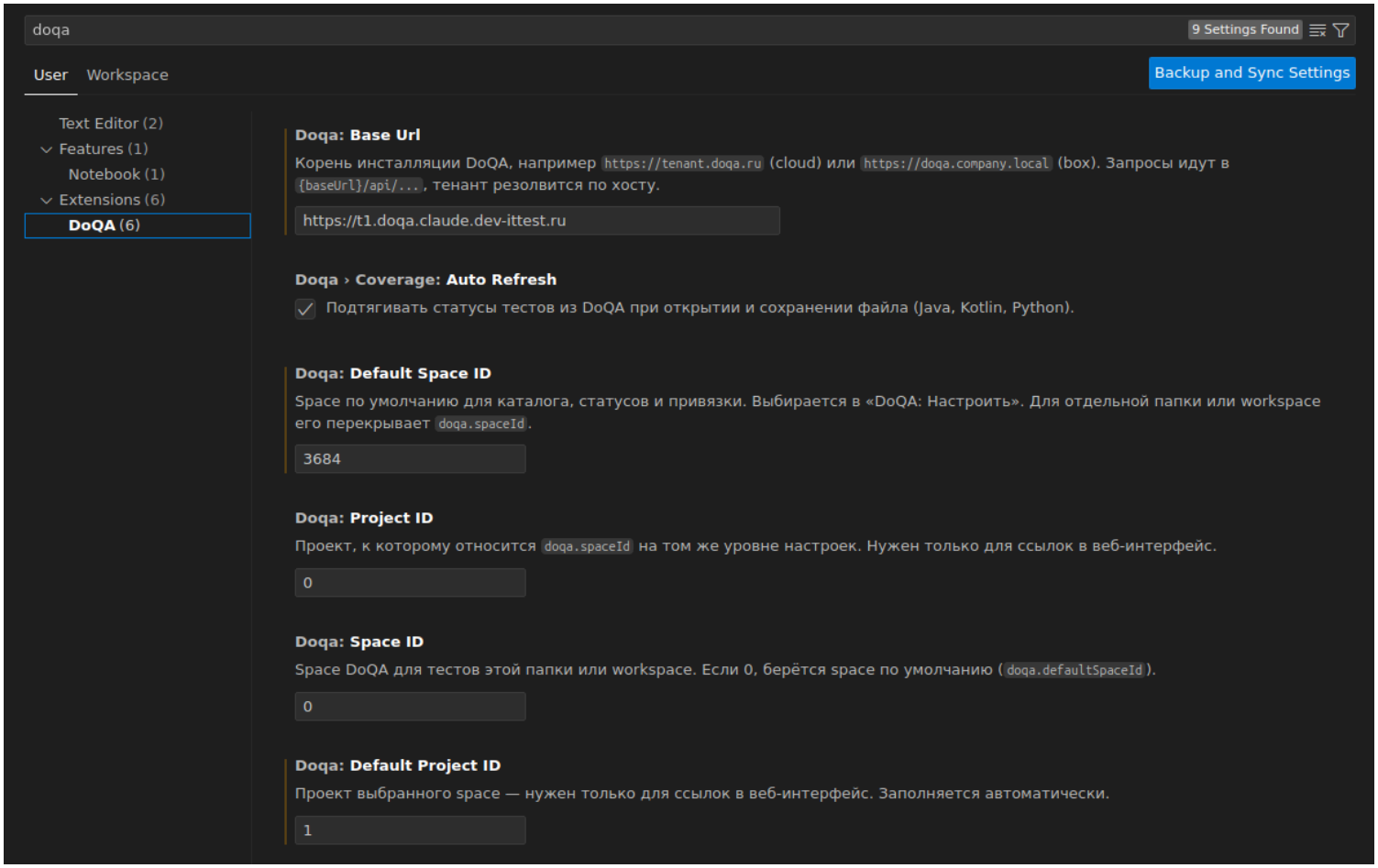
Повторы в других файлах плагин находит после того, как в фоне прочитает файлы рабочей области (кроме `node_modules`, `build`, `target`, `.venv` и подобных каталогов). Из файлов JavaScript и TypeScript заранее читаются только тестовые файлы по шаблонам Jest по умолчанию (`__tests__`, `*.test.*`, `*.spec.*`); файл с другим именем, например по своему `testMatch`, участвует в проверке, когда открыт в редакторе. Сравниваются ключи файлов, которые отчитываются в одно пространство: одинаковый ключ в разных пространствах означает два разных автотеста. Выключить эту проверку в настройках нельзя.

Смена ключа у существующего автотеста

Если автотест с этим ключом уже есть в DoQA, новый ключ станет новым автотестом, и история прогонов останется у старого. Меняйте ключ у того теста, который появился позже.

Настройки

Настройки плагина находятся в **File → Preferences → Settings** (**Ctrl+,**), в поиске наберите «doqa». Адрес, пространство и проект по умолчанию удобнее задавать командой «DoQA: Настроить»: она записывает их в пользовательские настройки.



Настройка	Что задаёт	Где действует
`doqa.baseUrl`	адрес DoQA, например `https://tenant.doqa.ru`. Если значение пустое, плагин не настроен	пользователь, рабочая область
`doqa.defaultSpaceId`	пространство по умолчанию; выбирается в «DoQA: Настроить». При `0` пространство не выбрано	пользователь, рабочая область
`doqa.defaultProjectId`	проект пространства по умолчанию, нужен только для ссылок в веб-интерфейс; заполняется сам	пользователь, рабочая область

Настройка	Что задаёт	Где действует
`doqa.spaceId`	пространство для тестов этой папки или рабочей области; перекрывает пространство по умолчанию. При `0` пространство не задано	пользователь, рабочая область, папка
`doqa.projectId`	проект, к которому относится `doqa.spaceId` на том же уровне; нужен только для ссылок	пользователь, рабочая область, папка
`doqa.coverage.autoRefresh`	подтягивать статусы тестов из DoQA при открытии и сохранении файла; включено по умолчанию	пользователь, рабочая область, папка

Токена среди настроек нет: он хранится в ключнице ОС, см. **Где хранится токен**.

Отличия от плагина для IntelliJ

Функции у плагинов одни и те же, а интерфейс устроен по-разному:

	IntelliJ	VS Code
Статус теста	значок в gutter	строка CodeLens над тестом
Подсказка	при наведении на значок	при наведении на объявление или аннотации теста
Меню действий	попап у значка, кнопка в тулбаре, **Tools → DoQA**, Alt+Enter	список быстрого выбора по щелчку на CodeLens или **Ctrl+Shift+Alt+D**, подменю DoQA, палитра команд
История прогонов, карточка кейса	всплывающие окна	вкладки рядом с редактором
Space проекта	`.idea/doqa.xml`, отдельно для модулей	`doqa.spaceId` в `.vscode/settings.json`, отдельно для папок рабочей области
Плагин не настроен	баннер над файлом	«DoQA: войти» в строке состояния

	IntelliJ	VS Code
Обновление статусов	при открытии файла	при открытии, переключении на вкладку и сохранении файла
Каталог автотестов	первые 200 автотестов, Enter без кейса копирует ключ	поиск на сервере, до 100 автотестов, Enter без кейса закрывает список
Быстрые действия Alt+Enter	привязать кейс, закрепить ключ, открыть кейс	нет; **Ctrl+.** только исправляет повторяющийся ключ
Проверка `doqa.properties`	нужен плагин Properties	работает сразу, для проверки токена нужен git
Повторяющийся ключ	инспекцию можно выключить	проверка всегда включена
Тесты Jest	значки в gutter только в IDE с плагином JavaScript (IntelliJ IDEA Ultimate, WebStorm)	CodeLens работает без дополнительных расширений

Ограничения

Ограничения плагина для IntelliJ относятся и к VS Code: плагин читает текст файла, не видит наследования из других файлов, в pytest знает только сам файл и конфигурацию pytest, в Jest только сам файл и ближайший конфиг Jest (без переменных окружения и флагов командной строки), поиск кейсов останавливается на 50 результатах. Проект Jest плагин ищет не выше папки рабочей области. Полный список приведён в разделе **Ограничения** страницы «Помощь в редакторе и черновики тестов».

Что есть только в VS Code:

- **CodeLens и подсказки есть только у файлов на диске.** У несохранённого нового файла и файлов из других источников (например, сравнение версий в git) строк CodeLens нет.
- **Похожие кейсы сверху только до начала ввода.** Пока в поле поиска есть текст, VS Code сортирует список сам, см. **Привязка кейса**.
- **Для ссылок в веб-интерфейс нужен номер проекта.** Если `doqa.spaceId` задан вручную без `doqa.projectId`, ссылок «Кейс в DoQA», «Автотест в DoQA» и «Открыть в DoQA» не будет.

- **Пока нет** баннера «DoQA не настроен» над файлом, проверки «кейс устарел» и панели с деревом покрытия.

Смотрите также

- **Плагины для IDE**: установка плагина и персональный токен
- **Работа с тестами в IntelliJ**: тот же набор функций в IntelliJ IDEA и PyCharm
- **Помощь в редакторе и черновики тестов**: разметка черновика, проверки `doqa.properties`, общие ограничения
- **Автотесты**: что такое внешний ID и как DoQA опознаёт автотест
- **Каталог автотестов**: тот же каталог в веб-интерфейсе
- **Нестабильные тесты и карантин**: как DoQA определяет стабильность и что делает карантин

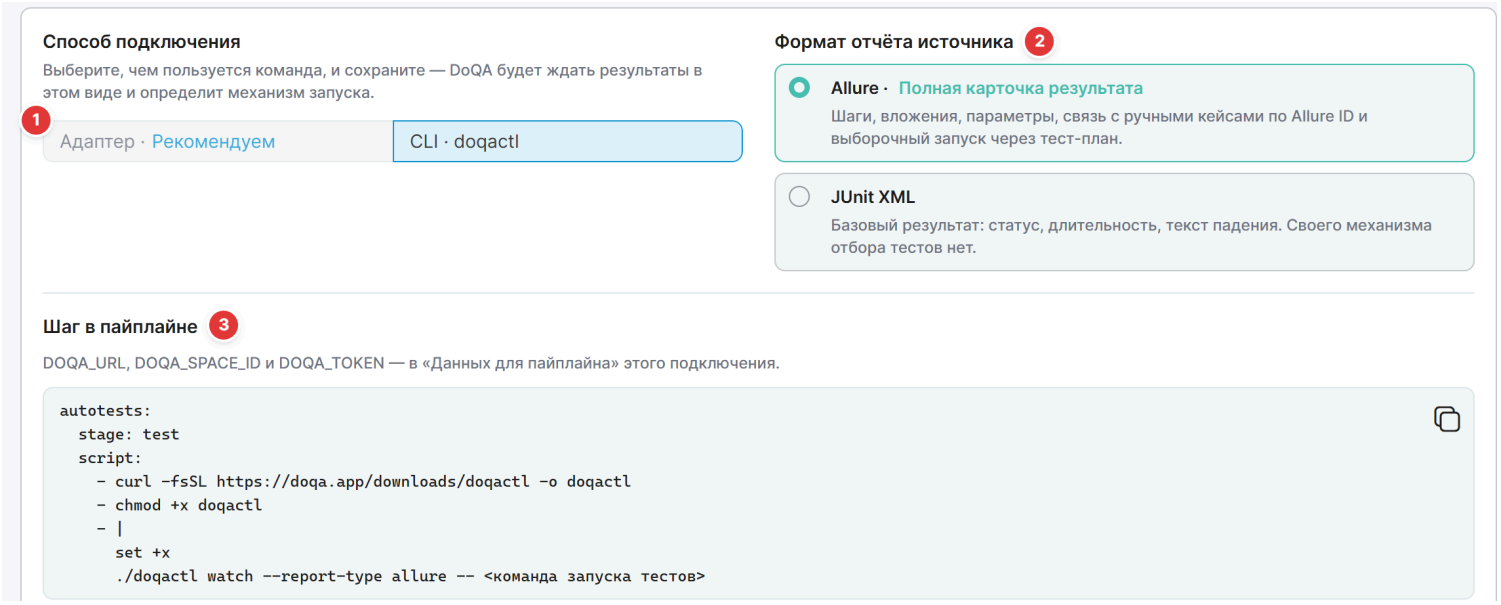
Утилита doqactl

`doqactl` — утилита командной строки: отправляет результаты автотестов в DoQA и возвращает вердикт **Quality Gate** кодом выхода. Нужна, когда для вашего фреймворка нет адаптера или результаты отправляются из пайплайна без правки проекта тестов.

Где находится

Утилиту не запускают из интерфейса. Её скачивают в пайплайн CI или на локальную машину. Ссылку на неё и готовый шаг пайплайна DoQA показывает здесь: Пространство → «Настройки» → «Подключение CI/CD» → карточка подключения → «Параметры источников» → «Способ подключения» → вкладка «CLI · doqactl».

Экран видят участники пространства с доступом к его настройкам, менять способ подключения может участник с правом управления CI/CD-подключениями. Сама утилита работает не от имени пользователя, а от токена проекта: ролей и прав интерфейса она не проверяет. Где взять токен и данные для пайплайна — **Подключение CI/CD**.



На скриншоте: 1 — вкладки «Адаптер» и «CLI · doqactl», 2 — «Формат отчёта источника», 3 — «Шаг в пайплайне» с командой doqactl.

Когда выбрать CLI, а не адаптер

	«Адаптер»	«CLI · doqactl»
Фреймворк	JUnit 5, JUnit 4, TestNG, pytest — см. Адаптеры	любой
Установка	зависимость в проекте тестов	один бинарь, скачивается в шаге пайплайна
Что видно в результате	шаги, фикстуры, вложения, параметры	Allure — полная карточка результата; JUnit XML — статус, длительность и текст падения
Выборочный запуск	адаптер получает набор тестов от DoQA (у JUnit 4 — не во всех режимах)	Allure — по тест-плану; JUnit XML — только нативным фильтром раннера, его задают в блоке «Выборочный запуск источника»

Установка

Бинарь называется `doqactl`; прежнее имя `doqa-cli` работает как алиас файла: старые пайплайны менять не обязательно. Весь вывод утилиты — по-английски.

```bash
curl -fsSL https://doqa.app/downloads/doqactl -o doqactl
chmod +x doqactl
```

Сборки есть для Linux (amd64, arm64), macOS (amd64, arm64) и Windows (amd64).

Номеров версий у утилиты нет: `doqactl version` печатает версию сборки.

Форматы отчётов

Формат	Что ищет `doqactl`	Каталоги по умолчанию (`watch`)
Allure	сырые файлы `*-result.json` (+ `*-container.json` с фикстурами)	`allure-results`, `target/allure-results`
JUnit XML	файлы `*.xml` (surefire)	`target/surefire-reports`, `surefire-reports`

Тип определяется по имени файла: оканчивается на `-result.json` → Allure; на `.xml` → JUnit. Смешанный набор обоих форматов в одном отчёте — ошибка. Ни того ни другого — тоже ошибка (тексты — в разделе **Частые ошибки**).

Быстрый старт

1. В блоке «Способ подключения» откройте вкладку «CLI · doqactl», в поле «Формат отчёта источника» выберите Allure или JUnit XML и нажмите «**Сохранить**». Пока способ не сохранён, у источника стоит бейдж «Способ не выбран» и DoQA не знает, в каком виде ждать результаты.

2. Перенесите значения из блока «Данные для пайплайна» в переменные своей CI-системы: `DOQA_URL`, `DOQA_SPACE_ID`, `DOQA_TOKEN`.

3. Добавьте в пайплайн шаг, который интерфейс печатает в блоке «Шаг в пайплайне»:

```bash
curl -fsSL https://doqa.app/downloads/doqactl -o doqactl
chmod +x doqactl
set +x
./doqactl watch --report-type allure -- <команда запуска тестов>
```

4. Запустите пайплайн. Результаты уходят в прогон по ходу тестов, ждать конца сборки не нужно.

Вместо переменных значения можно передать флагами `--base`, `--token`, `--space`. Токен — API-токен проекта, которому принадлежит пространство; его создают в админке: «Проекты» → карточка проекта → вкладка «API-токены» (нужны права администратора).

Общая конфигурация

Приоритет источников для одного и того же параметра: **флаг > переменная окружения > файл** `~/.doqactl.yaml` (создаётся командой `login`). Исключение: явно переданный пустой флаг (`--token ""`) к переменной окружения не откатывается: это защита от джобы с незаполненной переменной, которая иначе ушла бы с чужим токеном раннера.

Параметр	Флаг	Переменные окружения (по приоритету)
Базовый URL	`--base` (у `login` — `--url`)	`DOQA_API_BASE`, `DOQA_URL`, `DOQA_ENDPOINT`
Токен	`--token`	`DOQA_CI_TOKEN`, `DOQA_TOKEN`
Пространство	`--space` (у `gate` — `--space-id`, второе имя всегда есть скрытым алиасом)	`DOQA_SPACE_ID`, `DOQA_SPACE`
Пайплайн	`--pipeline-id`	см. таблицу автоопределения
Проект в CI	`--ci-project-id`	`DOQA_CI_PROJECT_ID`, `CI_PROJECT_ID`
Ветка	`--branch`	`DOQA_BRANCH`, `CI_COMMIT_REF_NAME`, `CI_BRANCH`, `GITHUB_REF_NAME`, `GIT_BRANCH`, `BRANCH_NAME`
Коммит (только `watch`)	`--commit`	`DOQA_COMMIT_SHA`, `CI_MERGE_REQUEST_SOURCE_BRANCH_SHA`, head PR из `GITHUB_EVENT_PATH`, `CI_COMMIT_SHA`, `GITHUB_SHA`, `GIT_COMMIT`, `BUILD_VCS_NUMBER`, `BUILD_SOURCEVERSION`, `CIRCLE_SHA1`, `BITBUCKET_COMMIT`, `bamboo_planRepository_revision`
Система-источник	`--system`	`DOQA_SYSTEM` (по умолчанию — хост базового URL)
Ключ CI-подключения (`upload`, `run`)	`--source-key`	`DOQA_SOURCE_KEY`
Correlation id (`upload`, `run`)	`--correlation-id`	`DOQA_CORRELATION_ID`
Прогон, заведённый DoQA (только `watch`)	—	`DOQA_TEST_RUN_ID` (только из окружения)

Хвостовой `/api` в базовом URL не удваивается: `--base https://doqa.example/api` и `--base https://doqa.example` дают один и тот же адрес. Токен вырезается из сообщений об ошибках и из тела ответа перед печатью, поэтому в лог CI-джобы он не попадает.

`--source-key` / `DOQA_SOURCE_KEY` и `--correlation-id` / `DOQA_CORRELATION_ID` нужны, когда у пространства несколько CI-подключений: без них результат отклоняется с 409. `doqactl` сам подставляет `DOQA_SOURCE_KEY`, если он есть в окружении. Адаптеры JVM и pytest этого пока не делают.

Как определяется CI-система и пайплайн

Значение обязано совпасть с тем, каким DoQA адресует запущенный ею прогон: похожее, но чужое значение прогон не найдёт и заведёт второй.

CI-система	Откуда берётся идентификатор пайплайна
Любая (явно)	`DOQA_PIPELINE_ID` — проверяется первым и перекрывает всё остальное
Jenkins	`JOB_NAME` + `BUILD_NUMBER` → `{путь job}#{номер}`, каждый `/` в `JOB_NAME` заменяется на `/job/`. Пример: `JOB_NAME=doqa-e2e/my-job`, `BUILD_NUMBER=12` → `doqa-e2e/job/my-job#12`
TeamCity	`%teamcity.build.id%`: сначала переменная `TEAMCITY_BUILD_ID`, иначе ключ `teamcity.build.id` (или `build.id`) из файла по пути `TEAMCITY_BUILD_PROPERTIES_FILE`
GitLab	`CI_PIPELINE_ID`
GitHub Actions, GitVerse	`GITHUB_RUN_ID`
Azure DevOps	`BUILD_BUILDID`
CircleCI	`CIRCLE_PIPELINE_ID`
Bitbucket	`BITBUCKET_PIPELINE_UUID`
Bamboo	`bamboo_buildResultKey`

`TEAMCITY_BUILD_ID` и `BUILD_NUMBER` принимаются, только если состоят из одних цифр. В TeamCity нужен именно идентификатор сборки `teamcity.build.id`, а не её номер `BUILD_NUMBER`. Там, где провайдер не отдаёт нужное значение, задайте его вручную, например `DOQA_PIPELINE_ID: "%teamcity.build.id%"`.

Команда watch

Прогоняет тестовую команду в CI и отправляет результаты по мере готовности — основная команда для пайплайна.

```bash
doqactl watch [флаги] -- <команда запуска тестов>
```

Алиас — `stream`. Всё после `--` — команда тестов; без неё `watch` только следит за каталогом результатов уже идущего прогона.

Как работает:

1. Находит прогон. С заданным `--pipeline-id` запрашивает у DoQA тест-план для прогона, запущенного из DoQA. Если плана нет, создаёт новый прогон. Без `--pipeline-id` заводит пустой прогон (локальный сценарий). Готовый прогон можно задать флагом `--run`, а прогон, заведённый DoQA, подхватывается из `DOQA_TEST_RUN_ID`.
2. Если план получен — пишет `testplan.json` и передаёт его тестовой команде; для `mvn` / `mvnw` сам добавляет `-Dallure.testplan.path=<путь>` и `-Dallure.filter.testplan=true`. Файл удаляется после прогона, если не задан `--keep-testplan`.
3. Перед запуском чистит каталоги отчётов (кроме `--no-clean`).
4. Запускает тестовую команду и отправляет каждый дописанный результат отдельным запросом сразу, как он перестал меняться. Если есть вложения, запрос уходит multipart'ом. Готовый результат ждёт свои файлы фикстур до `--fixture-wait` секунд.
5. После выхода команды дособирает оставшиеся результаты ещё до `--wait-results` секунд, затем возвращает код.

План не пишется, если он не выразит набор целиком: у части автотестов нет ни селектора, ни Allure ID, либо набор пуст. Тогда прогон идёт полностью, а сужает его нативный фильтр —

выражение в переменной `DOQA_NATIVE_FILTER`, которую подставляет рецепт пайплайна. Сам `doqactl` её не читает.

Отчёты JUnit XML `watch` отправляет только в новый прогон: в прогоне, запущенном из DoQA, элементы уже созданы тест-планом и сопоставляются по Allure-идентичности, которой в JUnit XML нет. <!-- TODO verify: что происходит с результатами, если прогон запущен из DoQA, а у источника выбран формат JUnit XML — интерфейс в этом случае всё равно печатает шаг с --report-type junit -->

Флаг	По умолчанию	Что делает
`--results-dir <каталог>`	—	корень, внутри которого искать стандартные каталоги отчётов
`--result <каталог>`	`allure-results`, `target/allure-results`	точный каталог результатов
`--report-type`	авто (сначала Allure, потом JUnit)	жёстко задать формат: `allure` или `junit`
`--run <id>`	—	отправлять по ходу выполнения в существующий прогон по его id, не определяя его
`--commit <sha>`	head-коммит из окружения CI	сузить тест-план анализом влияния, если он включён для пространства
`--title`	—	название создаваемого прогона
`--ci-type`	`gitlab`	тип провайдера при создании прогона
`--system`	хост базового URL	идентификатор системы-источника
`--interval`	`2` секунды	интервал опроса каталога результатов
`--idle`	`0` (до конца команды или Ctrl-C)	остановиться после N секунд без новых результатов
`--wait-results`	`300` секунд	сколько ждать первый результат после выхода команды и досылать зависшую доставку

Флаг	По умолчанию	Что делает
`--fixture-wait`	`10` секунд	сколько готовый результат ждёт свои файлы фикстур во время прогона
`--keep-testplan`	выключен	не удалять сгенерированный `testplan.json`
`--no-clean`	выключен	не чистить каталоги отчётов перед запуском

Плюс общие флаги: `--base`, `--token`, `--space`, `--pipeline-id`, `--branch`, `--ci-project-id`.

```bash
./doqactl watch --result target/allure-results -- mvn test
```

Коды возврата: `0` — успех; код упавшей тестовой команды пробрасывается наружу как есть; если команда прошла, а ни одного результата не доставлено — `2`.

Команда upload

Загружает уже готовый отчёт (файл, ZIP или каталог) одной командой, для локального запуска или когда отчёт уже собран в CI.

```bash
doqactl upload [флаги] <файл|каталог>
```

Ровно один позиционный аргумент: файл (`*.xml` или `*-result.json`), ZIP-архив или каталог результатов (каталог архивируется на лету).

Флаг	По умолчанию	Что делает
`--type`	`auto`	формат отчёта: `auto`, `allure` или `junit`
`--title`	—	название создаваемого прогона
`--system`	хост базового URL	идентификатор системы-источника

Флаг	По умолчанию	Что делает
`--external-id`	—	внешний идентификатор отчёта
`--work-items`	—	id тест-кейсов через запятую, которые привязать к отчёту
`--source-key`	—	ключ CI-подключения, обязателен при нескольких подключениях у пространства
`--correlation-id`	—	идентификатор, по которому DoQA находит свой прогон, пока CI не сообщил идентификатор пайплайна

Плюс общие флаги: `--base`, `--token`, `--space`, `--pipeline-id`, `--branch`, `--ci-project-id`. Лимит файла — **50 МБ**.

```bash
./doqactl upload --pipeline-id "$CI_PIPELINE_ID" allure-results/
```

Без `--pipeline-id` отчёт уйдёт в отдельный прогон

Загрузка отчёта не принимает `DOQA_TEST_RUN_ID`: отчёт адресуется только идентификатором пайплайна. Без `--pipeline-id` (или `DOQA_PIPELINE_ID`) результаты будут отправлены в новый прогон, а прогон, запущенный из DoQA, останется пустым. Команда предупреждает об этом в логе.

Если рядом с результатами лежит файл `doqa-reporting.properties` (его пишет JVM-адаптер при отказе доставки), `upload` берёт из него `pipelineId`, `ciRunId`, `sourceKey`, `correlationId`, `branch`, но не токен и не адрес сервера. Приоритет — явный флаг, затем переменная `DOQA_*`, затем данные из файла, затем общие переменные CI.

Коды возврата: `0` — успех, в stdout `Report accepted: runId=<id>`; `2` — ошибка (файл больше лимита, сервер не вернул `runId`, сетевая или HTTP-ошибка).

Команда run

Выполняет тестовую команду и после неё загружает готовый отчёт — короткий локальный сценарий. Для CI используйте `watch`.

```bash
doqactl run [флаги] --result <путь> -- <команда тестов>
```

stdout/stderr/stdin тестовой команды проксируются как есть. Отчёт из `--result` грузится независимо от исхода тестов, тем же путём, что `upload`; флаги — те же, что у `upload`, плюс:

Флаг	По умолчанию	Что делает
`--result <путь>`	—	путь к отчёту для загрузки после тестов; **обязателен**

```bash
./doqactl run --result allure-results/ -- pytest
```

Коды возврата: без `--result` — `2` (`run failed: --result is required to upload the report`); иначе наружу идёт код тестовой команды. Если тесты упали и загрузка отчёта тоже не удалась — наружу код тестов, а ошибка загрузки печатается отдельной строкой `doqactl: upload failed: …`.

Команда gate

Возвращает вердикт Quality Gate по пайплайну кодом выхода, чтобы блокировать пайплайн без похода в интерфейс.

```bash
doqactl gate [флаги]
```

Позиционных аргументов нет. Нужны токен, положительный целый `--space` / `--space-id` и `--pipeline-id` (без него — `pipeline id is required`). Статус `pending` опрашивается раз в секунду до **60 секунд**, дальше — `timed out waiting for quality gate result`.

Дополнительных флагов, кроме общих (`--base`, `--token`, `--space` / `--space-id`, `--pipeline-id`, `--ci-project-id`), у команды нет.

```bash
./doqactl gate --pipeline-id "$CI_PIPELINE_ID"
```

Вывод:

```text
Quality Gate: failed (blocking=true)
Violated criteria:
  - pass_rate: expected 95, actual 92.31
Evaluated metrics:
  duration_ms: 812000
```

При отсутствии нарушений или метрик печатается `Violated criteria: none` / `Evaluated metrics: none`.

Коды возврата: `0` — гейт не блокирует; `1` — блокирует; `2` — ошибка аргументов, сети, HTTP или разбора ответа (в том числе неизвестный статус в ответе — это не считается успешным прохождением гейта).

Вердикт — про весь **прогон**, а не про часть, затронутую текущим пайплайном: пайплайн, перезапустивший часть тестов (перезапуск выбранных, перезапуск одного автотеста, **авто-повтор**), получает `passed` только тогда, когда успешен весь прогон.

Команда sync-ids

Сопоставляет локальные тесты с идентификаторами автотестов в DoQA и, по флагу, проставляет их в исходники.

```bash
doqactl sync-ids [--dir <каталог>] [--lang <язык>] [--update-ids] [--create]
```

Без `--update-ids` команда ничего не меняет, а печатает расхождения: каким тестам назначены идентификаторы, какие остались без них, какие идентификаторы в DoQA остались без тестов.

Флаг	По умолчанию	Что делает
`--dir`	`.` (текущий каталог)	каталог с исходниками тестов

Флаг	По умолчанию	Что делает
`--lang`	все языки	ограничить языком: `py`, `go`, `java`, `ts`, `js`, `php`, `cs`
`--update-ids`	выключен	вписать выданный id в исходники (идемпотентно, уже помеченные тесты не трогает)
`--create`	выключен	разрешить DoQA создать недостающие автотесты и вернуть их id; без флага запрос только читает

Пропускаемые каталоги: `.git`, `vendor`, `node_modules`, `.idea`, `.vscode`, `target`, `bin`, `obj`, `dist`, `build`, `__pycache__`.

```bash
./doqactl sync-ids --dir ./src/test --lang java --update-ids
```

`--update-ids` правит файлы в рабочей директории

Команда дописывает идентификатор прямо в исходный код теста: это изменение рабочего дерева, как любая другая правка файла. Уже помеченные тесты не трогаются: распознаются маркеры `@doqa.id(...)`, `@pytest.mark.doqa_id(...)`, `@DoqaId(...)`, `[DoqaId(...)]`, `#[DoqaId(...)]`, `//doqa:…`, а также идентификатор в названии теста вида `[DOQA-123]`.

Что вставляется по языкам:

Язык	Вставляемая строка
Python	`@doqa.id("DOQA-123")`
Java	`@DoqaId(123)` — голое число для числового id, иначе `@DoqaId("…")`
C#	`[DoqaId(123)]`
PHP	`#[DoqaId(123)]`
Go	`//doqa:123`
TS / JS	`// @DoqaId(123)`

Коды возврата: `0` — расхождений нет (или всё прошло с `--update-ids`); `3` — локальные тесты и id в DoQA разошлись; `4` — не найдено ни одного теста (`sync-ids: no tests found in <каталог>`).

Команда login

Сохраняет базовый URL, токен и пространство в `~/.doqactl.yaml`, чтобы не передавать их флагами при каждом локальном запуске.

```bash
doqactl login --base <URL> [--token <токен>] [--space <id>]
```

`--url` — алиас `--base`. Файл создаётся с правами `0600` (права сужаются даже у уже существующего файла); путь переопределяется переменной `DOQACTL_CONFIG`.

Токен можно не светить в argv: он берётся из `--token`, иначе из `$DOQA_TOKEN` / `$DOQA_CI_TOKEN`, иначе из первой строки stdin, если stdin не терминал:

```bash
printf '%s' "$DOQA_TOKEN" | doqactl login --base https://doqa.example --space 42
```

Без базового URL или токена — ошибка `base URL (--base/--url) and token (--token, $DOQA_TOKEN, or stdin) are required` и код `2`. Успех — `Saved credentials for <URL> to <путь>`.

Устаревшая команда `report` этот конфиг игнорирует (кроме `spaceId` в старой форме аргументов).

Устаревшая команда report

Отправляет отчёт на URL, указанный первым аргументом, а не на адрес из `--base`. Оставлена для совместимости со старыми пайплайнами. Для новых используйте `upload`.

```bash
doqactl report <url> <spaceId> <token> <file> [type] [runName]
doqactl report <url> <token> <file> [type]
```

Форма распознаётся по второму аргументу: число — это `spaceId`, иначе `spaceId` берётся из `--space`, `DOQA_SPACE_ID` или файла `~/.doqactl.yaml`. `[type]` по умолчанию `zip` (файл или каталог упаковывается в архив); значение `allure` отправляет файл как есть. `[runName]` (или флаг `--title`) — название прогона. Передать его можно только вместе с `[type]`, потому что позиции аргументов фиксированы.

```bash
./doqactl report https://example.doqa.app/api/autotests/report 2 abc123 allure-results
```

Коды возврата — общие: `0` успех, `2` ошибка аргументов, сети или HTTP.

Прочие команды

Команда	Что делает
`version`	печатает версию сборки
`completion`	скрипт автодополнения оболочки
`help`	справка по команде; вызов `doqactl` без подкоманды печатает то же

Коды возврата (сводно)

Код	Значение
`0`	успех
`1`	блокирующий вердикт Quality Gate (`gate`)
`2`	ошибка аргументов, сети, HTTP или разбора ответа

Код	Значение
`3`	`sync-ids`: локальные тесты и id в DoQA разошлись
`4`	`sync-ids`: не найдено ни одного теста
код тестовой команды	`watch` / `run`: пробрасывается код возврата обёрнутой команды тестов

Частые ошибки

Сообщение	Причина	Действие
`Invalid token` / «Недействительный токен.» (401)	токен не передан или не найден	проверить `--token` / `DOQA_TOKEN`
`Token is not allowed for this space` / «Токен не имеет доступа к этому спейсу.» (403)	токен принадлежит другому проекту, чем пространство	использовать токен проекта, которому принадлежит это пространство
`Token is invalid` (403, `error_token`)	токен не прошёл проверку при загрузке отчёта	тот же токен, что и остальные запросы, — токен проекта
`file_too_large` — «Максимальный размер файла — 50 МБ» (422)	файл отчёта больше лимита	сузить отчёт или разбить на несколько загрузок
`upload payload exceeds the 50 MB limit`	то же самое при `upload` / `run`	то же самое
`invalid_space_id` — «spaceId must be an integer» (422)	`--space` не число	проверить значение `--space` / `DOQA_SPACE_ID`
`report contains both Allure results and XML files; set --type explicitly`	в отчёте одновременно `*-result.json` и `*.xml`	явно задать `--report-type` / `--type`
`could not detect report type; expected *-result.json`	ни Allure-, ни JUnit-файлов не найдено	проверить путь и содержимое каталога

Сообщение	Причина	Действие
`or *.xml`		результатов
`doqactl: warning: no pipeline id resolved; the report will go to a separate run instead of run <id> started by DoQA …`	нет `--pipeline-id` / `DOQA_PIPELINE_ID` при `upload`	задать `--pipeline-id` тем значением, которое CI-провайдер сообщает DoQA
`attachment … not found`	файл вложения результата не найден или превышен лимит вложений одного результата (100 МБ, только у `watch`)	проверить путь к вложению; результат уйдёт без него с пометкой «Вложения не приняты»
`X of Y Allure result(s) were not delivered to run Z: …`	часть результатов не доставлена после завершения команды	код возврата `2`; смотреть перечисленные причины в сообщении
`run failed: --result is required to upload the report`	не передан `--result` у команды `run`	добавить `--result <путь>`
`pipeline id is required` (`gate`)	не передан `--pipeline-id`	добавить `--pipeline-id` / `DOQA_PIPELINE_ID`
`timed out waiting for quality gate result`	статус гейта остаётся `pending` дольше 60 секунд	проверить, что результаты пайплайна уже доставлены в DoQA
`sync-ids: no tests found in <каталог>`	`--dir` указывает не туда или тесты не распознаны	проверить путь и `--lang`
`base URL (--base/--url) and token (--token, $DOQA_TOKEN, or stdin) are required` (`login`)	не передан URL или токен	добавить `--base` и `--token` (или переменные/stdin)

Особенности

- Запросы за данными и пофайловые отправки результатов повторяются при сетевой ошибке и ошибке сервера. Создание прогона и загрузка отчёта при сетевой ошибке не повторяются, чтобы не завести второй прогон.
- Сбой отправки одного файла не останавливает остальные; результат, отклонённый DoQA ответом 4xx (`rejected by DoQA`), повторно не отправляется.
- Фикстура (например, общий teardown класса), записанная позже отправки её результата, к уже отправленному результату не добавляется: такие данные видны только при загрузке готового отчёта целиком (`upload`).

Смотрите также

- **Отправка результатов** — как выбрать между адаптером, CLI и загрузкой отчёта через интерфейс
- **Запуск автотестов из DoQA** — как DoQA передаёт прогон и тест-план в пайплайн
- **Autotest API** — прямые HTTP-запросы вместо утилиты, для своего адаптера

Autotest API

Autotest API — прямое HTTP-API `/api/autotests/*`, через которое адаптеры тестовых фреймворков отправляют результаты в DoQA в реальном времени. Эта страница — для тех, кто пишет собственный адаптер или интеграцию, потому что **готового адаптера** под их язык или фреймворк нет.

Когда нужно Autotest API

По сравнению с **выгрузкой готового отчёта** прямое API даёт:

- результаты в прогоне по ходу выполнения, а не после его окончания;
- шаги и фикстуры со статусами, длительностью и вложениями;
- вложения (скриншоты, видео, логи), привязанные к результату;
- определения автотестов в каталоге: заголовок, описание, метки, ссылки, дерево шагов;
- привязку автотестов к тест-кейсам;
- управление тест-раном: создать прогон, узнать набор выбранных к запуску автотестов, следить за сводкой;
- конфигурацию, в которой получен результат.

Если из этого списка ничего не нужно — проще отправлять готовые отчёты.

Тест-ран в Autotest API — это обычный **прогон** DoQA; отдельной сущности нет.

Аутентификация

Все запросы адаптера авторизуются API-токеном проекта — как его создать, см. **Создание API-токена**.

- Токен передаётся в поле `token`: в теле POST-запросов и в query-параметре GET-запросов. Заголовок `Authorization` не используется.
- Токен привязан к проекту: работать можно с любым пространством этого проекта. Пространство чужого проекта — ошибка 403 `{"error":{"message":"Token is not allowed for this space","status":403}}`.

- Токен не передан или не найден — ошибка 401 `{"error":{"message":"Invalid token","status":401}}`.

Домен экземпляра в облачной поставке

В облаке запросы отправляйте на домен вашего экземпляра (например, `https://company.doqa.app`): токен ищется в базе конкретного клиента, и на другом адресе тот же токен вернёт 401.

Обзор эндпоинтов

Эндпоинты адаптера (авторизация — токен проекта):

Метод и путь	Назначение
`POST /api/autotests/test-runs`	создать пустой прогон (тест-ран)
`GET /api/autotests/test-runs/{id}`	статус и сводка прогона по статусам
`GET /api/autotests/test-runs/{id}/autotests`	какие автотесты числятся в прогоне — набор для выполнения
`POST /api/autotests/run-start`	создать прогон и сразу привязать его к пайплайну CI
`POST /api/autotests/upsert`	создать или обновить определения автотестов в каталоге
`POST /api/autotests/sync-ids`	сопоставить локальные тесты со стабильными ID DoQA
`POST /api/autotests/results`	загрузить пачку результатов в прогон
`POST /api/autotests/attachments`	загрузить файл вложения

Остальные эндпоинты `/api/autotests/*` — каталог, карточка и история автотеста, массовые действия, запуск выбранных, перестановка порядка тест-рана — обслуживают интерфейс DoQA и работают под авторизацией пользователя. Адаптеру они не нужны.

Типовой сценарий адаптера

Autotest API — не набор независимых эндпоинтов, а последовательность:

1. **Определения (необязательно).** `POST upsert` — зарегистрировать или обновить определения автотестов: заголовки, описания, метки, дерево шагов, привязку к кейсам. Можно делать до старта тестов или в конце прогона.
2. **Прогон.** Если задана переменная окружения `DOQA_TEST_RUN_ID` — прогон уже создан DoQA, репортите в него. Иначе создайте свой: `POST test-runs` → `runId`.
3. **Набор.** Если `DOQA_ADAPTER_MODE=0` (запуск выбранных из DoQA) — запросите `GET test-runs/{id}/autotests?ciRunId=$DOQA_CI_RUN_ID` и выполняйте только перечисленные автотесты, в отданном порядке.
4. **Результаты.** По ходу прогона отправляйте `POST results` порциями. Вложения загружайте заранее через `POST attachments` и ссылайтесь на них по `mediaFileId`.
5. **Завершение.** Эндпоинта завершения прогона в Autotest API нет. Прогон, привязанный к пайплайну CI, DoQA завершает сама по окончании пайплайна (см. **Запуск автотестов из DoQA**); прогон локального запуска завершают в интерфейсе.

Переменные `DOQA_TEST_RUN_ID`, `DOQA_ADAPTER_MODE`, `DOQA_CI_RUN_ID` DoQA передаёт пайплайну при запуске автотестов из интерфейса — полный список переменных и их смысл описаны на странице **Запуск автотестов из DoQA**.

Тест-раны

Создание: POST /api/autotests/test-runs

Тело запроса (имена полей — camelCase):

Поле	Обязательное	Смысл
`token`	да	API-токен проекта
`spaceId`	да	ID пространства
`name`	нет	название прогона; по умолчанию — «Autotest run» (с `externalKey` — «Autotest run {ключ}»)

Поле	Обязательное	Смысл
`configurationId`	нет	конфигурация прогона; должна принадлежать пространству
`externalKey`	нет	внешний ключ запуска — попадает в название прогона
`pipelineId`	нет	ID пайплайна CI (эхо `CI_PIPELINE_ID`) — привязывает прогон к пайплайну
`branch`	нет	ветка пайплайна

Ответ — `201 {"runId": <id>}`. Прогон создаётся пустым и наполняется результатами по мере их загрузки.

`POST run-start` делает то же самое с телом `{token, spaceId, title?, pipelineId?, branch?}`. Используйте любой из двух путей, оба привязывают прогон к пайплайну, если передан `pipelineId`. Без привязки прогон из CI не покажет пайплайн и джобы, и по нему не сработает автоматика финализации.

externalKey — не ключ идемпотентности

`externalKey` используется только в названии прогона. Повторный `POST test-runs` с тем же ключом создаст второй прогон. Не полагайтесь на него для дедупликации. Не повторяйте запрос создания, если ответ получен.

Сводка: GET /api/autotests/test-runs/{id}

Ответ: `{runId, spaceId, name, status, summary: {total, passed, failed, broken, blocked, skipped, initial}, createdAt, updatedAt}`. Токен передаётся query-параметром `token`.

Набор автотестов: GET /api/autotests/test-runs/{id}/autotests

Возвращает, какие автотесты числятся в прогоне: `{runId, configurationId, source, ordered, autotests: [{autotestId, externalId, name, position?}]}`. Адаптер должен выполнить только их и в порядке `position`, если `ordered = true`.

- Параметр `?ciRunId=` (эхо переменной `DOQA_CI_RUN_ID`) сужает набор до автотестов, назначенных именно этому пайплайну. Когда на один прогон запущено несколько

пайплайнов, без него не обойтись: пустой ответ с `ciRunId` означает «этому пайплайну выполнять нечего», а не «выполняй всё».

- Без `ciRunId` возвращаются ещё не прогнанные автотесты прогона, а если таких нет — весь набор.
- Для свежего пустого прогона возвращаются все автотесты пространства (`source = 'space-fallback'`).

Загрузка результатов: POST /api/autotests/results

Сохраняет пачку результатов в существующий прогон. Тело: `{token, spaceId, testRunId, configurationId?, ciRunId?, pipelineId?, results: [...]}` . Ответ: `{accepted, elementIds}` .

Поля элемента `results[]` :

Поле	Обязательное	Смысл
`externalId`	да	внешний ID автотеста, до 255 символов
`outcome`	да	исход: `passed` , `failed` , `skipped` или `broken`
`name`	нет	полное имя автотеста, до 255 символов. Обязательно, пока автотеста с таким `externalId` ещё нет в пространстве: безымянный результат для неизвестного автотеста пропускается
`startedOn` , `completedOn`	нет	время старта и окончания: ISO-8601 или epoch в миллисекундах
`durationMs`	нет	длительность в миллисекундах
`message` , `traces`	нет	сообщение и стек-трейс. Для `failed` / `broken` это ошибка результата; `message` у `passed` — информационное сообщение
`parameters`	нет	параметры запуска — **только списком** `[{name, value}]` ; форма-словарь `{"имя": "значение"}` отклоняется с 422

Поле	Обязательное	Смысл
`stepResults`, `setupResults`, `teardownResults`	нет	шаги, setup- и teardown-фикстуры: `{title, outcome?, durationMs?, message?, attachments?, steps?}`, вложенность через `steps`
`attachments`	нет	вложения результата: `[{mediaFileId, kind?}]` — ID заранее загруженных файлов
`links`	нет	ссылки результата: `[{url, type?, title?, description?}]`; `type` — `related`, `defect`, `requirement`, `blocked_by` или `repository`

Повторная доставка безопасна: существующий результат обновляется, только если новый строго новее по времени старта, а поздние повторы отбрасываются без создания дублей. Результаты одного теста с разными наборами параметров или в разных конфигурациях хранятся отдельными строками: набор параметров входит в ключ сопоставления.

Корреляция с пайплайном: передавайте `ciRunId` (эхо `DOQA_CI_RUN_ID`) или `pipelineId`. Если у прогона ровно один пайплайн, DoQA сопоставит результат и без них. Но при нескольких пайплайнах результат без эха останется не привязанным к своему пайплайну: его не учтут **критерии приёмки** и источники автотестов.

Вложения: POST /api/autotests/attachments

Multipart-запрос `{token, spaceId, file}`. Ответ — `201 {"mediaFileId": <id>}`. Полученный ID указывается в `results[].attachments`. Лимит размера файла — по умолчанию 100 МБ.

Сослаться можно только на файл, загруженный в то же пространство: чужой или несуществующий `mediaFileId` пропускается без ошибки.

Определения автотестов: upsert и sync-ids

`POST upsert` — массовое создание и обновление определений в **каталоге автотестов**. Тело: `{token, spaceId, autotests: [{externalId, name, title?, description?, namespace?,`

`classname?, labels?, tags?, links?, steps?, caseIds?}]}`; ответ — `{map: {"<externalId>": {autotestId, created}}}`.

- Обновление частичное: перетираются только присланные ключи, пустая строка очищает поле.
- `caseIds` привязывает автотест к существующим тест-кейсам (аддитивно); тест-кейсы не создаются.
- `steps` — авторское дерево шагов определения (что тест должен делать), оно не заменяет шаги результата из `results[]`.
- Элемент без `externalId` или без `name` пропускается.

`POST sync-ids` — сопоставление локальных тестов со стабильными ID DoQA. Тело: `{token, spaceId, create?, items: [{externalId?, name, signature?, file?, line?}]}`. Ответ — `{map: {"<ключ>": {autotestId, assignedId, caseIds, created}}, orphans: [...]}`:

- `assignedId` — стабильный внешний ID, который стоит записать в код теста разметкой адаптера;
- `caseIds` — тест-кейсы, уже привязанные к автотесту;
- `orphans` — автотесты пространства, которых не оказалось среди присланных (например, удалённые из кода);
- `create = true` создаёт недостающие автотесты, иначе запрос только читает.

Правила externalId

`externalId` — стабильный идентификатор автотеста, главный ключ всего API.

- Уникален в пределах пространства. За уникальность отвечает адаптер: два разных теста с одним `externalId` сольются в один автотест.
- Стабилен: пока `externalId` не меняется, переименование теста не рвёт историю и привязку к кейсам.
- Не длиннее 255 символов.

Поведение без `externalId`, перенос истории и переход на стабильные ID — **Автотесты**.

Рекомендации

- **Отправляйте результаты порциями.** Не отправляйте `results[]` отдельным запросом на каждый тест: адаптер DoQA для JVM шлёт до 100 результатов в одном запросе.
- **Повторяйте только идемпотентные запросы.** GET-запросы и ответы 429 можно повторять безопасно. `POST results` и `POST test-runs` повторяйте только если соединение вообще не установилось: потерянный ответ на доставленный запрос при повторе создаст дубль прогона.
- **Ограничивайте объем на своей стороне.** Обрезайте длинные стек-трейсы и сообщения до отправки (адаптер DoQA для JVM режет трейс до 100 000 символов, сообщения — до 10 000).
- **Не логируйте токен**, в том числе в сообщениях об ошибках HTTP-клиента.

Смотрите также

- Отправка результатов автотестов — готовые отчёты: интерфейс, doqactl, report-API
- Адаптеры для тестовых фреймворков — готовые адаптеры вместо собственного
- Запуск автотестов из DoQA — переменные пайплайна и финализация прогона
- Автотесты — внешний ID и связь автотестов с тест-кейсами
- Каталог автотестов — где видны определения, метки и привязки

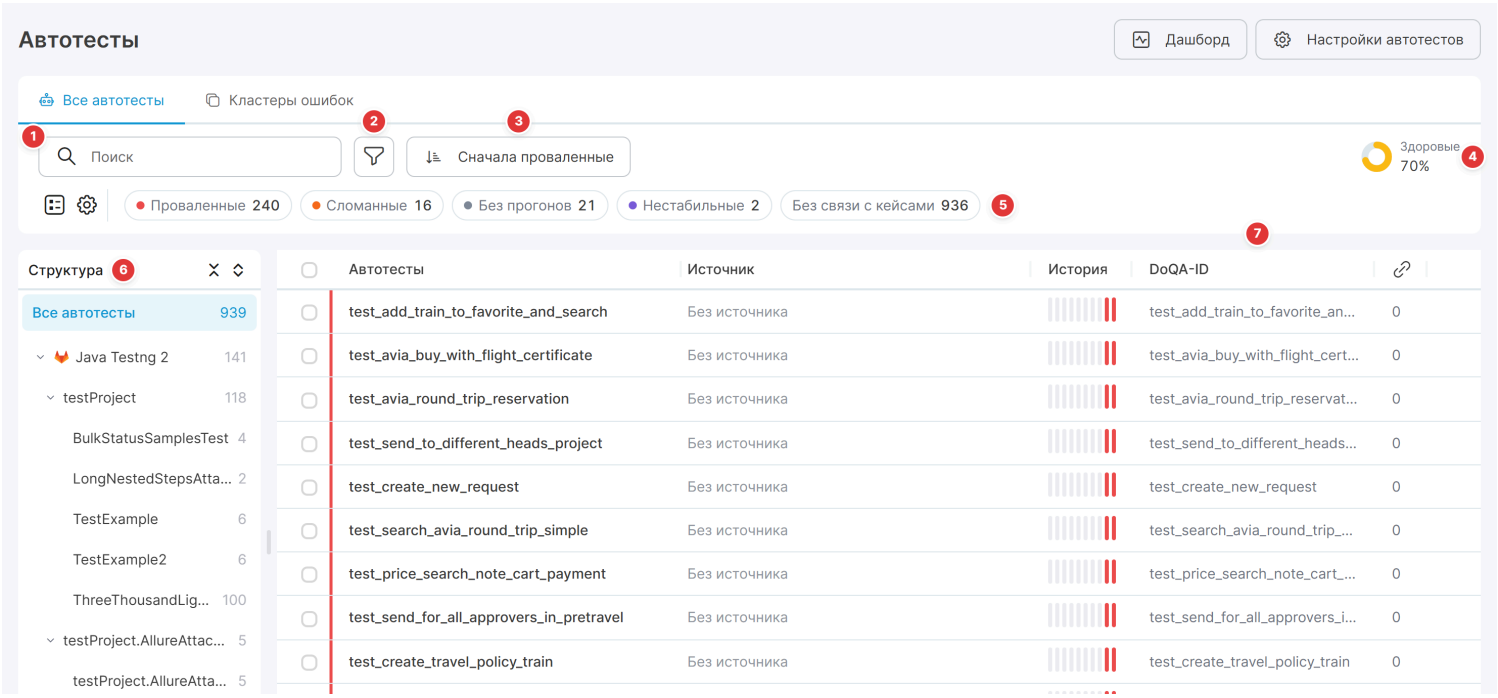
Каталог автотестов

Реестр всех автотестов пространства: что упало в последнем прогоне, насколько тест стабилен и к каким **тест-кейсам** привязан. Открывайте каталог для разбора падений, карантина, привязки кейсов и ручного запуска.

Где находится

Пространство → «Автотесты» → вкладка «Все автотесты». Раздел виден всем, у кого есть доступ к пространству.

Отмечать строки чек-боксами, ставить карантин, привязывать кейсы и требования, менять теги, назначать ответственного и запускать автотесты может только пользователь с правом на изменение прогонов. Без этого права каталог доступен на чтение: чек-боксы строк и кнопки действий не показываются.



На скриншоте: 1 — поле поиска, 2 — «Фильтры», 3 — сортировка, 4 — индикатор «Здоровые», 5 — ряд чипов, 6 — панель «Структура», 7 — таблица.

Раздел «Автотесты»

- Вкладка «Все автотесты» — каталог (эта страница), открывается по умолчанию.
- Вкладка «Кластеры ошибок» — группировка похожих падений, см. **Кластеры ошибок**.
- Кнопка «**Дашборд**» — видна всегда, ведёт на **дашборд** автотестов пространства.
- Кнопка «**Настройки автотестов**» — видна при доступе к настройкам пространства, открывает настройки сразу на разделе «**Подключение CI/CD**». Остальные разделы настроек автотестов — в левом навигаторе той же страницы.

Поиск

Поле поиска над таблицей ищет по названию автотеста, DoQA-ID, пакету и классу; максимум 100 символов. Список и панель «Структура» перезагружаются с начала. Крестик в поле очищает поиск.

Сводка над таблицей

Чипы-фильтры: клик по чипу фильтрует таблицу, повторный клик выключает фильтр.

Чип	Что отбирает
«Проваленные»	последний прогон завершился статусом «Провален»
«Сломанные»	последний прогон завершился статусом «Сломан» (заблокированные сюда не входят)
«Без прогонов»	прогонов ещё не было или последний прогон пропущен
«Нестабильные»	состояние стабильности — «Нестабильный» (автотесты в карантине сюда не входят)
«Без связи с кейсами»	нет ни одного привязанного тест-кейса

Справа — кольцевой индикатор «Здоровые»: процент автотестов, у которых последний прогон пройден. Подсказка при наведении — «Стабильность за 10 прогонов».

Панель «Структура»

Дерево слева от таблицы, три уровня: источник → пакет (namespace) → класс. Источник — это CI-подключение, из которого пришли результаты; автотесты, полученные без CI-подключения (Direct API, ручная загрузка), собраны в узле «Без источника». Рядом с каждым узлом — число автотестов в скобках.

Клик по узлу сужает таблицу до этого источника, пакета или класса; первая строка дерева «Все автотесты» возвращает полный список. Значок «Скрыть структуру» слева от чипов убирает панель, «Показать структуру» — возвращает. Ширина панели тянется мышью от 240 до 480 px; ширина и скрытое состояние запоминаются в браузере.

Если автотестов слишком много, дерево показывает «Дерево обрезано — уточните поиск»: сузьте выборку поиском или фильтрами.

Фильтры

Кнопка «Фильтры» с бейджем-счётчиком активных фильтров открывает диалог «Фильтр». Значения вступают в силу по кнопке «**Применить**», кнопка «**Сбросить фильтр**» очищает все поля диалога.

Поле	По умолчанию	Что делает
«Статус последнего прохождения»	не задано	множественный выбор: «Пройден», «Провален», «Сломан», «Пропущен», «Заблокирован», «Без прогонов»
«Запущен»	не задано	диапазон дат по дате последнего прогона
«Ветка»	«Все»	ветка CI; показатели стабильности в таблице считаются по выбранной ветке
«Источник»	«Все»	конкретное CI-подключение или «Без источника»; дублирует выбор узла в панели «Структура»
«Связь с тест-кейсами»	«Все»	«Все» / «Есть связь» / «Нет связи»

Поле	По умолчанию	Что делает
«Карантин»	«Все»	«Все» / «В карантине» / «Не в карантине»
«Стабильность»	«Все»	«Все» / «Стабильные» / «Нестабильные», одиночный выбор
«Теги»	пусто	теги пространства и группа «Теги фреймворка» (метки из кода теста); между строками условие «И», внутри строки — «ИЛИ»

- В фильтре тегов новый тег создать нельзя — только выбрать существующий.
- Значение «Все» ничего не фильтрует и не попадает ни в бейдж-счётчик, ни в адрес страницы.
- Ссылка «Сбросить фильтры» рядом с чипами сбрасывает фильтры и выбор в «Структуре», но не поиск. Поиск сбрасывает кнопка «**Сбросить фильтры**» в пустом состоянии «Ничего не найдено».

Сортировка

Кнопка сортировки справа от «Фильтров», подпись на кнопке — текущий вариант. По умолчанию — «Сначала проваленные»; остальные: «Сначала пройденные», «Стабильность: выше сверху», «Стабильность: ниже сверху», «Последний запуск: новые сверху», «Последний запуск: старые сверху».

Столбцы таблицы

Значок-шестерёнка «Настройка колонок» слева от чипов открывает панель: столбцы можно включать, выключать и переставлять перетаскиванием, ширину — тянуть за границу заголовка. Настройка сохраняется на сервере и переживает перезагрузку и смену устройства. Пресеты панели — «По умолчанию» и «Пользовательский».

Столбец	Что показывает
(чек-бокс)	выбор строки; не входит в настройку колонок
«Статус»	цветная риска исхода последнего прогона
«Автотесты»	название автотеста, рядом значки «В карантине» и «Нестабильный»
«Источник»	логотип и имя CI-подключения, `+N` при нескольких источниках
«История»	полоска из 10 ячеек — исходы последних 10 прогонов, слева направо от старых к новым
«DoQA-ID»	внешний идентификатор автотеста, «—» при отсутствии
«Связь с тест-кейсами»	число привязанных кейсов
«Последний запуск»	дата и время; «Никогда», если прогонов не было
(действия)	меню «три точки»; не входит в настройку колонок, всегда последний столбец

Действия в строке

Меню «три точки» в конце строки:

Пункт	Условие	Что делает
«Открыть историю»	всегда	открывает панель автотеста справа
«В карантин» / «Снять карантин»	право на изменение	переключает карантин у этого автотеста
«Связать с тест-кейсом»	право на изменение	открывает диалог выбора тест-кейсов, см. **Привязка тест-кейса**
«Связать требование»	право на изменение	открывает диалог выбора **требований**, можно выбрать несколько

Массовые действия

Отметьте автотесты чек-боксами. Внизу появится плавающая панель со счётчиком «Выбрано: N»:

Кнопка	Когда показана	Что делает
«Очистить выбор»	всегда	снимает отметки
«Связать с тест-кейсом»	всегда, но работает только с одним выбранным	открывает диалог выбора тест-кейсов
«В карантин»	среди выбранных есть хотя бы один не в карантине	ставит карантин всем выбранным
«Снять карантин»	среди выбранных есть хотя бы один в карантине	снимает карантин со всех выбранных
«Запустить выбранные»	всегда	открывает диалог запуска, см. **Запуск выбранных автотестов**

Чек-бокс «выбрать все» в шапке таблицы отмечает только уже загруженные строки, а не весь каталог.

Карантин применяется к ветке, выбранной в фильтре «Ветка»; если ветка не выбрана — сразу ко всем веткам. Снятие карантина без выбранной ветки снимает его и с каждой ветки отдельно.

Состояния автотеста

Состояние	Когда возникает
«Стабильный»	базовое состояние, сигналов нестабильности нет
«Нестабильный»	сработал хотя бы один сигнал нестабильности
«Сломан»	детерминированное падение
«В карантине»	карантин поставлен вручную или автоматически

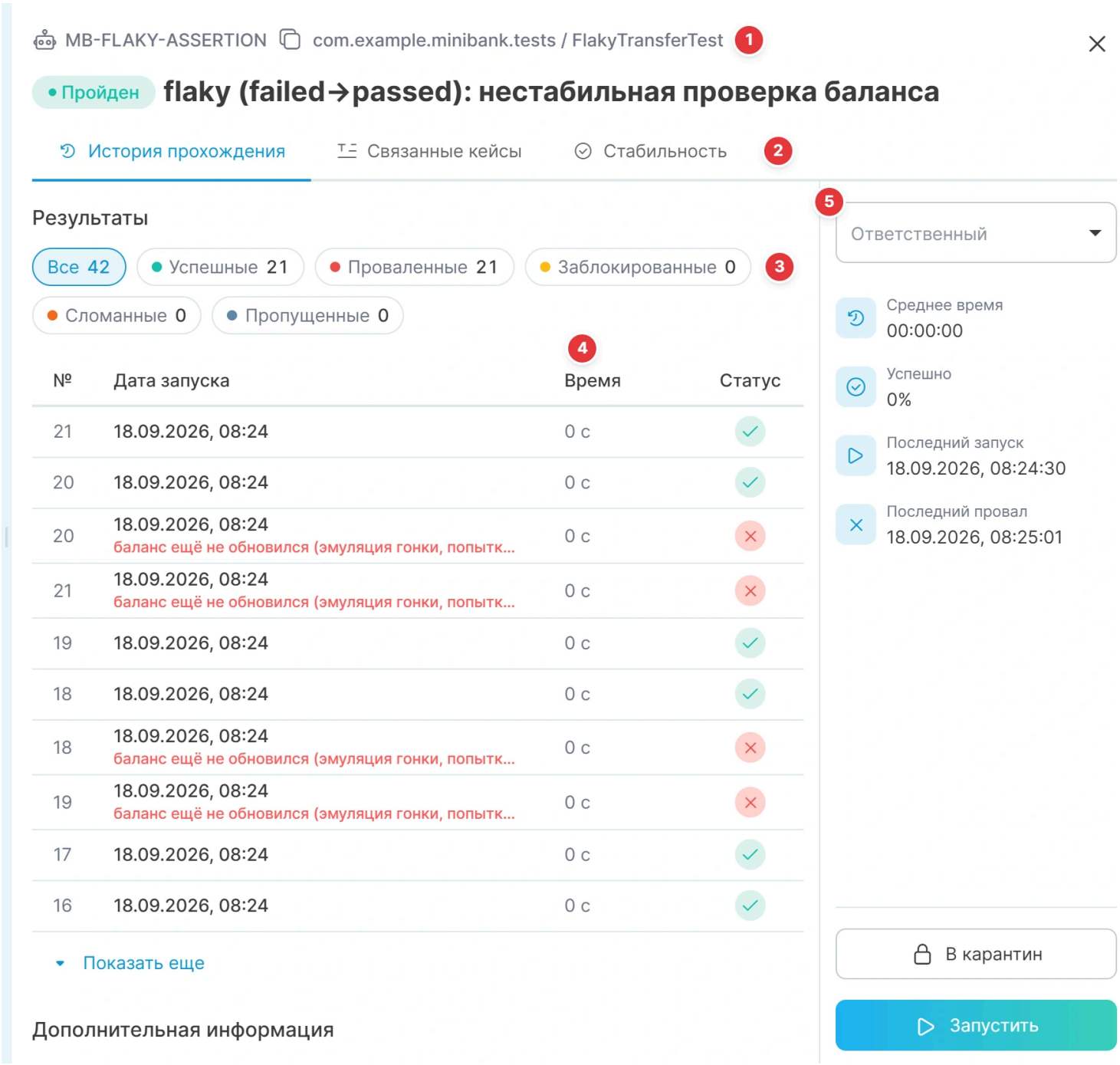
Состояние	Когда возникает
«Исправлен»	накоплено достаточно дней подряд без падений
«Отключён»	выставляется только через API, кнопок для этого в интерфейсе нет

Как считаются состояния, карантин и сигналы нестабильности — в **Нестабильные тесты и карантин**.

Панель автотеста

Клик по строке (или пункт «Открыть историю») открывает панель справа. Адрес страницы меняется: ссылку на открытую панель можно передать коллеге. Ширина панели тянется мышью и запоминается.

В шапке — DoQA-ID с кнопкой копирования, путь к автотесту (пакет и класс), значок статуса последнего прогона и название автотеста, плашки «В карантине» и «Нестабильный», если применимо. Панель разбита на три вкладки; при переключении на другой автотест она всегда открывается на первой.



На скриншоте: 1 — шапка с DoQA-ID и плашками, 2 — вкладки, 3 — чипы статусов, 4 — таблица «Результаты», 5 — правая колонка с ответственным и плитками.

Вкладка «История прохождения»

- Чипы-фильтры по статусу с числом на каждом: «Все», «Успешные», «Проваленные», «Заблокированные», «Сломанные», «Пропущенные».
- Таблица «Результаты»: «№», «Дата запуска», «Время», «Статус»; под датой у упавших результатов — текст ошибки. Клик по строке открывает слева панель «Результат прогона», повторный клик закрывает. Кнопка «**Показать ещё**» догружает следующие 20 строк.

- Блок «Дополнительная информация»: «Framework-теги» — метки из кода теста, только для чтения. «Теги DoQA» — множественный выбор из **тегов** пространства, сохраняется сразу по выбору. Доступно только при праве на изменение.
- Справа: селектор ответственного за автотест (назначается на весь автотест, всем веткам сразу), плитки «Среднее время», «Успешно», «Последний запуск», «Последний провал», и при праве на изменение — кнопки «**В карантин**» / «**Снять карантин**» и «**Запустить**» (тот же диалог, что и массовый запуск, на один автотест).

Вкладка «Связанные кейсы»

Таблица привязанных тест-кейсов: «ID» (ссылка на кейс), «Название», «Статус»; при наведении на строку — значок «Отвязать». Кнопка «**Привязать кейс**» открывает тот же диалог, что и «Связать с тест-кейсом» в каталоге. Плитка «Пройдены» — доля связанных кейсов в статусе «Пройден»; это статус тест-кейса, а не результат автотеста.

Вкладка «Стабильность»

Семь плиток: «Количество прогонов», «Состояние стабильности», «Балл нестабильности», «Смены состояний», «Последний успех», «Последний провал», «Успешно». Ниже — список «Смены состояния стабильности» (с фильтром по ветке, по умолчанию «Все ветки»), график и таблица «Балл нестабильности», таблица «Связанные кластеры ошибок». Подробнее о вычислении — **Нестабильные тесты и карантин**. О кластерах — **Кластеры ошибок**.

Привязка тест-кейса

Один тест-кейс может быть связан максимум с одним автотестом, а у одного автотеста связанных кейсов может быть несколько.

1. Отметьте в каталоге **один** автотест.
2. Нажмите «**Связать с тест-кейсом**» в нижней панели: откроется диалог выбора тест-кейсов. Тот же диалог открывают пункт «Связать с тест-кейсом» в меню строки и кнопка «**Привязать кейс**» на вкладке «Связанные кейсы».
3. Выберите один или несколько кейсов и нажмите «**Привязать**».

Внимание

Если выбранный кейс уже привязан к другому автотесту, привязка перенесёт его на текущий: прежний автотест останется без связи. DoQA сообщает об этом всплывающим уведомлением «Тест-кейсы перепривязаны с других автотестов: N», но подтверждения перед переносом не запрашивает.

Разорвать связь можно только из панели автотеста, вкладка «Связанные кейсы» — значком «Отвязать» в строке кейса, с подтверждением. Массовой отвязки в каталоге нет.

При двух и более выбранных автотестах кнопка «Связать с тест-кейсом» остаётся активной, но по нажатию показывает предупреждение «Тест-кейс может быть привязан только к одному автотесту — выберите один автотест».

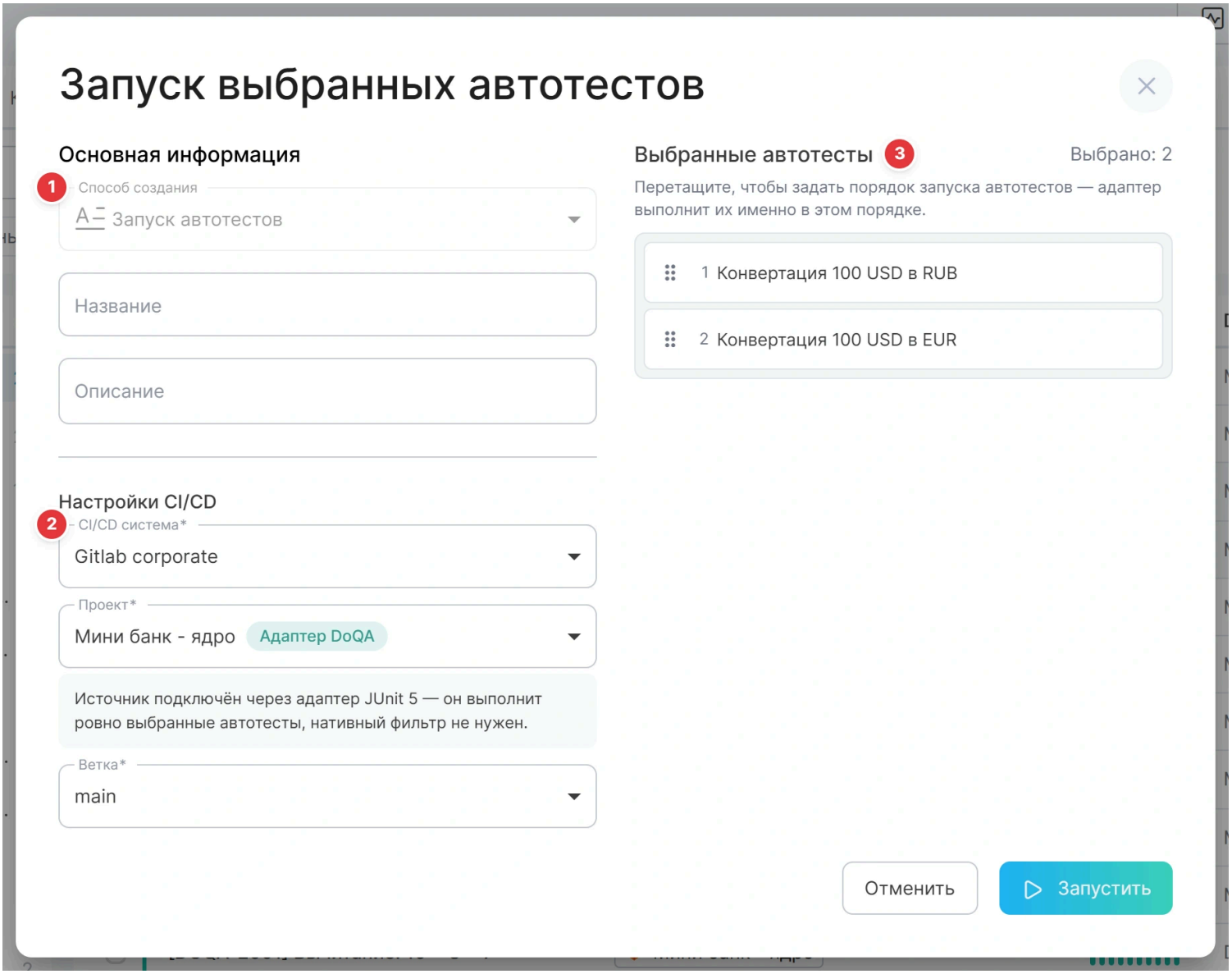
Запуск выбранных автотестов

1. Отметьте автотесты чек-боксами.
2. Нажмите «**Запустить выбранные**»: откроется диалог «Запуск выбранных автотестов».
3. Слева заполните форму, справа при необходимости задайте порядок прохождения.
4. Нажмите «**Запустить**». DoQA создаёт **прогон** и открывает его страницу.

Поля левой колонки диалога:

Поле	По умолчанию	Что делает
«Способ создания»	«Запуск автотестов»	заблокировано — этот диалог всегда запускает автотесты
«Название»	пусто	название будущего прогона
«Описание»	пусто	описание прогона
«Теги»	пусто	теги прогона; секция видна, только если в пространстве есть теги
«Конфигурация (необязательно)»	«Без конфигурации»	конфигурация прогона; секция видна, только если в пространстве есть конфигурации
Блок настроек CI/CD	последние использованные значения	подключение, проект, пайплайн, ветка и другие параметры запуска

В правой колонке — «Выбранные автотесты»: список с номерами по порядку, ручкой для перетаскивания и крестиком «Убрать из выбранного».



На скриншоте: 1 — форма слева, 2 — блок настроек CI/CD, 3 — список «Выбранные автотесты» справа.

Внимание

Порядок, заданный перетаскиванием, соблюдается только когда автотесты подключены через адаптер — тогда подсказка над списком заканчивается словами «адаптер выполнит их именно в этом порядке». Для остальных способов подключения порядок не гарантирован, хотя перетащить строки можно так же.

Если выбранные автотесты в прошлый раз запускались в разных местах, наверху диалога появляется предупреждение и выбор из двух режимов: «Одна конфигурация для всех» и

«Каждый там, где запускался последний раз». Второй режим бывает недоступен, тогда под ним написано почему: например, на один запуск допустимо не больше 10 пайплайнов.

Ошибку запуска DoQA показывает текстом над кнопками диалога. Если часть выбранных автотестов успели удалить — «N из выбранных автотестов больше не существуют. Обновите каталог и повторите попытку.»

Поля блока CI/CD и переменные пайплайна — в **Запуске автотестов из DoQA**.

Особенности

- Числа на чипах и процент в индикаторе «Здоровые» считаются по поиску, ветке, источнику и тегам; остальные поля фильтра на них не влияют. Если задан ещё и статус, стабильность, карантин, связь с кейсами или дата, число на чипе может не совпасть с числом строк.
- Чип подсвечивается активным, только когда фильтр в точности повторяет его: заданное чипом поле равно его значению, а «Статус последнего прохождения», «Стабильность» и «Связь с тест-кейсами» пусты. Выбрали в фильтре «Провален» и «Сломан» — не горит ни один чип.
- Панель «Структура» считается по тем же фильтрам, что и таблица, но не фильтрует себя по выбранному узлу, иначе узел исчезал бы из дерева сразу после выбора.
- Выбранная в фильтре ветка меняет и показатели стабильности в таблице: состояние, значки и полоска «История» считаются по этой ветке.
- В плитке «Состояние стабильности» панели автотеста показывается упрощённое значение — «Стабилен» или «Нестабилен». Полное состояние видно по значкам в строке каталога и в списке «Смены состояния стабильности».
- Карантин исключает автотест из **Quality Gate**, но не из статистики стабильности: окно и доля падений продолжают считаться как обычно.
- Каталог обновляется сам, когда меняется стабильность автотестов. Открытая панель автотеста при этом не обновляется. Закройте и откройте её заново.

Смотрите также

- **Запуск автотестов из DoQA** — диалог запуска, настройки CI/CD, переменные пайплайна
- **Нестабильные тесты и карантин** — как вычисляется стабильность, сигналы, ручной и автоматический карантин
- **Кластеры ошибок** — группировка похожих падений, категории ошибок

- **Отправка результатов** — как автотесты попадают в каталог
- **Матрица покрытия** — покрытие требований тестами

Подключение CI-систем

Чтобы **запускать автотесты из DoQA** и видеть статусы пайплайнов, пространству нужно подключение к CI-системе и привязка CI-проектов. Обе настройки находятся на странице настроек **пространства**: откройте раздел «Автотесты» и нажмите кнопку настроек — откроется секция «CI/CD connections». Понадобится доступ к настройкам пространства.

Два уровня подключений

Подключения существуют на двух уровнях:

- **Общее подключение тенанта** — создаёт администратор, см. **Интеграции с CI/CD**. Оно видно всем пространствам. В настройках пространства такое подключение помечено чипом «Shared (inherited)»: его нельзя изменить или удалить из пространства, но можно проверять соединение и привязывать к нему CI-проекты.
- **Подключения пространства** — создаются в настройках самого пространства и видны только в нём.

Подключений может быть несколько, в том числе одного типа, например несколько инстансов GitLab в одном пространстве.

Поддерживаемые CI-системы

DoQA работает с девятью CI-системами: GitLab, Jenkins, TeamCity, GitHub Actions, GitVerse (совместим с Gitea Actions), Bitbucket (облачные Pipelines), Azure DevOps, CircleCI, Bamboo.

Базовые операции — запуск пайплайна, чтение статуса, отмена пайплайна и ссылка на него — поддерживают все девять. Расширенные возможности различаются:

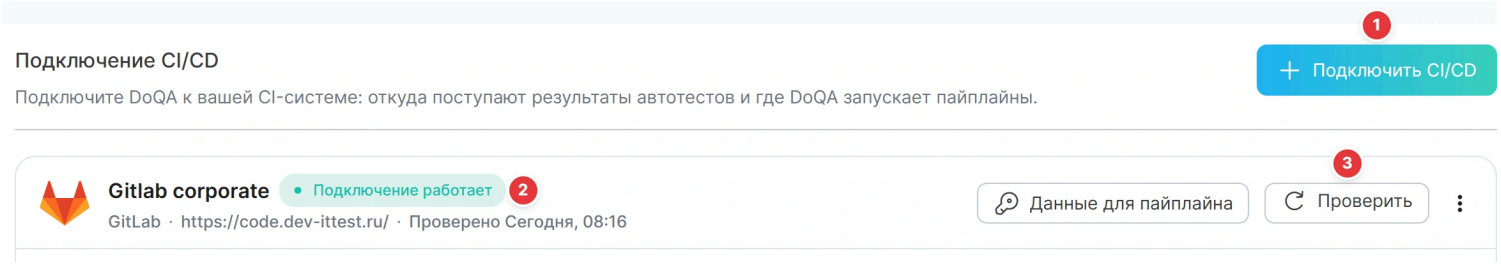
CI-система	Вебхук о завершении	Управление джобами	Скачивание результатов из артефактов	Авто-настройка CI-переменных	Аутентификация в поле «Токен»
GitLab	да	да	да	да	персональный access token
Jenkins	да	только просмотр стадий	—	—	`логин:api-токен`
TeamCity	да	только отмена	—	—	Bearer-токен
GitHub Actions	да	только перезапуск	—	—	токен (Bearer)
GitVerse	да	—	—	—	токен Gitea
Bitbucket	да	—	—	—	`логин:app_password` или Bearer-токен
Azure DevOps	да	—	—	—	Personal Access Token
CircleCI	да	—	—	—	API-токен
Bamboo	да	—	—	—	`логин:пароль` или Bearer-токен

Там, где нужна пара «логин + секрет», она вводится в поле «Токен» одной строкой через двоеточие (`логин:токен`). Значение без двоеточия Bitbucket и Bamboo трактуют как Bearer-токен.

Особенности провайдеров

Jenkins, GitHub Actions и GitVerse не сообщают идентификатор пайплайна в момент запуска: ссылка на пайплайн и его отмена могут быть недоступны, пока CI-система не пришлёт вебхук с реальным идентификатором. Bamboo умеет отменять сборку, только пока она стоит в очереди.

Создание подключения



На скриншоте: 1 — кнопка «**Подключить CI/CD**», 2 — бейдж проверки соединения (зелёный «Подключение работает»), 3 — кнопка «**Проверить**» повторной проверки соединения.

1. Откройте настройки пространства, секцию «CI/CD connections».
2. Добавьте подключение и заполните форму.
3. Перед сохранением проверьте соединение кнопкой «**Test connection**».
4. Сохраните подключение.

Поля формы:

Поле	Что указать
Тип	одна из девяти CI-систем. После сохранения тип изменить нельзя
Название	метка подключения, обязательна
Хост	адрес CI-системы, например `https://gitlab.example.com`
Токен	учётные данные для API CI-системы (что именно — см. столбец «Аутентификация» выше). При редактировании пустое поле означает «не менять токен»
Секрет вебхука	секрет для проверки подлинности входящих вебхуков, см. **Вебхук подключения**
Идентификаторы проекта	JSON-объект с идентификаторами проекта у провайдера, например `{"job": "autotests"}`

Какие ключи ожидаются в поле «Идентификаторы проекта»:

▶ **Ключи идентификаторов по провайдерам**

Рядом со списком подключений доступен диалог «Как подключить автотесты» — короткие сниппеты команд отправки результатов и аннотаций для быстрого старта.

Проверка соединения

- «**Test connection**» в форме — проверка до сохранения: DoQA обращается к CI-системе с введёнными хостом и токеном и показывает результат.
- «**Check**» на карточке сохранённого подключения — повторная проверка. Её результат и время сохраняются и показываются бейджем: зелёный означает успех, красный — ошибку, а серый «Not checked» означает, что подключение ещё не проверялось.

Глубина проверки зависит от провайдера: для GitLab проверяются токен, версия сервера и доступные проекты; для большинства остальных — только доступность хоста, учётные данные при этом не проверяются (об этом говорит текст результата).

Вебхук подключения

Вебхук — основной способ, которым DoQA узнаёт о завершении пайплайна; без него статус обновится только поллингом, с задержкой (см. **жизненный цикл пайплайна**). Карточка подключения показывает готовый адрес вебхука с кнопкой копирования. Настройте в CI-системе отправку событий пайплайна на этот адрес.

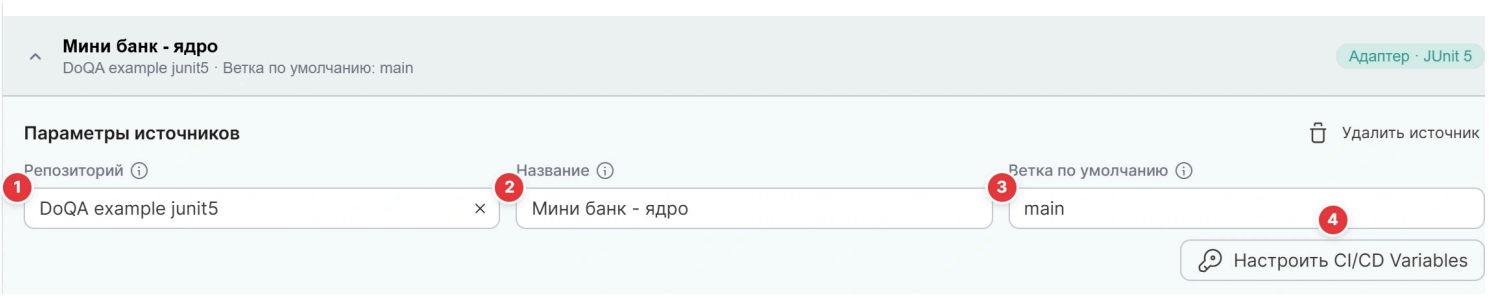
Подлинность вебхука проверяется секретом из поля «Секрет вебхука»: в GitLab тот же секрет указывается в поле «Secret token» настроек вебхука, у остальных провайдеров тело запроса должно быть подписано HMAC-SHA256 этим секретом. Если секрет в подключении не задан, входящие вебхуки отвергаются.

Привязка CI-проектов

Второй шаг той же секции — привязка репозиториев (CI-проектов) к пространству. Если подключений несколько, сначала выберите подключение. Каждая строка привязки:

- «Repository» — репозиторий; поиск подсказывает варианты начиная с 2 введённых символов;
- «Name» — человекочитаемая метка привязки, необязательна;
- «Subproject identifier» — необязательный идентификатор подпроекта.

Пара «репозиторий + идентификатор подпроекта» должна быть уникальна. Две строки с одним репозиторием и разными идентификаторами — это два раздельных источника автотестов: в **каталоге автотестов** они видны как разные источники, а идентификатор передаётся в пайплайн переменной `DOQA_SOURCE_KEY` (см. **переменные пайплайна**). Сохранение — кнопкой или сочетанием Ctrl/Cmd+S.



На скриншоте: 1 — поле «Репозиторий», 2 — поле «Название», 3 — «Ветка по умолчанию», 4 — кнопка «**Настроить CI/CD Variables**».

Основная ветка

На карточке подключения, в блоке «Default branches of linked projects», по каждому привязанному проекту можно задать основную ветку. Значение сохраняется по Enter или при уходе из поля; пустое поле сбрасывает настройку.

Основная ветка используется как ветка по умолчанию в формах запуска, когда у выбранных автотестов ещё нет истории запусков — подробнее в **пресетах запуска**.

Вид нативного фильтра

Колонка «Формат отчёта» в таблице источников задаёт вид нативного фильтра тестов для каждого источника (репозитория/джобы) отдельно: «Определять автоматически» либо один

из восьми видов — `surefire`, `gradle`, `pytest`, `gotest`, `playwright`, `phpunit`, `dotnet`, `jest`. Значение сохраняется вместе с остальными полями источника кнопкой «Сохранить».

- **Выбранный вручную вид** имеет высший приоритет в цепочке определения фильтра: он действует на все запуски из этого источника. Это единственная настройка, с которой выбор в диалоге запуска конфликтует: запуск с другим видом отклоняется с ошибкой, а превью фильтра в диалоге предупреждает об этом и показывает выражение в виде источника.
- **«Определять автоматически»** означает, что DoQA запомнит вид сам, по первым результатам прогонов источника. Запомненное значение показывается в селекте с пометкой «определён автоматически»; его можно переопределить вручную или сбросить обратно в автоопределение. Пока результатов ещё не было, вид определяется по самим выбранным автотестам при каждом запуске.
- Если в результатах одного источника встречаются тесты разных фреймворков, автоматически вид не запоминается (селект остаётся в «Определять автоматически» с подсказкой об этом). Для смешанного репозитория задайте вид вручную либо разделите его на несколько источников по подпроектам.
- Maven и Gradle по результатам не различаются: для JVM-проектов автоматически запоминается `surefire`; проектам на Gradle вид `gradle` нужно выбрать вручную.

Что такое нативный фильтр и как он влияет на запуск — **Нативный фильтр тестов**.

Авто-настройка переменных в CI-проекте

У каждой сохранённой привязки есть кнопка «**Настроить CI/CD Variables**». DoQA создаёт или обновляет в CI-проекте ровно три переменные:

- `DOQA_URL` — адрес вашего экземпляра DoQA;
- `DOQA_TOKEN` — project-токен (создаётся автоматически, переменная маскируется);
- `DOQA_SPACE_ID` — ID пространства.

Если в проекте остались переменные с прежними именами (`DOQA_ENDPOINT` — адрес, `DOQA_SPACE` — пространство), DoQA обновит и их: пайплайн, который их читает, продолжит работать с актуальным адресом. Новых переменных с прежними именами DoQA не создаёт, поэтому в свежем проекте их не появится. Об обновлении устаревшей переменной DoQA предупреждает в сообщении: пайплайн стоит перевести на `DOQA_URL` и `DOQA_SPACE_ID`.

После этого раннеру не нужно настраивать эти переменные вручную, чтобы обращаться к API DoQA — запрашивать тест-план и отправлять результаты.

Что важно знать:

- запись переменных поддерживается только для GitLab; для остальных провайдеров DoQA сообщит, что операция не поддерживается, и переменные придётся завести вручную;
- переменные пишутся токеном самого подключения: в проекте GitLab ему нужна роль Maintainer или выше; при нехватке прав DoQA покажет отдельную ошибку;
- перед записью в CI-систему запрашивается подтверждение.

Не переобъявляйте эти переменные в `.gitlab-ci.yml`

Переменные проекта и так доступны каждой джобе. Строка вида `DOQA_TOKEN: $DOQA_TOKEN` в секции `variables:` джобы — самоссылка: если переменной проекта нет или у неё снят флаг «Расширить ссылку на переменную», подставлять нечего, и в окружение попадёт литерал `$DOQA_TOKEN`. Адаптер примет его за токен и получит от DoQA 401.

Если такие строки уже есть, просто удалите их. Проверить фактическое значение в джобе можно так (маскирование значение не скроет, потому что вывод изменён):

```yaml
- printf %s "$DOQA_TOKEN" | od -c | head -3
```

Переменная `DOQA_TOKEN` создаётся маскированной и с включённой ссылочной подстановкой (`raw = false`). Флаг «Защитить переменную» не ставится: с ним переменная передавалась бы только в защищённые ветки, а запуск из DoQA обычно идёт по обычной ветке.

Project-токен

Project-токен — API-токен проекта DoQA. Им аутентифицируются запросы, которые пайплайн делает к DoQA без логина и пароля: тест-план, проверка критериев приёмки, отправка результатов, а также **Autotest API** адаптеров. Токен действует на все пространства проекта.

Как создать токен — **Создание API-токена**. Авто-настройка переменных создаёт и подставляет project-токен в CI-проект сама.

Смотрите также

- **Запуск автотестов из DoQA** — способы запуска, переменные пайплайна, жизненный цикл
- **Автоматика запусков** — расписания, авто-повтор, критерии приёмки прогона
- **Интеграции с CI/CD** — общие подключения на уровне тенанта
- **Отправка результатов автотестов** — доставка готовых отчётов без запуска из DoQA

Запуск автотестов из DoQA

DoQA запускает пайплайн в подключённой CI-системе и собирает его результаты в **прогон**. Для этого пространству нужны подключение CI и привязанные CI-проекты — см. **Подключение CI-систем**.

Способы запуска

Сценарий	Где	Что происходит
Прогон всего набора	вкладка «Прогоны» → способ создания «Запуск автотестов»	новый прогон + пайплайн на весь набор
Запуск выбранных автотестов	«**Запустить**» в **каталоге автотестов**	новый прогон из выбранных автотестов
Запуск и перезапуск в существующем прогоне	«**Запустить автотесты**» и «**Перезапустить автотесты**» в плеере прогона	новый пайплайн по элементам прогона
Перепрохождение одного автотеста	«**Перепройти**» в карточке результата	новая попытка одного теста
По расписанию	настройки пространства, секция «Расписания прогонов»	прогон создаётся по таймеру — см. **Автоматика запусков**
Со стороны CI	пайплайн стартует в самой CI-системе	DoQA создаёт под него прогон «Прогон из CI/CD»

Создание прогона способом «Запуск автотестов»

В окне создания прогона выберите способ «Запуск автотестов», CI-подключение, проект и ветку. Поля диалога описаны в **Создание прогона через запуск автотестов**.

DoQA создаёт пустой прогон в статусе «В работе» и запускает пайплайн в режиме «весь набор»: пайплайн прогоняет все тесты проекта, а результаты попадают в этот прогон (в

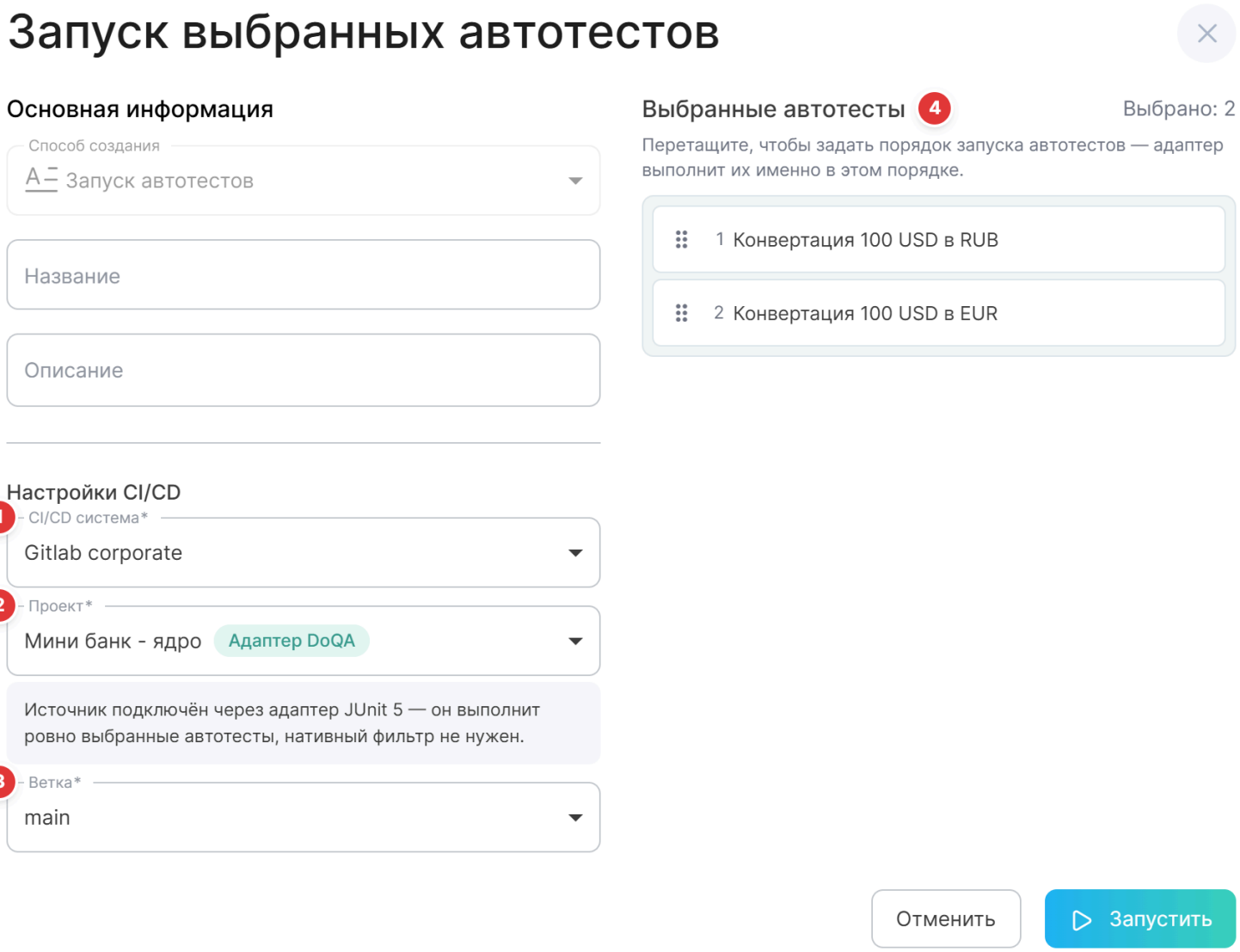
переменных пайплайна передаются ID прогона и режим — см. **Переменные пайплайна**). Тесты появляются в прогоне по мере поступления результатов. Когда завершается последний пайплайн, DoQA сама переводит прогон в статус «Завершен».

Запуск из каталога

Отметьте автотесты в **каталоге** и нажмите «**Запустить**» или нажмите «**Запустить**» в карточке одного автотеста. Откроется диалог «Запустить выбранные»:

- **CI-подключение, проект, ветка** — каскадный выбор; форма предзаполняется **пресетами запуска**. Если CI не настроен, диалог предложит перейти к настройкам пространства.
- **Порядок выполнения** — список выбранных автотестов можно переупорядочить перетаскиванием; порядок сохраняется в прогоне. Рядом — бейджи управляемости порядка: у `pytest` порядок управляем, у `surefire` / `gradle` — частично, у остальных фреймворков — не гарантирован; при параллельном выполнении в раннере порядок не гарантируется в любом случае.
- **Название прогона** и **конфигурация** — необязательны.
- **Нативный фильтр** — выбор формата отчёта («Автоопределение» или конкретный формат) и живое превью выражения с предупреждениями: неоднозначный формат, непокрытые фильтром тесты, усечение. Превью точно показывает, будет ли фильтр передан: выбранный формат — подсказка, и набор, который в нём невыразим, уйдёт полным прогоном. Тумблер управляет тем, передавать ли фильтр в пайплайн вообще. Подробнее — **Нативный фильтр тестов**.

После запуска DoQA создаёт прогон с выбранными автотестами и открывает его. Если между выбором и запуском набор изменился (например, автотест удалён), диалог сообщит, что набор устарел. Обновите каталог и повторите.



На скриншоте: 1 — CI/CD-подключение, 2 — проект, 3 — ветка, 4 — порядок запуска (перетаскивание).

Запуск и перезапуск из прогона

В плеере прогона с автотестами, доступными для запуска, есть кнопки «**Запустить автотесты**» и «**Retry failed tests now**»; обе активны только для прогона в статусе «В работе». «**Retry failed tests now**» перезапускает упавшие тесты по политике авто-повтора — см. Автоматика запусков.

«**Запустить автотесты**» открывает диалог с той же формой «подключение → проект → ветка» и пресетами; если у выбранных автотестов разные последние конфигурации, добавляется

выбор режима групп. Дополнительно в диалоге два тумблера: переопределить для этого прогона критерии приёмки и политику авто-повтора.

Уже завершённые элементы повторно этим способом не запускаются. Для них есть кнопка «**Перезапустить автотесты**» на панели таблицы: выбранные завершённые элементы конвертируются в новые попытки и уходят в новый пайплайн, а прежний результат сохраняется в истории попыток.

Если перед нажатием «**Запустить автотесты**» не выбран ни один элемент, запускаются все ещё не выполненные автотесты прогона: к пайплайну привязываются все элементы в статусе «Не прогнан». Нативный фильтр на такой запуск не строится: пайплайн выполняет весь набор проекта, а в прогон попадают результаты запланированных в нём автотестов.

Статус прогона при запуске из плеера не меняется. Такой прогон завершите сами. Автоматически DoQA завершает прогоны, созданные самим запуском: способом «Запуск автотестов», кнопкой «**Запустить**» из каталога или со стороны CI.

Перепрохождение одного автотеста

В карточке результата автотеста есть кнопка «**Перепройти**» с двумя вариантами:

- «Автоматически» — DoQA запускает новый пайплайн ровно под этот тест: используются та же интеграция, проект и ветка, что при прошлом запуске, а в пайплайн передаётся нативный фильтр на один тест. Прежний результат не стирается: он остаётся в истории как предыдущая попытка.
- «Вручную» — результат конвертируется в ручное прохождение. Если у автотеста были шаги, они материализуются в карточке с бейджем «Шаги из авто-прохождения»; дальше тест проходится как обычный тест-кейс.

Кнопка недоступна в завершённом прогоне. Вариант «Автоматически» требует, чтобы у автотеста был связанный пайплайн: если тест ни разу не запускался через CI из DoQA, перезапускать его нечем.

Все прохождения одного теста в прогоне собираются в цепочку попыток. Решающей считается последняя: именно она учитывается в статистике и критериях приёмки. Состав попытки и кнопка «**Подробнее**» — **Результаты и разбор падений**.

Прогон, созданный со стороны CI

Пайплайн может стартовать в самой CI-системе, по push или по расписанию CI. Чтобы результаты попали в DoQA, пайплайн вызывает `POST /api/integrations/ci/create-run`, передавая **project-токен**, ID пайплайна, ID пространства, тип CI-системы и ветку. DoQA создаёт прогон «Прогон из CI/CD», привязывает к нему пайплайн и отвечает `runId`; дальше пайплайн запрашивает **тест-план** как обычно. Если у привязки задан идентификатор подпроекта, передайте его в запросе: по нему DoQA определяет, к какой привязке относится пайплайн.

Пресеты запуска

Диалоги запуска предзаполняются последней использованной конфигурацией: по каждому выбранному автотесту DoQA смотрит, из какого подключения, проекта и с какой веткой он запускался в прошлый раз.

- Конфигурация у всех выбранных совпадает — форма молча предзаполняется ею.
- Конфигурации разные — диалог предупреждает и предлагает выбор: одна конфигурация на всех (укажите её вручную) либо «каждый там, где запускался». Второй вариант — мульти-запуск по группам, доступный при не более чем 10 разных конфигурациях.
- Истории запусков нет — конфигурация подставляется из **основной ветки**, и только если живая привязка в пространстве ровно одна и ветка у неё задана; иначе форма остаётся пустой.

Ветка определяется по цепочке: ветка последнего запуска → основная ветка привязки → пусто.

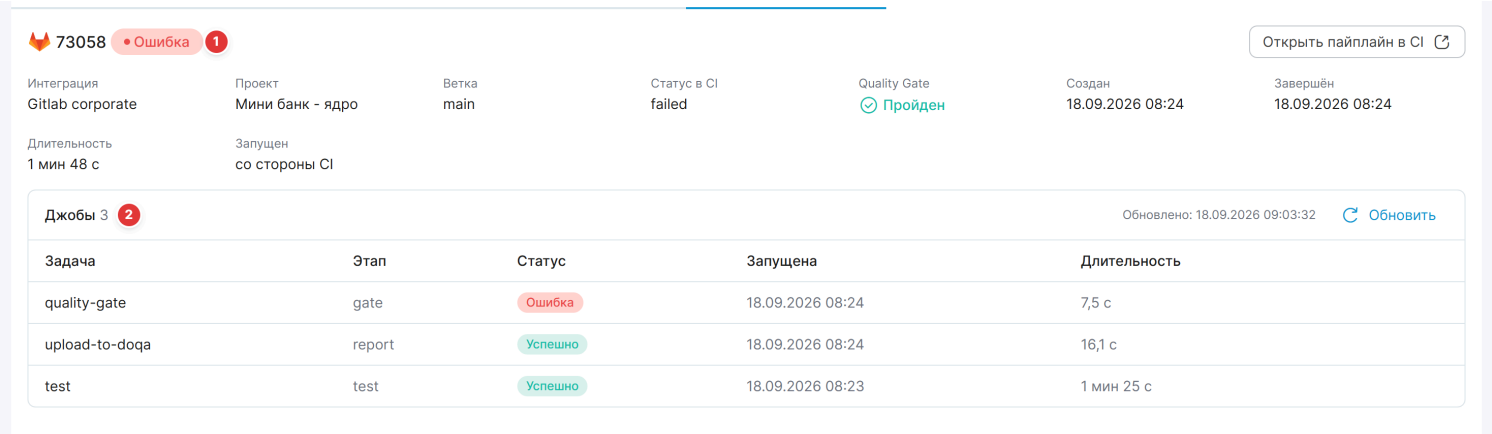
Несколько пайплайнов в одном прогоне

Один прогон может нести несколько пайплайнов: мульти-запуск по группам («каждый там, где запускался»), авто-повторы и перепрохождение отдельных тестов создают дополнительные пайплайны в том же прогоне. Прогон завершается только тогда, когда не осталось ни одного активного пайплайна; пока прогон в работе и пайплайнов больше одного, в шапке плеера видна сводка вида «(2 из 3 пайплайнов завершены)».

Все пайплайны прогона перечислены в разделе «Пайплайны и джобы» плеера и сгруппированы по запускам: «Последний запуск» и «Предыдущие запуски». Карточка пайплайна показывает провайдера, проект, ветку, время старта, длительность, статус, позицию в группе запуска (бейдж «2 из 3») и метку retry у служебных пайплайнов; под карточкой — таблица джоб этого пайплайна. По умолчанию виден только последний запуск. Переключатель «Все» раскрывает историю целиком.

Частичный старт групп не откатывается

Если при мульти-запуске по группам часть пайплайнов не стартовала, уже запущенные продолжают выполняться, а нестартовавшие помечаются неуспешными. Если не стартовала ни одна группа, прогон сразу завершается.



На скриншоте: 1 — карточка пайплайна, 2 — таблица джоб.

Как пайплайн узнаёт, что запускать

DoQA передаёт контекст запуска переменными пайплайна. Чтобы выполнить не весь набор, а выбранные тесты, есть два механизма: нативный фильтр и тест-план по API.

Переменные пайплайна

При старте пайплайна DoQA передаёт переменные `DOQA_*` (способ передачи зависит от CI-системы: переменные, параметры, inputs):

Переменная	Что содержит
`DOQA_SPACE_ID`	ID пространства — куда слать тест-план и результаты
`DOQA_TEST_RUN_ID`	ID прогона DoQA, в который должны попасть результаты
`DOQA_CI_RUN_ID`	ID пайплайна в DoQA — для корреляции результатов, когда у прогона несколько пайплайнов; передаётся вместе с `DOQA_TEST_RUN_ID`
`DOQA_ADAPTER_MODE`	режим адаптера: `1` — прогнать весь набор, `0` — только выбранные тесты
`DOQA_NATIVE_FILTER`, `DOQA_FILTER_KIND`	выражение нативного фильтра и его вид; передаются только вместе
`DOQA_TRIGGER`, `DOQA_SCHEDULE_ID`	признак запуска по расписанию и ID расписания
`DOQA_SOURCE_KEY`	идентификатор подпроекта привязки
`DOQA_CORRELATION_ID`	служебный ключ корреляции для провайдеров, не отдающих ID пайплайна при старте

DOQA_URL и DOQA_TOKEN при запуске не передаются

Адрес DoQA и project-токен настраиваются в CI-проекте заранее, вручную или **авто-настройкой переменных**.

Нативный фильтр тестов

Нативный фильтр — выражение отбора тестов в «родном» синтаксисе фреймворка (например, `Class#method` для Maven Surefire или выражение `-k` для pytest). Пайплайн подставляет `DOQA_NATIVE_FILTER` в команду запуска тестов, и выполняются только выбранные тесты, без обращения к API.

Формат отчёта (`DOQA_FILTER_KIND`) — один из восьми: `surefire`, `gradle`, `pytest`, `gotest`, `playwright`, `phpunit`, `dotnet`, `jest`. DoQA определяет его по цепочке приоритетов. Побеждает первый сработавший:

1. **формат, заданный вручную в настройках источника** — жёсткая настройка источника;

2. формат, выбранный в диалоге запуска;
3. формат источника, запомненный DoQA по результатам его прошлых прогонов;
4. определение по самим выбранным автотестам — префикс внешнего ID, метка фреймворка из последнего результата, имя класса и метода.

Фильтр либо передаётся целиком, либо не передаётся вовсе

Частичного фильтра не бывает: выражение либо адресует **все** выбранные автотесты, либо не формируется совсем. Переменные `DOQA_NATIVE_FILTER` и `DOQA_FILTER_KIND` при этом не передаются, а пайплайн выполняет весь набор проекта: лучше полный прогон, чем молча потерянный тест. Запуск при этом не отклоняется: прогон создаётся, автотесты в него материализуются, ответ содержит `nativeFilter: null` и список автотестов, которые фильтр адресовать не смог.

Фильтр не передаётся, когда:

- формат отчёта определить не удалось: сигналов нет;
- в наборе смешаны тесты разных фреймворков;
- хотя бы один автотест невыразим в этом формате: у него нет достоверного имени раннера, а отображаемое имя (заголовок из `@DisplayName`, `@allure.title`, docstring) селектором не является — по нему раннер не нашёл бы ни одного теста;
- выражение получилось длиннее 2048 символов;
- тумблер нативного фильтра в диалоге запуска выключен.

Ограничить состав **выполнения** в этих случаях могут только селективный режим адаптера (`DOQA_ADAPTER_MODE=0`) или **тест-план**. На состав **прогона** это не влияет: у запуска выбранных автотестов и у запуска из плеера набор пайплайна известен DoQA, поэтому результаты тестов вне набора отбрасываются при приёме — лишние прогнанные тесты в прогон не попадут. Прогон, созданный способом «Запуск автотестов» или со стороны CI, набора не имеет и принимает все результаты пайплайна.

Формат, выбранный в диалоге, — подсказка, а не гарантия

Если выбранный в диалоге формат не подходит части набора (например, выбран `pytest`, а среди выбранных есть JVM-тесты), запуск не отклоняется: вместо выборочного выполняется полный прогон по правилу выше. Диалог запуска показывает это заранее: в превью фильтра видно, будет ли фильтр передан, какие автотесты он не покрывает и откуда взят формат.

Единственное исключение — конфликт с форматом, **заданным вручную в настройках источника**: такой запуск отклоняется с ошибкой, потому что источник уже описан явно и выбор в диалоге его не переопределяет. Превью в этом случае не отказывает, а предупреждает и показывает выражение в формате источника.

Куда подставлять выражение

`DOQA_NATIVE_FILTER` несёт **значение** выражения без имени флага. Флаг добавляет сам рецепт пайплайна:

Формат	Раннер	Как подставить
`surefire`	Maven Surefire/Failsafe	`mvn test -Dtest="$DOQA_NATIVE_FILTER"`
`gradle`	Gradle	`gradle test $DOQA_NATIVE_FILTER` — без кавычек
`pytest`	pytest	`pytest -k "$DOQA_NATIVE_FILTER"`
`gotest`	go test	`go test ./... -run "$DOQA_NATIVE_FILTER"`
`phpunit`	PHPUnit	`phpunit --filter "$DOQA_NATIVE_FILTER"`
`dotnet`	dotnet test / VSTest	`dotnet test --filter "$DOQA_NATIVE_FILTER"`
`playwright`	Playwright	`npx playwright test -g "$DOQA_NATIVE_FILTER"`
`jest`	Jest	`npx jest -t "$DOQA_NATIVE_FILTER"`

Для gradle кавычки вокруг переменной ломают запуск

Только у `gradle` переменная несёт не одно значение, а готовый фрагмент аргументов — `--tests com.acme.LoginTest.login --tests com.acme.CartTest.add`. В кавычках оболочка передаст его Gradle одним аргументом, и разбор сломается («Unknown command-line option»). Пишите `$DOQA_NATIVE_FILTER` без кавычек. У остальных форматов кавычки, наоборот, обязательны: выражения содержат `|`, `(`, `)`, `^`, `$` и пробелы.

Тест-план по API

Второй механизм — пайплайн сам спрашивает у DoQA, что запускать:

```
GET {DOQA_URL}/api/integrations/ci/test-plan
    ?pipeline_id=$CI_PIPELINE_ID
    &ci_project_id=$CI_PROJECT_ID
    &space_id=$DOQA_SPACE_ID
    &token=$DOQA_TOKEN
```

Все четыре параметра обязательны. `token` — project-токен.

Ответ:

json

```
{
  "tests": [],
  "runId": 1770,
  "isNewRun": false,
  "version": "1.0",
  "autotests": [
    {
      "autotestId": 17,
      "externalId": "pytest:tests/test_login.py::test_valid",
      "name": "test_valid",
      "allureId": 99,
      "selector": "tests.test_login#test_valid"
    },
    {
      "autotestId": 24,
      "externalId": "pytest:tests/test_logout.py::test_session",
      "name": "test_session",
      "allureId": null,
      "selector": "tests.test_logout#test_session"
    }
  ],
  "incomplete": true
}
```

- `tests` — устаревший способ адресации: по Allure ID связанных тест-кейсов. Самостоятельный автотест (без связи с ручным кейсом) в этом поле выразить нечем. Если этот список беднее реального набора (`incomplete: true`), DoQA отдаёт `tests: []` . Пустой список прямо означает «этим полем набор не выражается», а по усечённому списку молча выполнился бы не тот набор;

- `autotests` — полный набор пайплайна. `externalId` адресует автотест независимо от того, есть ли у него связанный кейс (`allureId: null` — самостоятельный автотест), а `selector` — полное имя теста в том же виде, которым пользуется нативный фильтр (`null`, если достоверного имени у автотеста нет). Новые рецепты строят план по `autotests[]`;
- `incomplete` — `true`, когда набор нельзя адресовать целиком: часть строк без Allure ID либо часть автотестов без достоверного `selector`. Рецепт, который умеет фильтровать только по Allure ID, в этом случае обязан либо перейти на `autotests[]`, либо прогнать весь набор целиком, иначе выбранные тесты молча не выполнятся;
- `runId` — прогон DoQA, в который уйдут результаты;
- `isNewRun` — `true`, если прогон был создан способом «Запуск автотестов», и `false` при запуске в существующий прогон.

Пример шага, который забирает план:

```yaml
fetch_plan:
  stage: test
  script:
    - >
      curl -fsSL --get "$DOQA_URL/api/integrations/ci/test-plan"
      --data-urlencode "pipeline_id=$CI_PIPELINE_ID"
      --data-urlencode "ci_project_id=$CI_PROJECT_ID"
      --data-urlencode "space_id=$DOQA_SPACE_ID"
      --data-urlencode "token=$DOQA_TOKEN"
      -o test-plan.json
    - cat test-plan.json
```

Как отфильтровать тесты по этим ID, зависит от фреймворка и раннера. Без фильтрации (нативной или по тест-плану) пайплайн прогонит все тесты проекта: на состав прогона это не повлияет (см. выше), но выполнение займёт больше времени.

Жизненный цикл пайплайна

Статус пайплайна DoQA приводит к единому набору: в очереди, выполняется, успешно, с ошибкой, отменён, пропущен, таймаут. Статус доставляется тремя путями:

- **Вебхук** — основной путь: CI-система сообщает о завершении, и DoQA финализирует пайплайн сразу. Требуется **настроенный вебхук подключения**.
- **Поллинг** — страховка: DoQA ежеминутно опрашивает незавершённые пайплайны старше 5 минут. Пайплайн, застрявший в очереди дольше 60 минут, закрывается со статусом таймаута; пайплайн, статус которого узнать не удалось, — как неуспешный.
- **Скачивание результатов** — если пайплайн завершился, а результаты тестов так и не пришли в течение 15 минут, DoQA сама скачивает артефакт с результатами из джоб пайплайна и разбирает его. Работает только для GitLab.

Когда завершается последний активный пайплайн, происходит финализация прогона. Прогон, созданный самим запуском — способом «Запуск автотестов», из каталога или со стороны CI, — переводится в «Завершен» (пайплайн, запущенный в уже существующий прогон, его статус не меняет). Плановые, но не выполненные автотесты помечаются пропущенными. Затем отрабатывает автоматика: авто-повторы и критерии приёмки (**Автоматика запусков**), кластеризация ошибок и пересчёт стабильности (**Результаты и разбор падений**).

Остановка прогона отменяет все его активные пайплайны, запущенные из DoQA. Отдельный пайплайн отменяется в разделе «Пайплайны и джобы». Пользовательская остановка не считается CI-завершением: авто-повторы и критерии приёмки на неё не реагируют.

Управление CI-джобами

Джобы показываются в том же разделе «Пайплайны и джобы», справа, для выбранного в списке пайплайна: имя, стадия, статус, время старта, длительность, ссылка. Над таблицей — карточка пайплайна с его статусом в CI, временами и действиями «Открыть пайплайн в CI» и «Отменить пайплайн». По умолчанию открыт активный пайплайн, а если активных нет — самый свежий; выбранный пайплайн остаётся в адресной строке, поэтому ссылкой можно поделиться. Пока раздел открыт и работа в CI идёт, срез обновляется примерно каждые 15 секунд.

Джобу можно отменить или перезапустить кнопками на строке (с подтверждением); набор доступных действий зависит от возможностей CI-системы — см. **матрицу провайдеров**. Для провайдера без управления джобами вместо таблицы показывается пояснение. Сам пайплайн при этом открывается по ссылке из его карточки. Права на действия — те же, что на редактирование прогонов.

Таймлайны

По каждому результату автотеста доступен таймлайн запуска «от команды до отчёта», в карточке результата. По прогонам целиком — лента «Таймлайн запусков» на вкладке аналитики автотестов **дашборда**.

Смотрите также

- **Подключение CI-систем** — подключения, привязки проектов, вебхуки, переменные
- **Автоматика запусков** — расписания, авто-повтор упавших, критерии приёмки прогона
- **Каталог автотестов** — выбор автотестов для запуска, карантин, источники
- **Результаты и разбор падений** — карточка результата, попытки, кластеры ошибок
- **Прогоны** — статусы и завершение прогона

Автоматика запусков

DoQA умеет запускать автотесты по расписанию, автоматически перезапускать упавшие и оценивать итог прогона по критериям приёмки (Quality Gate). Все три механизма настраиваются на уровне пространства и выключены, пока вы их не включите.

Где настраивается

Откройте раздел «Автотесты» и нажмите кнопку настроек в шапке раздела либо выберите пункт «Настройки» в меню пространства. Нужные секции находятся в группе «Autotests» страницы «Настройки пространства»:

Секция	Что настраивает
«Расписания прогонов»	расписания запусков автотестов
«Авто-повтор»	автоматический перезапуск упавших автотестов
«Quality Gate»	критерии приёмки прогона

Активная секция синхронизирована с адресом страницы (параметр `?section=`), поэтому ссылку на конкретную настройку можно отправить коллеге. Страница доступна пользователям с правами на настройки пространства; без права изменения формы открываются только на чтение.

Расписания запусков

Расписание запускает пайплайн автотестов во внешней CI-системе по таймеру. Менять конфигурацию CI не нужно. Понадобится настроенное **подключение CI** с привязанным проектом.

Создание расписания

Нажмите «**New schedule**» и заполните форму:

- «Name» — имя расписания.
- Где запускать: CI-подключение → проект → ветка. Все три поля обязательны; если подключения ещё нет, кнопка перехода к настройкам внутри формы переключит вас на секцию «CI/CD connections».
- «Run title template» — шаблон названия прогона; поддерживаются плейсхолдеры `{name}`, `{date}`, `{time}`.
- «Repeat» — периодичность: «Daily» (время), «Weekly» (время и «Days of week»), «Interval» («Every N hours», от 1 до 23) или «Cron expression» — cron-выражение из 5 полей: минута, час, день, месяц, день недели (например, `0 3 * * 1-5`).
- «Timezone» — таймзона расписания; по умолчанию — зона вашего браузера.
- «If a previous run of this schedule is still active» — поведение при пересечении с ещё идущим прогоном этого расписания: «Skip this tick (recommended)» — пропустить срабатывание, «Start in parallel» — запустить параллельный прогон. Настройка смотрит **только на прогоны этого же расписания**: если параллельно работает прогон другого расписания или запущенный вручную, срабатывание не пропускается. Чтобы два расписания не наслаивались друг на друга, разведите их по времени.
- «Override quality gate for scheduled runs» и «Override auto-retry for scheduled runs» — свои критерии приёмки и политика перезапусков для прогонов этого расписания вместо настроек пространства.
- «Enabled» — новое расписание создаётся выключенным; не забудьте включить его.

Под формой DoQA показывает «Next 5 runs» — пять ближайших срабатываний с учётом таймзоны. Минимальный интервал между срабатываниями — 10 минут, лимит — 20 расписаний на пространство (значения по умолчанию, настраиваются администратором сервера).

Что происходит при срабатывании

DoQA создаёт прогон и запускает пайплайн так же, как при ручном **запуске автотестов из DoQA**. Прогон создаётся от имени автора расписания; если автор удалён или заблокирован, срабатывание завершается ошибкой. Пропущенные срабатывания (например, во время недоступности сервера) не навёрстываются.

Примечание

Сбои расписания копятся: после 10 сбоев подряд (значение по умолчанию) расписание автоматически выключается, а DoQA присылает уведомление. Сбоем считаются ошибка запуска и недоступность CI-подключения; пропуск из-за ещё идущего прогона — нет.

Список и история срабатываний

В списке расписаний: «Name», «Schedule» — периодичность и таймзона, «Next run» — следующее срабатывание (только у включённых), «Last status» — итог последнего срабатывания, тумблер «Enabled» и действия:

- «Run now» — запустить расписание немедленно. Если запуск невозможен, DoQA покажет предупреждение «Not started: …» с причиной.
- «Trigger history» — история срабатываний: «Planned for», «Status», «Run», «Error». Статусы: «Triggered» — прогон запущен (кнопка «Open run» ведёт в него), «Skipped — previous run active», «Skipped — integration unavailable», «Failed». История хранится 90 дней (по умолчанию).
- «Edit» и «Delete» — изменить или удалить расписание.

Авто-повтор упавших автотестов

Политика авто-повтора перезапускает упавшие автотесты силами DoQA, без изменений в пайплайне. По умолчанию выключена. Поля секции «Авто-повтор»:

Поле	Смысл
«Enable auto-retry of failed tests»	включить политику
«Max retry attempts»	число дополнительных попыток на тест, от 0 до 10; 0 — перезапусков нет
«Delay between attempts, sec»	пауза перед перезапуском — даёт окружению восстановиться
«Retry only tests with tags»	перезапускать только тесты с этими тегами; пусто — любые упавшие
«Never retry these cases»	исключения: id кейса или внешний ID автотеста

Поле	Смысл
«Fail the pipeline if all retries fail»	если попытки исчерпаны, а тесты так и не прошли — прогон проваливается

Как работает перезапуск

1. Пайплайн прогона завершается; DoQA отбирает упавшие и сломанные автотесты (по последней попытке каждого).
2. Набор фильтруется по тегам и исключениям политики.
3. DoQA запускает новый пайплайн только на эти тесты, с нативным фильтром, как при **запуске выбранных**. Если задана задержка, перезапуск откладывается на указанное время.
4. Цикл повторяется, пока тесты не пройдут или не исчерпается «Max retry attempts»; жёсткий потолок — 10 попыток.

Авто-повтор срабатывает только для прогонов, запущенных из DoQA (в том числе расписанием). Для пайплайнов, стартовавших со стороны CI, перезапуски не выполняются.

Перезапуски в прогоне

Каждая попытка сохраняется отдельно: список попыток, итог по последней и отметка «Пройден после повтора» описаны в **Результатах и разборе падений**. Пайплайны повторов видны в разделе «Пайплайны и джобы» прогона с меткой «Повтор».

Политику можно переопределить для отдельного прогона в диалоге дозапуска автотестов (см. **Запуск автотестов из DoQA**). В плеере прогона есть кнопка «**Retry failed tests now**» — вручную перезапустить упавшие прямо сейчас; она работает только при включённой политике.

Критерии приёмки прогона (Quality Gate)

Quality Gate — декларативные критерии «прогон прошёл / не прошёл» поверх результатов автотестов. Пока гейт не настроен, прогоны ведут себя как обычно. Критерии секции «Quality Gate»:

Критерий	Смысл
«Enable Quality Gate»	включить гейт
«Minimum success rate, %»	минимальная доля успешных тестов
«Maximum failures»	максимально допустимое число упавших
«Minimum tests count»	защита от полупустого прогона: выполнено меньше тестов — гейт провален
«Maximum duration, sec»	максимальная длительность прогона
«Fast fail (stop on first violation)»	остановить проверку на первом нарушении
«Запретить переход в зелёный через повторный прогон»	прогон не может пройти гейт, если его тесты прошли только после авто-повтора

Какие результаты учитываются

- только автотесты и только последняя попытка каждого;
- пропущенные и ещё не обработанные результаты не входят в знаменатель;
- исключаются автотесты в карантине (см. **каталог**) и падения, скрытые известными проблемами кластеров (см. **кластеры ошибок**).

Уровни настройки

Гейт пространства применяется ко всем CI-прогонам. Для отдельного прогона его можно переопределить в диалоге дозапуска автотестов, для прогонов расписания — тумблером «Override quality gate for scheduled runs».

Внимание

Override заменяет гейт пространства целиком, а не дополняет его: критерии, не заданные в override, не проверяются. То же относится к override политики авто-повтора.

Вердикт в прогоне

Гейт оценивается автоматически после завершения каждого пайплайна прогона; при включённом авто-повторе — после того, как отработали перезапуски. Вердикт виден в шапке

прогона (бейдж) и в панели «Quality Gate result»:

- статус: «Gate passed», «Gate failed» или «Missing data»;
- «Violated criteria» — нарушенные критерии в формате «Criterion» / «Expected» / «Actual»;
- «Evaluated metrics» — фактические метрики прогона;
- кнопка «**Evaluate now**» — пересчитать вердикт вручную по последнему завершённому пайплайну.

«Missing data» означает, что оценивать нечего: например, ни один тест не зачтён при заданном числовом критерии. По умолчанию такой вердикт тоже считается блокирующим.

При провале гейта прогон в списках и лентах показывается как проваленный. Статус пайплайна, полученный от CI-системы, DoQA при этом не переписывает.

Проверка вердикта из пайплайна

Джоба пайплайна может запросить вердикт гейта у DoQA: `GET /api/integrations/ci/quality-gate-check`, аутентификация project-токеном (см. **Подключение CI-систем**). По умолчанию метод отвечает `200` со статусом вердикта. Администратор сервера может включить режим, в котором блокирующий вердикт возвращает `422` (переменная `CI_QG_CHECK_HTTP_FAIL`). Тогда джобу можно завершать с ошибкой обычным `curl -f`.

Смотрите также

- **Запуск автотестов из DoQA** — способы запуска и жизненный цикл пайплайна
- **Подключение CI-систем** — подключения, привязка проектов, project-токен
- **Результаты и разбор падений** — попытки, кластеры ошибок, сравнение прогонов
- **Прогоны** — работа с прогонами и плеером

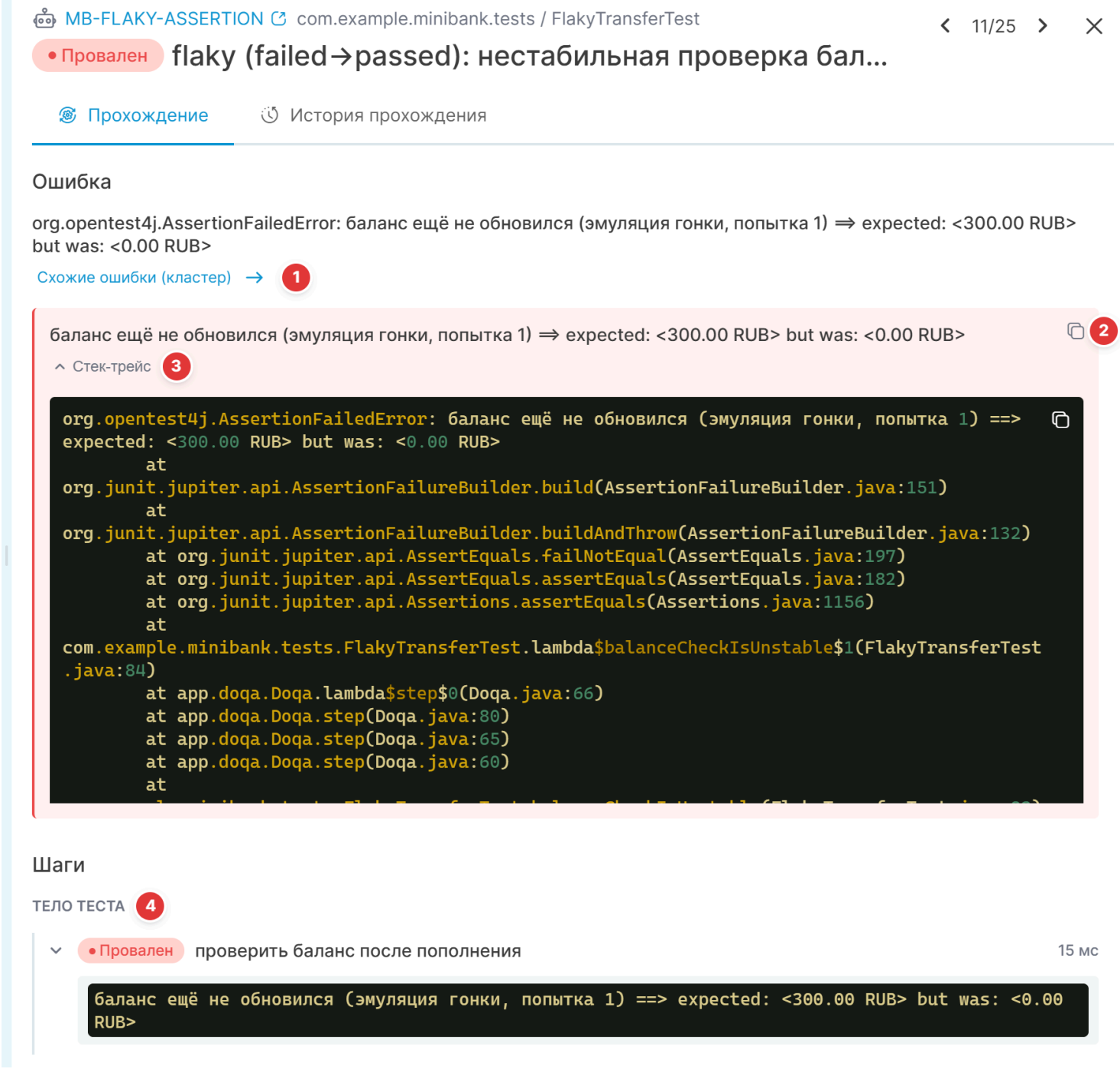
Результаты и разбор падений

Результаты автотестов находятся в **прогонах**. Эта страница о том, как читать карточку результата, разбирать причины падений через кластеры ошибок, сравнивать прогоны и следить за стабильностью. Пошаговый разбор упавшего прогона — в сценарии **Разбор упавших автотестов**.

Карточка результата автотеста

Откройте прогон и выберите автотест в таблице. Карточка результата откроется в панели справа. Эту же карточку в режиме чтения открывают история прогонов в карточке автотеста **каталога** и список тестов кластера ошибок. Кнопка «Открыть в прогоне» в ней ведёт к оригиналу.

У автотестов исполнителем указана «Автоматизация»: их результаты не редактируются вручную. Пока результаты обрабатываются, вместо карточки показывается заглушка «Результаты автотестирования все еще обрабатываются. Это может занять несколько минут — пожалуйста, подождите.»



На скриншоте: 1 — ссылка «Схожие ошибки (кластер)» на кластер, к которому отнесено падение, 2 — сообщение об ошибке с кнопкой копирования, 3 — переключатель «Стек-трейс», 4 — дерево шагов с упавшим шагом.

Состав карточки

Сверху вниз:

- конфигурация прогона и типизированные ссылки результата, если они переданы;
- сводка попыток — см. Попытки;

- таймлайн запуска — фазы от отправки команды в CI до получения отчёта, с длительностью каждой;
- карточка ошибки — см. ниже;
- дерево шагов и фикстур, если шаги были переданы: группы Setup → Test body → Teardown, упавшие узлы раскрыты сразу, у шагов видны длительность, параметры и вложения (шаги умеют отправлять **адаптеры** и **Autotest API**);
- «Output» — вывод теста; свёрнут и скрывается, если дословно дублирует стек-трейс;
- вложения: скриншоты открываются в полноэкранном просмотре, видео проигрываются в карточке, трейсы скачиваются файлом;
- параметры из отчёта;
- обсуждение: кнопки «**Добавить баг**» и «**Добавить комментарий**», добавленные **баг-репорты** и **комментарии**;
- свойства автотеста из отчёта.

Карточка ошибки

Для упавших, сломанных и заблокированных результатов секция называется «Ошибка». Для остальных — «Сообщение». Карточка при этом оформляется нейтрально: у прошедшего теста текст из отчёта — информационное сообщение, а не падение. Ошибка показывается всегда, когда она есть в отчёте, независимо от статуса результата.

В карточке: тип исключения, сообщение с кнопкой копирования и свёрнутый стек-трейс (переключатель «Стек-трейс») с подсветкой синтаксиса.

Если парсер не смог разобрать сам результат, DoQA показывает отдельную предупреждающую карточку «Ошибка обработки отчёта» — это проблема отчёта, а не теста.

Когда падение попало в кластер, над карточкой ошибки выводится блок кластера: заголовок, категория причины и ссылка «Similar failures (cluster)», открывающая этот кластер на вкладке «Кластеры ошибок».

Попытки

Один автотест в прогоне может пройти несколько раз: авто-повтор, перезапуск из DoQA, ручное перепрохождение или повтор средствами самого тестового фреймворка. Все попытки сохраняются; итог теста и вся статистика считаются по последней.

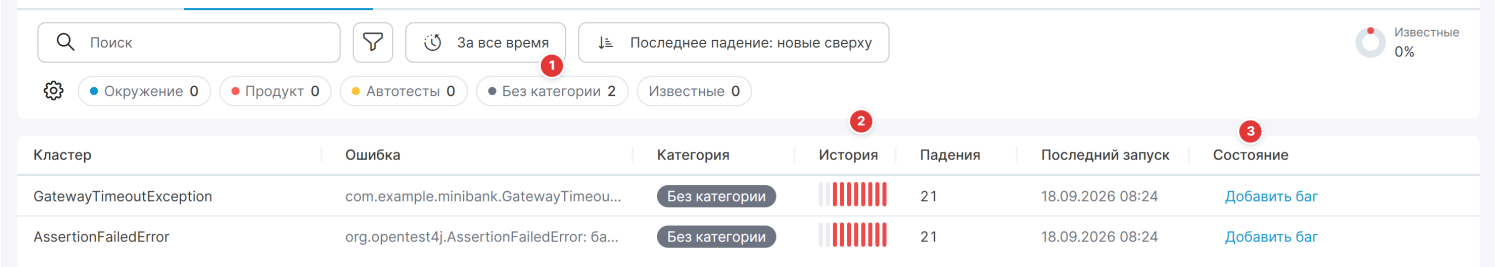
В списке попыток: номер, источник попытки, статус, отметка нестабильности, время, исполнитель и ссылка на пайплайн. У любой не прошедшей попытки есть кнопка «**Подробнее**» — внутри её собственные таймлайн, шаги, лог и вложения.

Тест, прошедший только после перезапуска, получает отметку «Пройден после повтора»: такие результаты учитывает детекция нестабильных. Правила автоматических перезапусков — на странице **Автоматика запусков**, перепрохождение одного автотеста — на странице **Запуск автотестов из DoQA**.

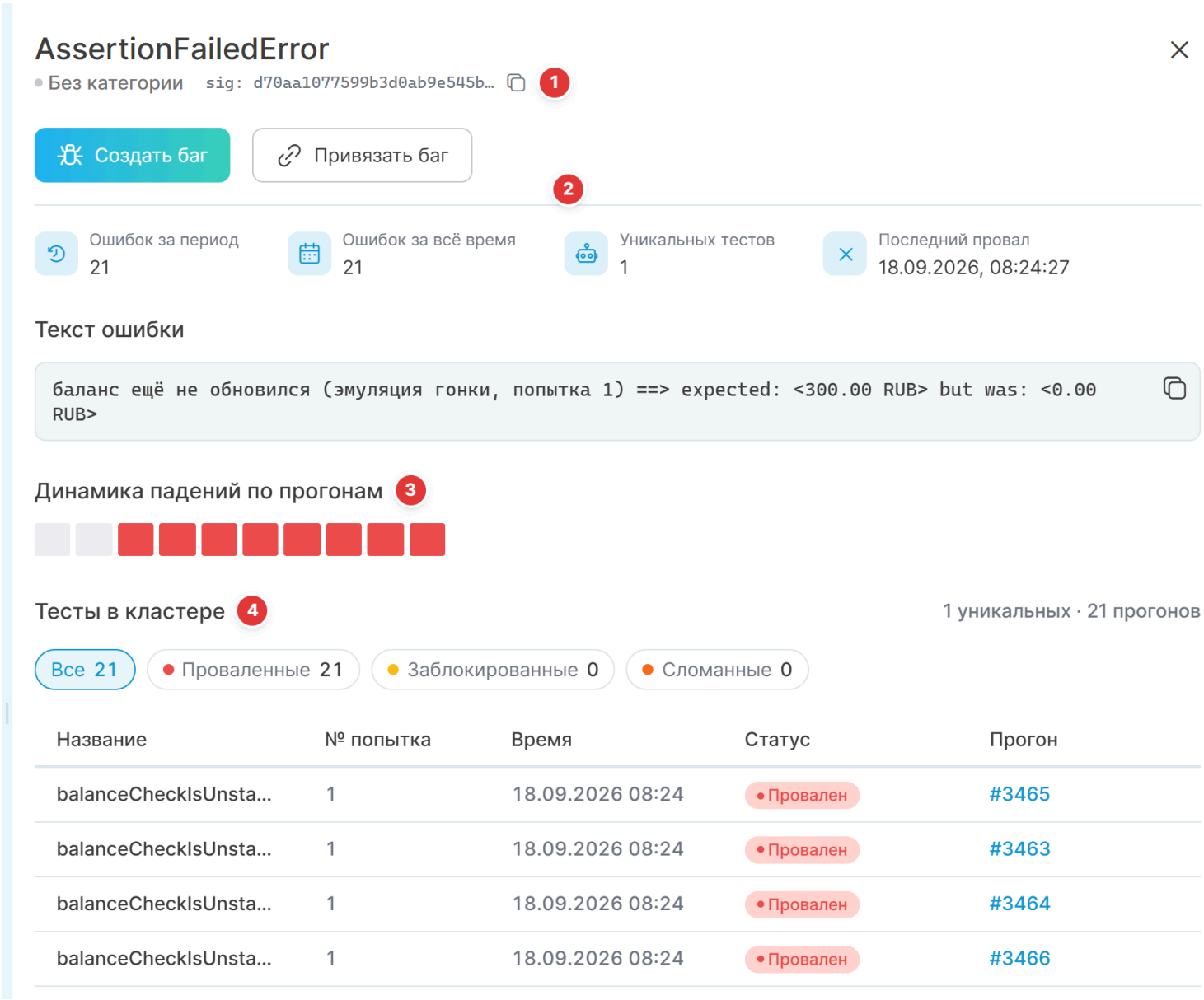
Кластеры ошибок

Вкладка «Кластеры ошибок» раздела «Автотесты» группирует падения автотестов пространства по сигнатуре ошибки: вместо сотен одинаковых упавших тестов — короткий список причин. Группировка детерминированная: нормализация текста ошибки плюс настраиваемые правила, без машинного обучения.

Кластеризация выполняется после завершения CI-прогона, поэтому свежие падения появляются на вкладке не мгновенно; список обновляется при повторном открытии страницы.



На скриншоте: 1 — плитка-фильтр категории, 2 — колонка «История» с мини-трендом падений, 3 — колонка «Состояние».



На скриншоте: 1 — сигнатура кластера с копированием, 2 — показатели кластера, 3 — тренд падений по прогонам, 4 — список тестов кластера с фильтром по статусу.

Категории причин

Каждому кластеру назначается категория — класс причины падения:

Категория	Смысл
«Продукт»	баг продукта
«Автотесты»	сломанный тест
«Окружение»	проблема окружения или сервиса

Категория	Смысл
«Без категории»	не совпало ни одно правило

Правила категоризации настраиваются в настройках пространства, секция «Категории ошибок»: регулярное выражение, к чему его применять (сообщение, стек-трейс или оба), категория и приоритет. Правила проверяются по возрастанию приоритета, побеждает первое совпавшее. Поле примера позволяет проверить паттерн на реальном сообщении прямо при вводе.

Работа со списком

- Плитки-фильтры по категориям: «Окружение», «Продукт», «Автотесты», «Без категории», «Известные» — на плитке счётчик, клик фильтрует список.
- Период (7, 30, 90 дней или всё время), поиск по типу и сообщению ошибки, сортировка по падениям за период или по свежести.
- Строка кластера: тип исключения и сообщение, бейдж категории, мини-тренд, падения за период и за всё время, последнее падение и состояние — бейдж связанного бага, метка «Reopened» либо ссылка «**Добавить баг**».

Клик по кластеру открывает панель: сигнатура с копированием, показатели (падения за период и всего, уникальных тестов, из них нестабильных, первое и последнее падение), крупный тренд по прогонам и список тестов кластера с фильтром «Только упавшие». Клик по тесту открывает карточку результата.

Кластер и баг-репорт

Баг кластера — это обычный баг-репорт раздела **«Баги»**. Оформить его можно двумя способами:

- «**Создать баг**» — открывает форму баг-репорта, заполненную по характерному падению кластера; созданный баг автоматически связывается с кластером;
- «**Привязать баг**» — связать кластер с существующим багом пространства или с тикетом трекера по ключу.

После оформления на кластере появляется плашка «Оформлен как дефект» с настройками связки:

- «Auto-hide in next runs» — падения кластера скрываются из списка (возвращаются тумблером «Показывать known issues») и исключаются из оценки критериев приёмки;
- «Reopen on regression» — если баг уже закрыт, а кластер упал снова, связка помечается «Reopened», и DoQA автоматически переоткрывает баг;
- «Отвязать кластер» — снимает связь; после отвязки последнего кластера связка удаляется, а баг остаётся обычным багом.

Внимание

Пока связка с включённым «Auto-hide in next runs» открыта, падения кластера не блокируют Quality Gate. Это удобно для известных проблем, но помните: гейт может оставаться пройденным при реально падающих тестах.

Связка видна и со стороны бага: в списке багов у багов из кластера есть отметка и фильтр «Источник», а в карточке бага — секция «Кластеры ошибок» с теми же настройками и отвязкой.

Стабильность и карантин

По каждому автотесту DoQA считает стабильность между прогонами: процент успешных прохождений, смены исхода и итоговое состояние. Смотреть их удобнее всего в **каталоге автотестов**: полоска последних десяти прогонов, процент стабильности и бейдж нестабильности в таблице, а в карточке автотеста — плитки статистики, история прогонов и журнал переходов состояния.

Хронически нестабильный тест можно отправить в карантин из каталога или карточки автотеста. Карантинный автотест исключается из статистики стабильности и из оценки критериев приёмки, пока карантин не снят.

Сравнение прогонов

Чтобы понять, что изменилось между двумя прогонами, выберите ровно два прогона в списке и нажмите «**Сравнить**». Сравнивать можно только прогоны одного пространства, и в обоих должны быть результаты автотестов.

Тесты двух прогонов сопоставляются по автотесту каталога (внешнему ID); результаты без привязки к каталогу показываются отдельными счётчиками. Каждому тесту присваивается класс изменения:

Класс	Смысл
«Regressed»	проходил в A, упал в B
«New»	появился только в B
«Still failing»	падает в обоих прогонах
«Fixed»	падал в A, прошёл в B
«Disappeared»	был в A, отсутствует в B
«Status changed»	прочие изменения статуса
«Unchanged»	без изменений; по умолчанию скрыты

На странице: карточки прогонов A и B (ветка, коммит, статус пайплайна), сводка с кликабельными счётчиками по классам, суммарная разница длительности, тумблеры «Only changes» и «Flaky only», фильтр окружения и поиск. В таблице — статусы A → B (клик открывает результат в соответствующем прогоне), разница длительности и число попыток каждой стороны.

Незавершённый прогон сравнивается по текущему состоянию. DoQA предупреждает, что результат сравнения может измениться. Всё состояние страницы хранится в адресе, поэтому ссылкой на сравнение можно поделиться.

Аналитика по автотестам

Вкладка «Автотесты» **дашборда** пространства собирает аналитику парка: доля автоматизации, доля нестабильных, полоса здоровья, покрытие по папкам, тренды прохождений и длительности, доля перезапусков и экономия времени CI, топ падающих тестов, топ кластеров ошибок, самые долгие тесты и лента CI-запусков.

Смотрите также

- **Разбор упавших автотестов** — пошаговый сценарий работы с упавшим прогоном
- **Каталог автотестов** — стабильность, карантин и карточка автотеста
- **Автоматика запусков** — авто-повтор и критерии приёмки
- **Прогоны** — плеер прогонов и работа с автотестами в прогоне
- **Баги** — работа с баг-репортами

Нестабильные тесты и карантин

DoQA по истории прогонов находит автотесты с непостоянным результатом и присваивает каждому состояние стабильности. Карантин выводит такой автотест из оценки **Quality Gate**, пока его не починили.

Где находится

Настройка детекции — «Автотесты» → «**Настройки автотестов**» → «Нестабильные тесты». Это единственный раздел настроек автотестов, который открывается без доступа к настройкам пространства: без права менять политику форма показывается заблокированной, сверху — плашка «У вас нет прав на изменение политики нестабильных тестов». Менять политику может тот, у кого есть право изменять прогоны или редактировать пространства проекта.

Состояние отдельного автотеста и кнопки карантина — в **каталоге автотестов**: значки «Нестабильный» и «В карантине» в строке, поля фильтра «Стабильность» и «Карантин», вкладка «Стабильность» в панели автотеста.

Состояния автотеста

Состояние	Когда возникает
«Стабильный»	базовое состояние: ни один признак нестабильности не сработал
«Нестабильный»	сработал хотя бы один признак нестабильности
«Сломан»	доля падений высокая, смен исхода почти нет и не сработал ни один признак без порогов — автотест падает детерминированно
«В карантине»	карантин поставлен вручную или автоматически
«Исправлен»	автотест был «Нестабильным», «Сломанным» или в карантине и накопил нужное число дней без падений: в окне нет ни падений, ни смен исхода
«Отключён»	ставится только через API, из интерфейса недостижимо; для Quality Gate работает как карантин, но автоматикой не снимается

Состояние «В карантине» и флаг карантина — не одно и то же, см. **Особенности**.

Признаки нестабильности

Автотест получает состояние «Нестабильный», если сработал любой из признаков:

Признак	Условие
Прошёл после повтора	автотест упал и прошёл при повторе внутри того же прогона
Разный исход на одном коммите	на одном и том же коммите есть и успешный, и упавший результат
Повтор выполнил фреймворк	тестовый фреймворк сам перезапустил упавший автотест
Доля падений выше порога	доля падений в окне ≥ «Доля падений для пометки нестабильным, %» **и** в окне есть смена исхода
Показатель нестабильности выше порога	показатель ≥ «Порог показателя нестабильности» **и** в окне есть смена исхода
Число падений выше порога	падений в окне не меньше значения поля «Число падений для пометки (необязательно)» **и** в окне есть смена исхода

Первые три признака срабатывают сразу, даже до того, как автотест накопил «Минимум прогонов до классификации»; остальные три — только после. Повышение порогов уже поставленную пометку не снимает, см. **Как снимается пометка**.

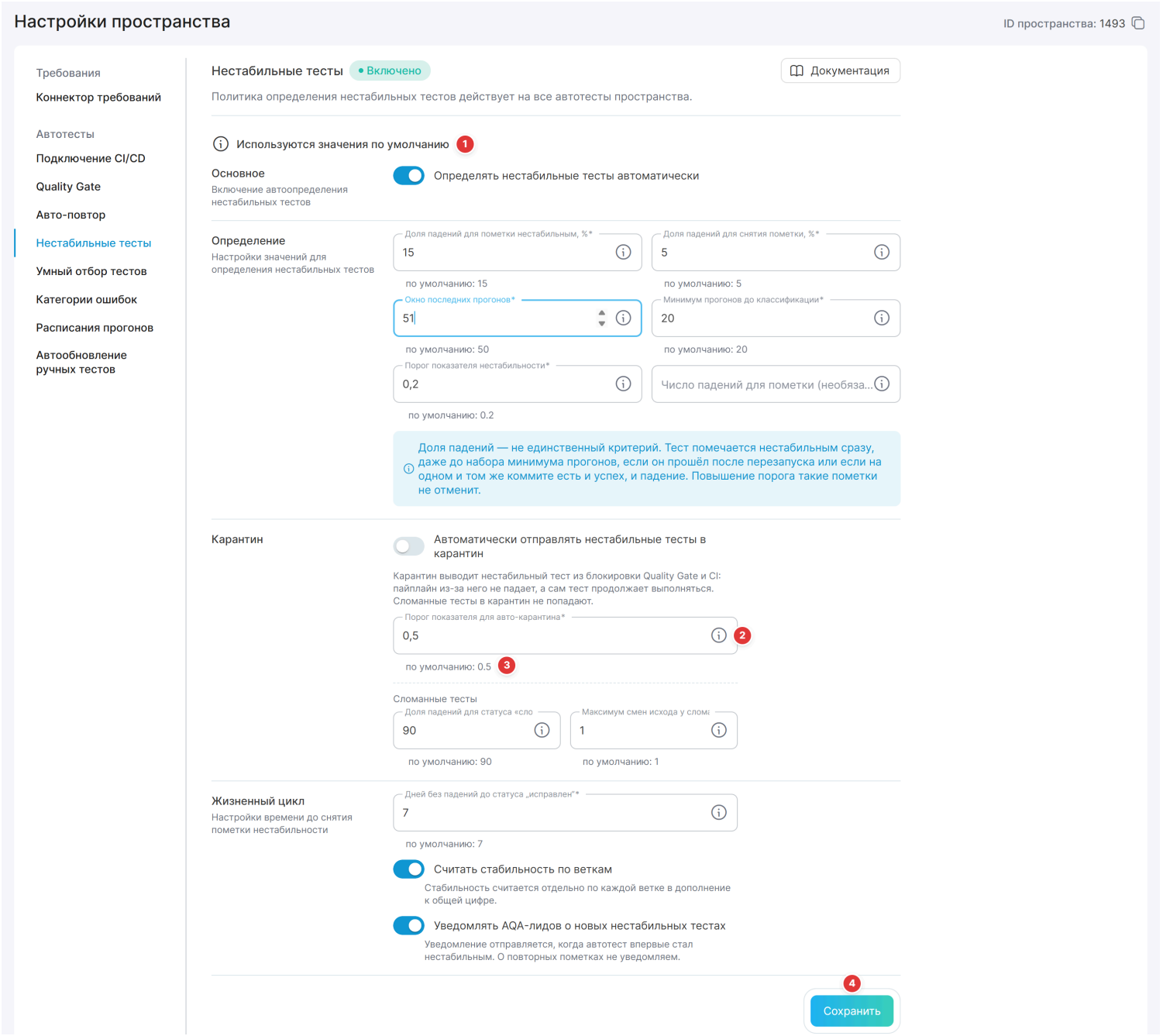
Элемент окна — прогон, а не строка результата: если параметризованный автотест дал в одном прогоне несколько строк, в окно идёт одна, с худшим исходом. Результаты ручных перепрохождений в окно не попадают.

Настроить детекцию

1. Откройте раздел «Нестабильные тесты». Форма заполнена текущими значениями; плашка «Используются значения по умолчанию» означает, что своя политика у пространства ещё не сохранялась.

2. Измените нужные поля. Внизу появляется кнопка «**Сохранить**».
3. Нажмите «**Сохранить**». Появляется всплывающее уведомление «Изменения сохранены».

При ошибке текст выводится под формой, а если сервер причину не назвал — «Не удалось сохранить политику нестабильных тестов. Попробуйте ещё раз.».



На скриншоте: 1 — плашка «Используются значения по умолчанию», 2 — значок-подсказка у поля, 3 — подпись «по умолчанию: X» под числовым полем, 4 — кнопка «Сохранить».

Основное

Поле	По умолчанию	Что делает
«Определять нестабильные тесты автоматически»	включено	пока включено, DoQA пересчитывает состояния; снимите — анализ стабильности останавливается полностью, уже проставленные состояния остаются как есть

Определение

Поле	По умолчанию	Что делает
«Доля падений для пометки нестабильным, %»	15 (0–100)	доля падений в окне, при которой автотест помечается нестабильным
«Доля падений для снятия пометки, %»	5 (0–100)	доля падений, до которой автотест должен опуститься, чтобы пометка снялась
«Окно последних прогонов»	50 (20–100)	сколько последних результатов автотеста анализируется; более старые выходят из окна и на вердикт не влияют
«Минимум прогонов до классификации»	20 (20–100)	пока автотест не накопил столько результатов, признаки с порогами считают его стабильным
«Порог показателя нестабильности»	0.2 (0–1, шаг 0.05)	показатель 0…1: как часто результат переключается между успехом и падением
«Число падений для пометки (необязательно)»	пусто (целое ≥ 1)	дополнительное условие по абсолютному числу падений в окне

Два ограничения проверяются при сохранении:

- «Доля падений для снятия пометки, %» — строго меньше «Доля падений для пометки нестабильным, %», иначе форма не сохраняется: «Доля падений для снятия пометки должна быть строго меньше доли для пометки.»;
- «Окно последних прогонов» — не меньше «Минимума прогонов до классификации», иначе «Окно последних прогонов не может быть меньше минимума прогонов до классификации.».

Карантин

Поле	По умолчанию	Что делает
«Автоматически отправлять нестабильные тесты в карантин»	выключено	автотест, получивший на пересчёте состояние «Нестабильный», уходит в карантин сам — условия ниже
«Порог показателя для авто-карантина»	0.5 (0–1, шаг 0.05)	показатель нестабильности, начиная с которого нестабильный автотест уходит в карантин автоматически
«Доля падений для статуса „сломан“, %»	90 (0–100)	начиная с этой доли падений автотест без смен исхода считается сломанным, а не нестабильным
«Максимум смен исхода у сломанного теста»	1 (целое ≥ 0)	сколько смен исхода ещё допустимо, чтобы автотест считался сломанным

Два последних поля стоят в подразделе «Сломанные тесты».

Жизненный цикл

Поле	По умолчанию	Что делает
«Дней без падений до статуса „исправлен“»	7 (1–90)	сколько дней подряд без падений нужно, чтобы автотест получил состояние «Исправлен» и вышел из карантина
«Считать стабильность по веткам»	включено	стабильность считается отдельно по каждой ветке в дополнение к общей цифре по всем веткам
«Уведомлять AQA-лидов о новых нестабильных тестах»	включено	включает уведомление «Автотест признан нестабильным»

Как снимается пометка

Только автоматически: снять пометку нестабильности вручную нельзя. Такой кнопки в интерфейсе нет.

- **Доля падений упала.** Доля падений опустилась до значения «Доля падений для снятия пометки, %» и смен исхода в окне нет — автотест становится «Стабильным».
- **Дни без падений.** Подряд накоплено «Дней без падений до статуса „исправлен“», в окне нет ни падений, ни смен исхода — автотест становится «Исправлен»; если он был в карантине, карантин снимается.
- **Детекция выключена.** Тумблер «Определять нестабильные тесты автоматически» снят — состояния перестают меняться вовсе, уже проставленные пометки остаются.

Карантин

Отправить автотест в карантин

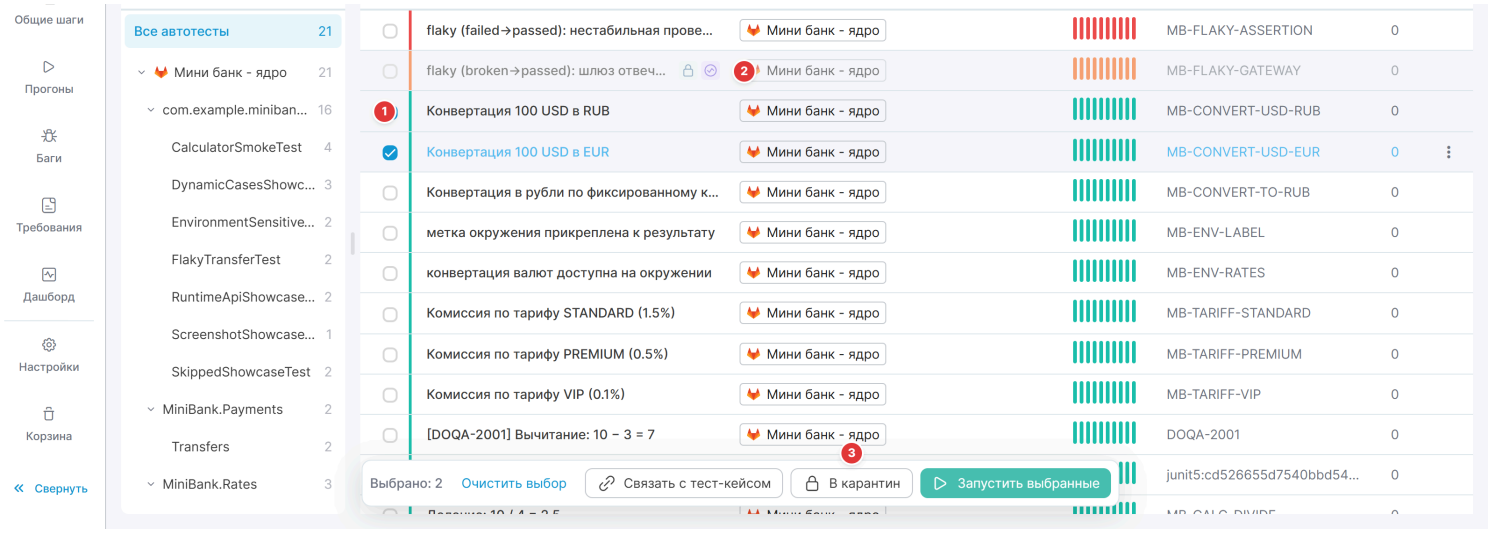
1. В каталоге автотестов отметьте чек-боксами один или несколько автотестов. Внизу появляется панель массовых действий.
2. Нажмите «**В карантин**». Появляется всплывающее уведомление «Карантин обновлён», в строке — значок «В карантине», состояние меняется на «В карантине».

Те же действия есть в меню строки («три точки») и в панели автотеста, на вкладке «История прохождения» в правой колонке. Карантин снимает кнопка «**Снять карантин**» там же; в панели массовых действий она появляется, если среди выбранных есть хотя бы один автотест в карантине. Без права изменять прогоны чек-боксы и кнопки не показываются.

Из каталога карантин ставится на ветку, выбранную в фильтре «Ветка»; если в фильтре «Все ветки» — сразу на все. Из панели автотеста карантин всегда ставится на все ветки.

Внимание

Снятие карантина по всем веткам снимает его и с каждой отдельной ветки, даже если в каталоге этот карантин уже не был виден. Обратной операции «снять со всех веток, оставив отдельные» нет.



На скриншоте: 1 — чек-боксы строк, 2 — значок «В карантине» в столбце «Автотесты» (у другого автотеста списка), 3 — кнопка «В карантин» в нижней панели массовых действий.

Автоматический карантин

Включается тумблером «Автоматически отправлять нестабильные тесты в карантин». Работает только для автотеста, который на этом пересчёте получил состояние «Нестабильный» и ещё не в карантине, при любом из двух условий:

- показатель нестабильности ≥ «Порог показателя для авто-карантина»;
- доля падений ≥ «Доля падений для пометки нестабильным, %» **и** предыдущее состояние уже было «Нестабильный».

Автотесты в состоянии «Сломан» в автоматический карантин не попадают никогда. Если карантин сняли вручную и после этого автотест ни разу не падал, автоматика не ставит карантин повторно.

Автоматически карантин снимается единственным путём: переходом в «Исправлен» по дням без падений.

Что карантин меняет

- Автотест в карантине (как и в состоянии «Отключён») не участвует в оценке Quality Gate: его падения не учитываются и не роняют пайплайн. Достаточно карантина на ветке прогона или на всех ветках.
- Автотест продолжает выполняться в каждом прогоне: карантин не отключает запуск, он снимает только блокировку Quality Gate.

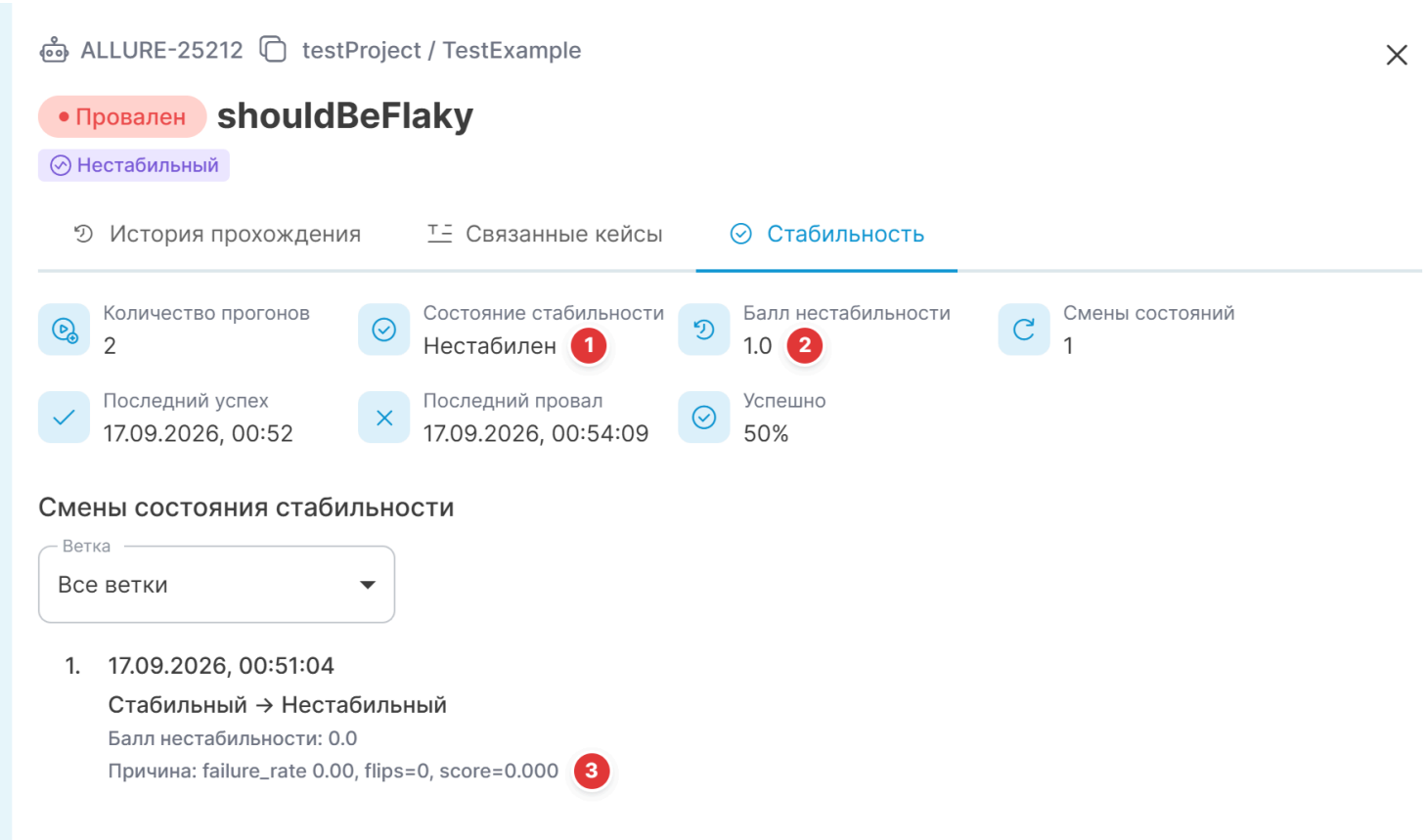
- Из статистики стабильности карантин **не** исключает: окно, доля падений и «Балл нестабильности» считаются как обычно.

Почему автотест стал нестабильным

1. Откройте панель автотеста в каталоге и перейдите на вкладку «Стабильность».
2. В секции «Смены состояния стабильности» посмотрите переходы: дата и время, «состояние → состояние», «Балл нестабильности» и, если есть, «Причина».

Выпадающий список «Ветка» в этой секции переключает историю между «Все ветки» и конкретной веткой.

Плитка «Состояние стабильности» показывает упрощённо: «Стабилен» — для состояний «Стабильный» и «Исправлен», «Нестабилен» — для всех остальных. Полное название состояния видно в списке переходов и в плашке в шапке панели. Текст в строке «Причина» формируется автоматически и не переводится, например: `auto-quarantine: score 0.610 ≥ 0.500`.



На скриншоте: 1 — плитка «Состояние стабильности», 2 — плитка «Балл нестабильности», 3 — строка перехода с «Причина» в списке «Смены состояния стабильности».

Ответственный и уведомления

1. Откройте панель автотеста в каталоге — вкладка «История прохождения».
2. В правой колонке выберите пользователя в селекторе ответственного: пока никто не выбран, в нём стоит «Не назначен».

Выбор сохраняется сразу. При ошибке значение возвращается прежнее и появляется всплывающее уведомление «Не удалось изменить ответственного». Ответственный назначается автотесту целиком, всем веткам сразу; отдельного ответственного за нестабильность или за отдельную ветку в интерфейсе нет.

Уведомление «Автотест признан нестабильным» получает ответственный за автотест, а если он не назначен — все, у кого есть право редактировать прогоны в пространстве. Уведомление уходит только при первом за всё время переходе автотеста в «Нестабильный» или «В карантине»; повторные переходы не уведомляют. Отправку выключает тумблер «Уведомлять AQA-лидов о новых нестабильных тестах».

Особенности

- Флаг карантина и состояние «В карантине» — разные вещи: снятие карантина гасит флаг, а состояние тут же пересчитывается по окну и может вернуться в «Нестабильный», если доля падений всё ещё высокая.
- Изменённая политика не пересчитывает состояния задним числом: автотест получает новое состояние при следующем своём прогоне.
- Пересчёт запускается по завершении прогона и по завершении **авто-повтора**, в фоне; пересчитываются только пары «автотест + ветка», встретившиеся в этом прогоне.
- Смена ответственного не меняет состояние автотеста и не запускает пересчёт.

Смотрите также

- **Каталог автотестов** — где смотреть состояния, историю и вкладку «Стабильность» каждого автотеста
- **Кластеры ошибок** — группировка падений по сигнатуре, в том числе падений нестабильных автотестов
- **Quality Gate** — как оценка прогона учитывает карантинные и отключённые автотесты

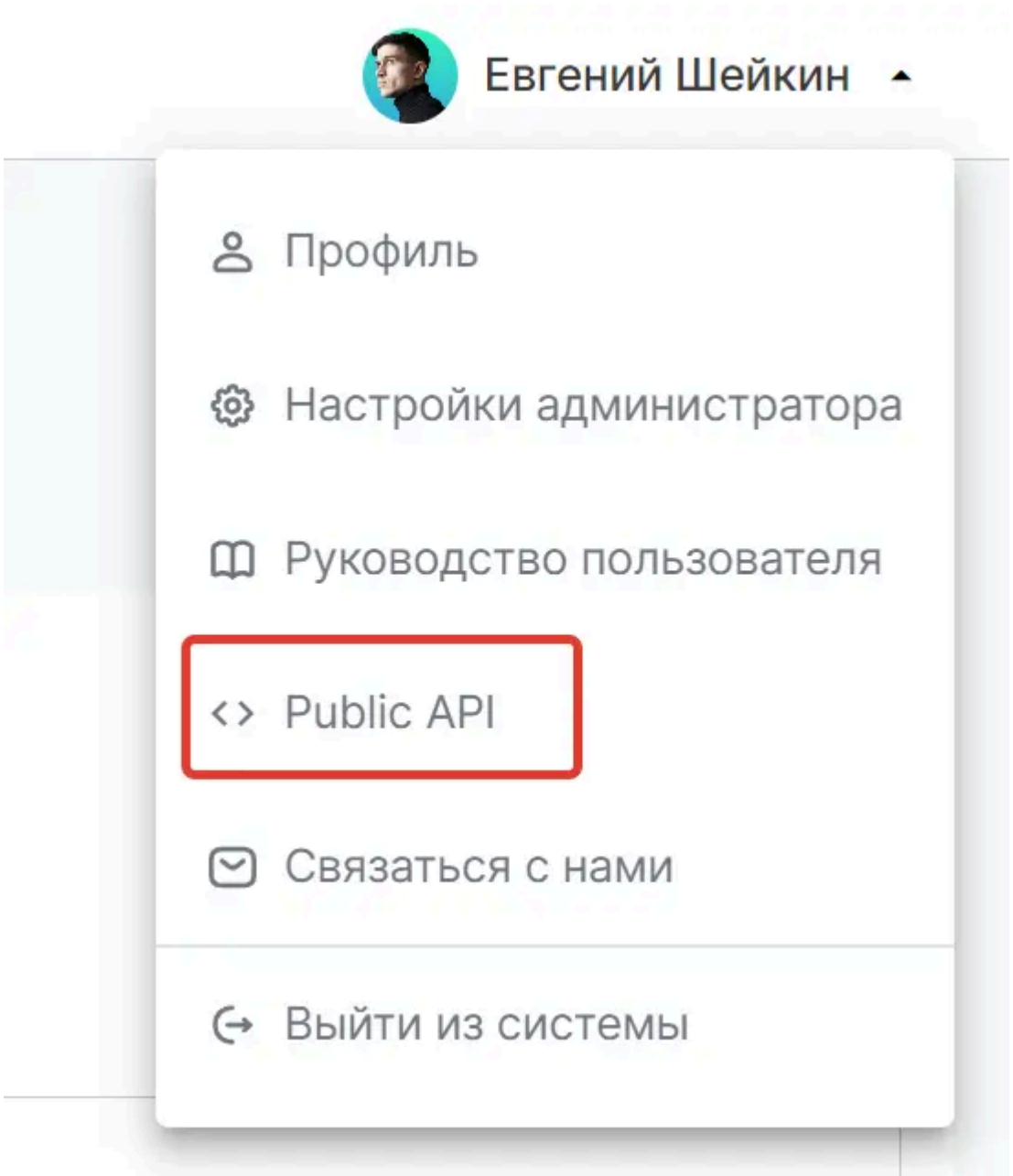
- **Авто-повтор** — повторный запуск упавших автотестов, после которого пересчитывается стабильность
- **Уведомления** — полный список событий и кто их получает

Публичный API

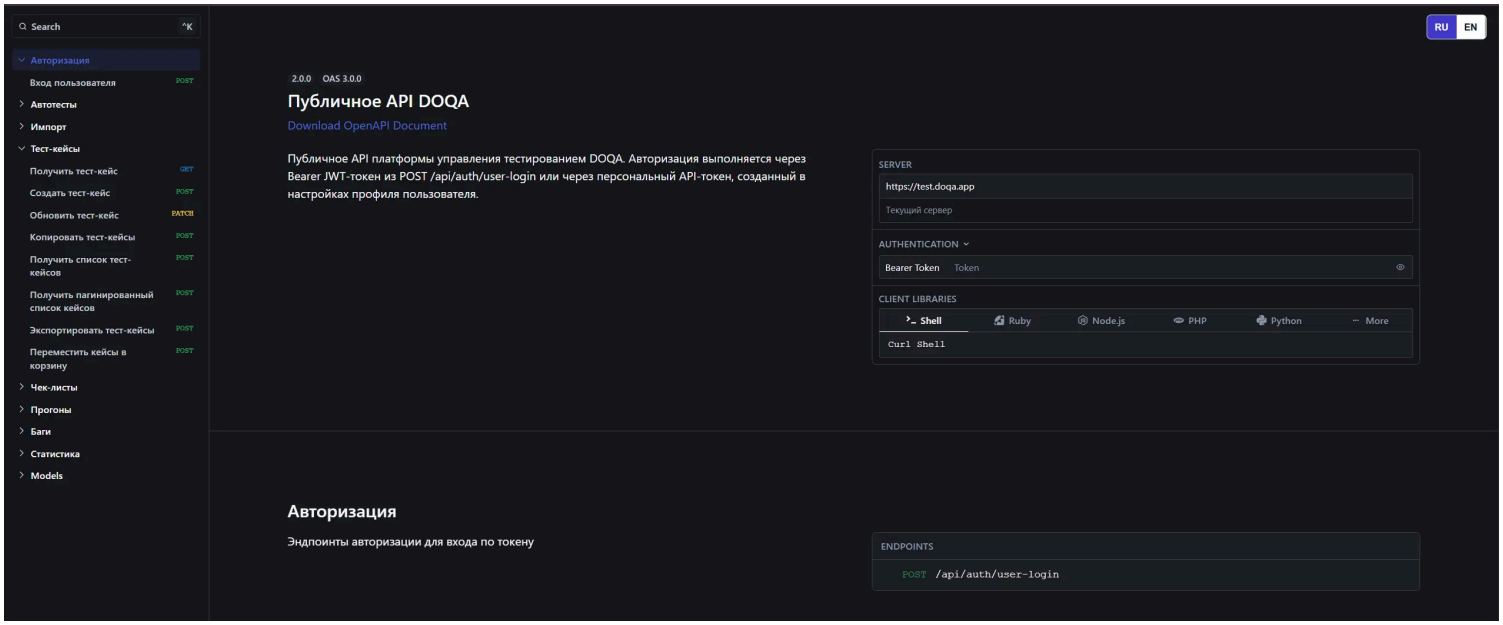
Через публичный API можно работать с тест-кейсами, чек-листами, прогонами и баг-репортами из скриптов и CI. Список методов с описанием параметров и возможностью отправить запрос прямо из браузера собран в интерактивном справочнике «Публичное API DOQA».

Доступ к справочнику

Нажмите на свое имя в правом верхнем углу и в открывшемся меню выберите пункт «**Public API**». Справочник откроется в новой вкладке.



Слева — список методов по группам, справа — блоки «SERVER» (адрес, на который уйдут запросы), «AUTHENTICATION» (токен) и «CLIENT LIBRARIES» (готовый фрагмент запроса для Shell, Ruby, Node.js, PHP, Python). Ссылка «Download OpenAPI Document» отдает спецификацию OpenAPI 3.0: ее можно передать генератору клиентов. Переключатель «RU»/«EN» в правом верхнем углу меняет язык описаний и названий групп.



Что доступно через API

Методы сгруппированы по разделам (в скобках — название группы в английской версии справочника):

Группа	Что в ней
«Авторизация» (Authorization)	получение токена по логину и паролю
«Тест-кейсы» (Test Cases)	**тест-кейсы**: создание, чтение, изменение, копирование, список, выгрузка в xlsx, удаление в корзину
«Чек-листы» (Checklists)	**чек-листы**: тот же набор операций
«Прогоны» (Runs)	**прогоны**: создание, список, детали, изменение, смена статуса, элементы прогона, комментарии, полный отчет, удаление в корзину

Группа	Что в ней
«Баги» (Bugs)	**баг-репорты** прогона и список баг-репортов пространства
«Статистика» (Statistics)	данные **дашборда**: артефакты, тренды прогонов, активность пользователей
«Импорт» (Import)	**импорт** проекта и статус импорта
«Автотесты» (Autotests)	создание прогона из отчета **автотестов**

Совет

Метод `POST /api/runs/from-autotest-report` — исключение. Он авторизуется не вашим токеном, а токеном интеграции проекта, который передается в теле запроса. Этот токен создается отдельно, см. **Создание API-токена**.

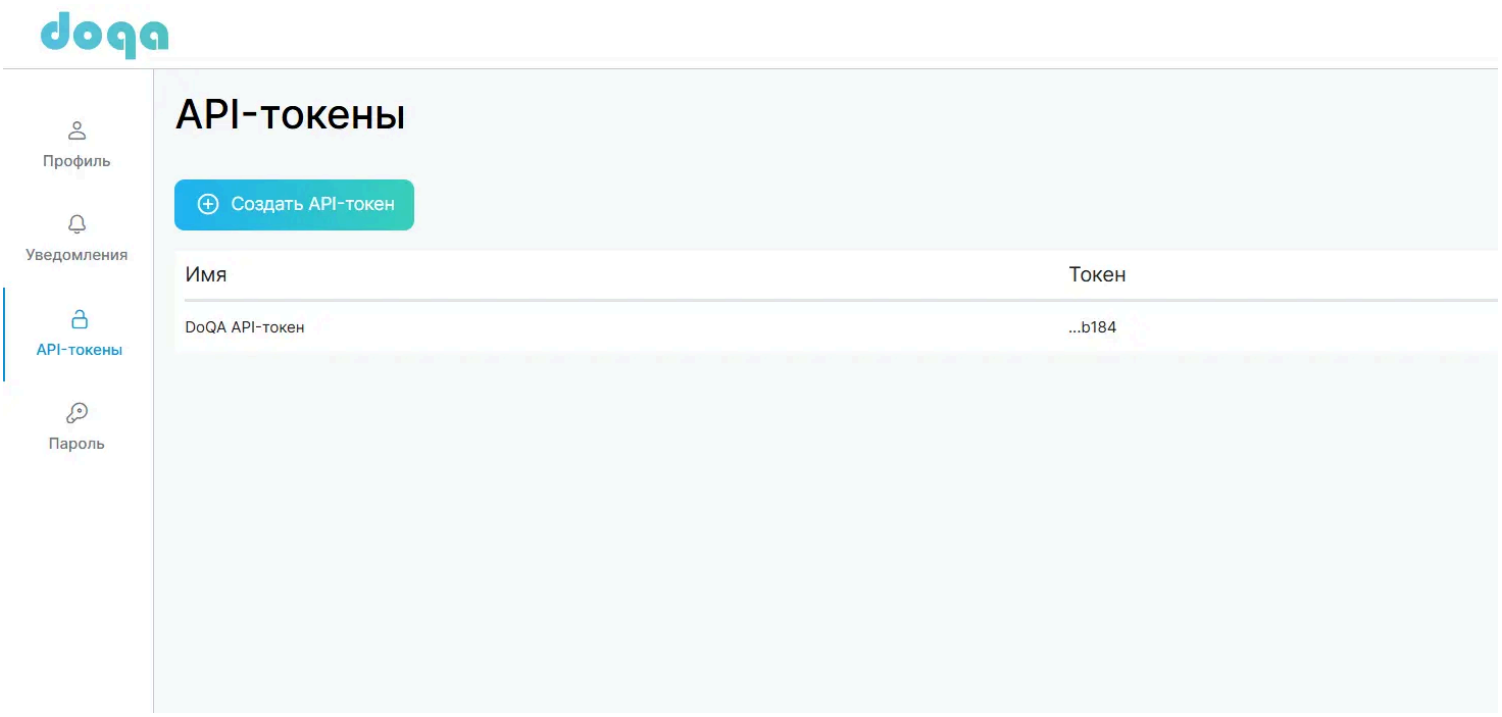
API-токены

Персональный API-токен - основной способ авторизации для скриптов и CI: он не привязан к сессии. Токены создаются в вашем профиле.

1. Нажмите на свое имя в правом верхнем углу и выберите пункт «**Профиль**».
2. Откройте раздел «API-токены» и нажмите на кнопку «**Создать API-токен**».
3. В поле «Название токена» введите название (до 255 символов).
4. В поле «Срок действия» выберите 30, 90 или 180 дней, 1 год, конкретную дату (не дальше года от сегодняшнего дня) или «Бессрочный». Бессрочный токен действует, пока вы его не отзовёте.
5. Если токену нужен не весь ваш доступ, ограничьте его:
   - «Где действует»: проекты и пространства, в которых токен будет работать. В списке только то, что видите вы сами. Пустое поле значит «везде, где у вас есть доступ»;
   - «Права»: какие из ваших прав получит токен. В списке только права, которые есть у вас. Пустое поле значит «все ваши права».
6. Нажмите «**Создать**».
7. В окне «API-токен создан» скопируйте значение токена и нажмите «**Скопировал, закрыть**».

Внимание

Значение токена показывается один раз: «Скопируйте токен сейчас. После закрытия этого окна токен больше нельзя будет просмотреть.» Если вы его не сохранили, токен придется удалить и создать новый.



В таблице токенов видно:

- «Имя» — название токена, его можно изменить;
- «Токен» — только последние 4 символа значения;
- «Действует до» - дата окончания срока или пометка «Истёк»;
- «Ограничения» - «Все ваши права» или число прав и мест, которыми ограничен токен;
- «Создан» — дата и время создания;
- «Последнее использование» — когда токеном последний раз обращались к API; прочерк, если токеном ни разу не пользовались.

Один пользователь может иметь до 10 токенов. Когда лимит исчерпан, кнопка «**Создать API-токен**» становится неактивной с подсказкой о лимите. Удалите ненужный токен, чтобы создать новый.

Токен удаляется по кнопке с корзиной, удаление нужно подтвердить. Восстановить токен нельзя: все скрипты, которые им пользовались, перестанут авторизовываться.

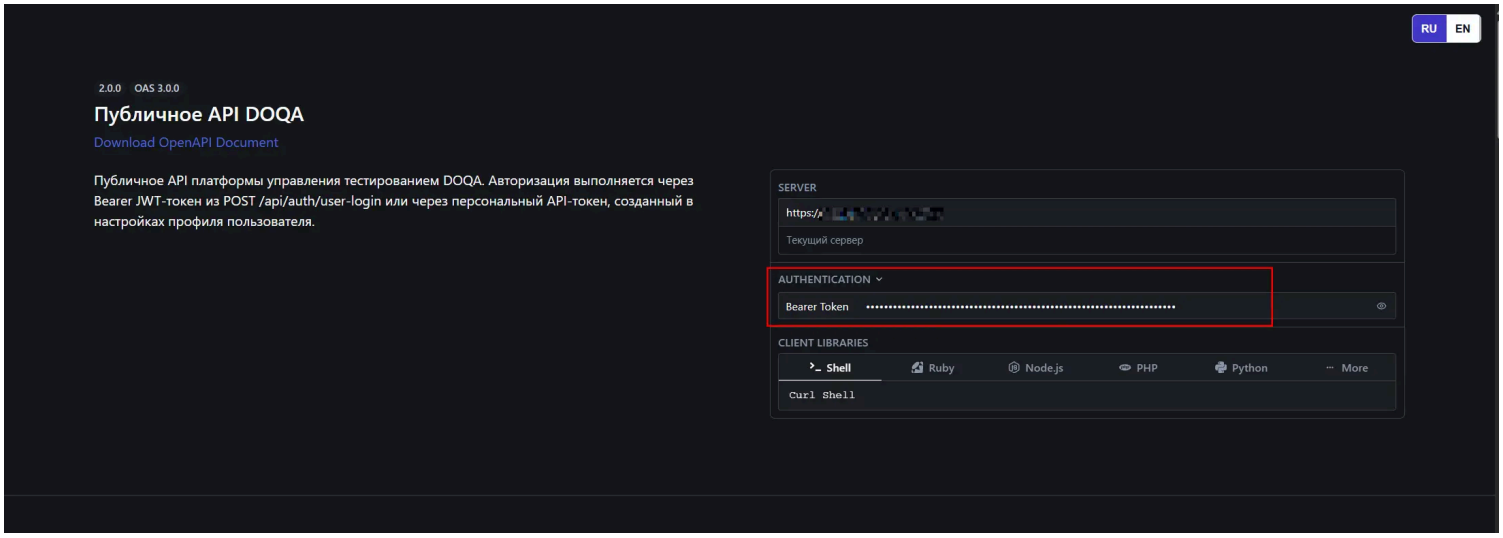
Передавайте токен в заголовке:

```
Authorization: Bearer <токен>
```

Токен действует от имени пользователя, которому он принадлежит. Он никогда не получает больше прав, чем есть у владельца: если владелец теряет доступ, токен теряет его тоже. Ограничения токена сужают эти права до выбранных мест и прав.

Авторизация в справочнике

Чтобы отправлять запросы прямо со страницы справочника, вставьте персональный API-токен в поле «Bearer Token» блока «AUTHENTICATION».



Токен сохраняется в браузере: при следующем открытии справочника вводить его заново не нужно. Чтобы перестать работать от имени этого токена, очистите поле.

Вместо персонального токена можно получить временный — по логину и паролю, методом `POST /api/auth/user-login` из группы «Авторизация». В отличие от персонального API-токена, который можно выпустить бессрочным, он всегда истекает: по умолчанию через 10 часов, в box-версии срок настраивает администратор. Когда это произойдет, получите новый тем же способом.

Сценарии

Справочные страницы отвечают на вопрос «что это». Сценарии отвечают на вопрос «как пройти путь целиком»: через несколько разделов продукта до конкретного результата.

У каждого сценария есть задача, список того, что понадобится, оценка времени и проверка результата на каждом шаге. Если вы уже знаете отдельные разделы, но не уверены, как собрать их в рабочий процесс, эти сценарии помогут.

Автоматизация

- **Первый прогон автотестов** — адаптер и GitLab CI, чтобы результаты приходили в нужный прогон.
- **Разбор упавших автотестов** — от ошибки в прогоне до баг-репорта.

Управление тестированием

- **От требования до решения о релизе** — сквозной путь: требование из трекера → тесты → прогон → баги.

Администрирование

- **Настройка корпоративного входа (SSO)** — подключение OIDC на сервере от `.env` до первого входа сотрудника.

Смотрите также

- **Быстрый старт** — если вы только начинаете работать с DoQA
- **Глоссарий** — расшифровка терминов

Первый прогон автотестов

Путь от «тесты гоняются только локально» до «результаты каждого запуска видны в DoQA в одном прогоне». Проходим его целиком: доступы → переменные в пайплайне → запуск из DoQA → просмотр результата. Отдельные шаги подробно описаны в **Автотестах**. Здесь мы собираем их в рабочий порядок.

Задача

Прогнать автотесты в GitLab CI так, чтобы результаты пришли в конкретный прогон DoQA, а не расползлись по новым, и увидеть статусы и ошибки в интерфейсе.

Решение

Запустить прогон из DoQA способом «Запуск автотестов»: DoQA сама создаёт прогон, стартует пайплайн и передаёт в него `DOQA_SPACE_ID`. Пайплайн скачивает doqa-cli, забирает у DoQA тест-план (в нём `runId` нужного прогона) и шлёт результаты через `watch`. Справочник по каждому шагу — **Автотесты**. Контракт тест-плана — **Тест-план по API**.

Что понадобится

- GitLab как CI/CD-система подключён администратором, а нужные проекты привязаны к пространству — см. **Интеграции с CI/CD**. Без этого в диалоге запуска не будет списка систем.
- Право на создание и редактирование прогонов в пространстве.
- API-токен проекта DoQA — им авторизуется отправка отчёта и запрос тест-плана. Как создать — **Создание API-токена**.
- ID пространства — **Как узнать ID пространства**.
- Тестовый фреймворк, который формирует сырые данные Allure (`allure-results`). Поддерживаемые форматы — **Поддерживаемые форматы отчетов**.
- doqactl — **Скачивание doqactl**.

Время: ≈30 мин

Шаг 1. Собрать доступы

Понадобятся две вещи, которые нужны и для отправки отчёта, и для запроса тест-плана:

- API-токен проекта. Настройки администратора → «Проекты» → нужный проект → вкладка «API-токены» → создать токен. Одним токеном можно загружать отчёты в любое пространство этого проекта. Подробно — **Создание API-токена**.
- ID пространства. Откройте пространство в браузере: в адресе `.../detail/1/2/cases` второе число — это ID. Подробно — **Как узнать ID пространства**.

> **Результат:** у вас на руках токен проекта и числовой ID пространства.

Шаг 2. Прописать переменные окружения в CI

В GitLab: Settings → CI/CD → Variables. Токен заведите с флагом Masked.

Переменная	Значение	Зачем
`DOQA_URL`	адрес вашего экземпляра, например `https://company.doqa.app`	куда слать отчёт и запрос плана (прежнее имя `DOQA_ENDPOINT` тоже читается)
`DOQA_TOKEN`	API-токен проекта из Шага 1	авторизация отправки и тест-плана
`DOQA_SPACE_ID`	ID пространства из Шага 1	в какое пространство идут результаты

`DOQA_SPACE_ID` вручную нужен только пайплайнам, которые стартуют сами, по коммиту или расписанию. При запуске из DoQA (наш случай, Шаг 4) значение приходит в пайплайн автоматически и перекрывает заданное в настройках CI. Полное описание переменных — **Переменные окружения**.

> **Результат:** в проекте CI заведены три переменные, токен замаскирован.

Шаг 3. Встроить doqa-cli в пайплайн

Добавьте в `.gitlab-ci.yml` этап, который скачивает doqa-cli, забирает тест-план и запускает тесты. Основной способ — `watch`: он отправляет результаты по мере готовности, и прогресс

виден в прогоне ещё до конца пайплайна.

watch (реальное время) report (в конце, всегда новый прогон)

yaml

```yaml
stages:
  - test

autotests:
  stage: test
  before_script:
    # Скачиваем актуальную doqa-cli и делаем исполняемой
    - curl -fsSL https://doqa.app/downloads/doqa-cli -o doqa-cli
    - chmod +x doqa-cli
    # Забираем тест-план: список Allure ID и runId прогона, куда уйдут результаты
    - >
      curl -fsSL --get "$DOQA_URL/api/integrations/ci/test-plan"
      --data-urlencode "pipeline_id=$CI_PIPELINE_ID"
      --data-urlencode "ci_project_id=$CI_PROJECT_ID"
      --data-urlencode "space_id=$DOQA_SPACE_ID"
      --data-urlencode "token=$DOQA_TOKEN"
      -o test-plan.json
    - cat test-plan.json
  script:
    # Запускаем тесты через watch — результаты уходят в прогон по мере появления файло
    # Фильтрацию по Allure ID из test-plan.json настройте под свой фреймворк и раннер.
    - ./doqa-cli watch -- mvn -B -Dallure.results.directory=allure-results test
  artifacts:
    when: always
    paths:
      - allure-results
    expire_in: 1 week
```

Что учесть:

- всё после двойного тире (`--`) — ваша исходная команда запуска тестов; сама doqa-cli её не меняет;
- если фреймворк формирует отчёты сразу в двух форматах, JUnit XML и Allure, doqa-cli берёт Allure;
- `watch` отправляет результаты по мере появления файлов: настройте фреймворк на генерацию отчётов в процессе, а не в самом конце;

- `report` — для случая, когда реальное время не нужно; он всегда создаёт новый прогон, поэтому для нашей задачи (результаты в запущенный из DoQA прогон) используйте вкладку `watch`.

Подробности по командам — **Команда watch** и **Команда upload**.

> **Результат:** в `.gitlab-ci.yml` есть этап, который скачивает doqa-cli, получает тест-план и отправляет результаты в DoQA.

Шаг 4. Запустить прогон из DoQA

Именно этот способ гарантирует, что результаты придут в один прогон.

1. На вкладке «Прогоны» нажмите «**Создать прогон**».
2. В поле «Способ создания» выберите «Запуск автотестов».
3. Укажите «CI/CD система», «Проект» и «Ветку».
4. Нажмите «**Сохранить**».

DoQA создаёт пустой прогон в статусе «В работе», стартует пайплайн и передаёт в него `DOQA_SPACE_ID`. Пайплайн запрашивает тест-план и получает `runId` этого прогона. Результаты стекаются именно в него, по мере выполнения тестов. Когда пайплайн в GitLab завершится, DoQA сама переведёт прогон в «Завершен»; вручную закрывать не нужно. Пайплайн, застрявший в pending дольше 60 минут, закрывается как неуспешный.

Подробнее о полях — **Создание прогона через запуск автотестов** и **Запуск автотестов из DoQA**.

> **Результат:** в списке появился прогон «В работе», и его таблица наполняется результатами по мере прохождения тестов.

Шаг 5. Посмотреть результат

Откройте прогон и выберите автотест в таблице. Пока результаты не пришли, карточка показывает «Результаты автотестирования все еще обрабатываются». Когда результат есть, в карточке видно:

- «Результаты» — заголовок ошибки и её описание со стек-трейсом;
- «Output» — вывод теста, если он есть в отчёте;
- «Свойства» — properties автотеста из отчёта.

Исполнителем автотеста указана «Автоматизация». По упавшему тесту прямо из карточки можно завести **баг-репорт** или оставить **комментарий**. Полное описание карточки — **Карточка результата автотеста**.

> **Результат:** вы видите статусы автотестов и тексты ошибок в том самом прогоне, который запустили из DoQA. Первый прогон настроен.

▶ **Возможные трудности**

Смотрите также

- **Автотесты** — обзор раздела: как результаты попадают в DoQA
- **Отправка результатов автотестов** — команды doqa-cli, форматы отчётов, токен и ID пространства
- **Запуск автотестов из DoQA** — способы запуска и контракт тест-плана
- **Интеграции с CI/CD** — подключение GitLab и привязка проектов к пространству
- **Прогоны** — где хранятся результаты и как проходит прогон
- **Публичный API** — токены и методы для работы из скриптов

Разбор упавших автотестов

Прогон отработал, часть автотестов упала. Этот сценарий — путь от списка упавших тестов до заведённого баг-репорта: прочитать ошибку, отделить одну общую причину от нескольких разных и связать баг с конкретным автотестом, чтобы его было видно и в прогоне, и в разделе «Баги».

Задача

Разобрать упавшие автотесты в прогоне: понять, что именно сломалось, сопоставить одинаковые ошибки между тестами и завести на них баг-репорты, не теряя связь с автотестом.

Решение

Открыть прогон, отфильтровать таблицу по статусам «Провален» и «Сломан», прочитать в карточке каждого автотеста блок «Результаты» со стек-трейсом. Одинаковый текст ошибки у нескольких тестов — общая причина: заводите один баг кнопкой «**Завести баг**» прямо из карточки. Падения CI-прогонов DoQA группирует по сигнатуре ошибки автоматически — см. вкладку **«Кластеры ошибок»**; для прогонов из загруженных отчётов сопоставляйте типовые ошибки по тексту сами. Дальше баг виден на вкладке «Баги» прогона и в разделе «Баги».

Что понадобится

- Прогон с упавшими автотестами — результаты уже пришли в DoQA. Как они туда попадают — **Автотесты**; весь путь запуска собран в сценарии **Первый прогон автотестов**.
- Право создавать баг-репорты в пространстве — редактирование или администрирование. Без него в карточке автотеста не будет кнопки «**Завести баг**».
- Если баги уходят во внешний трекер — настроенная администратором **интеграция с трекером**. Без неё баг создаётся только в DoQA.

Время: ≈20 мин

Шаг 1. Открыть прогон и отобрать упавшие тесты

Откройте прогон: нажмите на его название в таблице на вкладке «Прогоны». Тесты прогона перечислены на вкладке «Тестовый набор».

Чтобы не листать весь набор, сузьте таблицу до упавших тестов: в панели над таблицей примените фильтр по полю «Статус» и оставьте «Провален» и «Сломан». Автотест «Провален» — тест упал на проверке. Автотест «Сломан» — упал из-за ошибки в самом тесте или окружении. У всех автотестов в колонке исполнителя стоит «Автоматизация». Подробнее о панели и фильтрах таблицы — **Состав прогона** и **Фильтры**.

> **Результат:** в таблице остались только упавшие автотесты, с ними и работаем.

Шаг 2. Прочитать карточку ошибки автотеста

Нажмите на упавший автотест в таблице. Он откроется в боковой панели. Пока результаты не пришли, карточка показывает «Результаты автотестирования все еще обрабатываются». Когда результат есть, в карточке выводится:

- «Результаты» — заголовок ошибки (её тип и краткое сообщение) и описание со стек-трейсом. Каждый блок можно скопировать целиком по кнопке копирования;
- «Output» — вывод теста, если он есть в отчёте;
- «Свойства» — таблица properties автотеста из отчёта, если они были переданы.

Стек-трейс — главное, что отделяет одну причину от другой: смотрите на тип исключения и на строку, где тест упал. Полное описание карточки — **Карточка результата автотеста**.

Чтобы понять, тест упал впервые или падает регулярно, откройте вкладку «История прохождения» — она показывает прошлые попытки этого автотеста в прогоне. Стабильно падающий тест — это баг; тест, который то проходит, то падает, — скорее нестабильный (flaky), и его стоит чинить отдельно. Подробнее — **История прохождения теста в прогоне** и **Стабильность и карантин**.

> **Результат:** по каждому упавшему автотесту вы видите текст ошибки со стек-трейсом и знаете, падает он впервые или регулярно.

Шаг 3. Сопоставить типовые ошибки

Для CI-прогонов начните с вкладки **«Кластеры ошибок»**: похожие падения там уже собраны в группы с общей сигнатурой. Если кластера нет (прогон загружен отчётом или кластеризация

ещё не отработала), сопоставьте ошибки сами: пройдитесь по упавшим автотестам стрелками «‹» и «›» в шапке боковой панели и сравните блок «Результаты»:

- одинаковый тип исключения и одна и та же строка в стек-трейсе у нескольких тестов — почти наверняка одна причина (упал общий шаг, отвалился сервис, поменялся локатор). На такую группу заводят один баг-репорт;
- разные исключения или разные места падения — это разные дефекты, на каждый нужен свой баг.

Так вы отделяете «десять тестов упали от одной причины» от «десять тестов упали по десяти причинам» и не плодите дубли баг-репортов.

> **Результат:** упавшие тесты разложены на группы с общей причиной и на одиночные ошибки: понятно, сколько баг-репортов заводить.

Шаг 4. Завести баг-репорт из карточки автотеста

В карточке автотеста в блоке «Комментарии и баги» нажмите «**Завести баг**». Откроется форма баг-репорта. Часть полей уже предзаполнена данными автотеста.

Для автотеста баг привязывается к элементу прогона целиком. Завести баг на отдельный шаг, как в ручном тест-кейсе, нельзя: у автоматически пройденного элемента шагов для этого нет. В готовом баг-репорте вместо ссылки на тест-кейс выводится ссылка на автотест. Подробнее — **Баг-репорты на автотестах**.

Заполните форму:

1. В поле «**Фактический результат**» вставьте текст ошибки из блока «Результаты». Он скопирован на Шаге 2. Поле обязательно для заполнения.
2. Проверьте «**Имя дефекта**» и при необходимости поправьте, чтобы название читалось в списке багов. Поле обязательно.
3. Выставьте «Приоритет» и «Статус» (по умолчанию «Открыт»).
4. Если настроена интеграция с трекером, автоматически отмечается «**Связь с трекером**»: баг заведётся сразу и в DoQA, и в трекере. Для Jira и Битрикс24 дополнительно выберите «**Создатель**».
5. Нажмите «**Сохранить**».

Если на Шаге 3 несколько тестов упали от одной причины, заведите один баг-репорт на самом показательном из них, а в описании перечислите остальные затронутые автотесты. Так причина будет в одном месте, без дублей. Полное описание полей формы — **Создание баг-репортов**.

> **Результат:** на упавший автотест заведён баг-репорт со ссылкой на автотест; при включённой интеграции он появился и в трекере.

Шаг 5. Отслеживать баги

Заведённые баги видны в трёх местах:

- на вкладке «Баги» в плеере прогона — все баги этого прогона;
- в разделе «Баги» пространства — с фильтром по прогону, приоритету, статусу и создателю. Так вы соберёте баги конкретного прогона автотестов в один список — **Раздел «Баги»**;
- в карточке самого автотеста, в блоке «Комментарии и баги».

Статус баг-репорта проходит по значениям «Открыт» → «В работе» → «Тестирование» → «Закрыт». Если для трекера настроена обратная интеграция, статус и приоритет в DoQA обновляются автоматически вслед за трекером: вести статус вручную в двух местах не нужно. Подробнее — **Статусы и приоритеты**.

Когда причину починили, перезапустите автотест, не выходя из прогона: в его карточке «**Пройти заново**» → «Автоматически». DoQA сбросит элемент, очистит прежнюю ошибку и запустит пайплайн заново; в прогон придёт новый результат. Убедитесь, что тест стал успешным. Подробнее — **Перепрохождение одного автотеста**.

> **Результат:** баги упавших автотестов собраны и отслеживаются по статусу, а после фикса вы перепроверяете тест перезапуском прямо из прогона.

▶ **Возможные трудности**

Смотрите также

- **Результаты и разбор падений** — карточка результата, кластеры ошибок и стабильность

- **Запуск автотестов из DoQA** — как перезапустить упавший автотест
- **Прогоны** — плеер прогона, работа с автотестами и создание баг-репортов
- **Баг-репорты** — где искать и как читать заведённые баги
- **Комментарии** — обсудить причину падения прямо в карточке автотеста
- **Первый прогон автотестов** — как получить прогон с автотестами, с которого начинается этот разбор

От требования до решения о релизе

DoQA не хранит требования: они находятся во внешнем трекере, а в пространство попадают как локальные копии. Этот сценарий проходит весь цикл одного требования: от подключения трекера до ответа на вопрос «можно ли релизить». Отдельные разделы описаны на своих страницах; здесь они собраны в один рабочий процесс.

Задача

Провести требование от загрузки из трекера до решения о релизе: покрыть его тестами, прогнать, завести баги на провалах и оценить покрытие цифрами, а не на глаз.

Решение

Настроить **коннектор требований**, загрузить требование, сгенерировать по нему проверки с помощью AI, собрать из них **прогон**, завести **баг-репорты** на провалах и посмотреть метрики «Покрытие» и «Успешность» в списке требований и **матрице покрытия**.

Что понадобится

- Роль с правом редактировать тест-кейсы, чек-листы или прогоны в пространстве — этого достаточно, чтобы генерировать проверки, связывать их с требованиями и создавать прогоны. Настройка коннектора открывается на тех же правах либо на правах администратора проекта.
- Подключённая интеграция-трекер: источником требований может быть только **Jira Server** или **YouTrack**. Другие трекеры (Jira Cloud, Yandex Tracker, Bitrix, Redmine, GitLab) годятся для баг-репортов, но не для требований.
- Для AI-генерации — коины на балансе (облачная версия) или настроенный LLM-провайдер (серверная версия). См. **AI-функции**.

Время: от часа до дней (полный цикл)

В этом сценарии «проверка» — это тест-кейс или чек-лист: оба привязываются к требованию и оба считаются в покрытии.

Шаг 1. Настроить коннектор требований

Один раз на пространство. Если коннектор уже подключён, переходите к шагу 2.

Пространство → «Настройки» → вкладка «Коннектор требований». Выберите трекер и «Проект-источник», при необходимости сузьте выгрузку запросом-фильтром (JQL для Jira Server, запрос YouTrack). Затем задайте два поля трекера — «Поле для ID кейсов» (куда DoQA пишет ID связанных проверок) и «Поле статуса покрытия» — и заполните маппинг статусов. Включите «Принимать вебхуки»: этот тумблер работает как выключатель всего коннектора, без него ни загрузка, ни генерация тестов по требованию не запустятся.

Как настроить каждое поле, чем «Кастомное поле» отличается от «Системного статуса (workflow)» и как поставить вебхук на стороне трекера — в разделе **Коннектор требований**.

> **Результат:** рядом с заголовком «Коннектор требований» стоит «Подключено», а кнопка «**Обновить из трекера**» разблокирована.

Шаг 2. Загрузить требование из трекера

Откройте раздел «Требования» в пространстве и нажмите «**Обновить из трекера**». DoQA перечитает требования по фильтру коннектора, подтянет заголовки и статусы, а те, что ушли из выгрузки, архивирует в «Архивные записи».

Дальше вебхуки доставляют изменения сами: ручное обновление нужно, только когда вебхук не настроен или события были пропущены.

> **Результат:** требования появились в списке. У свежезагруженного требования, с которым ещё не связана ни одна проверка, в колонке «Покрытие» стоит «Нет тестов».

Шаг 3. Сгенерировать тесты по требованию с помощью AI

Раскройте строку требования. Откроется панель связанных проверок. Пока к требованию не привязано ни одной проверки, в ней есть кнопка «**Сгенерировать тесты с помощью ИИ**» (как только появится хотя бы одна связь, кнопка исчезает).

В диалоге выберите вид тестовой документации — «Тест-кейс», «Чек-лист» или «Набор тест-кейсов» — и папку для сохранения (новую можно создать прямо здесь). Нажмите

«**Сгенерировать**».

DoQA возьмёт текст требования из трекера — вводить контекст вручную не нужно — и создаст проверки в фоне за несколько минут. Со страницы можно уйти. Генерация расходует AI-коины (тест-кейс и чек-лист — 6, набор тест-кейсов — 30); при ошибке списание возвращается.

Подробности диалога и условия, при которых кнопки нет, — в разделе **Генерация тестов с помощью ИИ** и в описании **AI-функций**.

> **Результат:** созданные тест-кейсы или чек-листы лежат в выбранной папке и уже привязаны к требованию. Состояние покрытия ушло из «Нет тестов» в «Не прогонялись»: проверки есть, но результата прогона у них ещё нет.

Не генерировать, а привязать

Если проверки на это требование уже написаны, генерация не нужна: свяжите их вручную кнопкой «**Добавить кейс**» или «**Добавить чек-лист**» в той же панели. Способы связывания разобраны в разделе **Связывание требований с проверками**.

Шаг 4. Собрать прогон и пройти его

Нажмите «**...**» в строке требования и выберите «**Создать прогон**». В прогон попадут все связанные с требованием проверки — тест-кейсы и чек-листы одним смешанным набором, — и DoQA сразу перейдёт на страницу нового **прогона**. Пункт неактивен, пока с требованием не связана ни одна проверка.

Пройдите прогон в плеере. Проставьте каждому шагу или чеку результат: «Пройден», «Провален», «Сломан», «Заблокирован» или «Пропущен». Порядок прохождения кейсов, чек-листов и автотестов — в разделе **Прохождение прогона**.

> **Результат:** у проверок появился засчитанный результат. Требование, у которого все связанные проверки прошли, показывает «Все прошли»; если хотя бы одна провалена, сломана, заблокирована, пропущена или не прогонялась — «Не все прошли».

Шаг 5. Завести баги на провалах

На провале не выходя из прогона заведите **баг-репорт**: на шаге тест-кейса, на чеке чек-листа или на автотесте. Название можно сформировать по описанию проблемы кнопкой AI рядом с полем «Имя дефекта». Как именно завести баг-репорт — в разделе **Создание баг-репортов**.

Баг-репорт хранит ссылку на прогон и на проверку, при прохождении которой он найден. Поэтому позже его видно и в разделе «Баги», и на вкладке «Баги» самого прогона, и в редакторе тест-кейса или чек-листа.

> **Результат:** каждая проблема зафиксирована баг-репортом со ссылкой на прогон и проверку. Провал в прогоне теперь не «где-то что-то упало», а конкретный баг с автором, приоритетом и статусом.

Шаг 6. Оценить покрытие и принять решение о релизе

Теперь у требования есть и тесты, и результаты прогона: можно смотреть цифры, а не отдельные строки.

В шапке списка требований — два показателя, и считаются они по-разному:

- «Покрытие» — доля требований, у которых есть хотя бы одна связанная проверка. Отвечает на вопрос «что мы вообще проверяем».
- «Успешность» — доля требований в состоянии «Все прошли» среди **покрытых**. Отвечает на вопрос «из проверенного что прошло успешно».

Знаменатели разные, поэтому числа расходятся — это не ошибка. Формулы и разбор на примере — в разделе **Метрики: Покрытие и Успешность**.

Где именно дыры, видно в **матрице покрытия** (кнопка «**Матрица покрытия**» в шапке): строки соответствуют требованиям, столбцы — проверкам, а пустая ячейка означает, что связи нет. Выключите тумблер «Только связанные требования», чтобы непокрытые требования добавились в матрицу пустыми строками.

Решение о релизе собирается из трёх сигналов:

1. **«Покрытие» ниже целевого** — есть требования без тестов. Ищите их по чипу «Нет тестов» или в матрице с выключенным тумблером и возвращайтесь к шагу 3.
2. **«Успешность» ниже 100%** — есть проваленные проверки. Каждой должен соответствовать баг-репорт из шага 5; по ним и принимается решение «блокеры или можно жить».
3. **Есть требования в состоянии «Требуется актуализация»** — аналитик изменил требование в трекере, и старые результаты по нему больше не считаются. Это состояние перебивает все остальные, даже если тесты были успешными: пока пометка висит, требование считается непроверенным. Что с этим делать — в разделе **Актуализация**.

> **Результат:** решение о релизе опирается на две метрики и список открытых баг-репортов, а не на ощущение готовности. Каждое требование либо покрыто и успешно пройдено, либо у него есть понятная причина не быть таким.

▶ **Возможные трудности**

Смотрите также

- **Требования** — состояния покрытия, актуализация, генерация и создание прогона из требования
- **Коннектор требований** — подключение Jira Server или YouTrack, поля и вебхуки
- **Матрица покрытия** — где именно дыры в покрытии
- **Прогоны** — прохождение и завершение прогона
- **Баг-репорты** — где искать и как читать заведённые баги

Настройка корпоративного входа (SSO)

Путь администратора серверной версии: подключить корпоративный провайдер OIDC, включить автоматическое создание учётных записей, закрыть вход по паролю и убедиться, что сотрудник заходит по корпоративной кнопке. Всё делается правкой `.env` и перезапуском приложения.

Задача

Перевести вход в DoQA на корпоративный провайдер: сотрудники заходят своей рабочей учётной записью, новые пользователи заводятся сами, локальный пароль остаётся только у аварийного списка.

Решение

Настроить единственный OIDC-коннектор в `.env`, включить JIT-провижининг по списку доменов, а затем `SSO_ENFORCE` с заранее заполненным break-glass. Справочник по каждому параметру — **Корпоративный вход (SSO)**; здесь мы проходим его по порядку.

Что понадобится

- Доступ к серверу box: правка `.env` и перезапуск через `./doqa`.
- Клиент (application), заведённый в вашем IdP — Keycloak, AD FS или Entra (Azure AD): `client_id`, `client_secret` и разрешённый redirect URI.
- Список email для break-glass — учётные записи с локальным паролем, которым вход остаётся при недоступном провайдере.

Время: ≈40 мин

Шаг 1. Прописать OIDC в `.env`

Параметры `SSO_*` в поставляемый шаблон не входят: добавьте строки в `.env` вручную (см. раздел **Аутентификация**). Возьмите значения из карточки клиента в вашем IdP:

```
SSO_ENABLED=true
SSO_ISSUER=https://idp.company.ru/realms/company
SSO_CLIENT_ID=doqa
SSO_CLIENT_SECRET=<секрет клиента из IdP>
SSO_REDIRECT=https://doqa.company.ru/api/auth/sso/callback
SSO_SCOPES="openid email profile"
SSO_LABEL="Corporate SSO"
```

`SSO_ISSUER` — это базовый адрес провайдера. Остальные endpoint'ы (authorize, token, jwks) DoQA читает сама по `{issuer}/.well-known/openid-configuration`, отдельно их указывать не нужно. `SSO_REDIRECT` должен совпадать с адресом `https://<APP_URL>/api/auth/sso/callback` и быть в списке разрешённых redirect URI на стороне IdP.

Перезапустите приложение, чтобы `.env` применился:

bash

```
./doqa stop && ./doqa start
```

> **Результат:** на странице входа появилась кнопка «**Корпоративный вход (SSO)**». Форма логина и пароля пока остаётся рядом: оба способа работают одновременно. Детали и тонкая настройка кэшей/таймаутов — раздел **Включение**.

Шаг 2. Включить автоматическое создание учётных записей (JIT)

Без JIT войти смогут только те, кого вы завели заранее. Чтобы сотрудник появлялся при первом входе, добавьте:

```
SSO_JIT_ENABLED=true
SSO_JIT_MODE=domain_allowlist
SSO_JIT_ALLOWED_DOMAINS=company.ru
```

`domain_allowlist` создаёт учётную запись только для email из `SSO_JIT_ALLOWED_DOMAINS`; значение `open` пропустит любого, кого вернул провайдер. Отдельного значения

«выключено» у `SSO_JIT_MODE` нет: за это отвечает `SSO_JIT_ENABLED`. Домены перечисляются через запятую или пробел.

Роли из провайдера не читаются и не синхронизируются. Новый пользователь получает роли групп по умолчанию и групп, подходящих по правилу или по claim `groups`, остальные права выдаёт администратор внутри продукта (это Шаг 5). Заранее заведите нужные **группы**.

Перезапустите приложение после правки.

> **Результат:** сотрудник с доменом из списка, войдя через провайдера впервые, заводится сам, попадает в группы по умолчанию и без локального пароля. Порядок сопоставления и создания - **Автоматическое создание пользователей (JIT)**.

Шаг 3. Собрать break-glass и включить enforce

`SSO_ENFORCE=true` запрещает вход по локальному паролю тем, кто считается управляемым через SSO (домен из `SSO_JIT_ALLOWED_DOMAINS` или есть привязанная внешняя учётная запись), и скрывает форму логина. Автоматических исключений нет: владелец инстанса и суперадмин попадают под enforce на общих основаниях.

Поэтому сначала заполните аварийный список, и только потом включайте enforce:

```
SSO_ENFORCE=true
SSO_BREAKGLASS_EMAILS=admin@company.ru
```

Убедитесь, что у перечисленных учётных записей задан пароль. Задать его можно из консоли сервера (ввод скрытый):

```bash
docker compose exec api php artisan users:set-password admin@company.ru
```

Форма пароля под enforce скрыта, но не удалена: чтобы открыть её для break-glass, добавьте к адресу входа `?password` — `https://<APP_URL>/auth/login?password`. Вход по этой форме пройдёт только для email из `SSO_BREAKGLASS_EMAILS`.

Перезапустите приложение.

> **Результат:** SSO-управляемые пользователи заходят только через провайдера, форма пароля на обычной странице входа скрыта, а у break-glass остаётся аварийный вход. Подробно — Обязательный вход через SSO (enforce).

Шаг 4. Проверить вход

1. Откройте страницу входа. Нет кнопки «**Корпоративный вход (SSO)**» — значит `SSO_ENABLED` не применился: проверьте, что приложение перезапущено после правки `.env`.
2. Нажмите кнопку. Браузер должен уйти на страницу аутентификации провайдера.
3. Войдите рабочей учётной записью. Провайдер вернёт вас в DoQA.

Если вместо возврата страница показывает «Не удалось войти через внешнего провайдера», причину смотрите в параметре `error=sso_*` адресной строки и в логах (`docker compose logs -f api`). Расшифровка кодов и порядок диагностики — **Проверка настройки**.

> **Результат:** сотрудник вошёл через корпоративный провайдер и оказался в DoQA. Если он завёлся впервые, он в списке пользователей с ролью «просмотр».

Шаг 5. Выдать роли и лицензии

Пользователь, созданный через JIT, места в лицензии не занимает и прав на редактирование не имеет. Дальше всё делается внутри продукта, в разделе «Пользователи» администрирования:

- Проверьте, что пользователю хватило лицензии: лицензии распределяются сами, при нехватке пользователь работает только на просмотр. См. **Добавление пользователей** и **Лицензии (серверная версия)**.
- Выдайте доступ группам или людям: **Доступ к проектам и пространствам**. Для групп из SSO сопоставьте claim `groups` с группой DoQA, см. **Группы**.

> **Результат:** сотрудник входит по корпоративной кнопке, имеет лицензию и нужные права. Инсталляция переведена на корпоративный вход.

▶ **Возможные трудности**

Смотрите также

- **Корпоративный вход (SSO)** — полный справочник параметров и кодов ошибок
- **Описание параметров среды (.env)** — где и как править `.env`
- **Добавление пользователей** — приглашения, роли и лицензии внутри продукта

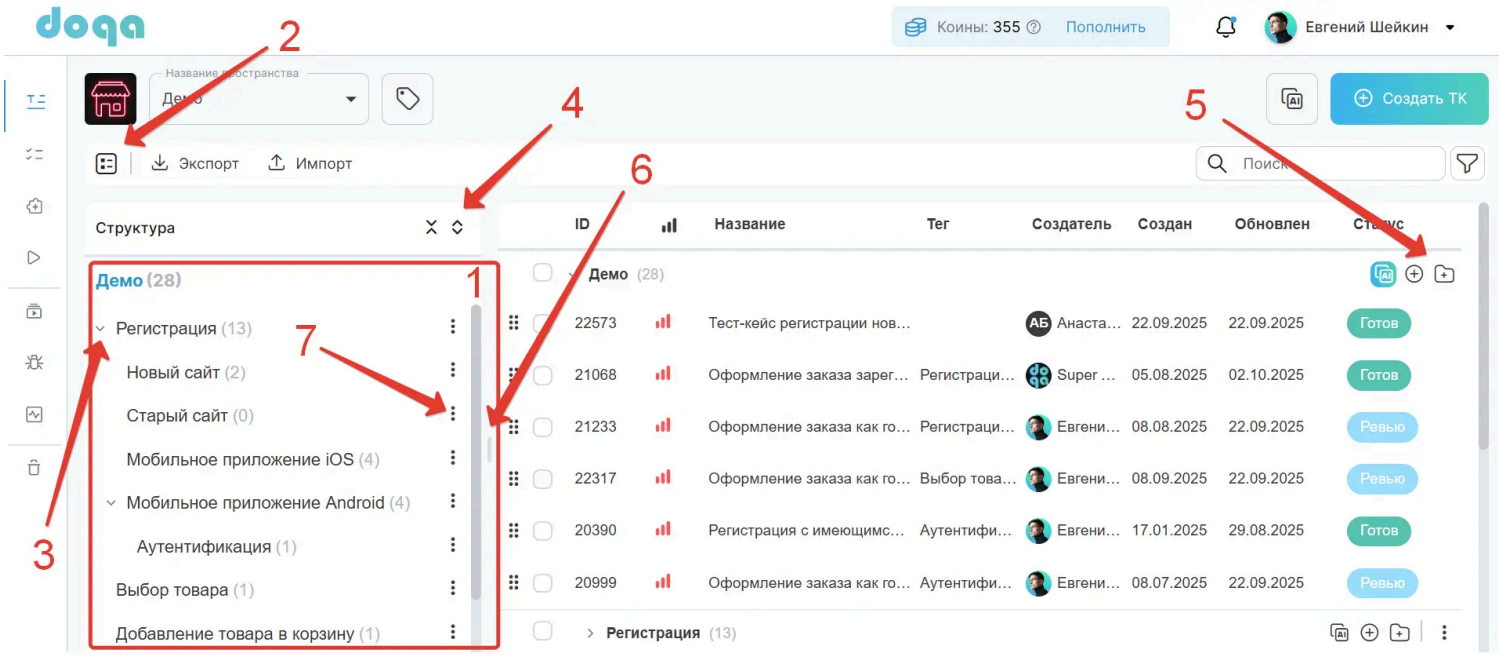
Папки

Папки организуют хранение тестовой документации внутри пространства. Они есть во вкладках «Кейсы», «Чек-листы» и «Прогоны»; у каждой вкладки своё дерево папок.

Структура папок

Слева от таблицы находится панель «Структура» — дерево папок текущей вкладки.

1. Сверху расположена корневая папка пространства. Клик по ней показывает в таблице всё содержимое вкладки.
2. Рядом с названием каждой папки в скобках указано количество элементов в ней.
3. У папок с вложенными папками слева отображается стрелка. Она сворачивает и разворачивает ветку.
4. В шапке панели две кнопки: «Свернуть всё» и «Развернуть всё».
5. Ширину панели можно изменить, перетащив её правую границу.
6. Всю панель можно скрыть кнопкой «Скрыть структуру папок» над таблицей.



Создание папки

Папка создаётся из таблицы, а не из панели структуры. Наведите курсор на строку папки (или на строку корневой папки). Справа появятся кнопки действий. Нажмите кнопку с иконкой папки со знаком «плюс»: новая папка будет создана внутри той, на строке которой вы нажали. Соседняя кнопка со знаком «плюс» создаёт в этой же папке элемент — тест-кейс, чек-лист или прогон.

Чтобы создать папку в корне пространства, используйте кнопку на строке корневой папки.

В открывшемся окне:

1. Введите название — от 1 до 255 символов. Сочетание «//» недопустимо.
2. Нажмите «**Сохранить**».

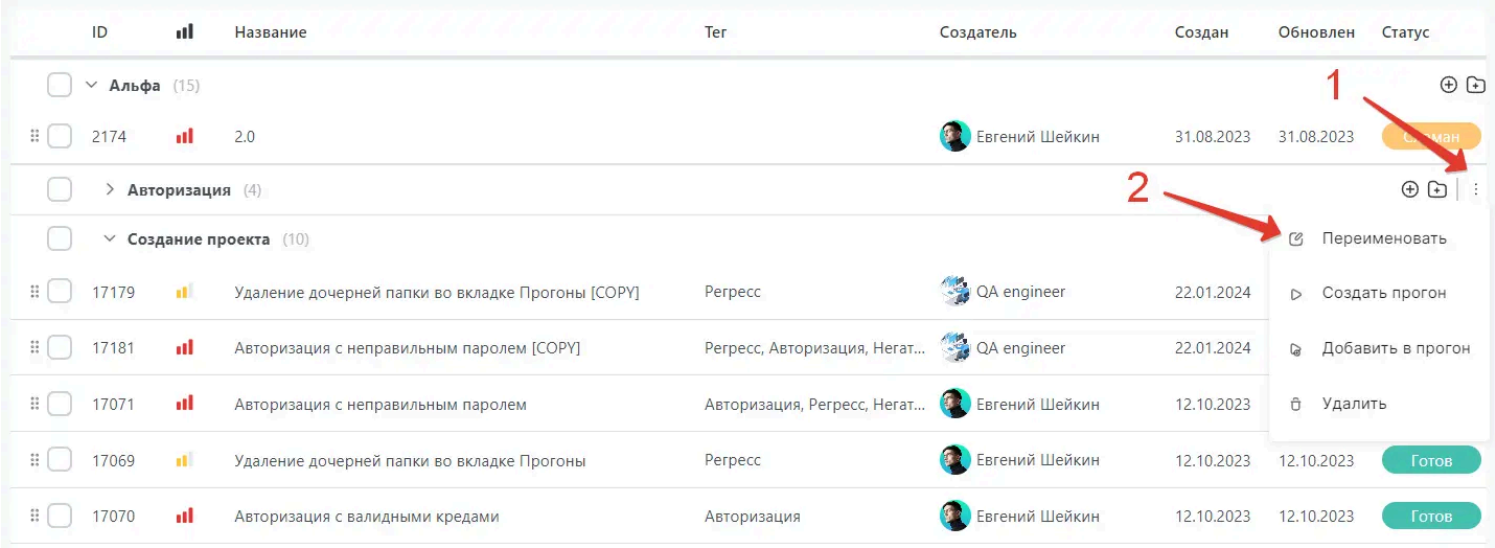
Переименование папки

Переименовать папку можно двумя способами, и работают они по-разному.

В панели структуры. Откройте меню «⋮» на строке папки и выберите «Редактировать». Название превратится в поле ввода прямо в дереве. Подтвердите изменение галочкой или клавишей Enter, отмените — крестиком.

В таблице. Откройте меню «⋮» на строке папки и выберите «Переименовать». Откроется окно с полем названия и кнопкой «**Сохранить**».

Ограничения на название те же, что и при создании: от 1 до 255 символов, без сочетания «//».



Действия с папкой в таблице

Меню «⋮» на строке папки в таблице содержит:

- «Переименовать».
- «Создать прогон» — собрать **прогон** из содержимого папки. Пункт появляется, только если папка не пуста, и не показывается во вкладке «Прогоны».
- «Добавить в прогон» — добавить содержимое папки в существующий прогон. Условия те же.
- «Удалить».

Меню и кнопки создания видны только при наличии прав на редактирование в пространстве. У строки корневой папки меню «⋮» нет: её нельзя ни переименовать, ни удалить.

Перемещение

Элементы перемещаются между папками перетаскиванием: перетащите элемент на нужную папку в панели структуры или на строку папки в таблице. Если перед этим вы отметили несколько элементов чек-боксами, перетащатся все отмеченные. Тем же способом элемент возвращается и в корневую папку пространства — перетащите его на неё.

Папки перетаскиваются в панели структуры: перетащите папку на другую папку, чтобы она стала вложенной, или на нужную позицию, чтобы изменить порядок. Перетаскивание доступно только при правах на редактирование и недоступно, пока открыто поле переименования.

Удаление папки

Откройте меню «⋮» на папке (в структуре или в таблице) и выберите «Удалить».

Папка удаляется вместе с вложенными папками и всем содержимым. Сами элементы при этом не пропадают: они попадают в **корзину** и могут быть восстановлены оттуда. Восстановленный элемент вернётся в исходную папку, а если её уже нет — в папку «Restored».

Фильтры

Фильтры доступны на вкладках «Кейсы», «Чек-листы», «Прогоны» и «Баги», в окне добавления тестов в прогон и в списке тестов внутри плеера **прогона**. Они позволяют находить нужные элементы по заданным параметрам.

Как работает фильтр

Кнопка фильтра на панели над таблицей открывает окно «Фильтр». Набор полей зависит от вкладки: на «Кейсах» и «Чек-листах» это «Статус», «Приоритет», «Создан», «Обновлен», «Создатель», «Ответственный» и «Теги», а на «Кейсах» дополнительно «Статус автоматизации».

Поля различаются по типу ввода: «Статус» и «Приоритет» — переключатели; «Создатель» и «Ответственный» — выпадающие списки; «Создан» и «Обновлен» — диапазоны дат; «Теги» — строки с деревом тегов.

Пользовательские **атрибуты** проекта выводятся в окне фильтра отдельным блоком «Атрибуты», ниже стандартных полей.

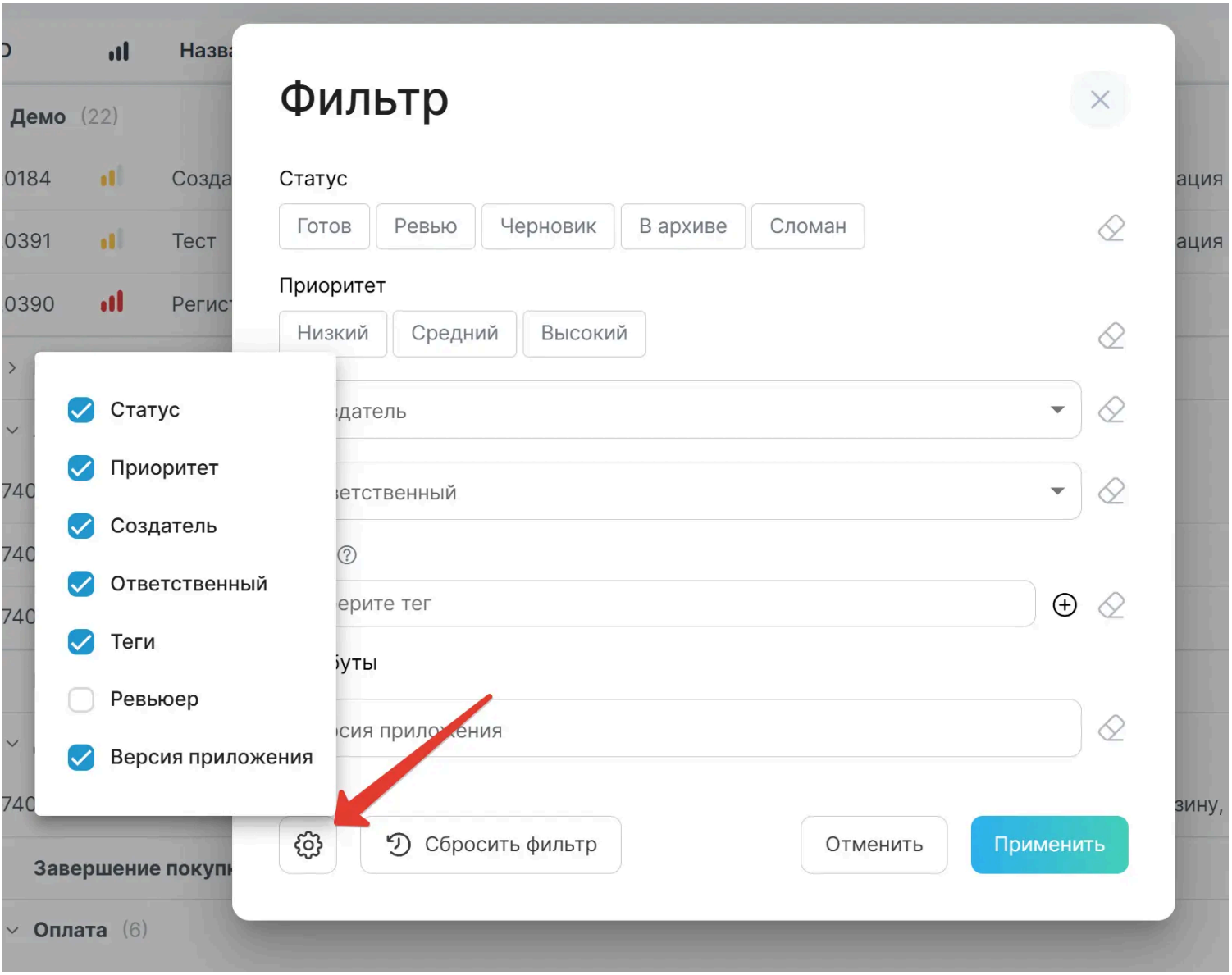
Работа с окном:

- Кнопка с ластиком справа от поля очищает только это поле.
- «**Сбросить фильтр**» очищает все поля сразу.
- «**Применить**» применяет фильтр к таблице. Пока в окне ничего не менялось, кнопка неактивна.
- «**Отменить**» закрывает окно, не меняя выборку.

Выборка меняется только по кнопке «**Применить**». Правки в полях сами по себе на таблицу не влияют.

Настройка полей фильтра

Поля, отображаемые в фильтре, можно настроить: нажмите на иконку шестерёнки в окне фильтра и в открывшемся окне отключите или подключите нужные поля.



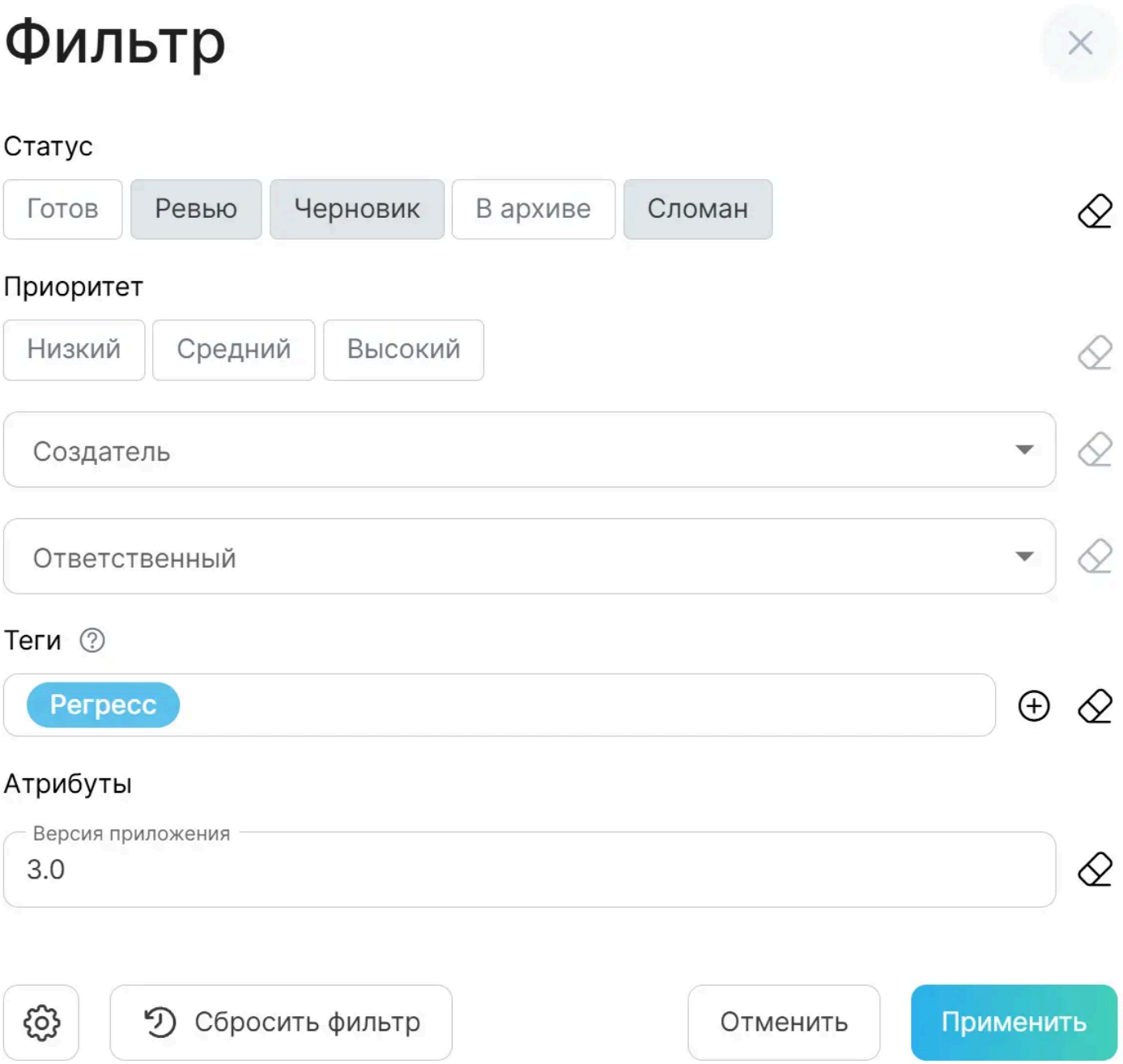
Набор полей настраивается отдельно для каждой вкладки и сохраняется при обновлении страницы. Настройка хранится в текущей вкладке браузера: при её закрытии набор полей вернётся к значению по умолчанию, на другое устройство или в другой браузер он не переносится.

Фильтр внутри прогона

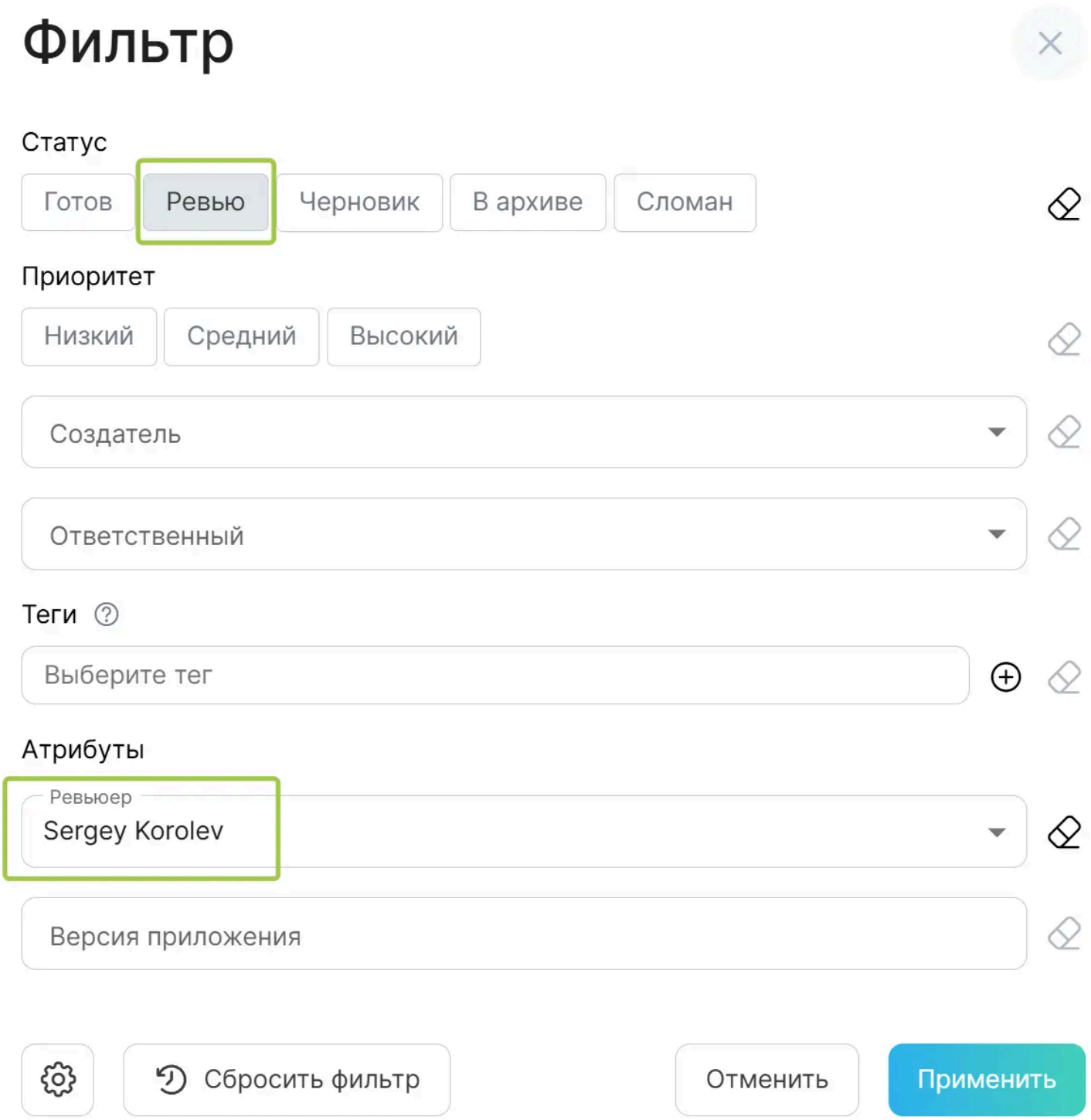
В плеере прогона у списка тестов есть собственный фильтр с полями «Путь», «Приоритет», «Статус» и «Исполнитель». Он фильтрует только список тестов открытого прогона, а не таблицу вкладки «Прогоны».

Примеры использования фильтров

К примеру, перед очередной итерацией тестирования вы можете найти тестовую документацию, которую нужно актуализировать для регрессионного тестирования определённой версии приложения:



Или найти тест-кейсы, назначенные на вас для ревью:



Смотрите также

- Теги — логика «ИЛИ» в одной строке и «И» между строками.
- Баги — фильтр на вкладке «Баги».

Глобальный поиск

Поиск по содержимому тест-кейсов, чек-листов и общих шагов сразу во всех доступных пространствах. Нужен, когда неизвестно, в каком пространстве или папке лежит элемент.

Где находится

Строка поиска — в шапке приложения, между логотипом DoQA и колокольчиком уведомлений; она есть на любой странице после входа. Страница результатов открывается только из этой строки, своего пункта в боковом меню у неё нет.

Строку поиска и страницу результатов видят все авторизованные пользователи. В выдачу попадает только то, что лежит в доступных пользователю пространствах.

При ширине окна меньше 768 px строка сворачивается в иконку-лупу: клик по ней открывает полноэкранную панель поиска с тем же поведением.

Что ищет

- **Типы: тест-кейсы**, **чек-листы**, **общие шаги**. Прогоны, баги, требования и автотесты глобальный поиск не находит.
- **Поля:** «Название», «Описание», «Предусловия», «Шаги и пункты», «Ожидаемый результат», «Постусловия», «Теги и комментарии» — по умолчанию все сразу. Фильтр «Искать в полях» сужает поиск до выбранных групп.

Подсказки в строке поиска

1. Поставьте курсор в строку поиска или нажмите Ctrl + K (на MacOS — Command + K). Сочетание работает с любой страницы.
2. Введите запрос: не меньше двух символов, не больше 100. Под строкой открывается панель подсказок.
3. Выберите подсказку кликом или стрелками и Enter. Откроется боковая панель самой сущности: совпадение подсвечено, нужный шаг раскрыт.

В панели не больше девяти подсказок, сгруппированных по типам — «Тест-кейсы», «Чек-листы», «Общие шаги», — и не больше трёх на тип. У каждой подсказки иконка типа, подсвеченное совпадение и путь «Пространство / Папка» (у общих шагов вместо пути стоит «Общие шаги»). У группы — свой счётчик и кнопка «**Смотреть все**».

Страница результатов открывается тремя способами: Enter без выбранной подсказки, кнопка «**Показать все результаты — N**» внизу панели или «**Смотреть все**» у группы (тогда на странице сразу выбрана вкладка этого типа).

Страница результатов

Заголовок страницы — «Поиск», под ним счётчик «N результатов по запросу „запрос“» или «Ничего не найдено по запросу „запрос“». Счётчик считает результаты только по выбранной вкладке.

- Вкладки «Все», «Тест-кейсы», «Чек-листы», «Общие шаги» ограничивают выдачу одним типом; у каждой свой счётчик.
- В карточке результата под названием — ряд чипов «Найдено в:», по одному на каждое поле с совпадением: «Название», «Описание», «Шаг 3», «Сабчек 1.1», «Комментарий», «Тег». Выбранный чип определяет и показанный фрагмент, и то, на каком шаге откроется сущность.
- Клик по названию карточки открывает сущность так же, как клик по подсказке.
- Запрос в поле можно править прямо на странице: выдача обновляется сама, кнопка «**Найти**» или Enter применяет запрос сразу.
- Выдача подгружается прокруткой, по 30 результатов. После возврата из открытой сущности список остаётся на том же месте.
- Запрос, вкладка, фильтры и сортировка хранятся в адресе страницы, поэтому выдачу можно переслать ссылкой.

Сузить выдачу фильтрами

1. Нажмите кнопку-воронку над списком результатов. На кнопке — число активных фильтров.
2. В окне «Фильтры» задайте нужные поля — их состав в разделе **Фильтры и сортировка**.
3. Нажмите «**Применить**». Кнопка неактивна, пока в окне ничего не изменили.

4. Заданные фильтры выводятся чипами над списком. Крестик на чипе снимает один фильтр, «**Сбросить фильтры**» рядом — все сразу; запрос, вкладка и сортировка при этом остаются.

Порядок выдачи задаёт список «Сортировка» рядом с кнопкой-воронкой.

Фильтры и сортировка

Поле	По умолчанию	Что делает
«Искать в полях»	ничего не выбрано = все 7 групп	сужает поиск до выбранных групп полей
«Создатель»	не задан	оставляет элементы, созданные выбранными пользователями; до 100 значений
«Пространство»	не задано	ограничивает выдачу выбранными пространствами; до 100 значений
«Только целое слово»	выключено	ищет слово целиком, а не как часть другого
«Учитывать регистр»	выключено	различает заглавные и строчные буквы
«Сортировка»	«По релевантности»	порядок выдачи: «По релевантности», «Сначала новые», «Сначала старые»

Особенности

- Поиск в таблице раздела устроен иначе: он ищет по названию и ID в пределах открытого пространства и папки. Глобальный поиск ищет по содержимому, включая шаги, описание, теги и комментарии, сразу во всех доступных пространствах.
- Отметить все 7 групп в «Искать в полях» — то же самое, что не отмечать ни одной: такой фильтр даже не показывается чипом.
- Шестерёнка в окне «Фильтры» прячет и показывает отдельные фильтры. Спрятанный фильтр теряет заданное значение.

- Если результатов меньше пяти и запрос похож на другое слово, под полем появляется «Возможно, вы имели в виду: ...». Клик подставляет исправление и повторяет поиск.
- После перехода на любую страницу приложения строка поиска в шапке очищается.
- Если очередная порция результатов не загрузилась, под списком появляется «Не удалось загрузить остальные результаты.» и кнопка «**Повторить**». Пока её не нажать, подгрузка при прокрутке не возобновляется.
- Пока идёт переиндексация поиска (запускает суперадминистратор), поиск недоступен.

Смотрите также

- **Фильтры** — фильтры таблиц разделов и поиск внутри пространства
- **Пространства** — доступ к пространству определяет, что попадёт в выдачу
- **Горячие клавиши** — остальные сочетания клавиш DoQA

Корзина

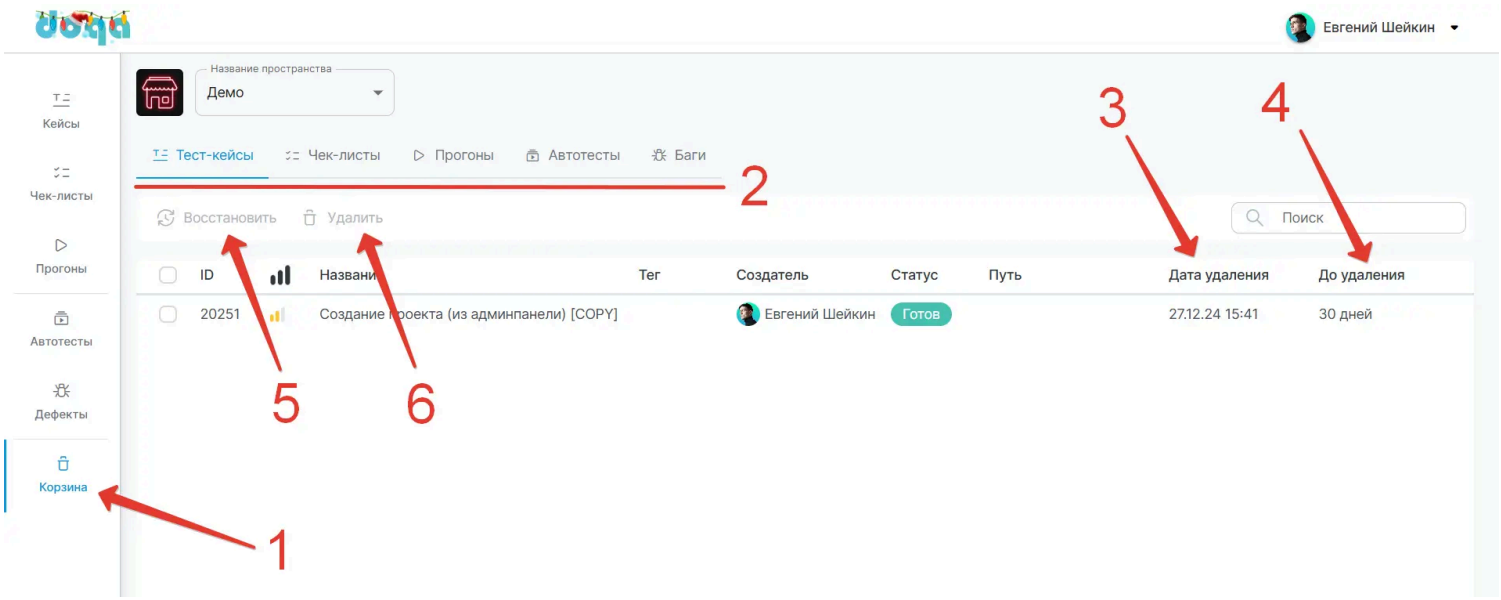
Корзина хранит удалённые объекты пространства и позволяет вернуть их обратно. Она своя у каждого **пространства**: в неё попадают только объекты этого пространства, и открывается она из бокового меню пространства пунктом «Корзина».

Что попадает в корзину

В корзине пять вкладок — по одной на каждый тип объектов, которые можно восстановить:

- «Тест-кейсы» — см. **Тест-кейсы**.
- «Чек-листы» — см. **Чек-листы**.
- «Общие шаги» — см. **Общие шаги**.
- «Прогоны» — см. **Прогоны**.
- «Баги» — см. **Баги**.

Объектов других типов в корзине нет. Удалённые проекты в корзину не попадают: для них есть отдельный архив, см. **Проекты**.



Кто имеет доступ

Корзина открывается тем, у кого есть право `bin.view` в этом пространстве: оно входит в роли «Тестировщик», «Автоматизатор», «Тест-лид» и «Администратор проектов». Восстановление доступно по правам на удаление соответствующих объектов. Без права пункт «Корзина» в боковом меню остаётся неактивным.

Кнопка «**Удалить**» - это досрочное окончательное удаление. Она показывается тем, у кого есть права на удаление навсегда (`case.purge`, `checklist.purge`, `run.purge` и аналогичные): они есть только у роли «Администратор проектов». См. **Справочник прав**.

Таблица корзины

Внутри вкладки доступны:

- Поиск по названию и ID объекта.
- Сортировка кликом по заголовку колонки.
- Подгрузка следующей порции кнопкой «**Показать еще**» внизу таблицы.

Набор колонок зависит от вкладки. Общие для всех — «ID», «Название», «Дата удаления» и «До удаления». Кроме них:

- «Тест-кейсы» и «Чек-листы»: приоритет, «Тег», «Создатель», «Статус», «Путь».
- «Общие шаги»: «Создатель».
- «Прогоны»: прогресс, «Тег», «Статус».
- «Баги»: приоритет, «Прогон», «Создатель», «Статус».

Восстановление

1. Откройте нужную вкладку.
2. Отметьте чек-боксами объекты, которые нужно вернуть.
3. Нажмите «**Восстановить**» и подтвердите действие.

Тест-кейсы и чек-листы восстанавливаются в те **папки**, в которых они были до удаления. Если папку успели удалить, объект восстановится в папку «Restored» в корне пространства. DoQA

создаёт её автоматически, если её ещё нет. Остальные объекты просто возвращаются в рабочее состояние.

Досрочное удаление

Отметьте объекты чек-боксами и нажмите «**Удалить**».

> **Внимание**
>
> Удаление из корзины необратимо: восстановить такие объекты уже нельзя. Вместе с прогоном удаляются история его прохождения, а также оставленные в нём баг-репорты и **комментарии**.

Сроки хранения

Колонка «До удаления» показывает, сколько дней осталось до окончательного удаления объекта. По истечении срока тест-кейсы, чек-листы, прогоны и баги удаляются автоматически, без предупреждения и подтверждения.

По умолчанию срок хранения — 30 дней. В коробочной поставке он задаётся переменной окружения `DAYS_UNTIL_PERMANENT_DELETION`, поэтому на вашей инсталляции может отличаться. Ориентируйтесь на колонку «До удаления». Переменные окружения описаны в **описании параметров среды**.

Дашборд

Дашборд показывает статистику пространства: сколько артефактов создано за период, сколько прогонов идет прямо сейчас, кто чем занимался и как менялись результаты прогонов. Чтобы открыть его, нажмите на иконку дашборда в левом боковом меню.

Все виджеты показывают данные одного **пространства**. Переключить его можно в поле «Название пространства» в шапке дашборда: в списке доступны пространства всех ваших проектов, сгруппированные по проектам, и есть поиск по названию.

Настройки виджетов сохраняются в браузере. При следующем заходе с этого же устройства и в этом же браузере они восстановятся; в другом браузере, на другом компьютере или после очистки данных сайта виджеты откроются с настройками по умолчанию.

Период

Виджеты «Новые артефакты», «Активность пользователей» и «Тренды результатов прогона» строятся за выбранный период. В поле «Период» доступны фиксированные значения: 1 день, 7 дней, 14 дней, 30 дней, 90 дней, 180 дней.

Последний пункт списка — «Задать период». При его выборе рядом появляется поле с календарем, где задается произвольный диапазон дат. Выбрать дату в будущем нельзя.

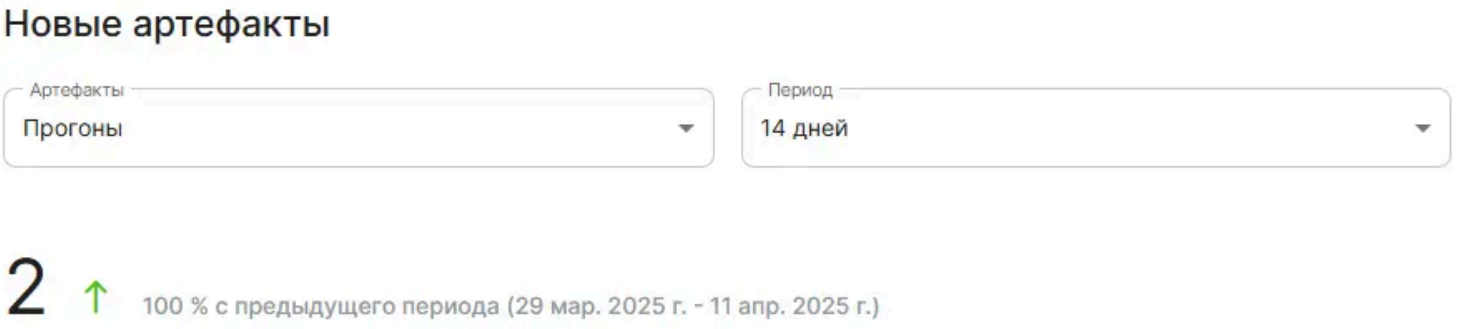
В виджете «Тренды результатов прогона» список периодов короче: значения «1 день» в нем нет, тренд строится минимум за 7 дней. Там же дата окончания диапазона должна быть строго позже даты начала. В остальных виджетах диапазон из одного дня допустим.

Новые артефакты

Виджет показывает, сколько тестовых артефактов одного типа создано за период. В поле «Артефакты» выберите тип: «Тест кейсы», «Чеклисты», «Прогоны» или «Баг-репорты».

Число элементов отображается вместе со сравнением с предыдущим аналогичным периодом. Даты периода, с которым идет сравнение, указаны рядом.

Артефакты, помещенные в **корзину**, учитываются в статистике до момента их окончательного удаления.



Активные прогоны

Виджет показывает количество незавершенных **прогонов** в пространстве (в статусе «Готов», «В работе», «Пауза»). Период здесь не выбирается: число считается на текущий момент.



Активность пользователей

Виджет сравнивает активность нескольких пользователей по одному показателю.

1. В поле «Пользователи» выберите до 20 человек, работу которых нужно сравнить (цифра 1).
2. В поле «Активность» выберите показатель (цифра 2):
   - «Результаты тестов» — сколько тестов пользователь прошел, с разбивкой по статусам. Как единица учитывается **тест-кейс** или один чек в **чек-листе**.

- «Создано кейсов» — сколько тест-кейсов создал пользователь.
- «Создано чеклистов» — сколько чек-листов создал пользователь.
- «Создано прогонов» — сколько прогонов создал пользователь.
- «Создано баг-репортов» — сколько **баг-репортов** создал пользователь.
- «Активные прогоны» — в скольких активных прогонах пользователь является исполнителем.

3. Выберите период (цифра 3).
4. Переключателем выберите единицы отображения: «Количество» или «Процент» (цифра 4).

На диаграмме отображается один выбранный показатель: увидеть несколько показателей одновременно нельзя.

Для активности «Активные прогоны» поле «Период» исчезает из виджета: этот показатель считается на текущий момент, а не за период.

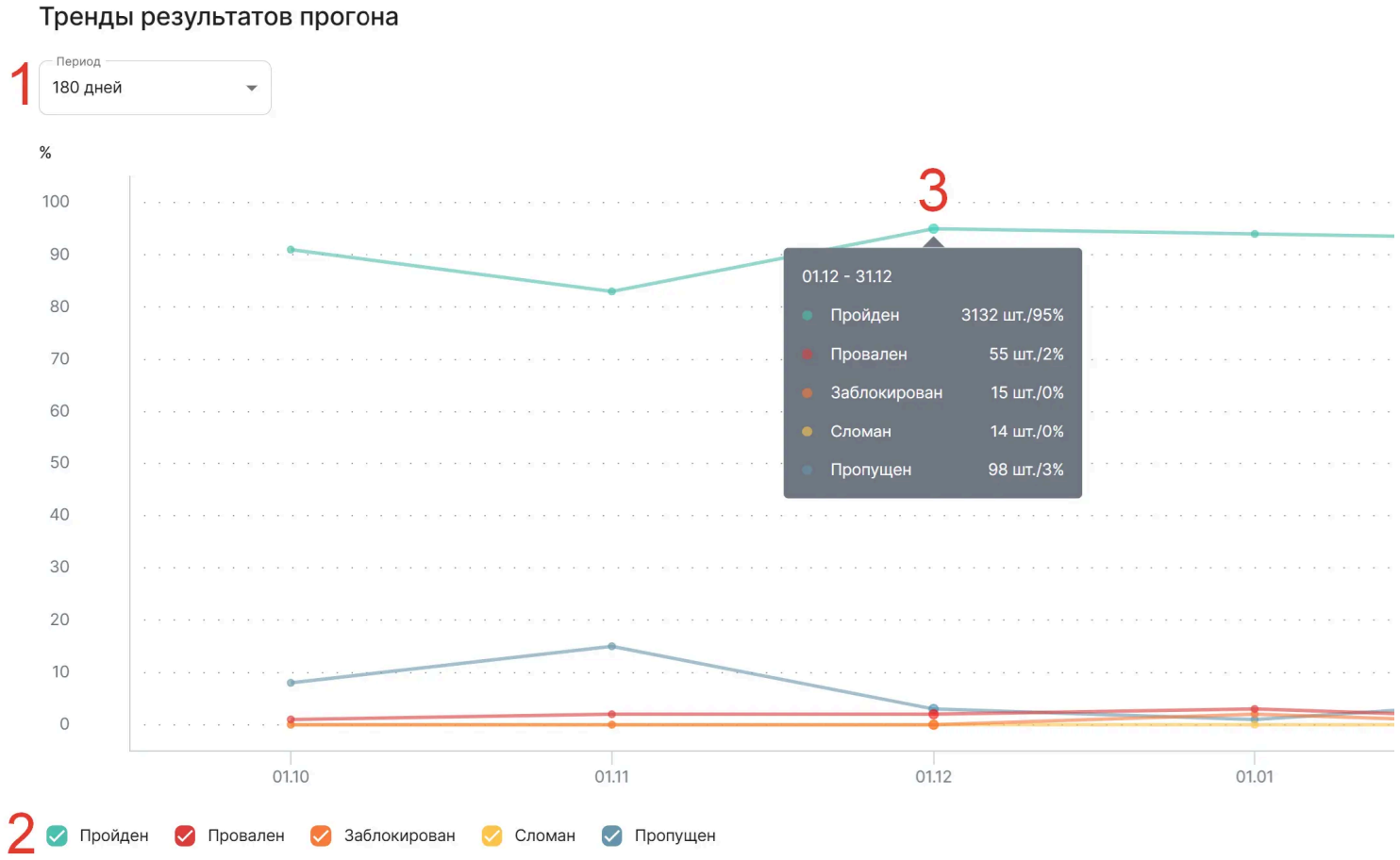
Для активности «Результаты тестов» можно скрыть или отобразить определенные статусы в диаграмме (цифра 5).



Тренды результатов прогона

Виджет показывает, как за выбранный период менялось процентное соотношение статусов пройденных тестов. На диаграмме пять статусов: «Пройден», «Пропущен», «Заблокирован», «Сломан», «Провален».

Выберите период (цифра 1): значения «1 день» здесь нет, тренд строится минимум за 7 дней. Чек-боксами под диаграммой можно скрыть или отобразить отдельные статусы (цифра 2). При наведении курсора на один из временных интервалов отображается точное количество тестов с разбивкой по статусам (цифра 3).



Экспорт

Тест-кейсы и чек-листы можно выгрузить в файл xlsx. Кнопка «**Экспорт**» находится на панели действий над таблицей в разделах «Кейсы» и «Чек-листы». Она доступна тем, у кого есть право на выгрузку (`case.export` , `checklist.export`), а кнопка «**Импорт**» - тем, у кого есть право на загрузку. Оба права входят в уровень «Полный», например в роль «Тест-лид». См. **Справочник прав**.

Шаг 1. Выберите, что выгружать

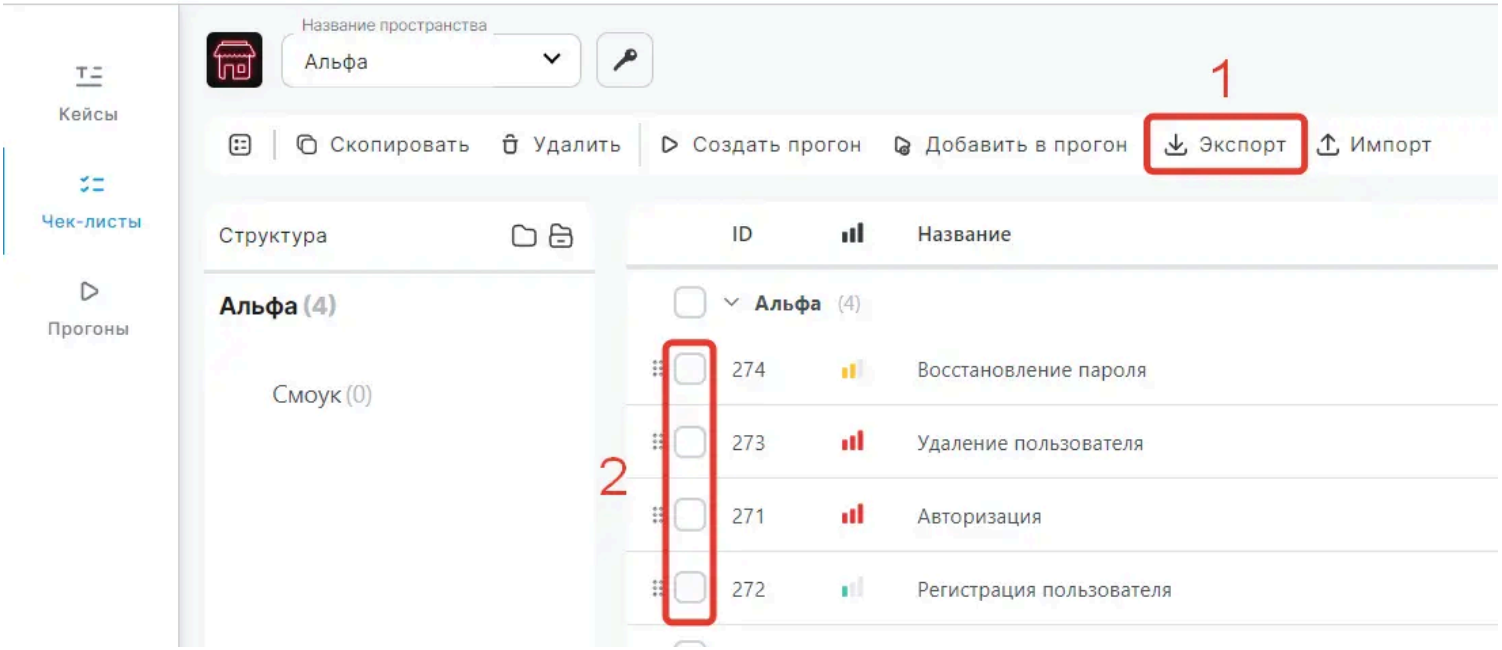
Объем выгрузки определяется тем, что отмечено в таблице:

1. Отмечены чек-боксы (цифра 2) — выгружаются только отмеченные элементы и папки.
2. Ничего не отмечено, но открыта **папка** — выгружается содержимое текущей папки.
3. Ничего не отмечено и папка не выбрана — выгружается вся вкладка.

> **Внимание**
>
> Экспорт всегда учитывает активные **фильтры** и поисковый запрос: выгрузится только то, что попало в выдачу таблицы. Чтобы выгрузить всё, сначала сбросьте фильтры и очистите поиск.

Шаг 2. Нажмите «Экспорт»

Нажмите на кнопку «**Экспорт**» (цифра 1). Начнется загрузка xlsx-файла.



Результат

Файл называется по шаблону `Проект_Пространство_дд-мм-гг.xlsx`, например `Альфа_Смоук_16-07-26.xlsx`.

Заголовки колонок в файле — на английском языке. Для тест-кейсов выгружаются колонки:

Колонка	Что в ней
Number	порядковый номер строки в выгрузке
ID	идентификатор тест-кейса
Path	путь по папкам, включая проект и пространство
Title	название
Creator	автор
Creation date	дата создания
Priority	приоритет
Status	статус

Колонка	Что в ней
Keys	теги
Description	описание
Parameters	параметры
Preconditions	предусловия
Postconditions	ожидаемый результат тест-кейса
Attachments	ссылки на вложения
Step	шаг
Expected result	ожидаемый результат шага
Test data	тестовые данные шага
Automation status	статус автоматизации
Autotest	связанный автотест
Attributes	атрибуты

Один тест-кейс занимает столько строк, сколько в нем шагов: первый шаг попадает в строку самого тест-кейса, каждый следующий — в отдельную строку, где колонки тест-кейса пустые.

Выгрузка чек-листов устроена так же: один чек-лист занимает столько строк, сколько в нем чеков. Отличаются только колонки — вместо `Preconditions`, `Postconditions`, `Step`, `Expected result`, `Test data`, `Automation status` и `Autotest` в ней две: `Check number` (номер чека) и `Check` (текст чека).

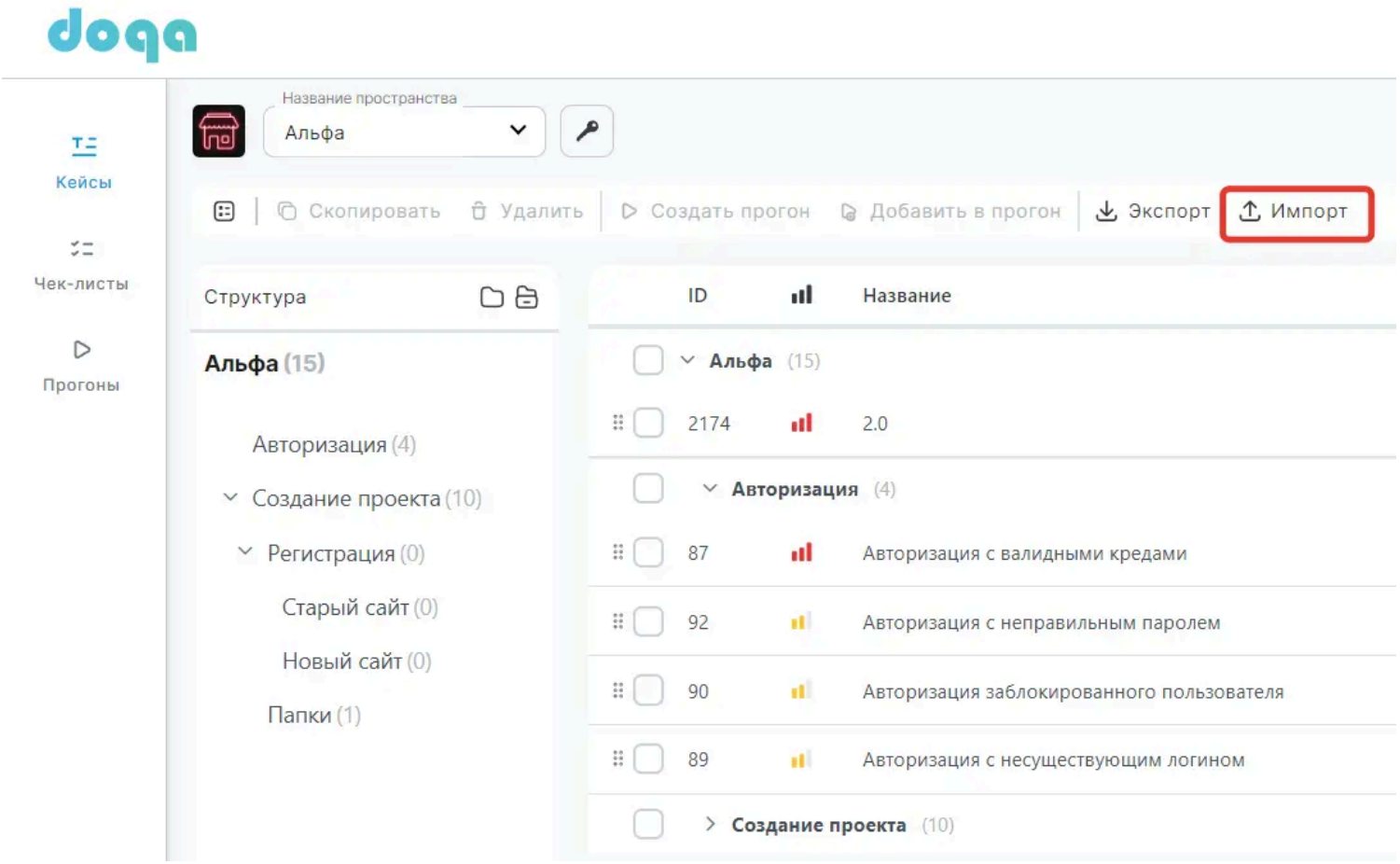
Смотрите также

Выгруженный файл можно загрузить обратно в DoQA: при **импорте** выберите источник «DoQA».

Импорт

Импорт тест-кейсов и чек-листов

Кнопка «**Импорт**» находится на панели действий над таблицей в разделах «Кейсы» и «Чек-листы». Она доступна тем, у кого есть право на загрузку (`case.import` или `checklist.import`, уровень «Полный», например роль «Тест-лид») - см. **Справочник прав**.

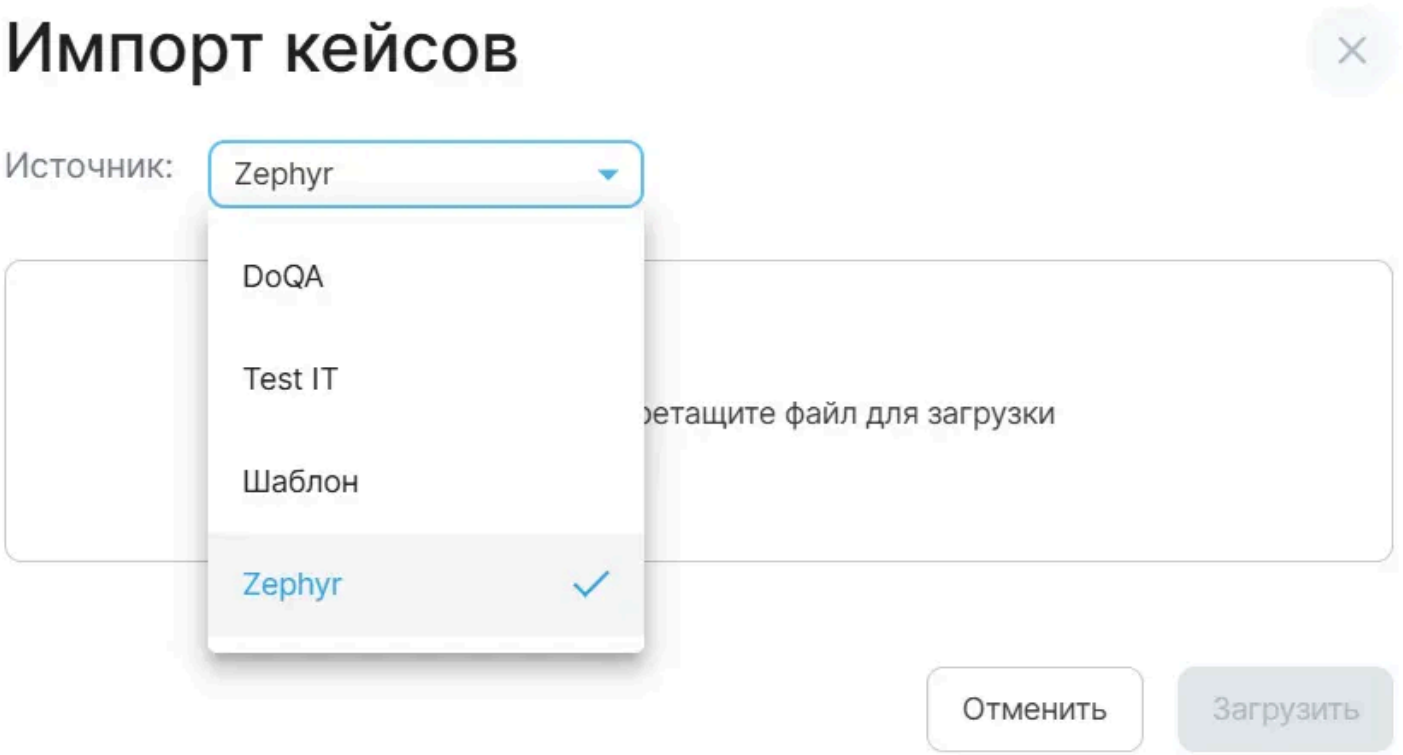


Выбор источника импорта

В окне импорта в поле «Источник:» выберите один из вариантов:

- «DoQA» (выбран по умолчанию) — файл, ранее выгруженный из DoQA через **экспорт**.
- «Test IT» — файл экспорта из TMS Test IT.
- «Шаблон» — пустой шаблон DoQA, заполненный вручную.
- «Zephyr» — файл экспорта из TMS Zephyr.

Источник «Zephyr» есть только в разделе «Кейсы»: для чек-листов доступны «DoQA», «Test IT» и «Шаблон».



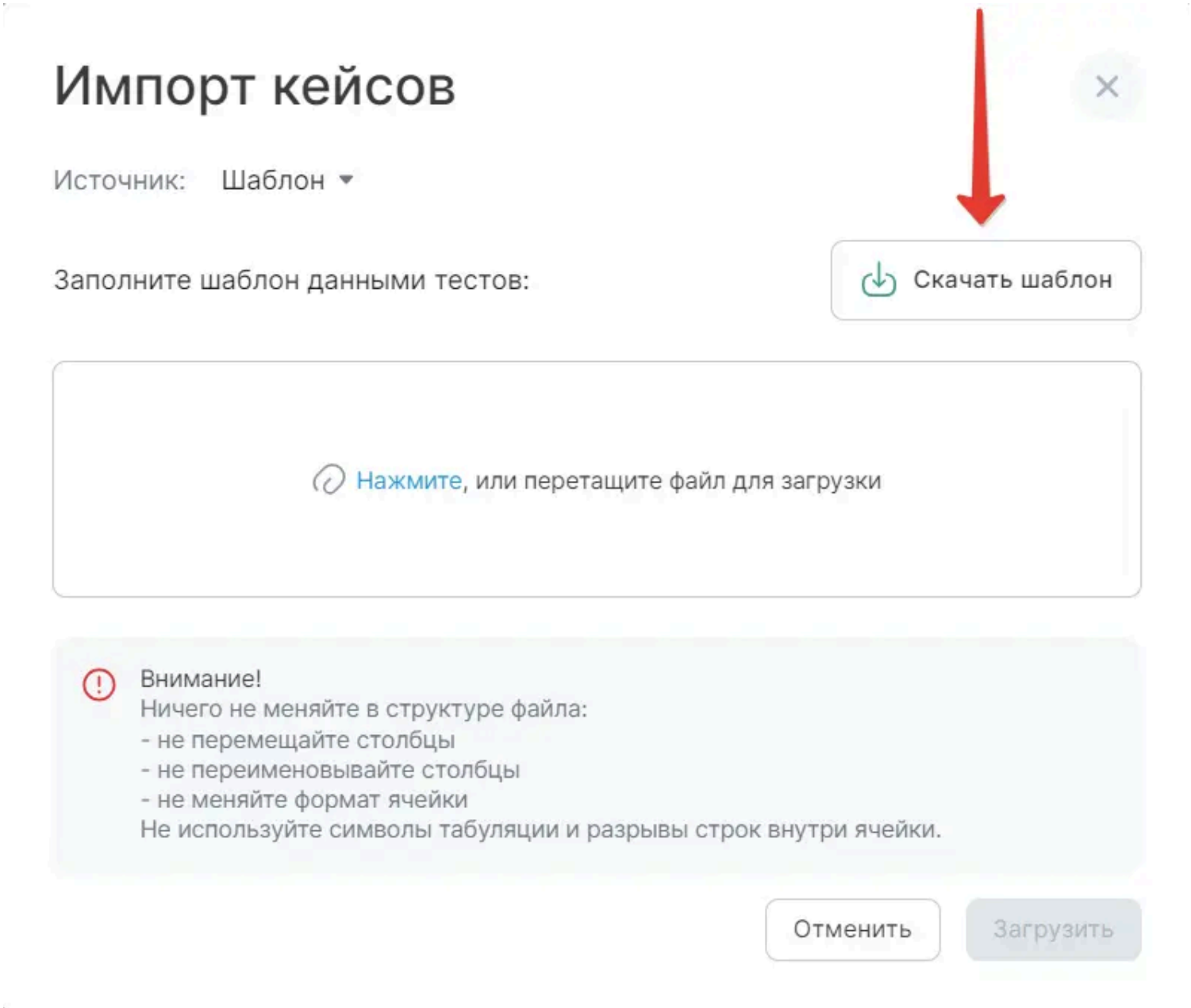
Шаблон для импорта

Выберите источник «Шаблон» и нажмите на кнопку «**Скачать шаблон**». В зависимости от раздела скачается файл `template_case.xlsx` или `template_checklist.xlsx`.

В качестве шаблона можно использовать и выгруженный из DoQA файл: «Шаблон» и «DoQA» — это один и тот же формат.

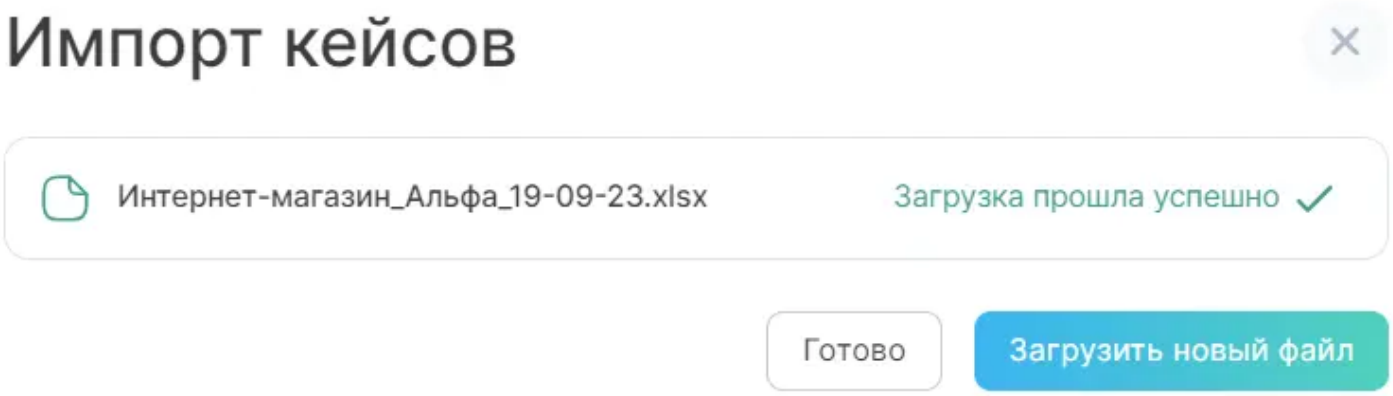
Правила заполнения выводятся в окне в блоке «Внимание!»:

- ничего не меняйте в структуре файла: не перемещайте столбцы, не переименовывайте столбцы, не меняйте формат ячейки;
- не используйте символы табуляции и разрывы строк внутри ячейки.



Процесс импорта

1. Выберите или перетащите в окно импорта файл. Принимается один файл формата .xlsx размером не больше 10 МБ.
2. Нажмите кнопку «**Загрузить**». Во время загрузки в окне отображается имя файла и прогресс-бар с процентами.



В одно пространство одновременно можно загружать только один файл. Если импорт в это пространство уже запустил другой пользователь, вместо области загрузки появится сообщение «Импорт невозможен так как в данное пространство уже загружается файл».

Что переносится при импорте

- **Папки** — по колонке `Path`. Недостающие папки создаются.
- **Теги** — по колонке `Keys`. Недостающие ключи и теги создаются.
- **Вложения** — по колонке `Attachments`, только для источников «DoQA» и «Шаблон». Из файлов Test IT и Zephyr вложения не переносятся.

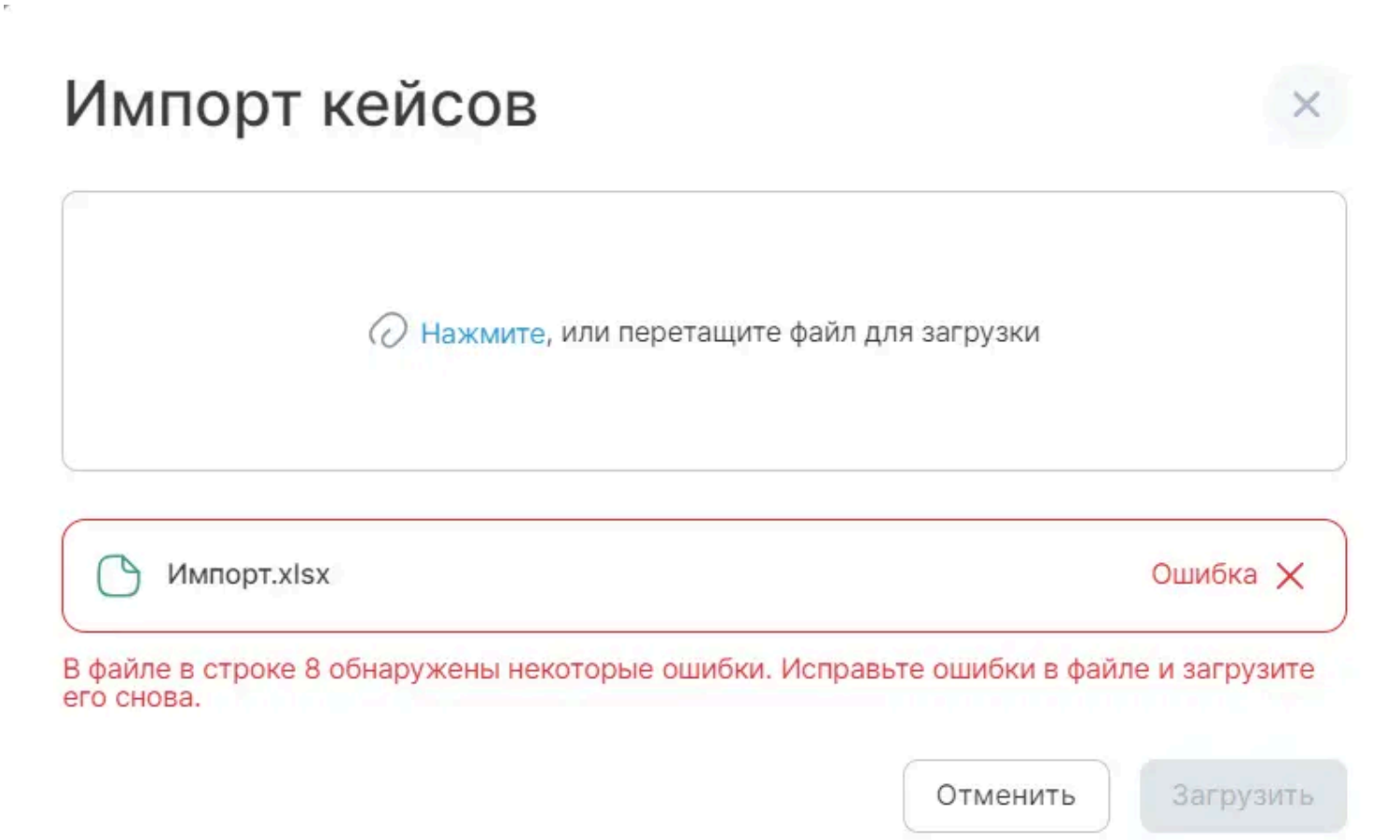
> **Совет**
>
> При импорте из Zephyr теги попадают в ключ с названием «Импорт zephyr»: ищите их в **тегах** под этим ключом.

Общие шаги при импорте тест-кейсов и чек-листов не создаются: они переносятся только при импорте проекта из Test IT.

Результаты импорта

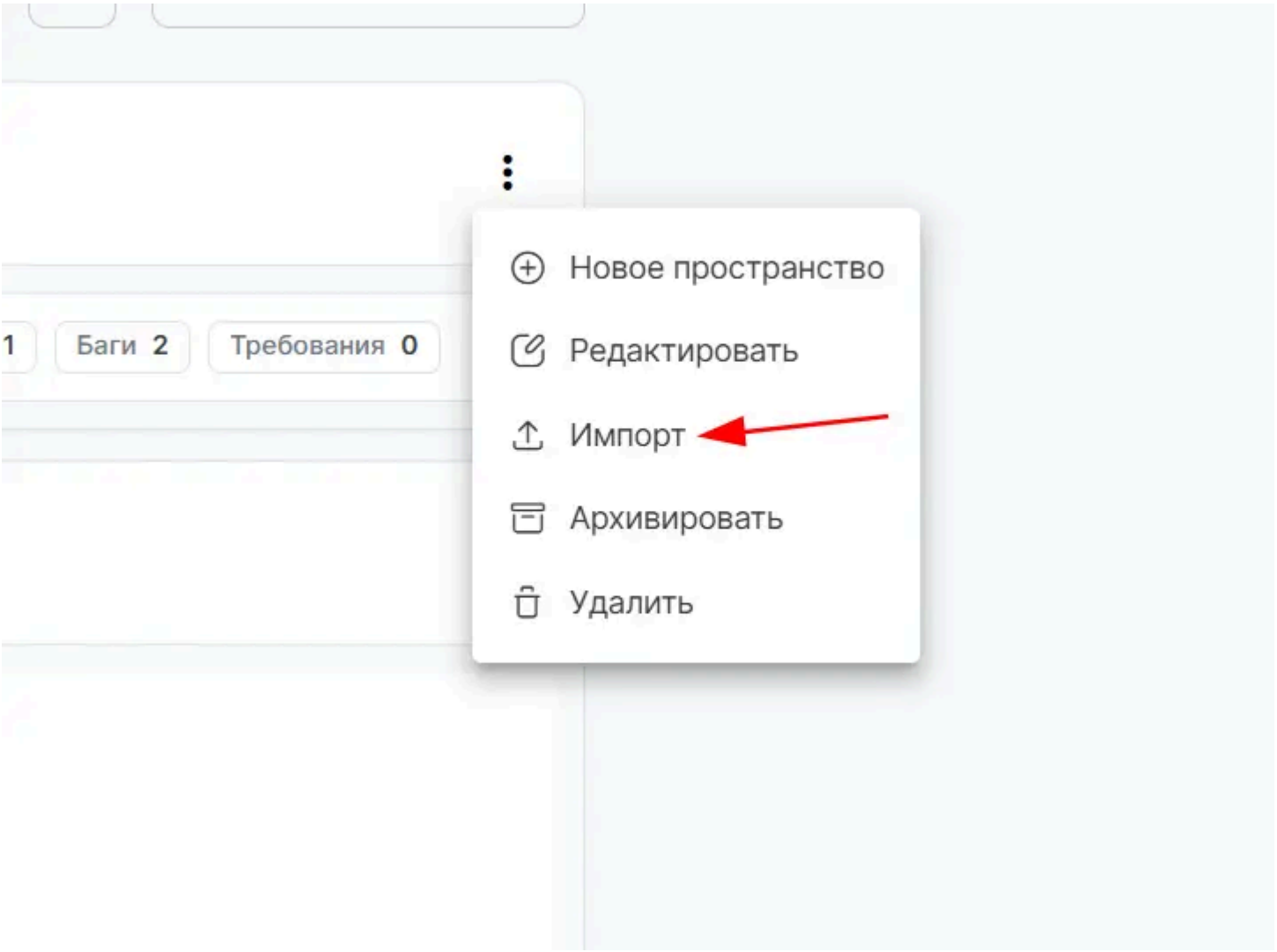
- **Импорт прошёл успешно.** У файла отображается статус «Загрузка прошла успешно». Нажмите «**Готово**», чтобы закрыть окно, или «**Загрузить новый файл**», чтобы выбрать следующий файл.
- **Файл не соответствует шаблону.** У файла отображается статус «Ошибка», а под ним — что именно не так и в какой строке, например: «В файле в строке 8 обнаружены некоторые ошибки. Исправьте ошибки в файле и загрузите его снова.» Исправьте файл и загрузите его заново.

- **Ошибка при обработке.** Если сбой произошёл уже после загрузки, во время обработки файла, откроется окно «Ошибка!» с текстом «При загрузке файла возникла ошибка, попробуйте обратиться в поддержку, либо загрузите другой файл.» и кнопками «**Перейти к форме обратной связи**» и «**Загрузить новый файл**».



Импорт проекта

DoQA импортирует проекты из TMS Qase и TMS Test IT. Перейдите на страницу «Проекты», нажмите на кнопку меню одного из проектов и выберите пункт «Импорт». Меню проекта доступно пользователям с правами на администрирование проекта.

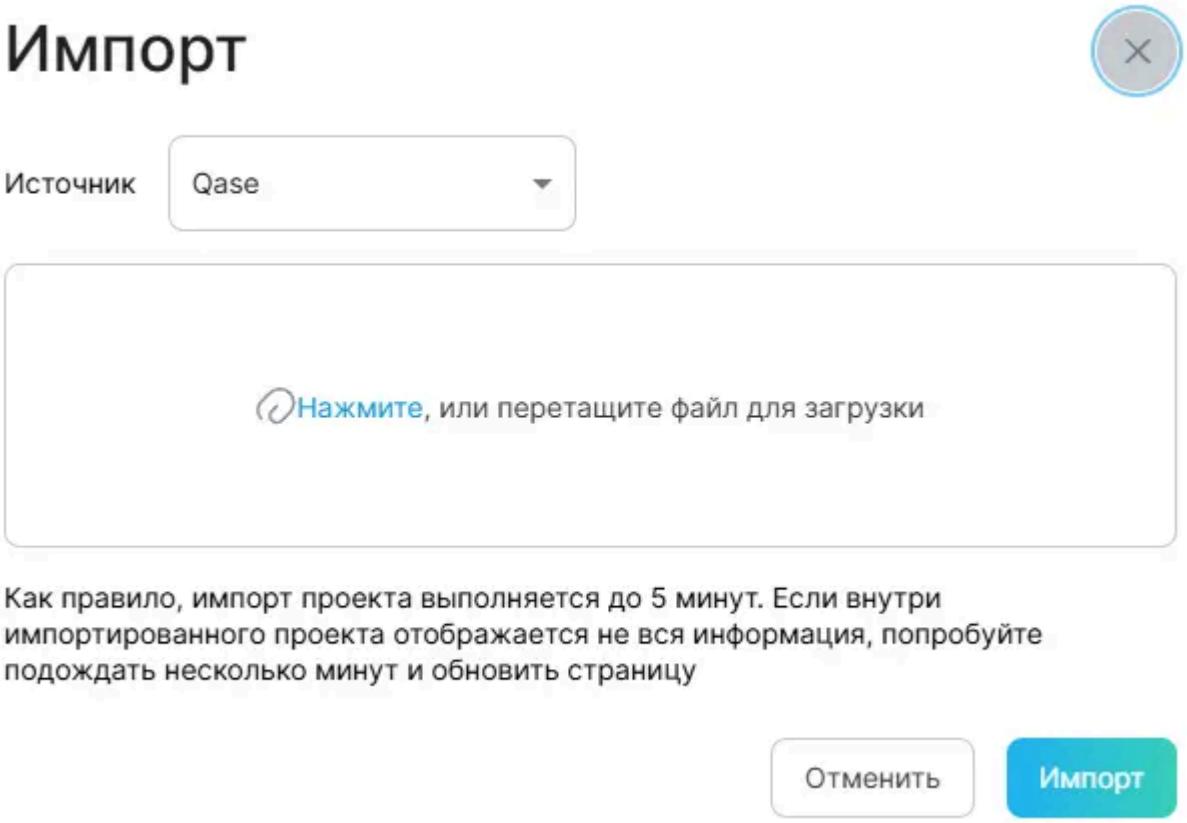


Выберите источник и файл, затем подтвердите действие:

- «Qase» — файл экспорта .json;
- «Test IT» — архив .zip.

Принимается один файл размером не больше 50 МБ.

Проект из сторонней системы переносится в DoQA как новое **рабочее пространство** в выбранном вами проекте. В зависимости от размера проекта импорт может занять до 5 минут. Если внутри импортированного проекта видны не все данные, подождите несколько минут и обновите страницу.


Импорт
Источник
Qase
Нажмите, или перетащите файл для загрузки
Как правило, импорт проекта выполняется до 5 минут. Если внутри импортированного проекта отображается не вся информация, попробуйте подождать несколько минут и обновить страницу
Отменить
Импорт

Комментарии

Комментарии можно оставлять в редакторе тест-кейса или чек-листа (чтобы обсудить содержание теста) и при прохождении теста в **прогоне**.

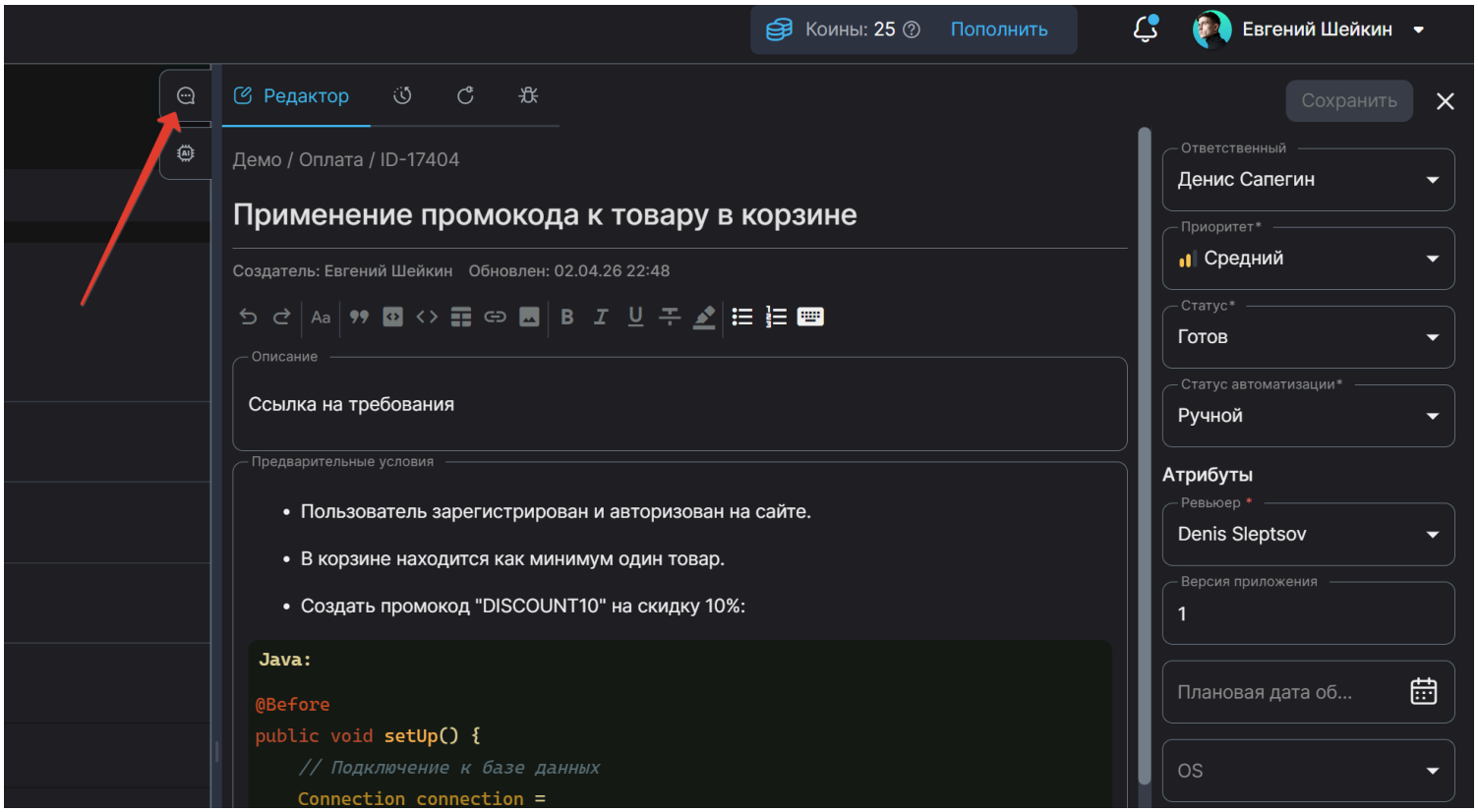
В редакторе есть две отдельные панели, их не надо путать:

- «Комментарии» — обсуждение самого тест-кейса или чек-листа.
- «Комментарии прогонов» — комментарии, оставленные при прохождении этого теста в прогонах. Здесь их только читают.

Комментарии в редакторах тест-кейсов и чек-листов

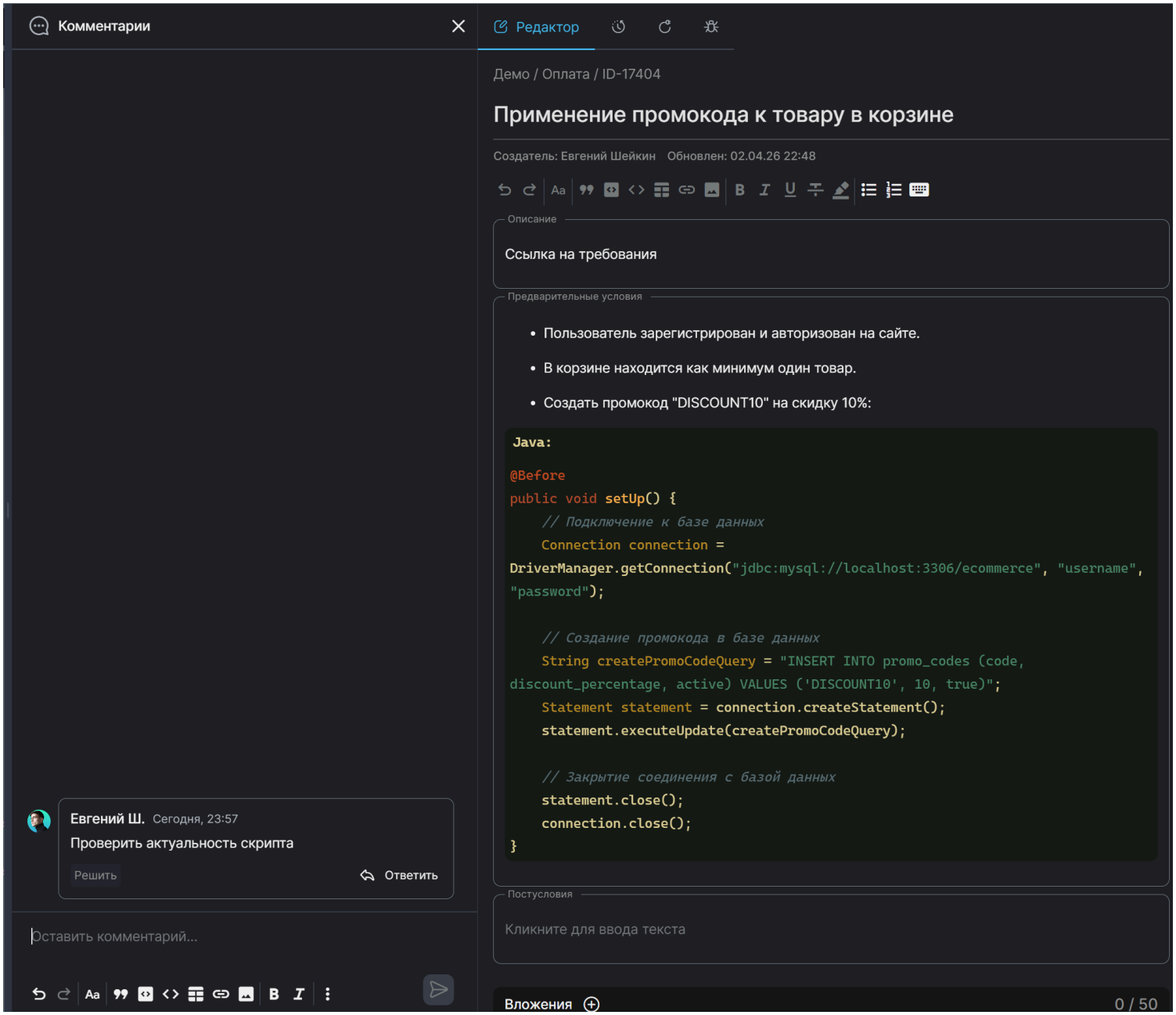
Как оставить комментарий или начать обсуждение

Панель комментариев открывается закладкой с иконкой сообщения справа от редактора тест-кейса или чек-листа. Заголовок открытой панели — «Комментарии».



Напишите текст и нажмите на кнопку отправки или на сочетание клавиш `Ctrl + Enter` (Windows) или `Command + Enter` (macOS). В комментарии доступно форматирование текста и

вставка изображений. Длина комментария — до 5000 символов.



Чтобы начать обсуждение, нажмите на кнопку «**Ответить**» под комментарием. Ответы сворачиваются и разворачиваются кнопками «**Показать ответы**» и «**Скрыть ответы**».

Ветка обсуждения одноуровневая

Ответить можно только на корневой комментарий. Если вы отвечаете на ответ, ваше сообщение попадает в ту же ветку — к корневому комментарию, а не под ответ. Вложенных веток не бывает.

Панель работает в реальном времени: комментарии и ответы коллег появляются без перезагрузки страницы. Если комментарий не отправился, он помечается ошибкой с кнопкой «**Повторить отправку**».

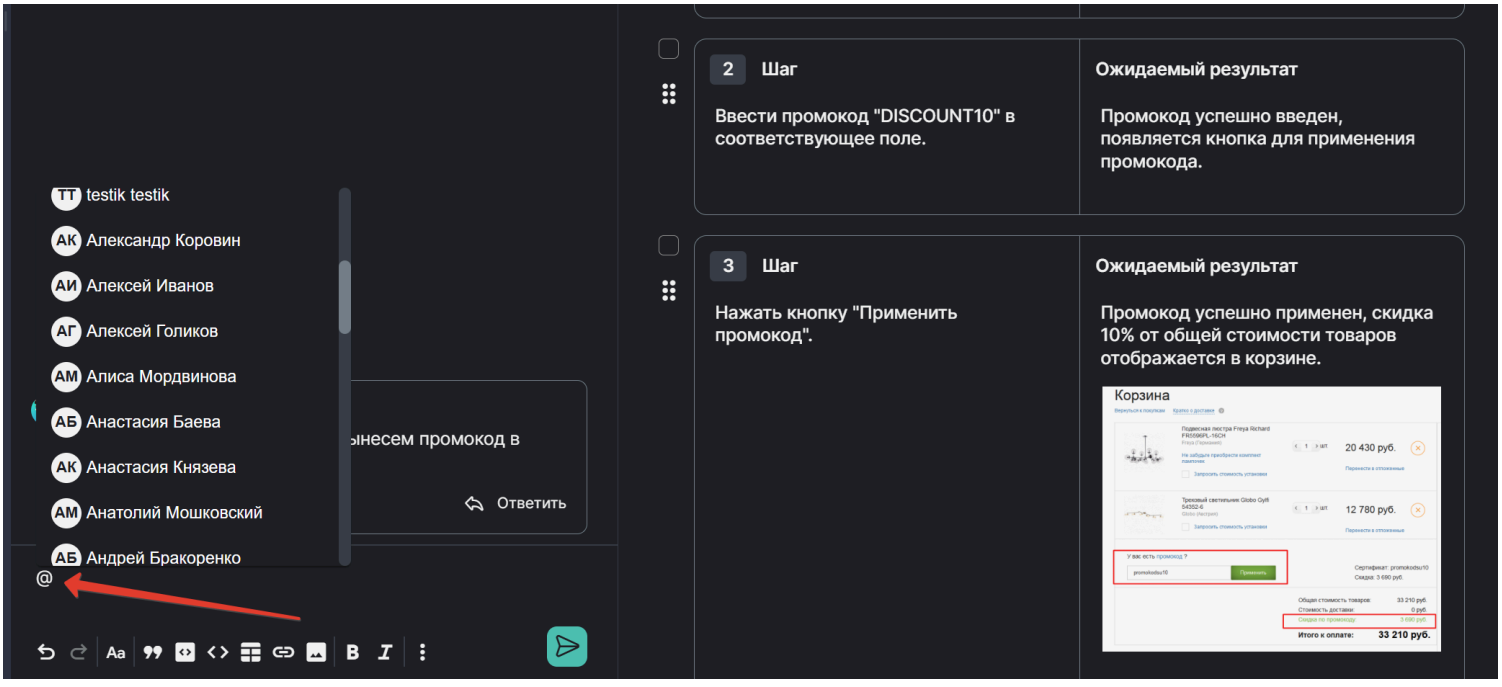
Отредактированный комментарий помечается меткой «| изменён». Редактировать и удалять свой комментарий может автор. Чужие комментарии правят и удаляют те, у кого роль даёт эти права без ограничения «только свои» (уровень «Полный», например «Тест-лид») - см. **Роли и права**.

Писать комментарии (`comment.create`) могут только пользователи с лицензией: без неё роль работает на просмотр, комментарии можно только читать. Лицензии: **облачная версия**, **серверная версия**.

Упоминания и уведомления

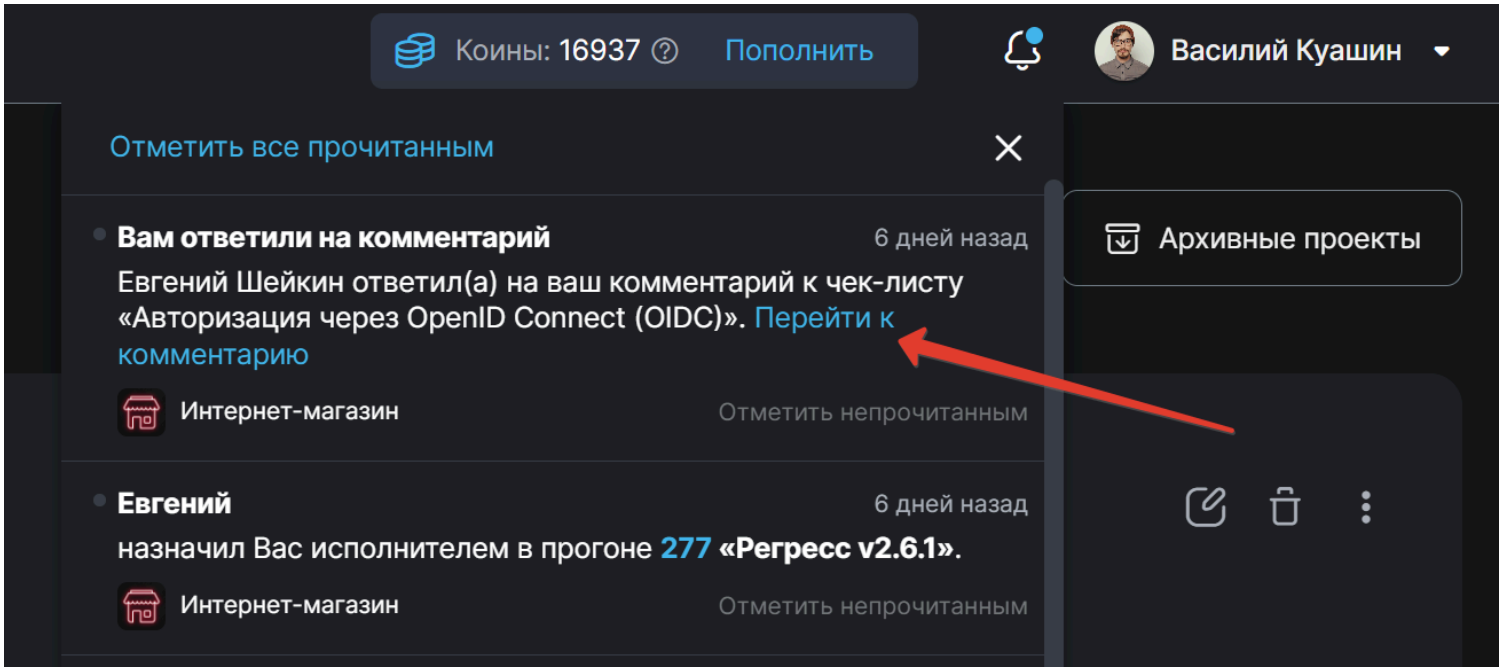
Чтобы привлечь внимание конкретного коллеги, используйте упоминания. Введите символ @ в поле ввода комментария и выберите нужного пользователя из выпадающего списка.

> **Примечание**
>
> В списке отображаются пользователи, завершившие регистрацию в системе и имеющие доступ к пространству на уровне просмотра или выше. Заблокированные пользователи в списке не выводятся.



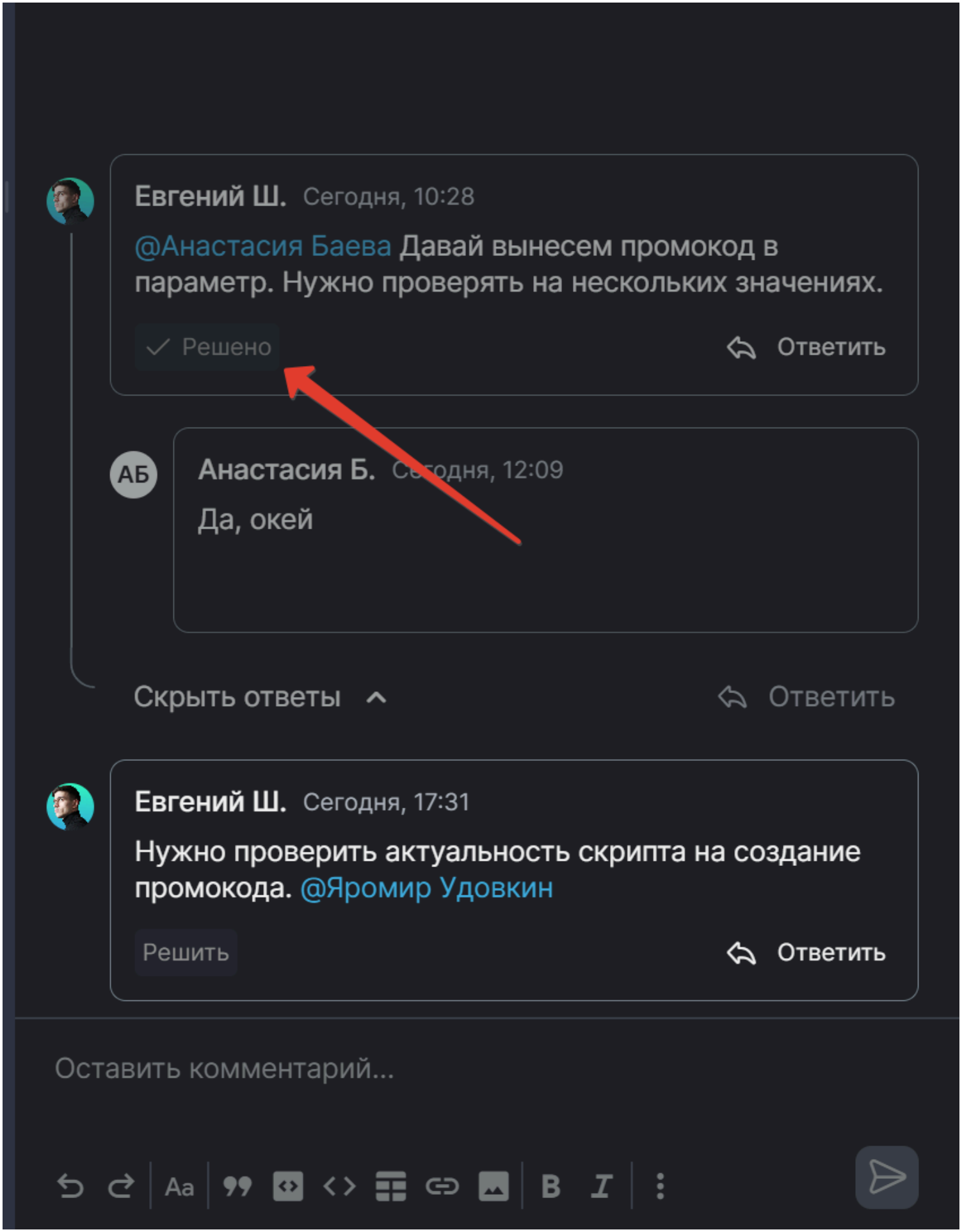
DoQA отправляет уведомление упомянутому пользователю, как только вы нажмете на кнопку отправки. Автор комментария получает уведомление, когда кто-то отвечает в его ветку обсуждения, даже если его не упомянули через @. Уведомление содержит ссылку для быстрого перехода к комментарию или к ответу на него.

Уведомления приходят по трем каналам — «DoQA», «E-mail» и «Telegram». Каждый пользователь выбирает каналы сам или отключает эти уведомления — см. **Уведомления**.



Работа со статусом комментариев

Когда обсуждение закончено, нажмите на кнопку «**Решить**» под комментарием. Комментарий получит отметку «Решено»: он останется в истории, но станет менее заметен.


Евгений Ш. Сегодня, 10:28
@Анастасия Баева Давай вынесем промокод в параметр. Нужно проверять на нескольких значениях.
Решено
Ответить
АБ
Анастасия Б. Сегодня, 12:09
Да, окей
Скрыть ответы
Ответить
Евгений Ш. Сегодня, 17:31
Нужно проверить актуальность скрипта на создание промокода. @Яромир Удовкин
Решить
Ответить
Оставить комментарий...
Aa

Отметить решенным можно только корневой комментарий ветки, но не ответ. Отметить и снять отметку может тот, у кого есть право `comment.resolve`.

Комментарии прогонов в редакторе

Панель «Комментарии прогонов» открывается второй закладкой справа от редактора. В ней собраны комментарии, оставленные при прохождении этого тест-кейса или чек-листа во всех прогонах.

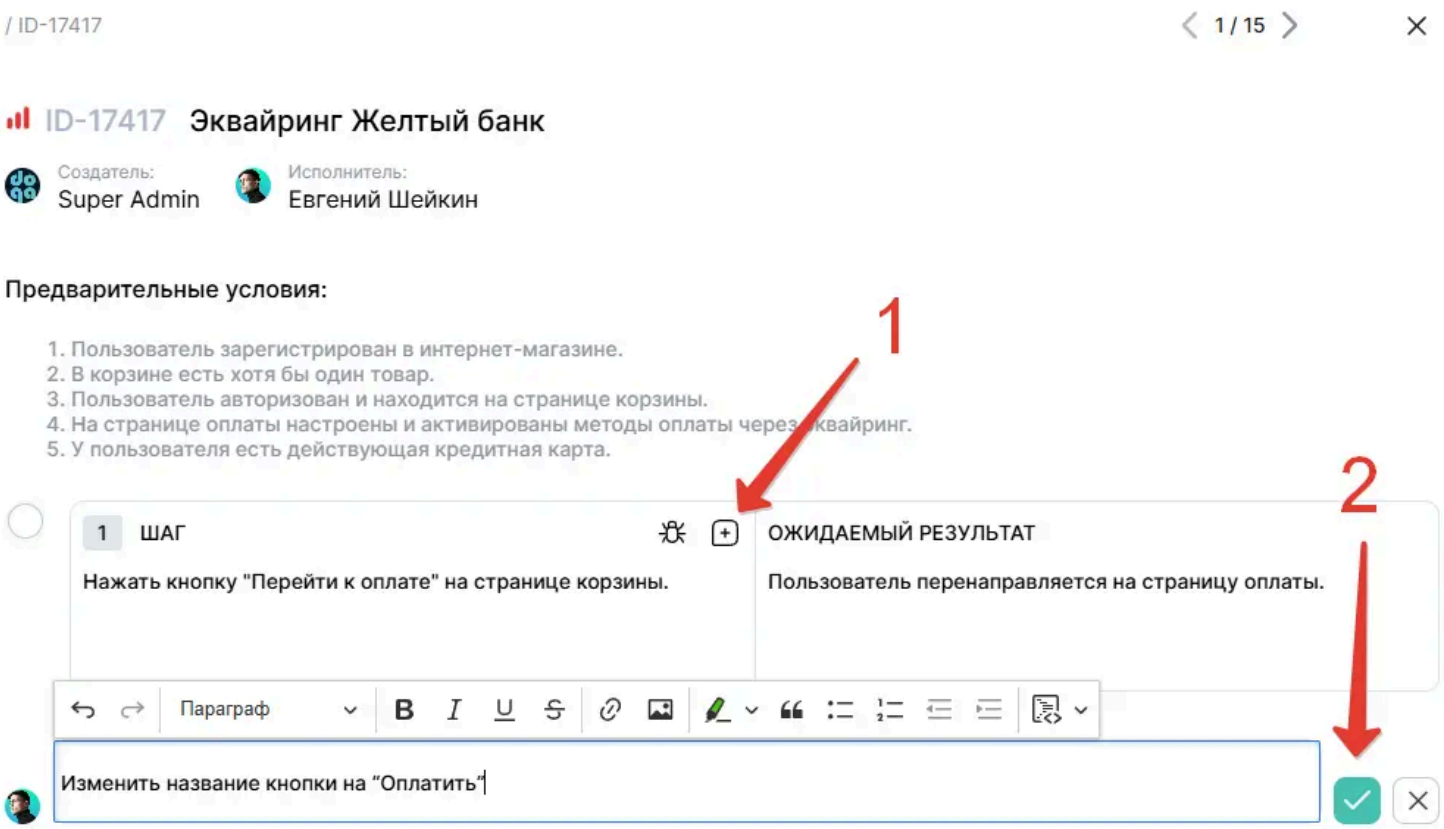
У каждого комментария есть ссылка на прогон, в котором он был оставлен, и раскрывающийся блок с содержанием шага или чека. У комментария, оставленного на автотесте, ссылки нет: выводится только пометка «Автотест».

Комментарии в прогонах

При прохождении тест-кейсов или чек-листов в прогоне можно оставлять комментарии. Например, чтобы отметить, что текущая версия теста не актуальна и требует доработок перед очередным тестовым циклом.

Это отдельный механизм: упоминания через @, ветки обсуждения, отметка «Решено» и уведомления, описанные выше для редакторов, в комментариях прогонов не поддерживаются.

На экране прохождения теста в плеере прогона нажмите на кнопку добавления комментария (цифра 1), введите комментарий и подтвердите его создание (цифра 2).

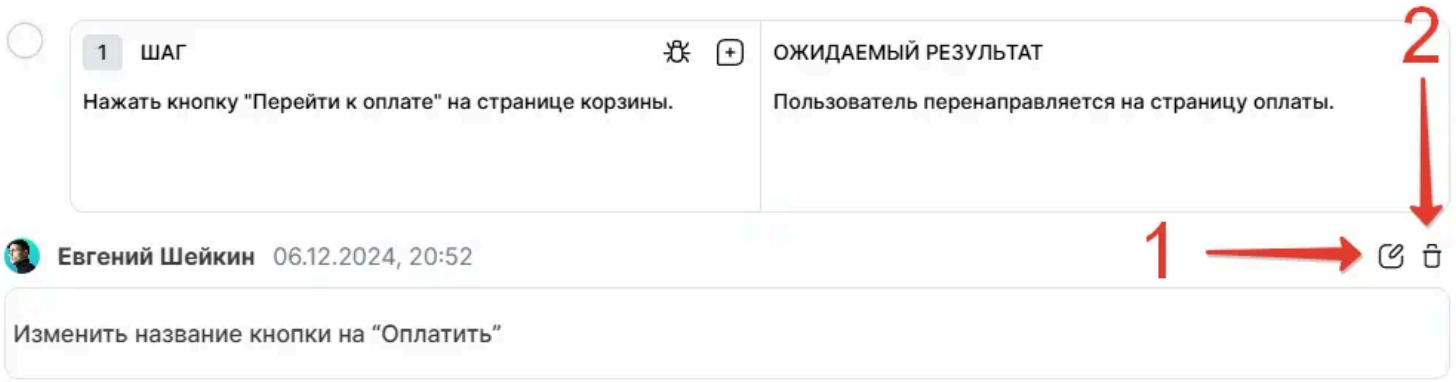


Комментарий можно оставить на шаге тест-кейса или на чеке, а на **автотесте** — кнопкой «**Добавить комментарий**» в карточке автотеста. Если элемент прогона пройден автоматически, комментарий привязывается к элементу целиком: оставить его на отдельном шаге или чеке нельзя.

Созданный комментарий отображается:

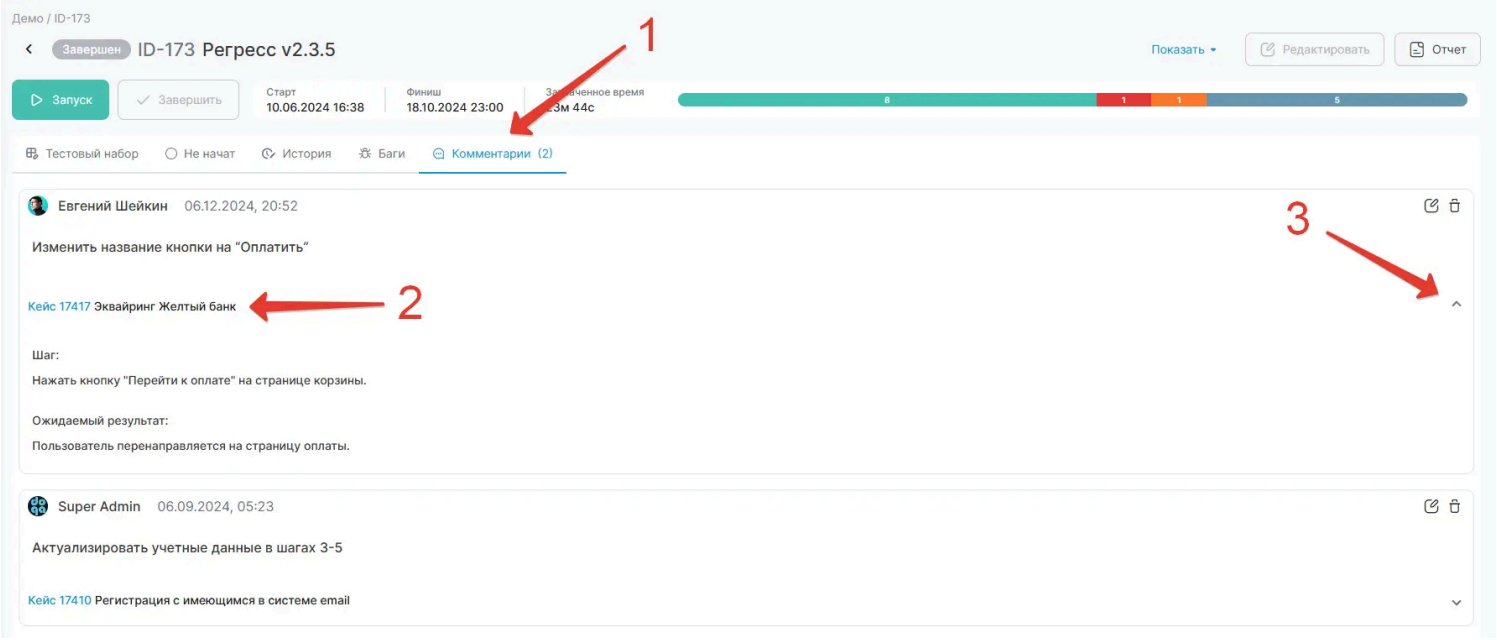
- На экране прохождения теста в плеере прогона, где он был создан.
- В плеере прогона на вкладке «Комментарии».
- В редакторе тест-кейса или чек-листа на панели «Комментарии прогонов».

Комментарий может быть отредактирован (цифра 1) или удален (цифра 2) его автором либо пользователем, у которого роль разрешает менять любые комментарии.



Отображение в плеере прогона

Чтобы просмотреть все комментарии, созданные в прогоне, нажмите на вкладку «Комментарии» в плеере прогона (цифра 1). Помимо данных о самом комментарии, указаны ID и название тест-кейса или чек-листа, в котором он был оставлен (цифра 2). При нажатии на иконку стрелки (цифра 3) раскрывается содержание шага или чека, на котором был оставлен комментарий.



Уведомления

DoQA отправляет уведомления о событиях в системе. Каждое событие настраивается отдельно, по каждому из трех каналов — «E-mail», «Telegram» и «DoQA» (уведомления внутри приложения, с колокольчиком).

Список событий:

Событие	Когда приходит
«Назначение ответственным за тест-кейс»	Вас назначили ответственным за **тест-кейс**.
«Назначение ответственным за чек-лист»	Вас назначили ответственным за **чек-лист**.
«Назначение исполнителем в прогоне»	Вас назначили исполнителем теста в **прогоне**.
«Ежедневная сводка по прогону (8:00 МСК (UTC+3))»	Ежедневно в 8:00 МСК (UTC+3) — сводка по незавершенным прогонам.
«Уведомление за 30 дней об истечении срока лицензии»	До окончания срока **лицензии** осталось 30 дней.
«Завершение созданного прогона»	Завершился прогон, который вы создали.
«Упоминание пользователя»	Вас упомянули через @ в **комментарии**.
«Ответы на комментарии»	Кто-то ответил на ваш комментарий.
«Необходимость актуализации теста»	Изменилось **требование**, связанное с тест-кейсом или чек-листом, за который вы отвечаете.
«Отключение расписания автотестов»	**Расписание запусков автотестов** отключилось само после серии сбоев подряд.

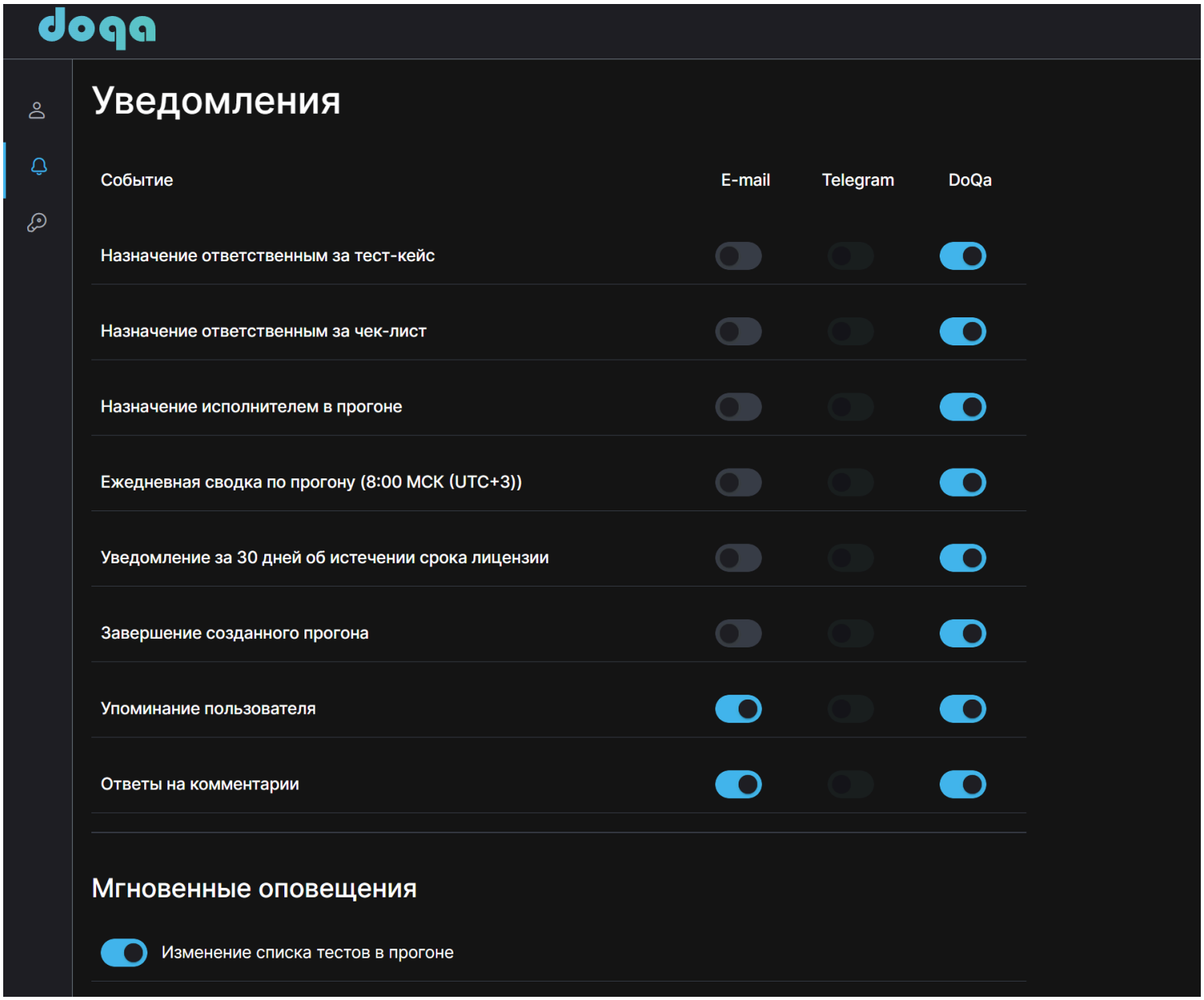
Отдельно от них настраиваются «Мгновенные оповещения» — событие «Изменение списка тестов в прогоне». Оно приходит только по каналу «DoQA», и его можно отключить.

Настройка уведомлений

Перейдите в раздел «Уведомления» в профиле пользователя. В таблице по строкам — события, по колонкам — каналы «E-mail», «Telegram» и «DoQA». Включите нужные переключатели.

Примечание

Переключатели в колонке «Telegram» неактивны, пока не привязан бот. Как его привязать — ниже.



Уведомление «Необходимость актуализации теста»

Уведомление получает только тот, кто указан ответственным за тест-кейс или чек-лист, если было изменено связанное с тестом требование.

Если у одного пользователя затронуто несколько тестов, он получает одно уведомление со списком (в списке выводится до 10 тестов) и ссылкой на изменившееся требование. Если у связанных проверок ответственных нет, уведомление не отправляется никому.

Что делать с таким уведомлением — в разделе **Актуализация** страницы «Требования».

Панель уведомлений

Уведомления по каналу «DoQA» приходят в панель, которая открывается по иконке колокольчика в шапке приложения. Счетчик на колокольчике показывает число непрочитанных.

В панели:

- Нажатие на уведомление открывает объект, к которому оно относится.
- Под текстом уведомления показаны логотип проекта и название пространства.
- Ссылка «**Отметить прочитанным**» / «**Отметить непрочитанным**» меняет состояние отдельного уведомления.
- Кнопка «**Отметить все прочитанным**» гасит счетчик целиком.
- Кнопка «**Показать еще**» подгружает следующую порцию.

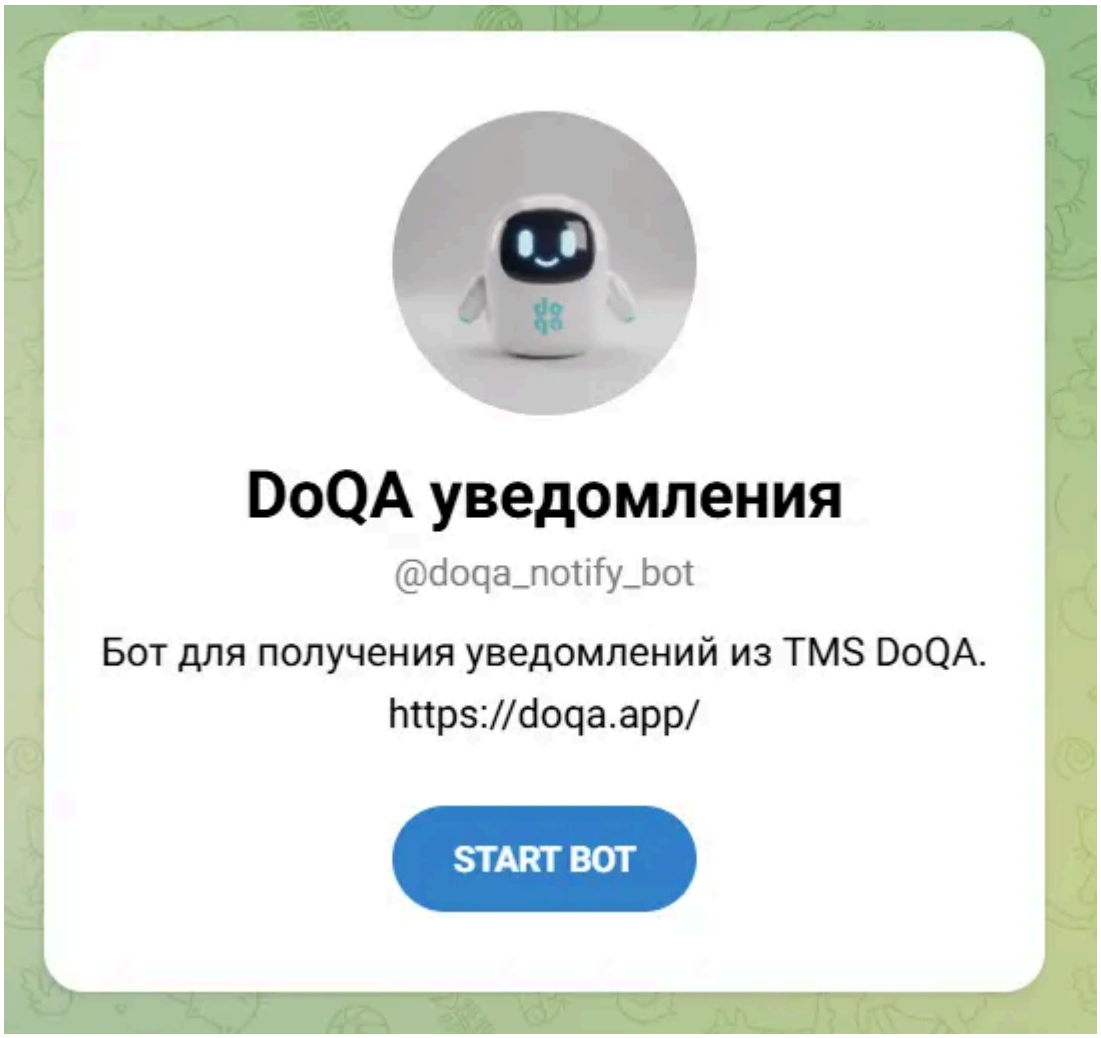
Когда уведомлений нет, в панели выводится «Нет новых уведомлений».

Кроме того, при получении нового уведомления по каналу «DoQA» в правом верхнем углу появляется всплывающее оповещение. Оно закрывается само через 10 секунд (рядом идет обратный отсчет); закрыть его можно и вручную крестиком. Если уведомлений пришло несколько, оповещения показываются по очереди, по одному.

Подключение telegram-бота

Блок подключения показывается в разделе «Уведомления» профиля, пока бот не привязан.

1. Скопируйте код: нажмите на него в блоке подключения.
2. Откройте чат с Telegram-ботом по ссылке из этого же блока.
3. Отправьте скопированный код сообщением в чат с ботом.



После привязки бота переключатели в колонке «Telegram» станут активными. Отметьте события, о которых хотите получать уведомления в Telegram.

Имя бота

Имя бота задается на сервере, поэтому ссылку берите из блока подключения, а не из этой инструкции.

Настройку бота в DoQA-box выполняет администратор сервера — см. **Описание параметров среды (.env)**.

Горячие клавиши

Сочетания клавиш DoQA и текстового редактора, в котором пишутся описания и шаги.

Дальше в таблицах: первая колонка — Windows и Linux, вторая — MacOS. Разница почти всегда одна: вместо Ctrl нажимается Command.

Редакторы

Действие	Windows/Linux	MacOS	Где работает
добавить шаг или чек в конец списка	Ctrl + Shift + D	Command + Shift + D	редакторы **тест-кейса**, **чек-листа** и **общего шага**
сохранить	Ctrl + S	Command + S	редакторы тест-кейса и чек-листа; редактор **баг-репорта** и окно создания баг-репорта в **прогоне**
отправить **комментарий**	Ctrl + Enter	Command + Enter	поле комментария в тест-кейсе, чек-листе и прогоне
закрыть боковую панель	Escape	Escape	любая открытая панель в пространстве

Совет

В редакторе общего шага Ctrl + S не работает: там есть только Ctrl + Shift + D. Сохраняйте общий шаг кнопкой.

Таблицы

Действие	Как	Где работает
отметить диапазон чек-боксов	отметьте чек-бокс начала диапазона, затем зажмите Shift и отметьте конечный	таблицы **тест-кейсов**, **чек-листов**, **прогонов**

Действие	Как	Где работает
открыть элемент в новой вкладке	Ctrl + клик по строке	таблицы тест-кейсов и чек-листов
открыть элемент в новом окне	Shift + клик по строке	таблицы тест-кейсов и чек-листов

Выделение диапазона захватывает только элементы: **папки**, попавшие между начальным и конечным чек-боксом, остаются неотмеченными. В таблице **общих шагов** диапазон и открытие в новой вкладке не работают: там чек-боксы отмечаются по одному.

Текстовый редактор

Часть форматирования срабатывает по сочетанию клавиш и по Markdown-синтаксису: наберите разметку прямо в тексте, и редактор превратит её в оформление.

Форматирование текста

Команда	Markdown-синтаксис	Windows/Linux	MacOS
Жирный	`**текст**` или `__текст__`	Ctrl + B	Command + B
Курсив	`*текст*` или `_текст_`	Ctrl + I	Command + I
Подчеркнутый	нет	Ctrl + U	Command + U
Зачеркнутый	`~~текст~~`	Ctrl + Shift + X	Command + Shift + X
Код (встроенный)	`` `текст` ``	Ctrl + E	Command + E
Выделение цветом	`==текст==`	Ctrl + Shift + H	Command + Shift + H

Форматирование абзаца и блоков

Команда	Markdown-синтаксис	Windows/Linux	MacOS
Заголовки 1–5	`#` ... `#####` + пробел	Ctrl + Alt + 1...5	Command + Alt + 1...5

Команда	Markdown-синтаксис	Windows/Linux	MacOS
Цитата	`>` + пробел	Ctrl + Shift + B	Command + Shift + B
Блок кода	``` ``` ``` или `~~~` + пробел	Ctrl + Alt + C	Command + Alt + C
Горизонтальная линия	`---` или `***`	нет	нет

В редакторе доступны заголовки пяти уровней. Сочетания Ctrl + Alt + 6 нет.

Списки

Команда	Markdown-синтаксис	Windows/Linux	MacOS
Маркированный список	`*`, `-` или `+` + пробел	Ctrl + Shift + 8	Command + Shift + 8
Нумерованный список	`1.` + пробел	Ctrl + Shift + 7	Command + Shift + 7

Основные команды и навигация

Команда	Windows/Linux	MacOS
Отменить	Ctrl + Z	Command + Z
Повторить	Ctrl + Shift + Z	Command + Shift + Z
Добавить разрыв строки	Shift + Enter	Shift + Enter
Выйти из блока кода	Ctrl + Enter	Command + Enter
Вставить параметр	Ctrl + Shift + K	Command + Shift + K

Ctrl + Shift + K вставляет пустой параметр `[]` и открывает подсказку с выбором. Тот же список появится, если набрать `[]` вручную. Параметры доступны только в редакторах тест-кейса и чек-листа. В редакторе общего шага, в баг-репорте и в комментариях этого сочетания нет.

В поле комментария символ `@` открывает список пользователей для упоминания.

Работа с таблицами

Действие	Клавиша
Переход к следующей ячейке	Tab
Переход к предыдущей ячейке	Shift + Tab

Глоссарий

Термины DoQA в алфавитном порядке. Каждый ведёт на страницу, где он разбирается подробно.

А

Автотест — тест, результаты которого приходят в DoQA из CI или загружаются вручную; вручную автотест не создаётся. Отображается в **прогоне** рядом с ручными тестами. См. Автотесты.

Атрибут — дополнительное поле, которое администратор заводит для тестовой документации проекта (строка, дата, выбор из списка, пользователь). См. **Атрибуты** и **настройку атрибутов**.

Б

Баг-репорт — запись о дефекте, которую заводят при прохождении теста в прогоне. Может уходить в подключённый трекер. См. **Баги**.

Г

Глобальный параметр — параметр, доступный во всём пространстве, а не только в одном тесте. См. **Параметризация тестов**.

К

Коин — внутренняя единица оплаты AI-функций в облачной версии. В серверной версии вместо коинов расходуются токены вашего LLM-провайдера. См. **AI-функции** и **Настройки DoQA AI**.

Коннектор требований — связь пространства с внешним трекером (Jira Server или YouTrack), из которого загружаются требования. См. **Подключение трекера требований**.

Корзина — раздел, куда попадают удалённые тест-кейсы, чек-листы, общие шаги, прогоны и баги до окончательного удаления. См. **Корзина**.

М

Матрица покрытия — таблица «требования × проверки»: видно, какие требования покрыты тестами, а какие нет. См. **Матрица покрытия**.

О

Общий шаг — набор шагов, который хранится отдельно и переиспользуется в нескольких тест-кейсах; правится в одном месте. См. **Общие шаги**.

П

Параметр — именованный набор значений, который подставляется в текст теста; при добавлении в прогон DoQA создаёт по слепку на каждую комбинацию значений. См. **Параметризация тестов**.

Покрытие — доля требований, у которых есть связанные тесты. Показывается в разделе требований и в матрице. См. **Требования**.

Проект — верхний уровень в DoQA: содержит пространства, задаёт участников, права и атрибуты. См. **Проекты**.

Прогон — набор тест-кейсов и чек-листов, которые проходят в одном цикле тестирования; сюда же попадают результаты автотестов. См. **Прогоны**.

Пространство — рабочая область внутри проекта: в нём находятся тест-кейсы, чек-листы, общие шаги, прогоны, требования, автотесты и баги. См. **Пространства**.

С

Слепок — копия параметризованного теста под конкретную комбинацию значений параметров, создаётся в прогоне. См. **Параметризация тестов**.

Стратегия комбинаций — правило, по которому из значений параметров собираются комбинации: «Все комбинации», «Каждый выбор», «Линейное сопоставление», «Попарное тестирование (pairwise)». См. **Параметризация тестов**.

Т

Тег — метка для поиска и фильтрации тестовой документации. См. **Теги**.

Тест-кейс — сценарий проверки: шаги с ожидаемым результатом, ответственным и статусом автоматизации. См. **Тест-кейсы**.

Требование — загруженная из внешнего трекера единица, к которой привязываются тесты; в DoQA требования не создаются, а синхронизируются. См. **Требования**.

Ч

Чек-лист — дерево проверок без шагов и ожидаемых результатов; подходит там, где подробный сценарий избыточен. См. **Чек-листы**.

Латиница

SSO — корпоративный вход через внешнего провайдера (OIDC) в серверной версии. См. **SSO**.

Смотрите также

- **Руководство пользователя** — с чего начать
- **Как устроен продукт** — общая картина

Администрирование

Раздел «Администрирование» - рабочее место администратора: здесь заводят проекты и людей, раздают доступ, управляют лицензиями, подключениями и безопасностью. Он открывается из меню продукта и имеет собственную шапку со ссылкой «К проектам», боковую навигацию по группам и поиск по разделу.

Раздел виден тому, у кого есть право хотя бы на один из его пунктов, хотя бы в одном месте. Каждый пункт меню показывается, только если для него есть право: тест-лид, например, видит «Проекты» только по своим проектам. Как права складываются в доступ - на странице Как устроены права.

Разделы

Группа	Раздел	Нужное право	Страница
	Обзор	Любое право раздела	см. ниже
Люди	Пользователи	`user.view`	Пользователи
Люди	Группы	`group.view`	Группы
Люди	Роли	`role.view`	Роли и права
Люди	Сервисные аккаунты	`service_account.manage`	Сервисные аккаунты
Люди	Проекты	Любое из: `project.create`, `project.edit`, `project.archive`, `project.delete`, `space.manage`, `project_settings.manage`, `tracker.manage`, `access.view`	Доступ к проектам и пространствам
Подключения	Интеграции	`integration.view` (менять: `integration.manage`)	Интеграции
Подключения	API-токены	`project_token.view` или `project_token.manage`	API-токены

Группа	Раздел	Нужное право	Страница
ИИ	Провайдер, Промпты	`ai_settings.manage`	Настройки DoQA AI
ИИ	Использование	`ai_usage.view`	Настройки DoQA AI
	Лицензии и оплата	`license.view` (оплата: `billing.view`, `billing.manage`)	Лицензии, серверная версия
Безопасность	Запросы доступа	`access.manage`	Запросы доступа
Безопасность	Проверка доступа	`access.view`	Проверка доступа
Безопасность	Пересмотр доступа	`access.manage`	Пересмотр доступа
Безопасность	Журнал аудита	`audit.view`	Журнал аудита
Безопасность	Вход и SSO	`sign_in.view`	Вход и SSO
Система	Поиск	`search.reindex`	переиндексация поискового индекса
Система	Уведомления	`notification.manage`	каналы и шаблоны уведомлений организации

Права входят в **роли**: «Администратор организации» видит всё, «Администратор пользователей» - людей, лицензии и оплату, «Аудитор» - всё только для чтения. Полный список прав: **Справочник прав**.

Обзор

Первая страница раздела показывает состояние организации и то, что требует внимания: истекающие лицензии, ошибки подключений, неработающий вход. Рядом - сводки по

пользователям (активные, приглашённые, заблокированные), лицензиям, подключениям, ИИ и последние действия администраторов с переходом в журнал аудита. Блоки, к которым у вас нет права, не показываются.

Права на разделы и кнопки

Если у вас нет права на действие, кнопка недоступна, а в подсказке названа причина: запрет, нет роли, место закрыто для доступа сверху, не выполнено условие роли или нужна лицензия. Свои права и их источники видно на странице **Мой доступ**.

Навигация по темам

Люди и доступ

- Как устроены права
- Пользователи: добавление, редактирование, сброс пароля
- Роли и права и Справочник прав
- Группы
- Доступ на всю организацию
- Доступ к проектам и пространствам и доступ к пространству
- Закрытые проекты и пространства
- Запреты
- Запросы доступа
- Пересмотр доступа
- Проверка доступа
- Мой доступ
- Владелец организации

Токены и сервисные аккаунты

- API-токены

- Сервисные аккаунты

Безопасность

- Журнал аудита
- Вход и SSO

Лицензии

- Лицензии и оплата (облачная версия)
- Лицензии (серверная версия)

Проекты и настройки

- Создание проекта
- Атрибуты

Интеграции

- Интеграции с трекерами
- Интеграции с CI/CD

DoQA AI

- Настройки DoQA AI
- Настройка прокси для LLM-провайдеров

Как устроены права

Эта страница объясняет, из чего складывается доступ в DoQA. Подробности по каждому понятию - на отдельных страницах раздела.

Модель в четырёх правилах

1. Права живут только в **ролях**. Человеку отдельное право не выдаётся.
2. Доступ - это роль, выданная в каком-то месте. Роль, выданная выше, действует ниже, в том числе в новых проектах и пространствах. Исключение - **закрытый** проект или пространство.
3. Доступ дают группам, исключения выдают лично. Человек получает роли своих **групп** и свои личные роли. Итог - их сумма. **Запрет** сильнее любой роли.
4. Без лицензии - только просмотр. Лицензия не меняет ролей, она включает права на изменение.

Понятия

Понятие	Что это
Право	Одно действие над одним видом объекта, например `case.edit`. Полный перечень: **Справочник прав**
Роль	Именованный набор прав. Бывает системной (от DoQA) и своей
Группа	Набор пользователей. Состав ведётся вручную, берётся из SSO или LDAP либо задаётся правилом
Сервисный аккаунт	Не человек: CI, скрипт, интеграция. Получает роли как человек, в интерфейс войти не может
Место (узел)	Организация, проект или пространство
Выдача	Роль, выданная человеку, группе или сервисному аккаунту в определённом месте
Запрет	Отнимает набор прав или весь доступ в месте и ниже

Понятие	Что это
Условие	Сужает право внутри роли: только свои, назначенные мне, в нужных статусах
Лицензия	Включает права на изменение. Выдаётся пользователю
Владелец организации	Один человек с полным доступом, который нельзя отозвать или ограничить. Это свойство организации, а не роль

Дерево мест

```
Организация
├── Проект «Мобильный банк»
│   ├── Пространство «Android»
│   └── Пространство «iOS»
└── Проект «CRM»
    └── Пространство «Core»
```

Роль на организации действует во всех проектах и пространствах, роль на проекте - во всех его пространствах, роль на пространстве - только в нём. Архивный проект остаётся в дереве, его права действуют после восстановления.

У каждого права есть уровень: организация, проект или пространство. Право организации (например, `user.invite`) работает только при роли, выданной на всю организацию.

Как принимается решение

Шаг	Проверка	Если не прошла
1	Вы владелец организации?	Можно
2	Есть запрет на это право в этом месте или выше?	Нельзя, причина: запрет
3	Есть ли среди ролей (личных и групповых) в этом месте и выше, но не выше ближайшего закрытого, роль с этим правом, и	Нельзя, причина: нет роли или место

Шаг	Проверка	Если не прошла
	выполнены ли условия?	закрыто
4	Право требует лицензии, а её нет?	Нельзя, причина: нет лицензии
5	Иначе	Можно

Причину отказа видно в интерфейсе и на странице **Проверка доступа**.

Где что настраивать

Задача	Страница
Пригласить людей, выдать лицензию	**Пользователи**
Собрать набор прав	**Роли и права**
Дать доступ команде	**Группы**
Дать доступ к проекту или пространству	**Доступ к проектам и пространствам**
Спрятать проект или пространство	**Закрытые проекты и пространства**
Исключить человека из доступа	**Запреты**
Ответить на просьбу о доступе	**Запросы доступа**
Регулярно проверять выданное	**Пересмотр доступа**
Понять, почему доступ есть или нет	**Проверка доступа**
Доступ для CI и скриптов	**Сервисные аккаунты, API-токены**
Посмотреть свои права	**Мой доступ**

Создание проекта

Проект создают в разделе «Проекты» администрирования или на странице **«Проекты»**. Форма на странице «Проекты» продукта дополнительно запрашивает поле «ID родителя» (префикс из 3-4 латинских букв). Создать проект может тот, у кого есть право `project.create`: роли «Создатель проектов» и «Администратор организации».

Форма «Новый проект»

1. Нажмите «Создать проект».
2. Заполните поля:
   - «Название проекта» - обязательное, например «Мобильный банк».
   - «Описание» - необязательно.
   - «Логотип» - необязательно, файл изображения.
3. Нажмите «Создать проект».

В форме администрирования ключ проекта не запрашивается: проект опознаётся по названию, а удаление подтверждается вводом названия.

Доступ к новому проекту

Создатель проекта получает на нём роль «Тест-лид». Остальным доступ выдают на вкладке «Доступ» проекта: людям, группам и сервисным аккаунтам. Люди и группы с рабочей ролью на всю организацию получают доступ автоматически. См. **Доступ к проектам и пространствам**.

Карточка проекта

Вкладки карточки и действия с проектом (архивирование, удаление) описаны на странице **Доступ к проектам и пространствам**.

Внимание

Удаление проекта необратимо. Архивация обратима: проект скрывается от участников и возвращается из архива, см. **Архивные проекты**.

Владелец организации

Владелец организации - один человек с полным доступом, который нельзя отозвать или ограничить. Это не роль, а свойство организации. Владельцем по умолчанию становится первый зарегистрированный пользователь.

Что даёт

- Полный доступ ко всему, независимо от ролей и **запретов**.
- Доступ к разделу «Лицензии» и оплате даже при истёкших лицензиях, чтобы продлить их, см. **Лицензии и оплата (облачная версия)**. Владелец всегда первый в очереди на лицензию и работает с полным доступом, даже когда лицензий нет; сделать его наблюдателем нельзя.
- Закрытые проекты и пространства владелец видит и может выдавать в них доступ, см. **Закрытые проекты и пространства**.

Что нельзя

Владельца нельзя заблокировать, удалить или ограничить запретом. Свою роль владельца нельзя потерять иначе, чем передав владение.

В списке пользователей владелец помечен.

Передача владения

Только владелец может передать владение, и только активному пользователю.

1. Откройте карточку нужного пользователя.
2. Нажмите «Передать владение...».
3. Подтвердите. Новый владелец получает полный контроль, прежний сохраняет только выданные ему роли.

Передача записывается в **журнал аудита**.

Если нужно вернуть администрирование

Если в организации не осталось людей, управляющих доступом, на серверной версии можно выдать «Администратора организации» командой `access:grant-org-admin <почта>` на сервере приложения.

Пользователи

Раздел «Люди» → «Пользователи»: список всех, кто работает в организации, с ролями, группами и статусом. Раздел виден тем, у кого есть право `user.view`.

Список

Столбец	Содержание
Пользователь	Имя, под ним почта. Пометки « · владелец», « · вы». Метка «Только просмотр» или «Наблюдатель» у тех, кто работает без лицензии
Роль и группы	Роль администрирования, если она есть (если их несколько, первая и «+N»), и группы через запятую или «Нет групп»
Последний вход	Дата последнего входа или «не входил». Если человек не входил больше 30 дней, дата выделена и под ней написано, сколько дней прошло
Статус	Активен, Приглашён, Заблокирован

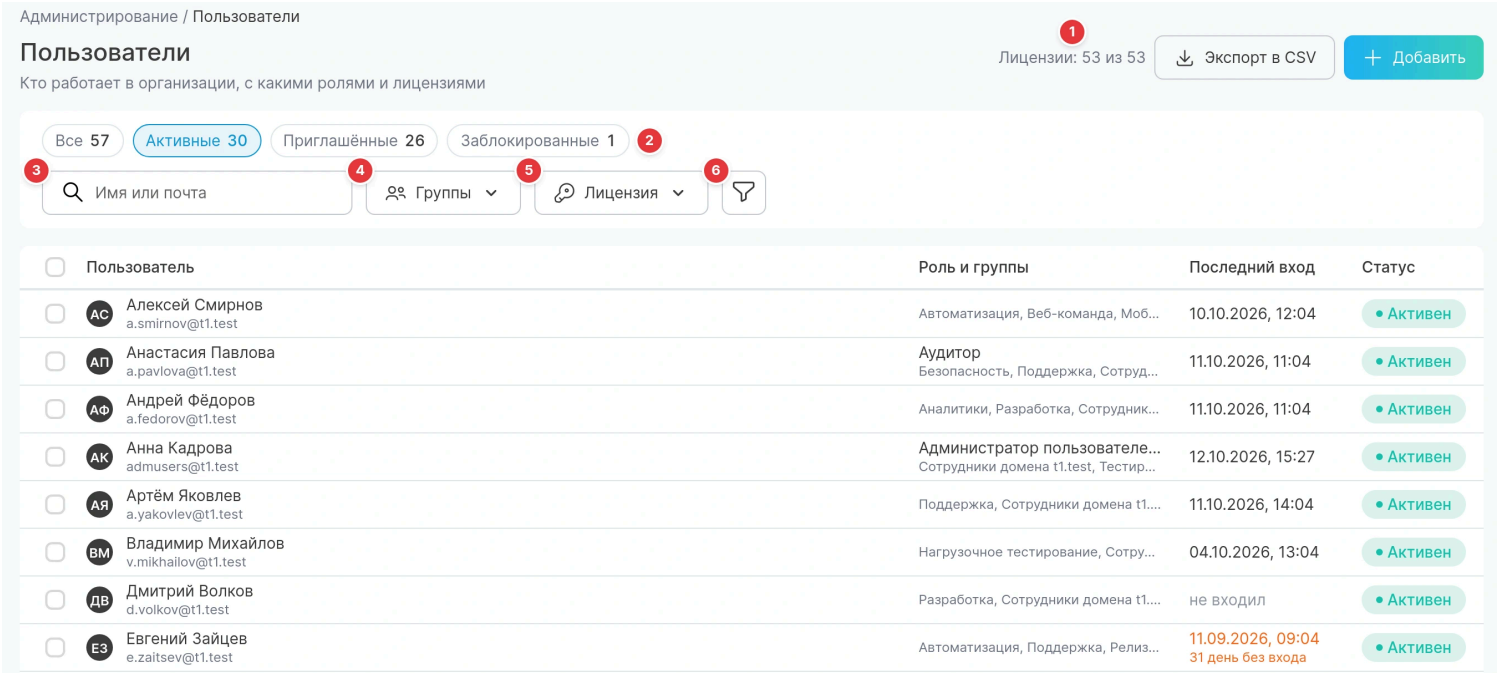
Вкладки над списком: «Все», «Активные», «Приглашённые», «Заблокированные», у каждой число людей (цифра 2).

Фильтры складываются друг с другом:

- поиск по имени или почте (цифра 3);
- «Группы» (цифра 4): отметьте одну или несколько групп и нажмите «Готово». В список попадут те, кто состоит хотя бы в одной из выбранных. Пункт «Без групп» показывает людей вне групп, его можно отметить вместе с группами. Справа у каждой группы число людей;
- «Лицензия» (цифра 5, видна с правом `license.view`): «С лицензией», «Без лицензии» (все, кто работает только на просмотр), «Не хватило лицензии», «Наблюдатели»;
- кнопка фильтров (цифра 6): роль администрирования, проект, способ входа, доступ вне групп.

«Сбросить фильтры» снимает все фильтры сразу. Список можно выгрузить в CSV с текущими фильтрами.

Клик по строке открывает **карточку пользователя** справа. Список остаётся виден: клик по другой строке переключает карточку.



Лицензии

Лицензии распределяются сами: каждый добавленный пользователь занимает свободную лицензию, а если их не хватило, работает «Только просмотр». Подробно: **Как распределяются лицензии**.

- Рядом с кнопкой «Добавить» тихая ссылка «Лицензии: X из N» (цифра 1 на скриншоте списка) ведёт в раздел лицензий (видна с правом `license.view`).
- Если кому-то не хватило лицензий, над списком плашка «N пользователей работают только на просмотр: не хватило лицензий». В облаке кнопка «Докупить N» открывает покупку с нужным числом лицензий (нужно `billing.manage`, без права плашка подсказывает, кто может докупить). В серверной версии кнопка «Запросить продление» открывает заявку на новый ключ.
- У имени метка «Только просмотр» (не хватило лицензии, доступ включится сам после докупки) или «Наблюдатель» (назначен вручную и не занимает лицензию). Подсказка у метки объясняет причину. У владельца и заблокированных меток нет.
- «Сделать наблюдателем» (в карточке пользователя или массово) освобождает лицензию человека: она сразу переходит первому ждущему. Роли и доступ сохраняются. «Вернуть полный доступ» ставит человека в конец очереди. Нужно право `license.assign`. Сделать наблюдателем владельца или себя нельзя.

После оплаты метки снимаются без перезагрузки страницы.

Приглашение

Кнопка «Добавить» открывает панель «Добавить пользователей» (нужно право `user.invite`).

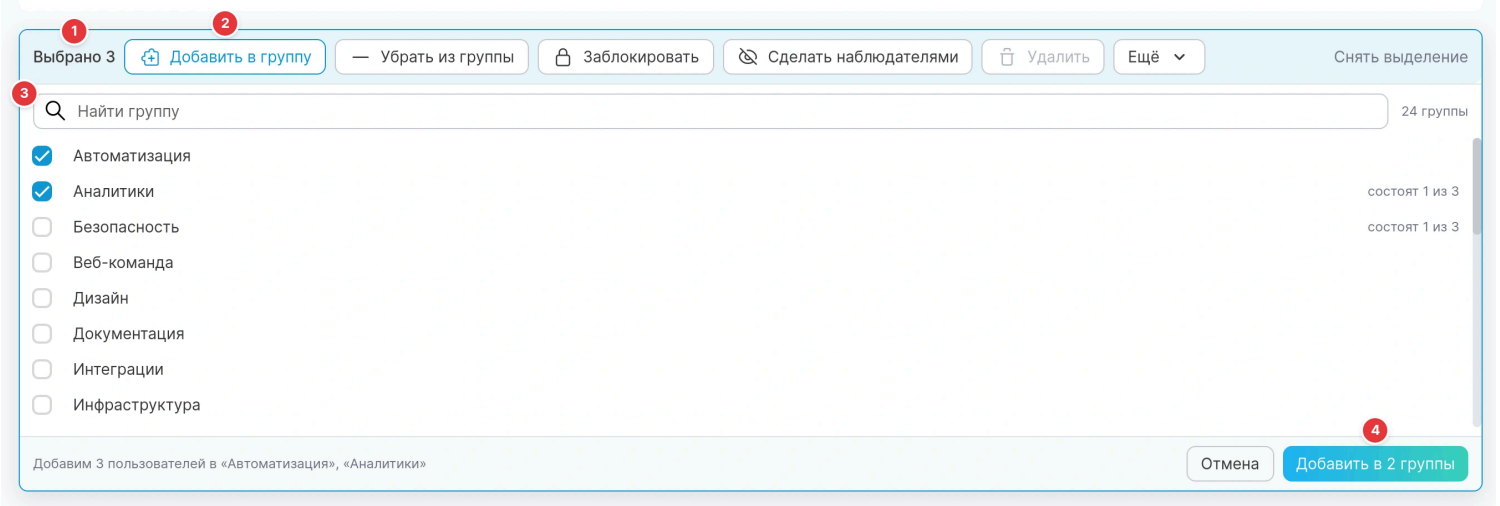
1. Введите почты через запятую, пробел или перенос строки. Можно вставить столбец из таблицы. За раз - не больше 100 адресов. Дубли, неверные адреса, уже приглашённые и уже состоящие в организации помечаются и пропускаются.
2. «Группы»: выберите группы. Группы по умолчанию отмечены заранее. Группы из SSO, LDAP и по правилам заполняются автоматически.
3. «Рабочие роли»: выберите роли и в блоке «Где» проекты. Оставьте проекты пустыми, чтобы выдать роль на всю организацию.
4. «Роли администрирования»: при необходимости.
5. Нажмите «Добавить N».

Лицензии выбирать не нужно: новые пользователи займут свободные сами. Если свободных лицензий меньше, чем добавляемых, под полями появится строка «Свободно N лицензий. M пользователей будут работать только на просмотр, пока вы не докупите лицензии». После добавления DoQA сообщит «Добавлено: N. Приглашения отправлены на почту».

Если включена политика «только группы», рабочие роли людям напрямую не выдаются: выберите группы. Если включён SSO с автоматическим созданием пользователей, люди могут войти и без приглашения.

Массовые действия

Отметьте строки, и над списком появится панель «Выбрано N» (цифра 1). Она остаётся на месте при прокрутке, «Снять выделение» убирает отметки. Кнопки действий (цифра 2) работают сразу со всеми отмеченными. «Добавить в группу» раскрывает в панели выбор групп (цифра 3), кнопка «Добавить в N групп» (цифра 4) применяет выбор.

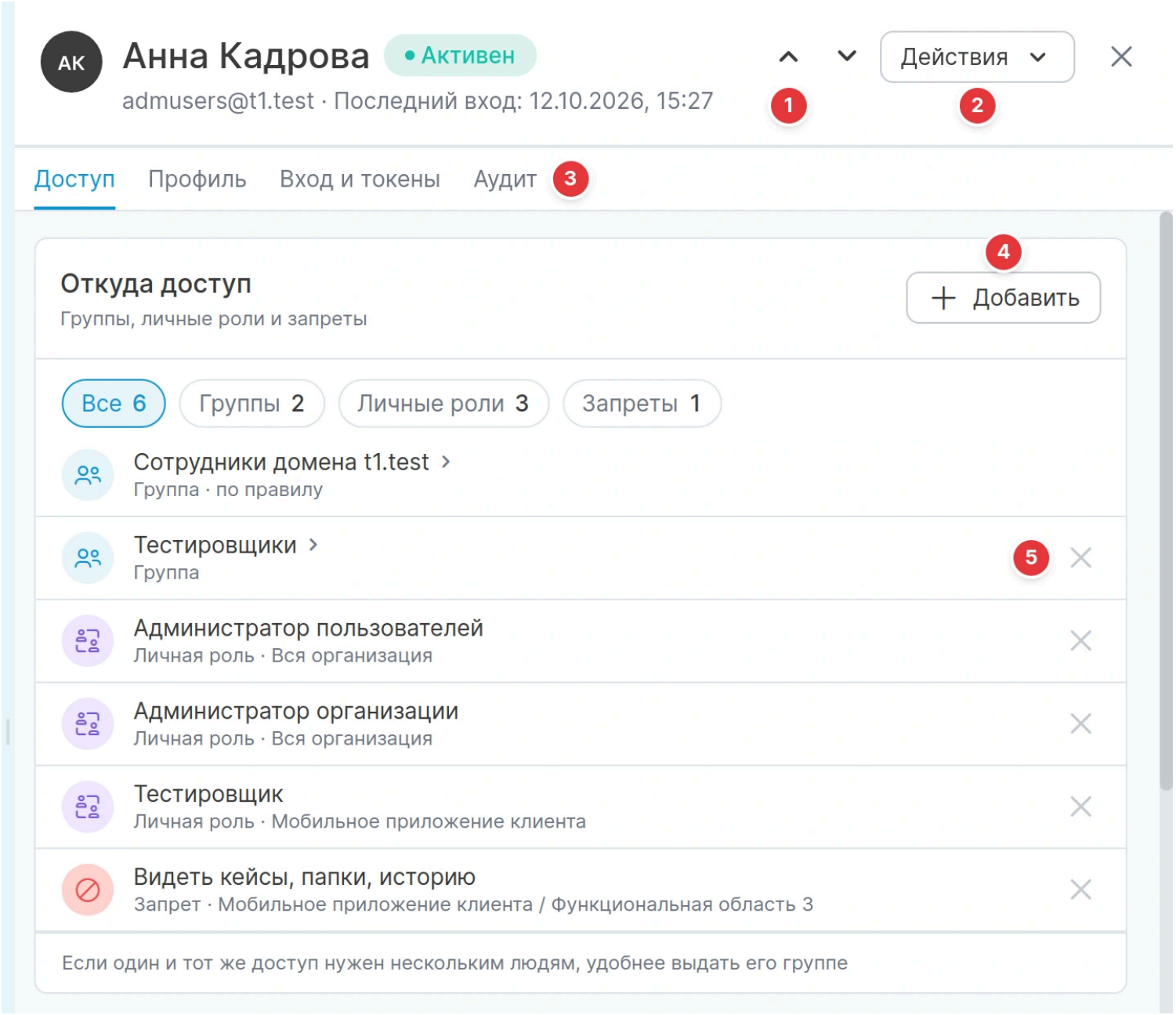


Действие	Что происходит
Добавить в группу	В панели раскрывается выбор групп: можно отметить несколько. Рядом с группой видно, сколько выбранных в ней уже состоит. Группы, состав которых задаётся правилом, SSO или LDAP, выбрать нельзя
Убрать из группы	Тот же выбор, но только из групп, где состоит кто-то из выбранных
Заблокировать	Подтверждение прямо в панели: «Заблокировать N пользователей? Их сессии завершатся, лицензии освободятся». Владельца организации и себя заблокировать нельзя, они пропускаются
Разблокировать	Появляется, если среди выбранных есть заблокированные. Без подтверждения
Сделать наблюдателями	Наблюдатели освобождают лицензии для первых в очереди. Без подтверждения
Удалить	Удаляются только приглашённые, подтверждение в панели. Зарегистрированных удалить нельзя: заблокируйте их
Ещё	«Выдать роль...», «Сбросить пароль», «Отправить приглашение снова», «Отозвать приглашение», «Вернуть полный доступ»

После действия отметки снимаются, а сообщение называет итог и пропущенных с причиной.

Карточка пользователя

Карточка открывается справа от списка. Вкладок нет: это одна страница с разделами «Доступ», «Профиль», «Вход и токены», «Аудит», ссылки на них над содержимым (цифра 3). Раздел без права не показывается.



В шапке: имя, статус, метка лицензии, почта и последний вход. Стрелки ↑ и ↓ (цифра 1) открывают предыдущего и следующего человека в текущем списке с учётом фильтров. Закрыть карточку можно крестиком или клавишей Esc. У карточки свой адрес, ссылку на неё можно переслать. Старые ссылки на страницы пользователя тоже открывают карточку.

Действия

Меню «Действия» (цифра 2):

- «Отправить приглашение снова» и «Скопировать ссылку-приглашение» для приглашённых;
- «Сделать наблюдателем» или «Вернуть полный доступ» (право `license.assign`; владельца, себя и заблокированного сделать наблюдателем нельзя);
- «Скопировать доступ другому»: выберите человека, посмотрите, что он получит, и выдайте. Копируются ручные группы и личные роли, запреты не копируются;
- «Посмотреть как пользователь» открывает **проверку доступа** с этим человеком: что он видит в каждом проекте и почему;
- «Передать владение» (только для владельца), см. **Владелец организации**;
- «Заблокировать» или «Разблокировать»;
- «Отозвать приглашение» для приглашённого. Зарегистрированного пользователя удалить нельзя: заблокируйте его, лицензия освободится.

Доступ

Раздел «Откуда доступ» собирает в один список группы, личные роли и запреты человека. Правила групп здесь не показаны: название группы ведёт в её карточку, где видны роли и участники. Группы с составом по правилу, из SSO или LDAP помечены и здесь не снимаются.

Кнопка «Добавить» (цифра 4):

- «Добавить в группу»: основной способ, доступ как у коллег. Можно выбрать несколько групп;
- «Выдать личную роль»: исключение для одного человека. Выберите роль и место: вся организация или проект. «Больше настроек» открывает полную панель выдачи с пространствами и несколькими местами;
- «Поставить запрет»: запрет сильнее любых ролей, см. **Запреты**;
- «Как у другого пользователя»: группы и личные роли коллеги.

Крестик в строке (цифра 5) убирает из группы, снимает роль или запрет сразу, а в сообщении есть «Отменить». Клик по личной роли открывает её изменение. Если источников больше пяти, над списком появляется фильтр по типу, длинный список свёрнут до шести строк. Свой собственный доступ менять нельзя: попросите другого администратора.

Профиль

Имя и фамилию меняют с правом `user.edit`. Кнопки «Сохранить» и «Отменить» появляются после изменения. Почту может изменить только сам пользователь, с подтверждением.

Вход и токены

- Способ входа и последний вход.
- «Сбросить пароль» отправляет ссылку для установки нового пароля, текущий пароль перестаёт работать. Для SSO и LDAP пароль управляется во внешней системе. См. **Сброс пароля**.
- «Завершить все сессии» разлогинивает пользователя на всех устройствах.
- Личные **API-токены** (нужно `user.tokens_manage`): когда созданы и использованы, «Отозвать» и «Отозвать все». Токены и сессии владельца организации видит только он сам.

Аудит

Последние события «С пользователем» и «Действия пользователя» (нужно `audit.view`), ссылка «Открыть в журнале аудита» с фильтром по человеку. Любое изменение в карточке сразу попадает в журнал.

Блокировка

«Заблокировать» закрывает вход и завершает сессии, отзывает личные API-токены, а лицензия освобождается для следующего в очереди. Группы и роли сохраняются: после разблокировки доступ вернётся, а пользователь встанет в конец очереди на лицензию. Если нужно только освободить лицензию, сделайте пользователя наблюдателем вместо блокировки. Владельца организации и себя заблокировать нельзя.

Добавление пользователей

Пользователей приглашают по почте в разделе «Люди» → «Пользователи». Для этого нужно право `user.invite` (роль «Администратор пользователей» или «Администратор организации»).

1. Нажмите «Добавить».
2. Введите почты через запятую, пробел или перенос строки (до 100 за раз).
3. Выберите группы, рабочие роли с местом и роли администрирования.
4. Нажмите «Добавить N».

Каждый получает письмо со ссылкой для регистрации. Пока приглашение не принято, пользователь отображается со статусом «Приглашён». Повторить, отозвать приглашение или скопировать ссылку-приглашение можно в карточке пользователя, меню «Действия».

Все параметры панели добавления, ограничения и массовые действия описаны на странице **Пользователи**.

> **Совет**
>
> Доступ удобнее давать через **группы**: новый человек сразу получает права команды. Группы по умолчанию отмечаются в приглашении заранее.

Лицензию выдавать не нужно: новые пользователи занимают свободные лицензии сами. Если лицензий не хватит, панель предупредит, сколько человек будут работать только на просмотр. Лицензии: **облачная версия**, **серверная версия**.

Редактирование пользователей

Откройте «Люди» → «Пользователи» и выберите человека: откроется его карточка. Вкладки и действия описаны на странице **Пользователи**.

Что можно изменить

Что	Где	Нужное право
Имя и фамилию	Вкладка «Профиль»	`user.edit`
Роли и группы	Вкладка «Доступ»	`access.manage` в нужном месте
Лицензию	Вкладка «Лицензия»	`license.assign`
Блокировку	Вкладка «Профиль», «Заблокировать...»	`user.block`
Пароль	«Сбросить пароль»	`user.reset_password`
Удаление	«Опасная зона»	`user.delete`

Почту может изменить только сам пользователь, с подтверждением.

Блокировка

Заблокированный пользователь не может войти, его сессии завершаются, личные API-токены и лицензия отзываются. Группы и роли сохраняются, поэтому после разблокировки доступ возвращается. Лицензию нужно выдать заново. Все вкладки заблокированного доступны только для чтения.

Владельца организации и свой аккаунт заблокировать нельзя.

Владелец организации

Передача владения описана на странице **Владелец организации**.

Сброс пароля

Администратор с правом `user.reset_password` может отправить пользователю ссылку для установки нового пароля.

1. Откройте карточку пользователя («Люди» → «Пользователи», клик по строке).
2. В разделе «Вход и токены» нажмите «Сбросить пароль».
3. Подтвердите: «Отправить ссылку».

Ссылка придёт на почту пользователя. Текущий пароль работает, пока его не сменят. Чтобы разлогинить человека на всех устройствах, нажмите «Завершить все сессии».

Сброс можно выполнить сразу для нескольких людей: отметьте их в списке и выберите «Ещё» → «Сбросить пароль».

Если человек входит через SSO или LDAP, пароль управляется во внешней системе, сброс для него недоступен. Не получится сбросить пароль и приглашённому, который ещё не зарегистрировался: для него есть повторное приглашение.

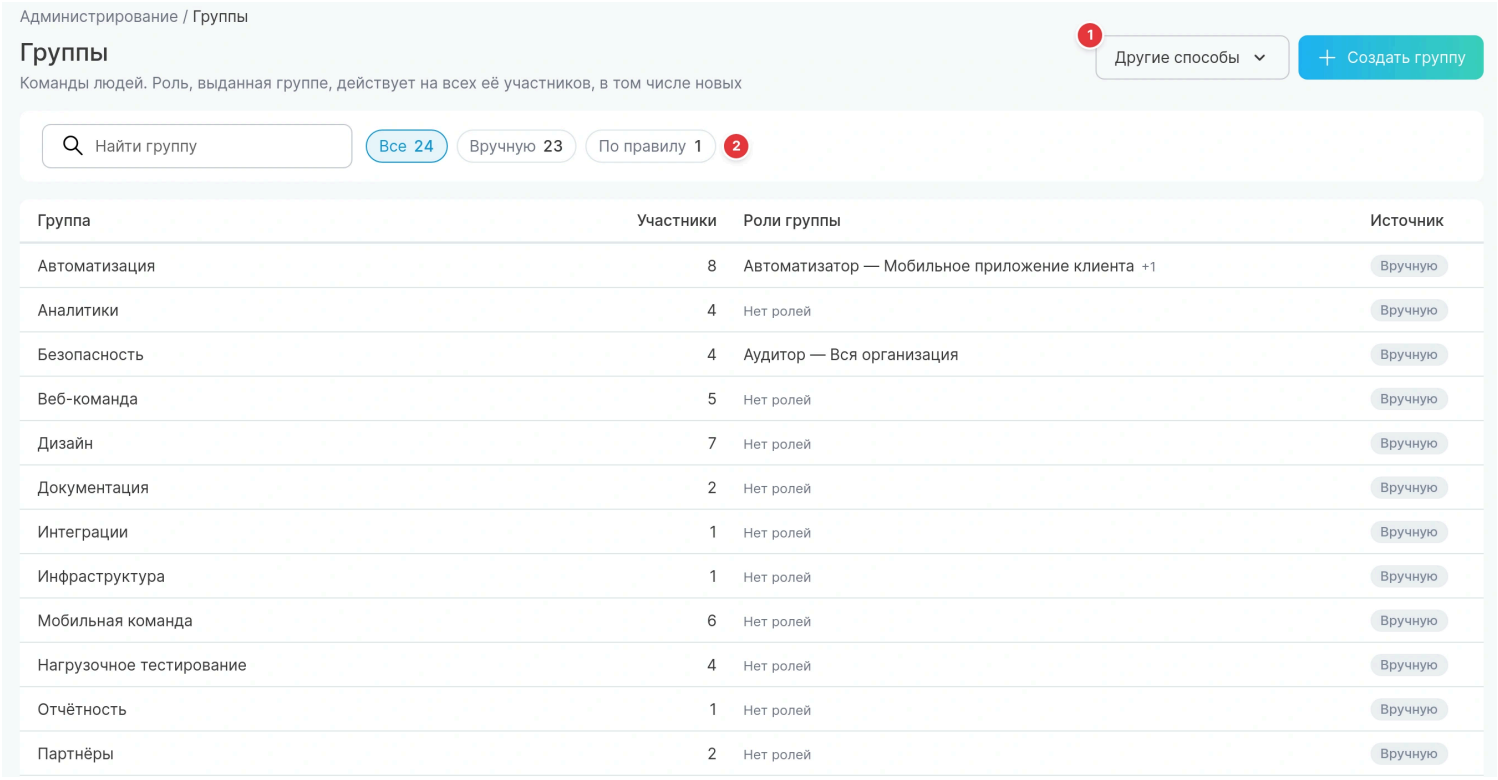
Группы

Группа - команда людей. Роль, выданная группе, действует на всех участников, в том числе на тех, кто вступит позже. Группы - основной способ давать доступ: новый человек получает права тем, что попадает в нужные группы.

Раздел «Люди» → «Группы» виден тем, у кого есть право `group.view`. Создавать группы, менять состав и настройки может `group.manage`.

Список групп

Список ищет по названию и описанию («Найти группу») и делится чипами источника (цифра 2): «Все», «Вручную», «По правилу»; чипы «SSO», «LDAP» и «Ошибка синхронизации» появляются, когда такие группы есть.



Колонка	Что показывает
Группа	Название и описание. У группы по правилу без описания показано правило, например «По правилу: почта на @t1.test». У группы по умолчанию пометка «по умолчанию»
Участники	Число участников

Колонка	Что показывает
Роли группы	Первая роль и где она выдана, остальные числом «+N»; «Нет ролей», если ролей нет
Источник	«Вручную», «По правилу», «SSO» или «LDAP»; красный чип «Ошибка синхронизации» с текстом ошибки в подсказке

Клик по строке открывает карточку группы справа. Список остаётся видимым и кликабельным: клик по другой строке переключает карточку.

Создание группы

Кнопка «Создать группу» (цифра 1) сразу создаёт ручную группу «Новая группа» (если имя занято, «Новая группа 2» и так далее) и открывает её карточку на настройках: дайте группе название и добавьте участников.

Группу по правилу, из SSO или из LDAP создают через «Другие способы» рядом с ней: «По правилу...», «Из SSO...», «Из LDAP...». В панели укажите название, описание и условие состава, затем нажмите «Создать группу». После создания открывается карточка новой группы.

Источник состава

Источник	Как формируется состав	Менять вручную
Вручную	Людей добавляют и убирают администраторы и владельцы группы	Да
SSO	Состав следует значению claim `groups` из единого входа. Нужно указать значение claim, например `qa-leads`	Нет
LDAP	Состав следует группе в каталоге LDAP. Нужно указать DN группы, например `CN=QA-Mobile,OU=Groups,DC=example,DC=com`	Нет

Источник	Как формируется состав	Менять вручную
По правилу	Состав следует условию на свойства пользователя	Нет

Источник нельзя изменить после создания группы. Состав группы из внешнего источника и по правилу пересчитывается при входе людей и по кнопке «Синхронизировать» в карточке группы.

Группы по правилу

Правило сравнивает свойство пользователя со значением (оператор «равно»):

Свойство	Пример значения
Домен почты	`contractor.example.com`
Атрибут SSO	название атрибута и значение, например `department = QA`
Атрибут LDAP	название атрибута и значение
Способ входа	пароль, SSO или LDAP

Перед сохранением виден предварительный результат: сколько людей подходит под правило. Правило меняют в карточке группы: «Настройки» → «Изменить правило». Если сменить правило, состав будет пересчитан, а роли группы последуют за новым составом.

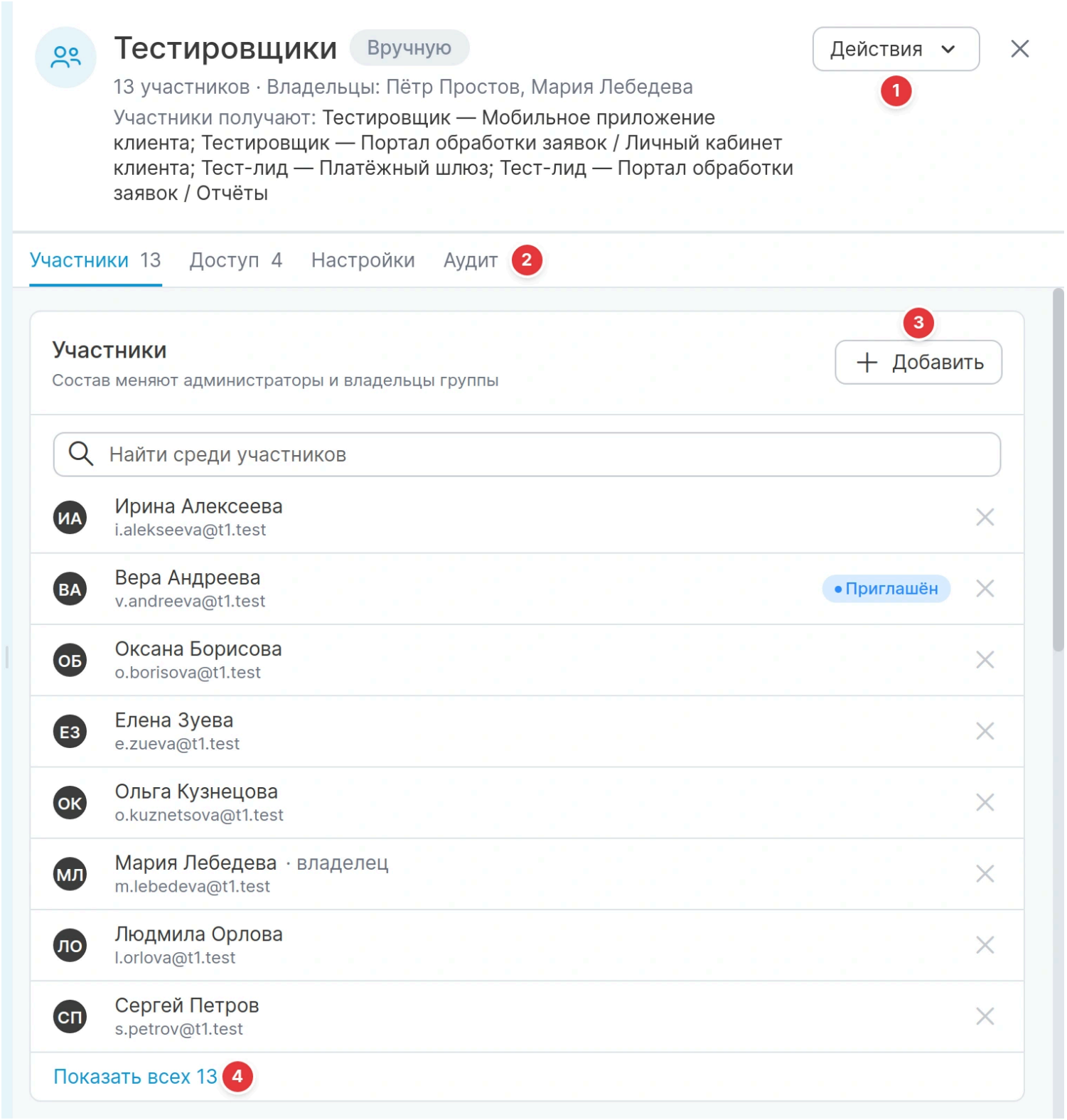
Как включить SSO и LDAP: **Корпоративный вход (SSO)**, **Авторизация через LDAP**. Состояние входа видно в разделе **Вход и SSO**.

Карточка группы

Карточка открывается справа от списка, без затемнения, и закрывается крестиком или клавишей Esc. В шапке: название, источник, число участников и владельцы, строка «Участники получают» со списком ролей группы. Меню «Действия» (цифра 1):

«Синхронизировать сейчас» (для группы не вручную), «Копировать группу», «Удалить группу».

Карточка состоит из секций с якорями «Участники · Доступ · Настройки · Аудит» над ними (цифра 2): клик по якорю прокручивает к секции.

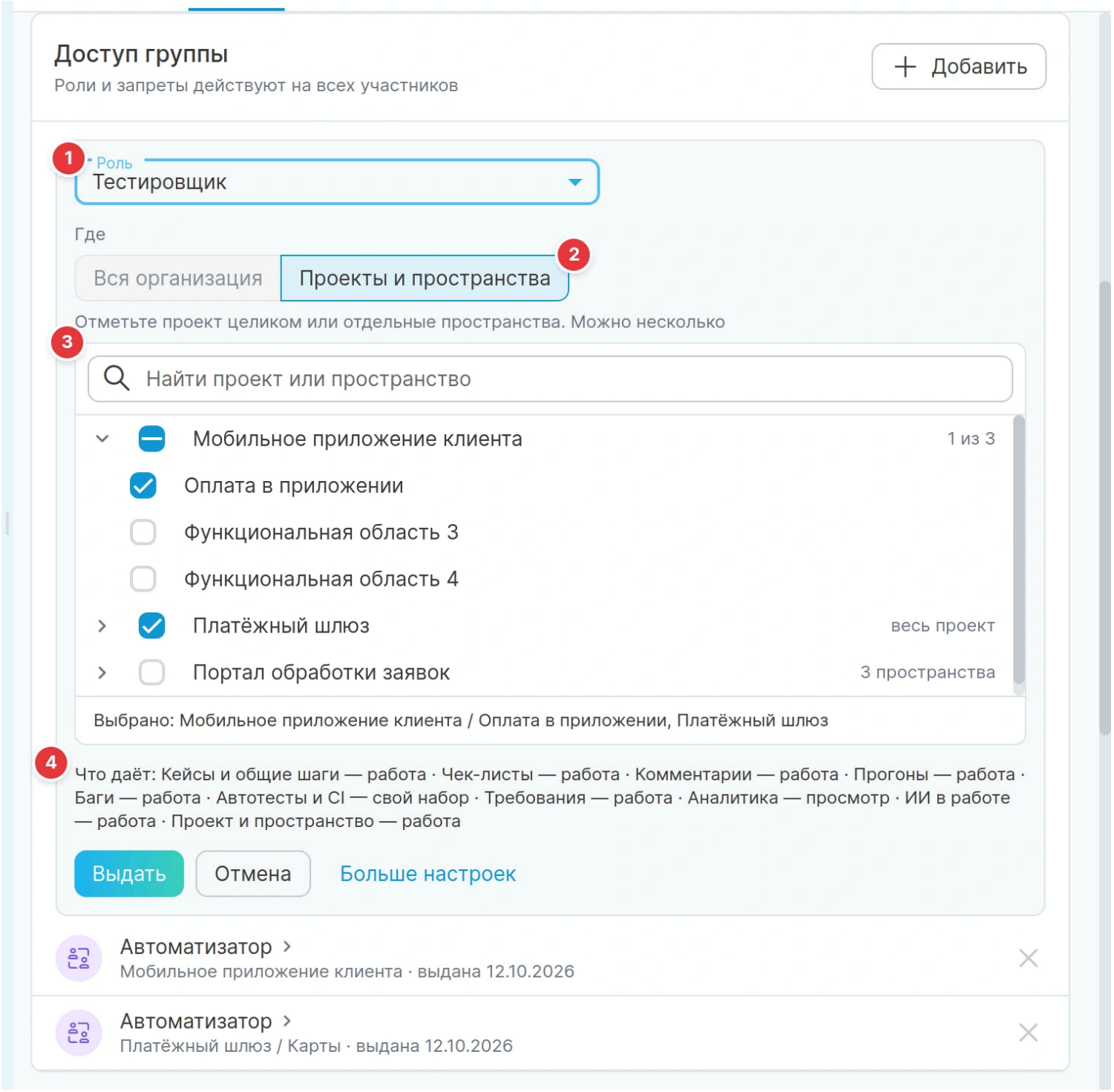


В секции «Участники» кнопка «Добавить» (цифра 3) открывает выбор людей, «Показать всех» (цифра 4) раскрывает весь состав.

Секция	Что в ней
Участники	Состав: имя, почта, отметка владельца, статус «Заблокирован» или «Приглашён». Видны первые восемь человек, остальные по кнопке «Показать всех», поиск среди участников. У ручной группы «Добавить» открывает выбор людей с поиском по имени и почте, крестик убирает человека (в тосте можно отменить). У группы по правилу, из SSO и LDAP вместо этого кнопка «Синхронизировать», состав руками не меняется
Доступ	Роли группы и запреты на неё: роли и запреты действуют на всех участников. «Добавить» → «Выдать роль» (роль и «Где» прямо в карточке) или «Поставить запрет». Клик по названию роли открывает её карточку, клик по подписи под ролью позволяет сменить роль или место. Крестик забирает роль или снимает запрет, в тосте можно отменить
Настройки	Название, описание, владельцы, «Группа по умолчанию», источник состава. Кнопки «Сохранить» и «Отменить» появляются, когда есть изменения
Аудит	Последние события группы: создание, участники, владельцы, выдачи и снятия ролей, запреты. Ссылка «Открыть в журнале аудита» ведёт в журнал с фильтром по группе (нужно право `audit.view`)

Где действует роль группы

При выдаче роли выберите роль (цифра 1) и «Где» (цифра 2): «Вся организация» (во всех проектах и пространствах, включая новые) или «Проекты и пространства». Во втором режиме отметьте проекты целиком или отдельные пространства, можно несколько (цифра 3). Каждая отмеченная область становится отдельной выдачей. Роли администрирования выдаются только на всю организацию. Под формой видно, что даёт роль по разделам (цифра 4): «Кейсы и общие шаги — работа», «Автотесты и CI — свой набор», и предупреждения, например что выдача уже есть.



Без права `access.view` роли группы видны только для чтения, а запреты не показываются.

Владельцы группы

У ручной группы могут быть владельцы. Они добавляют и убирают участников без прав администратора, но роли группе выдать не могут: это делает тот, у кого есть право

`access.manage` в нужном месте. Так тимлид сам ведёт состав своей команды и не может расширить её доступ.

Владельцем назначают только участника группы. Если человека убрать из группы, он перестаёт быть и владельцем. Права видеть список пользователей (`user.view`) владельцу не нужно: людей для добавления он ищет прямо в своей группе. Права `group.view` тоже не нужно: раздел «Администрирование → Группы» открывается владельцу и показывает только его группы.

Копия группы

Копия группы всегда ручная и не бывает группой по умолчанию. В неё попадают текущие участники, владельцы, роли и запреты исходной группы, название по умолчанию «{название} (копия)». Нужно право `group.manage`, а каждую роль и каждый запрет группы вы должны уметь выдать сами в том же месте. Если хоть одну роль или запрет выдать нельзя, копия не создаётся, и DoQA показывает, какая роль и где мешает.

Группа по умолчанию

Ручную группу можно отметить как группу по умолчанию. Новые люди автоматически попадают в неё при приглашении и при первом входе через SSO. В окне приглашения такие группы отмечены заранее, отметку можно снять.

Удаление группы

«Действия» → «Удалить группу». Под шапкой карточки появится вопрос «Удалить группу «...»? Роли группы исчезнут, у участников останется только личный доступ» с кнопками «Отмена» и «Удалить». Группа исчезает вместе со своими ролями и запретами, участники теряют роли, которые получали через неё.

Как снять доступ, пришедший через группу

Способ	Результат
Убрать человека из группы	Он теряет все роли группы, во всех проектах
Поставить **запрет**	Человек остаётся в группе, но конкретный доступ теряет

Сервисные аккаунты

Сервисный аккаунт - субъект для CI, скриптов и интеграций. Он получает доступ так же, как человек, но войти в интерфейс не может и к API обращается с **токенами**. Лицензия сервисным аккаунтам не нужна.

Раздел «Люди» → «Сервисные аккаунты» виден тем, у кого есть право `service_account.manage`.

Создание

1. Нажмите «Создать аккаунт».
2. Введите название: короткое, уникальное, легко узнаваемое в логах, например `ci-android` или `bi-export`.
3. Добавьте описание: для чего аккаунт и кто им пользуется.
4. Выберите ответственного. Если оставить пустым, ответственным станете вы.
5. Сохраните, затем выдайте аккаунту роль и выпустите токен.

Без роли аккаунт ничего не может, даже с токеном.

Вкладки

Вкладка	Что в ней
Доступ	Роли аккаунта и где они выданы. Выдать, сменить, отозвать роль
Токены	Токены аккаунта
Аудит	События по аккаунту (нужно право `audit.view`)

Доступ

Рекомендуется роль «Агент CI» (отправка результатов и чтение плана прогона) или «Читатель API» (просмотр без выгрузок). Можно использовать и свои роли. Отозвав роль, вы отнимаете права и у токенов аккаунта: токен не может быть шире аккаунта.

Токены

Нажмите «Выпустить токен».

Параметр	Описание
Название	Для чего токен, например bi-export
Срок действия	30 дней, 90 дней, 1 год или бессрочный. Бессрочный работает, пока его не отзовут, поэтому лучше выбирать ограниченный срок
Где действует	Вся организация либо проекты и пространства. Токен на проекте действует и в его пространствах
Права	Весь набор аккаунта либо выбранные права. Нельзя выбрать то, чего нет у аккаунта; опасные права помечены

После создания скопируйте токен: он показывается один раз. Отозвать токен можно в списке; всё, что его использует, сразу перестанет работать.

Ответственный

Ответственный отвечает за аккаунт: токены и доступ пересматриваются вместе с ним. Если ответственный заблокирован, токены аккаунта продолжают работать, но аккаунт помечается «Ответственный заблокирован. Назначьте другого».

Управляемые аккаунты

Некоторые аккаунты создаёт и обновляет подключение CI или система: они помечены «управляется подключением» или «управляется системой». Изменить или удалить их здесь

нельзя, но доступом и токенами управлять можно. Аккаунты, созданные из токенов проектов при обновлении, помечены «перенесён из токена проекта» и не имеют собственного ограничения: они могут всё, что может аккаунт.

Удаление

Удаление записывается в журнал аудита. Роли аккаунта снимаются, токены отзываются, скрипты с ними перестают работать. Чтобы остановить аккаунт ненадолго, отзовите его токены: аккаунт и доступ сохранятся.

API-токены

Токен - секрет, которым скрипт, задача CI или интеграция обращается к API. В DoQA три вида токенов.

Вид	Кому принадлежит	Где создаётся
Личный	Пользователю	Профиль, вкладка API-токены
Токен проекта	Проекту (для отправки результатов автотестов)	Администрирование → проект → «Токены»
Токен сервисного аккаунта	Сервисному аккаунту	Администрирование → сервисный аккаунт → «Токены»

Общие правила:

- Секрет показывается один раз, сразу после создания. Закройте окно, не скопировав его, и увидеть его уже не сможет никто, включая администраторов. Хранятся только хэш и маска.
- Токен передают заголовком `Authorization: Bearer <токен>`.
- Токен никогда не шире прав владельца и пересчитывается при каждом запросе: если у владельца права пропали, токен их тоже теряет.
- У токена видно дату последнего использования.
- Отозванный токен восстановить нельзя: выпустите новый и обновите скрипты. Отзыв действует сразу.

Личные токены

Личный токен действует от вашего имени с подмножеством ваших прав.

Параметр	Правило
Количество	До 10 токенов на пользователя
Срок действия	Необязателен: 30, 90 или 180 дней, 1 год, своя дата (до года вперёд) или «Бессрочный». Бессрочный токен действует, пока его не отзовут, поэтому лучше выбирать ограниченный срок

Параметр	Правило
Где действует	Можно ограничить местами (организация, проекты, пространства). Пустой список означает все ваши места
Права	Можно ограничить набором прав. Пустой набор означает все ваши права. Выбрать право шире своего нельзя
Название	До 255 символов

Блокировка пользователя отзывает его личные токены. Администратор с правом `user.tokens_manage` видит личные токены другого человека в его карточке (раздел «Вход и токены») и может отозвать один или все. Создать токен за другого администратор не может.

Токены, выпущенные до обновления

Личные токены, созданные до перехода на новую систему прав, продолжают работать с правами владельца на всех его местах. Токены без срока остаются бессрочными.

Токены проекта

Токены проекта отправляют в проект результаты автотестов из CI, см. **Отправка результатов**. Вкладка «Токены» проекта видна тем, у кого есть право `project_token.view`, создавать и отзывать токены можно с `project_token.manage`.

- Создание: «Создать токен», название (например, gitlab-ci). Секрет показывается один раз.
- «Переименовать» меняет название. «Отозвать» останавливает отправку результатов с этим токеном: запросы получат 401.
- В таблице видны автор токена, подключение CI, дата создания и последнее использование. Если автор заблокирован, строка помечена. Токены, которые не использовались 90 дней, подсвечены.

Токен проекта работает как сервисный аккаунт с ролью «Агент CI» на этом проекте: он может отправлять результаты и читать план прогона. Отозвать его доступ можно, сняв роль у аккаунта.

Раздел «Подключения» → «API-токены»

Показывает токены всех проектов: что использует CI и что стоит отозвать. Есть поиск по названию токена или проекта, фильтры, пометка «Не использовался более 90 дней» и массовый отзыв. Личные токены находятся в карточке пользователя.

Токены сервисных аккаунтов

См. Сервисные аккаунты.

Роли и права

Доступ в DoQA строится на ролях. Роль - это набор **прав**. Человеку, группе или сервисному аккаунту выдают роль в нужном месте: на всю организацию, на проект или на пространство. Права по одному никому не выдаются. Если нужен особый набор, создайте свою роль, обычно копией готовой.

Роли находятся в разделе «Люди» → «Роли» администрирования. Раздел виден тем, у кого есть право `role.view`.

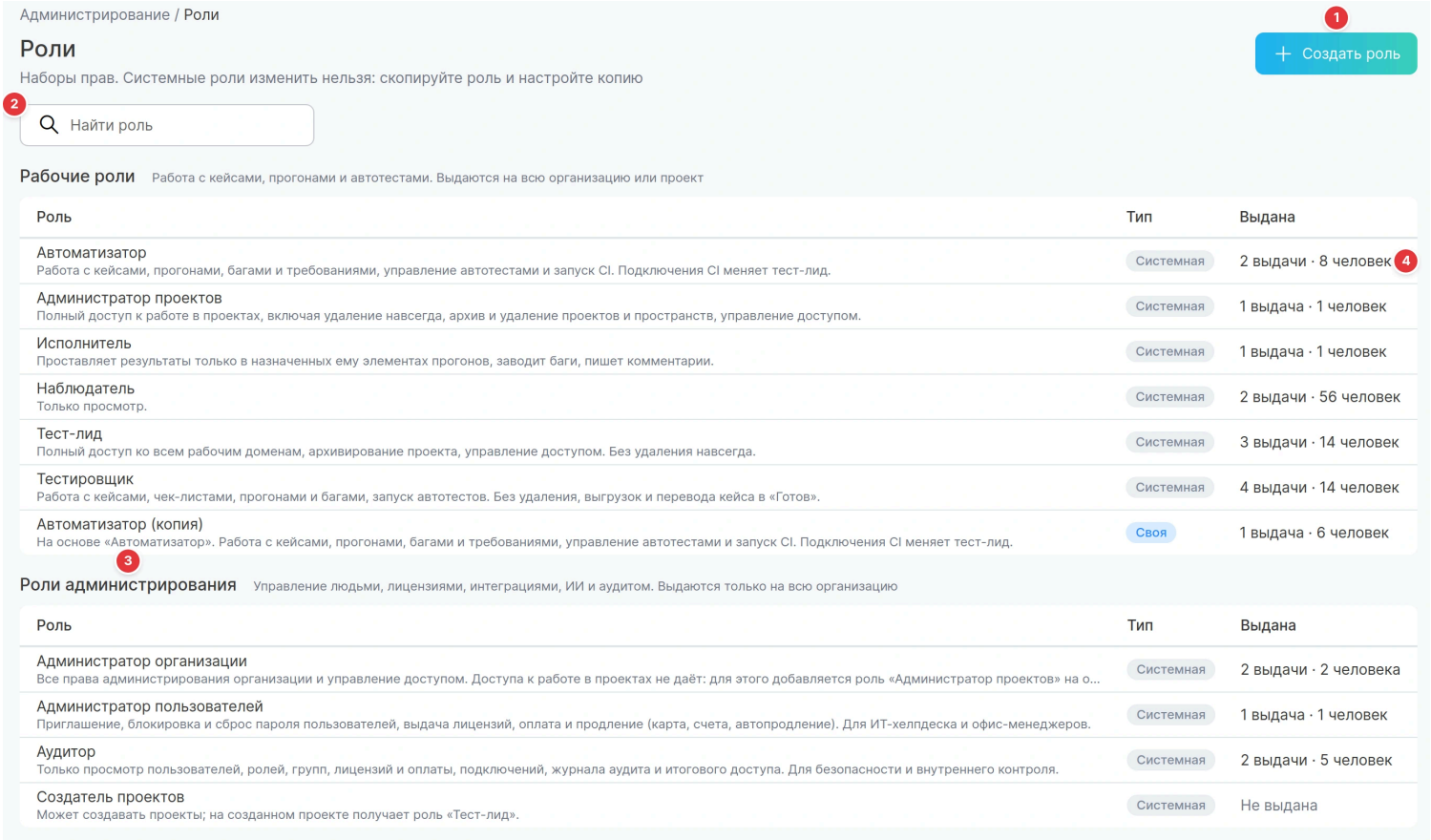
Два вида ролей

Вид	Что содержит	Где выдаётся
Роли администрирования	Права организации: люди, группы, лицензии, подключения, ИИ, аудит, вход	Только на всю организацию
Рабочие роли	Работа с кейсами, прогонами, автотестами, требованиями, настройки проекта и пространства	На организацию (действует во всех проектах, включая новые), на проект или на пространство
Роли сервисных аккаунтов	Узкие наборы для CI и скриптов	Сервисным аккаунтам, см. **Сервисные аккаунты**

Вид роли задаётся при создании по базовой роли и позже не меняется.

Список ролей

Роли показаны тремя блоками по видам (цифра 3). Поиск «Найти роль» (цифра 2) ищет по названию и описанию, пустые при поиске блоки скрываются. В строке: название и описание (у своей роли описание начинается с «На основе «X».»), тип «Системная» или «Своя» и «Выдана»: число выдач и людей без повторов, с учётом участников групп, например «3 выдачи · 12 человек», или «Не выдана» (цифра 4). Клик по строке открывает карточку роли справа от списка.



Системные роли

Системные роли поставляются с DoQA. Изменить или удалить их нельзя, но можно скопировать и настроить копию.

Роли администрирования

Роль	Что может
Администратор организации	Все права администрирования и управление доступом. Доступа к работе в проектах не даёт: для этого добавляется роль «Администратор проектов» на организации
Администратор пользователей	Приглашает, блокирует, сбрасывает пароль, выдаёт лицензии и управляет ими, оплачивает и продлевает лицензии в облаке, видит группы и роли
Аудитор	Только просмотр: пользователи, роли, группы, лицензии, счета и платежи, подключения, расход ИИ, журнал аудита, состояние входа, итоговый доступ

Роль	Что может
Создатель проектов	Создаёт проекты. На созданном проекте получает роль «Тест-лид»

Рабочие роли

Роль	Что может
Администратор проектов	Полный доступ к работе, удаление навсегда, архивирование и удаление проектов и пространств, управление доступом
Тест-лид	Полный доступ ко всем рабочим доменам, архивирование проекта, управление доступом. Без удаления навсегда
Автоматизатор	Кейсы, прогоны, баги, требования, автотесты и запуск CI. Подключения CI меняет тест-лид
Тестировщик	Кейсы, чек-листы, прогоны, баги, запуск автотестов. Без удаления, выгрузок и перевода кейса в «Готов»
Исполнитель	Проставляет результаты только в назначенных ему элементах прогонов, заводит баги, пишет комментарии
Наблюдатель	Только просмотр

Роли сервисных аккаунтов

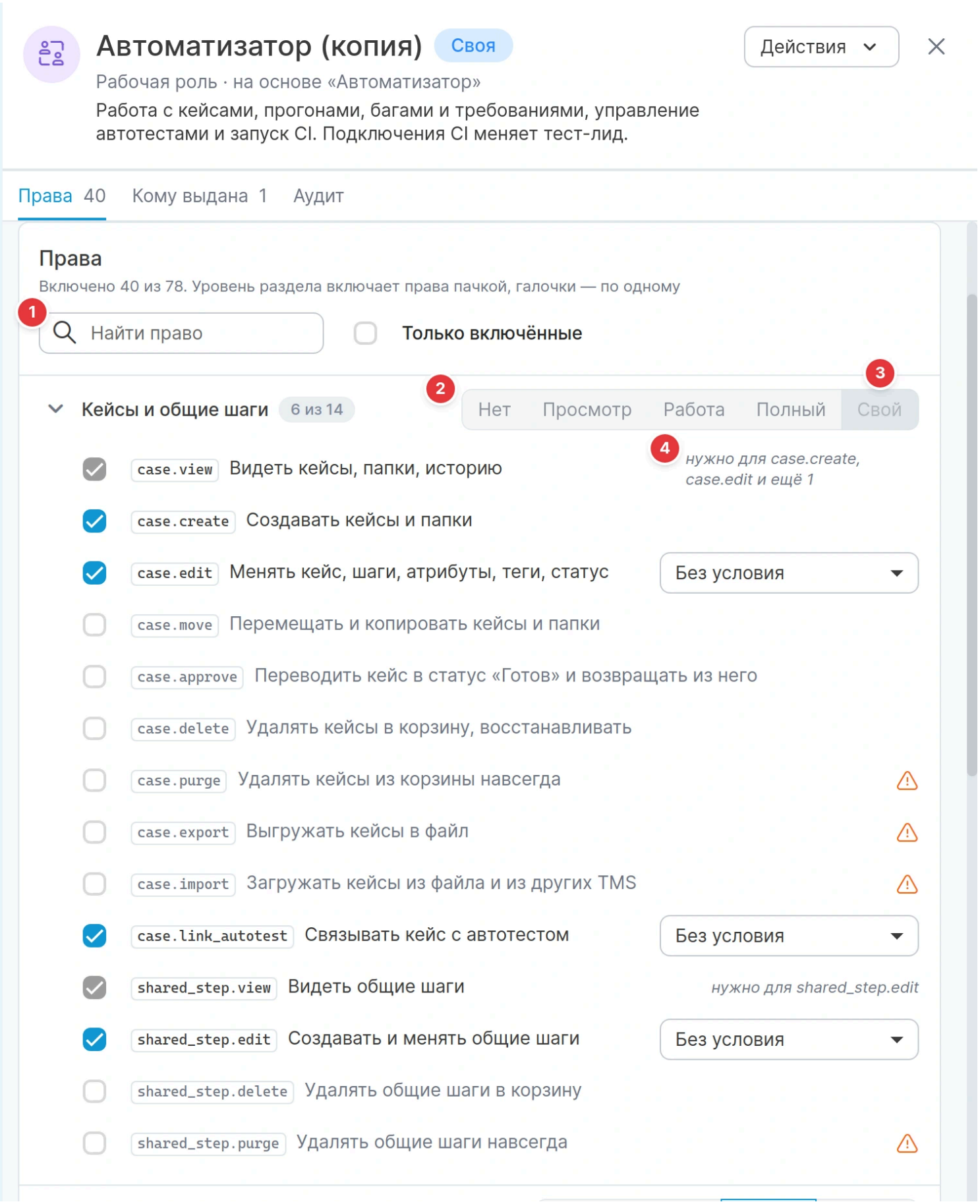
Роль	Для чего
Агент CI	Отправка результатов автотестов и чтение плана прогона
Читатель API	Скрипты, которым нужен просмотр во всех рабочих доменах, без выгрузок

Свои роли

«Создать роль» (цифра 1 на рисунке списка) сразу создаёт роль «Новая роль» копией системной роли «Наблюдатель» и открывает её карточку: остаётся задать название и включить нужные права. Роль администрирования или роль сервисного аккаунта создают так

же, копией системной роли своего вида: «Действия» → «Создать копию» в её карточке. У своей роли название и описание сохраняются сами, отдельной кнопки нет.

Права в карточке роли:



- Права сгруппированы по разделам. Для каждого раздела можно выбрать уровень «Нет», «Просмотр», «Работа» или «Полный» (цифра 2), либо отметить права по одному: тогда раздел помечается как «Свой» (цифра 3). Под заголовком видно, сколько прав включено.

- Уровни есть и у рабочих разделов, и у разделов администрирования. «Просмотр» даёт только видеть, «Работа» добавляет повседневные действия, «Полный» добавляет остальное. Удаление навсегда, архивирование и удаление проекта, удаление пространства, управление доступом (`access.manage`) и отправка результатов автотестов (`autotest_result.ingest`) в уровни не входят: их включают только отдельной галочкой, и на уровень раздела они не влияют.
- Право можно сузить условием: «Только свои», «Где я ответственный», «Назначенные мне», «В статусах...» (статусы вводятся через запятую). Какие условия доступны, зависит от права, см. **Справочник прав**.
- Если право требует другого (например, правка требует просмотра), нужное включится само. Право, на котором держатся другие, снять нельзя: рядом написано, для каких прав оно нужно (цифра 4).
- Значок предупреждения отмечает необратимые и массовые действия.
- Поиск «Найти право» и отметка «Только включённые» (цифра 1) помогают в длинном списке. В роли должно быть хотя бы одно право.

Изменения прав своей роли сохраняются сразу. Если роль выдана 50 людям или больше, правки копятся: появляется строка «Изменено прав: N» с кнопками «Проверить и сохранить» и «Отменить». Проверка показывает, сколько выдач и людей затронет изменение и какие права люди получат или потеряют, затем нужно подтвердить сохранение. Права пересчитываются в течение нескольких секунд, изменение попадает в **журнал аудита**.

Рекомендация

Своих ролей рекомендуется не больше 30: когда их много, выбрать роль при выдаче доступа трудно. Похожие роли можно найти и объединить в **пересмотре доступа**.

Карточка роли

Карточка открывается справа от списка, без затемнения. В шапке: название, «Системная» или «Своя», вид роли и базовая роль, описание. У системной роли вверху плашка: изменить её нельзя, нужен другой набор, создайте копию. Меню «Действия»: «Создать копию» и у своей роли «Удалить роль».

Секция	Что в ней
Права	Права роли по разделам, правка для своих ролей (см. выше)
Кому выдана	Группы, люди и сервисные аккаунты, которым выдана роль, и где: например «Группа · 6 участников · Вся организация» или «Личная роль · Проект». «Выдать» открывает форму прямо в секции: кому (группы и люди, у роли сервисного аккаунта сервисные аккаунты) и где (вся организация или отмеченные проекты и пространства, каждое место отдельной выдачей). Клик по группе открывает её карточку, клик по подписи позволяет сменить роль или место, крестик забирает роль (в тосте можно отменить)
Аудит	Последние события роли: создание, изменения прав («Включены права: ...», «Выключены права: ...»), выдачи и снятия. Ссылка ведёт в журнал аудита с фильтром по роли

Удалить можно только свою роль, которая никому не выдана. Если роль выдана, DoQA напомнит, сколько выдач нужно сначала снять. Иначе под шапкой появится вопрос «Удалить роль «...»? Её никому не выдавали» с кнопками «Отмена» и «Удалить». Удаление записывается в журнал аудита, роль можно восстановить оттуда в течение 30 дней.

Защита от повышения прав

- Выдать роль можно, только если каждое право роли есть у вас в этом месте.
- Изменить роль можно, только если вы сами имеете все права, которые добавляете.
- Нельзя снять последнюю выдачу, дающую управление доступом на всей организации.
- Права администрирования выдаются и запрещаются только на всю организацию.

Лицензия и роли

Лицензия не меняет ролей. Без неё любая роль работает только на просмотр: права, которым нужна лицензия, не действуют (они отмечены в **справочнике прав**). Оплата устроена отдельно: её выдают права `billing.view` и `billing.manage`, а `license.manage` теперь относится к ключу серверной версии и заявкам на продление. Подробнее: **Лицензии и оплата (облачная версия)** и **Лицензии (серверная версия)**.

Справочник прав

Полный перечень прав, из которых собираются **роли**. Право описывает одно действие над одним видом объектов. Человеку права по одному не выдаются: их получают через роль.

Как читать таблицы

Столбец	Что означает
Уровень	Где право действует: организация, проект, пространство или любой узел. Права организации действуют только при роли, выданной на всю организацию
Условия	Как можно сузить право внутри роли: `own` (только свои), `responsible` (где я ответственный), `assigned` (назначенные мне), `status_in` (только в выбранных статусах)
Лицензия	Право меняет данные и работает только у людей с **лицензией**. Без лицензии доступен просмотр
Опасное	Право помечено как опасное: оно попадает в отчёт «Опасные права» **пересмотра доступа**

Уровни прав

В каждом разделе права собраны в три уровня. На них построены системные роли, ими же раздел своей роли включается целиком:

Уровень	Рабочие разделы	Разделы администрирования
Просмотр	Только чтение	Только видеть: списки, настройки, журналы
Работа	Просмотр плюс создание и изменение. Комментарии и баги на этом уровне правятся только свои	Повседневные действия: приглашать и блокировать людей, менять группы, назначать лицензии, оплачивать, создавать проекты, менять подключения и настройки ИИ

Уровень	Рабочие разделы	Разделы администрирования
Полный	Работа плюс удаление в корзину, выгрузки, перевод кейса в «Готов», настройки. Комментарии и баги правятся любые	Необратимое и сама система прав: удалять людей, управлять чужими токенами, своими ролями и сервисными аккаунтами, ключом серверной версии, переиндексацией поиска

Некоторые права в уровни не входят: их включают только отдельной галочкой, и на уровень раздела они не влияют. Это удаление навсегда (`*.purge`), архивирование и удаление проекта (`project.archive`, `project.delete`), удаление пространства (`space.delete`), управление доступом (`access.manage`) и отправка результатов автотестов (`autotest_result.ingest`).

Кейсы и общие шаги

Право	Уровень	Условия	Лицензия	Опасное	Описание
`case.view`	пространство	-	нет	нет	Видеть кейсы, папки, историю
`case.create`	пространство	-	да	нет	Создавать кейсы и папки
`case.edit`	пространство	own, responsible, status_in	да	нет	Менять кейс, шаги, атрибуты, теги, статус
`case.move`	пространство	own	да	нет	Перемещать и копировать кейсы и папки
`case.approve`	пространство	-	да	нет	Переводить кейс в статус «Готов» и возвращать из него
`case.delete`	пространство	own, status_in	да	нет	Удалять кейсы в корзину, восстанавливать

Право	Уровень	Условия	Лицензия	Опасное	Описание
`case.purge`	пространство	-	да	да	Удалять кейсы из корзины навсегда
`case.export`	пространство	-	нет	да	Выгружать кейсы в файл
`case.import`	пространство	-	да	да	Загружать кейсы из файла и из других TMS
`case.link_autotest`	пространство	own	да	нет	Связывать кейс с автотестом
`shared_step.view`	пространство	-	нет	нет	Видеть общие шаги
`shared_step.edit`	пространство	own	да	нет	Создавать и менять общие шаги
`shared_step.delete`	пространство	own	да	нет	Удалять общие шаги в корзину
`shared_step.purge`	пространство	-	да	да	Удалять общие шаги навсегда

Чек-листы

Право	Уровень	Условия	Лицензия	Опасное	Описание
`checklist.view`	пространство	-	нет	нет	Видеть чек-листы
`checklist.create`	пространство	-	да	нет	Создавать чек-листы
`checklist.edit`	пространство	own, responsible	да	нет	Менять пункты и свойства чек-листов

Право	Уровень	Условия	Лицензия	Опасное	Описание
`checklist.move`	пространство	own	да	нет	Перемещать и копировать чек-листы
`checklist.delete`	пространство	own	да	нет	Удалять чек-листы в корзину, восстанавливать
`checklist.purge`	пространство	-	да	да	Удалять чек-листы навсегда
`checklist.export`	пространство	-	нет	да	Выгружать чек-листы
`checklist.import`	пространство	-	да	да	Загружать чек-листы

Комментарии

Право	Уровень	Условия	Лицензия	Опасное	Описание
`comment.view`	пространство	-	нет	нет	Видеть комментарии к кейсам, чек-листам, прогонам
`comment.create`	пространство	-	да	нет	Писать комментарии
`comment.edit`	пространство	own	да	нет	Менять комментарии (на уровне «Работа» только свои, на «Полном» любые)
`comment.delete`	пространство	own	да	нет	Удалять комментарии (на уровне «Работа» только свои, на «Полном» любые)

Право	Уровень	Условия	Лицензия	Опасное	Описание
`comment.resolve`	пространство	-	да	нет	Закрывать обсуждения

Прогоны

Право	Уровень	Условия	Лицензия	Опасное	Описание
`run.view`	пространство	-	нет	нет	Видеть прогоны, элементы, попытки, отчёт, историю
`run.create`	пространство	-	да	нет	Создавать прогоны, в том числе из отчёта и копией
`run.edit`	пространство	own, status_in	да	нет	Менять свойства прогона, состав, порядок
`run.execute`	пространство	assigned	да	нет	Проставлять результаты элементов, шагов, проверок
`run.assign`	пространство	-	да	нет	Назначать исполнителей
`run.close`	пространство	own	да	нет	Завершать и переоткрывать прогон
`run.delete`	пространство	own	да	нет	Удалять прогоны в корзину, восстанавливать

Право	Уровень	Условия	Лицензия	Опасное	Описание
`run.purge`	пространство	-	да	да	Удалять прогоны навсегда
`run.export`	пространство	-	нет	да	Выгружать прогон и отчёт
`scheduled_run.view`	пространство	-	нет	нет	Видеть плановые прогоны
`scheduled_run.manage`	пространство	own	да	нет	Создавать, менять, включать, удалять плановые прогоны
`scheduled_run.run_now`	пространство	own	да	нет	Запускать плановый прогон вне расписания

Баги

Право	Уровень	Условия	Лицензия	Опасное	Описание
`bug.view`	пространство	-	нет	нет	Видеть баги и кластеры ошибок
`bug.create`	пространство	-	да	нет	Заводить баги том числе в трекер
`bug.edit`	пространство	own	да	нет	Менять баги (уровне «Рабо только свои, н «Полном» любые)

Право	Уровень	Условия	Лицензия	Опасное	Описание
`bug.delete`	пространство	own	да	нет	Удалять баги корзину, восстанавлив
`bug.purge`	пространство	-	да	да	Удалять баги навсегда
`error_cluster.reclassify`	пространство	-	да	нет	Переносить ошибки межд кластерами и категориями

Автотесты и CI

Право	Уровень	Условия	Лицензия	Опасное	Описание
`autotest.view`	пространство	-	нет	нет	Видеть каталог автотестов, стабильность, запуски CI, влияние
`autotest.manage`	пространство	-	да	нет	Теги, карантин, связи с кейсами, известные проблемы
`autotest.launch`	пространство	-	да	нет	Запускать пайплайны, перезапускать упавшие
`autotest.control`	пространство	own	да	нет	Отменять и повторять запуски CI

Право	Уровень	Условия	Лицензия	Опасное	Описание
`ci.view`	пространство	-	нет	нет	Видеть CI-подключения, привязки, расписания CI
`ci.manage`	пространство	-	да	да	Подключать CI к пространству, менять привязки, переменные, расписания
`autotest_policy.manage`	пространство	-	да	нет	Quality Gate, авто-ретрай, flaky, анализ влияния, категории ошибок, автосоздание кейсов
`autotest_result.ingest`	пространство	-	да	нет	Отправлять результаты автотестов и читать план прогона

Требования

Право	Уровень	Условия	Лицензия	Опасное	Описание
`requirement.view`	пространство	-	нет	нет	Видеть тр покрытие синхрони
`requirement.link`	пространство	-	да	нет	Связыват отвязыва требован

Право	Уровень	Условия	Лицензия	Опасное	Описание
`requirement.actualize`	пространство	-	да	нет	Актуализ кейсы по требован
`requirement.export`	пространство	-	нет	да	Выгружат
`requirement.connector_manage`	пространство	-	да	нет	Настраив коннекто пространс пересинх
`requirement.purge`	пространство	-	да	да	Удалять в требован пространс

Аналитика

Право	Уровень	Условия	Лицензия	Опасное	Описание
`dashboard.view`	пространство	-	нет	нет	Дашборды качества и автоматизации
`dashboard.view_people`	пространство	-	нет	нет	Активность отдельных пользователей
`dashboard.export`	пространство	-	нет	да	Выгружать данные дашбордов

ИИ в работе

Право	Уровень	Условия	Лицензия	Опасное	Описание
`ai.use`	пространство	-	да	нет	Генерация и ревью кейсов, чек-листов, багов,

Право	Уровень	Условия	Лицензия	Опасное	Описание
					отчётов, тестовых данных
`ai.bulk_generate`	пространство	-	да	да	Массовая генерация кейсов по требованиям или описанию

Проект и пространство

Право	Уровень	Условия	Лицензия	Опасное	Описание
`project.view`	проект	-	нет	нет	Видеть проект его пространст
`project.edit`	проект	-	да	нет	Название, описание, лого проекта
`project.archive`	проект	-	да	да	Архивировать восстанавлива проект
`project.delete`	проект	-	да	да	Удалять проек навсегда
`space.manage`	проект	-	да	нет	Создавать, переименовыв архивировать пространства
`space.delete`	проект	-	да	да	Удалять пространство навсегда
`project_settings.manage`	проект	-	да	нет	Атрибуты, клю тегов, конфигурации, параметры проекта

Право	Уровень	Условия	Лицензия	Опасное	Описание
`tracker.manage`	проект	-	да	нет	Привязка прое к трекеру и соответствия полей
`project_token.view`	проект	-	нет	нет	Видеть токены проекта
`project_token.manage`	проект	-	да	да	Создавать, переименовыв отзывать токен проекта
`outbound_webhook.manage`	пространство	-	да	нет	Исходящие вебхуки пространства
`bin.view`	пространство	-	нет	нет	Видеть корзин пространства
`access.view`	любой узел	-	нет	нет	Видеть привяз итоговый досту на узле и ниже
`access.manage`	любой узел	-	да	да	Выдавать и снимать досту на узле и ниже ставить запрет

Люди и доступ

Право	Уровень	Условия	Лицензия	Опасное	Описание
`user.view`	организация	-	нет	нет	Список пользователей организации с почтами и статусами

Право	Уровень	Условия	Лицензия	Опасное	Описание
`user.invite`	организация	-	да	нет	Приглашать, повторять и отзывать приглашения
`user.edit`	организация	-	да	нет	Менять имя и фамилию
`user.block`	организация	-	да	да	Блокировать и разблокировать
`user.reset_password`	организация	-	да	да	Отправлять сброс пароля, завершать сессии
`user.delete`	организация	-	да	да	Удалять пользователя с обезличиванием данных
`user.tokens_manage`	организация	-	да	да	Видеть и отзывать личные токены других
`role.view`	организация	-	нет	нет	Видеть роли и их состав
`role.manage`	организация	-	да	да	Создавать, менять, удалять свои роли
`group.view`	организация	-	нет	нет	Видеть группы и состав
`group.manage`	организация	-	да	да	Создавать группы, менять состав, настраивать синхронизацию

Право	Уровень	Условия	Лицензия	Опасное	Описание
`service_account.manage`	организация	-	да	да	Сервисные аккаунты и их токены

Лицензии и оплата

Право	Уровень	Условия	Лицензия	Опасное	Описание
`license.view`	организация	-	нет	нет	Видеть лицензии, сроки и кто ждёт лицензию
`license.assign`	организация	-	да	нет	Делать наблюдателем и возвращать полный доступ
`license.manage`	организация	-	нет	да	Замена ключа и запрос продления в серверной версии, заявка на продление пробного периода в облаке
`billing.view`	организация	-	нет	нет	Платежи и счета, документы, реквизиты и автопродление: только просмотр. Только облако
`billing.manage`	организация	-	нет	да	Покупка и оплата, выставление и отмена счетов, реквизиты, привязка и отключение автопродления, коины. Только облако

`billing.manage` включает `billing.view`, а `billing.view` включает `license.view`, поэтому отдельно выдавать просмотр не нужно. Права оплаты не требуют лицензии. В серверной

версии `billing.*` в редакторе ролей не показываются.

Проекты организации

Право	Уровень	Условия	Лицензия	Опасное	Описание
`project.create`	организация	-	да	нет	Создавать проекты; создатель получает «Тест-лида» на новом проекте

Подключения

Право	Уровень	Условия	Лицензия	Опасное	Описание
`integration.view`	организация	-	нет	нет	Видеть подключения трекеров и CI и где они используются
`integration.manage`	организация	-	да	да	Подключать, менять, проверять и отключать

ИИ организации

Право	Уровень	Условия	Лицензия	Опасное	Описание
`ai_settings.manage`	организация	-	да	да	Провайдер, модель, токены провайдера, промпты
`ai_usage.view`	организация	-	нет	нет	История запросов и расход монет

Безопасность и система

Право	Уровень	Условия	Лицензия	Опасное	Описание
`audit.view`	организация	-	нет	нет	Журнал аудита организации
`sign_in.view`	организация	-	нет	нет	Состояние SSO и LDAP
`notification.manage`	организация	-	да	нет	Каналы и шаблоны уведомлений организации, повтор доставки
`search.reindex`	организация	-	да	да	Переиндексация поиска

Доступ на всю организацию

Рабочая роль, выданная на всю организацию, действует во всех проектах и пространствах, в том числе в тех, что появятся позже. Так даётся доступ «ко всем проектам», который раньше задавался переключателем уровня доступа.

Как выдать

1. Откройте карточку пользователя («Откуда доступ» → «Добавить» → «Выдать личную роль») или карточку группы (секция «Доступ» → «Добавить» → «Выдать роль»).
2. Выберите рабочую роль и в «Где» оставьте «Вся организация».
3. Нажмите «Выдать». Для полной панели выдачи нажмите «Больше настроек».

В окне приглашения нового пользователя то же самое делается блоком «Где»: оставьте проекты пустыми, чтобы выдать роль на всю организацию.

Соответствие прежним уровням

При обновлении бывшие общие уровни переведены в роли:

Было	Стало
Просмотр всех проектов	«Наблюдатель» на организации
Редактирование всех проектов	«Тестировщик» на организации
Администрирование всех проектов	«Тест-лид» на организации
Роль «Супер админ»	«Администратор организации» и «Администратор проектов» на организации
Роль «Администратор пользователей»	«Администратор пользователей»
Роль «Создатель проектов»	«Создатель проектов»

Было	Стало
Уровень «Скрыть»	**Запрет** «весь доступ» на проекте

Полная таблица перехода: **Переход на новую систему прав**.

Ограничения

- Роли администрирования выдаются только на организацию.
- Закрытые проекты и пространства роль с организации не пропускают, см. **Закрытые проекты и пространства**.
- **Запрет** на проекте сильнее роли на организации.
- Права на изменение работают только у людей с лицензией.

Доступ к проектам и пространствам

Доступ к проекту - это роль, выданная человеку, группе или сервисному аккаунту на проекте. Роль на проекте действует во всех его пространствах. Более узкий доступ выдают на пространстве, более широкий - на всю организацию, см. **Доступ на всю организацию**.

Вкладка «Доступ» проекта

Откройте «Проекты» → нужный проект → вкладка «Доступ». Она видна тем, у кого есть право `access.view` на проекте.

Блок	Что показывает
Выданный доступ	Привязки ролей на этом проекте: кто, какая роль, когда и кем выдана. Роли, пришедшие сверху (с организации), помечены «сверху» и меняются там
Итоговый доступ	Все, у кого есть доступ в проекте, и их итоговые роли с указанием источника: лично или через группу
Запреты	**Запреты**, действующие в проекте
Закрытость	Признак **закрытого проекта**

Как выдать доступ

1. Нажмите «Выдать доступ».
2. Выберите человека, группу или сервисный аккаунт. Людей ищут по имени или почте среди всех активных пользователей организации. Тест-лиду для этого не нужен доступ к разделу «Пользователи»: в поиске он видит только имя, почту и аватар.
3. Выберите роль. Рабочие роли выдаются на проект, роли администрирования на проект выдать нельзя.
4. Подтвердите.

Выдать роль можно, только если все её права есть у вас в этом проекте. Если нужен особый набор прав, нажмите «Создайте роль на основе существующей» - вы перейдёте к **созданию**

своей роли.

Группы или личные роли

Группы - основной путь, личные роли - исключения. Если тот же доступ нужен нескольким людям, выдайте его группе. Если личная роль повторяет то, что человек уже получает через группу, она помечается «дублирует группу», и её можно снять действием «Снять дубль». Организация может включить политику «только группы»: тогда рабочие роли людям напрямую не выдаются, только группам.

Выдать как у коллеги

В карточке пользователя «Добавить» → «Как у другого пользователя» копирует этому человеку личные роли и ручные группы коллеги в пределах мест, где вы управляете доступом. Обратное действие «Скопировать доступ другому» есть в меню «Действия». Запреты не копируются. Роли, которые уже есть, не дублируются. Перед выдачей показано, что будет скопировано, а что нет.

Как снять доступ

Что снять	Действие
Личную роль	«Снять» в строке выдачи. Перед снятием показано, что у человека останется в этом месте
Роль группы	Снимите её в карточке группы, секция «Доступ». Роль перестанет действовать для всех участников
Весь доступ человека к проекту	«Проверка доступа» с выбранным человеком, блок «Что видит», действие «Убрать доступ...» в строке проекта: убрать из группы или поставить запрет на проект

Что нужно для управления доступом

Право	Что даёт
`access.view`	Видеть выдачи, запреты и итоговый доступ в месте и ниже
`access.manage`	Выдавать и снимать роли, ставить запреты, закрывать и открывать место, отвечать на **запросы доступа**

Эти права входят в роли «Тест-лид» и «Администратор проектов»: тест-лид управляет доступом в своём проекте. Чтобы выдать доступ, ему хватает `access.manage` на проекте: поиск людей и список ролей с их составом открываются в панели «Выдать доступ» без прав на разделы «Пользователи» и «Роли».

Создание проекта

Раздел «Проекты» → «Создать проект». Укажите название, при желании описание и логотип. Отдельного ключа у проекта нет. Создать проект может тот, у кого есть право `project.create` (роль «Создатель проектов» или «Администратор организации»). Создатель получает на новом проекте роль «Тест-лид». Доступ остальным выдаётся на вкладке «Доступ» созданного проекта.

Карточка проекта

Вкладка	Что в ней
Обзор	Сводка, администраторы, последнее событие, общие сведения, архивирование и удаление
Пространства	Пространства проекта: создание, переименование, архивирование, удаление, переход к доступу пространства
Доступ	Выдачи, итоговый доступ, запреты
Атрибуты	**Атрибуты** кейсов и чек-листов
Трекер	Привязка проекта к **трекеру**

Вкладка	Что в ней
Токены	API-токены проекта
Аудит	События проекта (нужно право `audit.view`)

Архивирование скрывает проект и его пространства от участников, данные сохраняются, проект можно восстановить. Токены проекта при этом перестают принимать результаты. Удаление необратимо: для подтверждения нужно ввести название проекта.

Доступ к пространству

Роль можно выдать на отдельное пространство: тогда человек работает только в нём. Так, например, внешний подрядчик получает «Тестировщика» только в пространстве «Android».

Вкладка «Доступ» пространства

Откройте проект → вкладка «Пространства» → нужное пространство → «Доступ». Здесь те же блоки, что и у проекта: выданный доступ, итоговый доступ, запреты. Роли, выданные на проекте или организации, видны с пометкой «сверху» и меняются там. Пометка «только в пространстве» отмечает роли, выданные именно на пространство.

Как выдать

1. Нажмите «Выдать доступ».
2. Выберите человека, группу или сервисный аккаунт и рабочую роль.
3. Подтвердите.

Для доступа к пространству нужно право `access.manage` на нём или выше.

Если нужен частичный доступ

Прежние переключатели «Кейсы», «Чек-листы» и «Прогоны» заменены ролями. Для частичных наборов создайте свою роль, копию «Тестировщика» с нужными областями на уровнях «Нет», «Просмотр» или «Работа», см. **Роли и права**. Роли вида «Тестировщик (только кейсы)» при обновлении создаются автоматически для тех, у кого был такой набор.

Закрытое пространство

Пространство можно закрыть: доступ, выданный выше, внутрь не проходит. Подробности: **Закрытые проекты и пространства**.

Закрытые проекты и пространства

Роль на всю организацию проходит во все проекты и пространства. Закрытие нужно там, куда она проходить не должна: например, пространство «Безопасность» внутри общего проекта только для троих.

Что значит «закрыт»

Что	Поведение
Роли, выданные выше	Внутрь не проходят. Для закрытого проекта это роли с организации, для закрытого пространства - с проекта и организации
Роли, выданные на сам закрытый узел и ниже	Работают как обычно
Владелец организации и те, у кого `access.manage` выше	Видят закрытое место и выдают в нём роли, но работать с содержимым без своей роли внутри не могут
Запреты, поставленные выше	Проходят внутрь
Закрытое пространство в списке проекта	Не видно тем, у кого нет доступа внутрь

Организацию закрыть нельзя.

Как закрыть

Нужно право `access.manage` на проекте или пространстве.

1. Откройте проект (или пространство) → вкладка «Доступ».
2. Нажмите «Закрыть проект» (или «Закрыть пространство»).
3. Изучите сводку: «Потеряют доступ» и «Сохранят доступ». Сохранят его те, у кого роли выданы на этом месте или ниже.
4. Чтобы сохранить уходящего, сначала выдайте ему роль в этом месте, затем закройте.

5. Подтвердите.

Закрытое место помечается в заголовке: «Закрытый проект» или «Закрытое пространство».

Как открыть

Кнопка «Открыть проект» (или «Открыть пространство») возвращает наследование. Сводка покажет, кто снова получит доступ.

Закрытие и запреты

Задача	Что использовать
Доступ нужен только части людей	Закройте место и выдайте роли в нём
Вся группа, кроме одного человека	**Запрет** на этого человека

Закрытые места в пересмотре

Каждое закрытое проект и пространство попадает в отчёт «Запреты и закрытые узлы» **пересмотра доступа**: убедитесь, что закрытие всё ещё нужно.

Запреты

Запрет нужен для исключений, которые роли не выражают: «вся группа QA, кроме Ивана, в проекте „Безопасность“» или «группе подрядчиков нельзя выгружать данные».

Правила

- Запрет действует в месте, где поставлен, и во всех местах ниже.
- Запрет сильнее любой роли, в том числе выданной ниже. Отменить его нижестоящей ролью нельзя: единственное правило приоритета.
- Запретить можно весь доступ или выбранные права.
- Запрет «весь доступ» на проекте прячет проект целиком: он не виден в списках и поиске.
- Запреты, поставленные выше, проходят и в **закрытые** проекты и пространства.
- Владельца организации ограничить нельзя.
- Права администрирования можно запретить только на всю организацию.
- Запретов должно быть немного: если их больше 20 на организацию, интерфейс предупреждает, что роли нарезаны неудачно.

Как поставить

Нужно право `access.manage` в нужном месте.

1. Откройте карточку пользователя («Откуда доступ» → «Добавить» → «Поставить запрет») либо вкладку «Доступ» проекта или пространства и нажмите «Добавить запрет».
2. Выберите, кому ставится запрет (человек, группа или сервисный аккаунт) и место: организация, проект или пространство.
3. Выберите «Весь доступ» или «Выбранные права».
4. Укажите причину. Она необязательна и видна другим администраторам.
5. Сверьтесь со списком «Кто что потеряет» и подтвердите.

Запрет, поставленный на группе, снимается в карточке группы (секция «Доступ»), а запрет выше текущего места - там, где он задан.

Как снять

Строка запрета в блоке «Запреты» → «Снять». Снять запрет может тот, у кого есть `access.manage` на месте запрета.

Запрет или закрытие

Нужно	Выберите
Убрать доступ одному человеку или группе	Запрет
Оставить доступ только избранным	**Закрытие** и роли внутри

Все запреты попадают в отчёт «Запреты и закрытые узлы» **пересмотра доступа**.

Запросы доступа

Если у человека нет доступа к проекту или пространству, он может попросить его из интерфейса, не разыскивая администратора.

Как запросить доступ

Запрос можно отправить на странице **«Мой доступ»** кнопкой «Запросить доступ», из сообщения об отсутствии доступа или со страницы «Проекты», если у вас ещё нет ни одного проекта.

1. Выберите проект или пространство из видимых вам или вставьте ссылку на нужное место.
2. Напишите, зачем нужен доступ.
3. Нажмите «Отправить запрос».

Повторный запрос на то же место не создаётся, пока первый не рассмотрен. Ответ придёт уведомлением. Если на запрос не ответили, он закрывается через 14 дней, и автор получает уведомление.

Кто и как отвечает

Запросы получают те, у кого есть право `access.manage` в этом месте: уведомлением и в разделе «Безопасность» → «Запросы доступа». Вы видите запросы только в проекты и пространства, которыми управляете.

Вкладка	Что показывает
Новые	Ожидающие запросы: кто, куда, зачем, до какого срока, есть ли у автора лицензия
Рассмотренные	Одобренные, отклонённые и истёкшие запросы

Одобрить

1. Нажмите «Одобрить...».

2. Выберите роль для выдачи.
3. Нажмите «Одобрить и выдать».

Доступ выдаётся сразу, автор получает ответ. Если закрытое место, в строке будет пометка «Закрытый узел: доступ сверху сюда не проходит».

Отклонить

Нажмите «Отклонить», укажите причину (обязательно). Автор запроса её увидит.

Пересмотр доступа

Доступ выдаётся без срока, поэтому его нужно регулярно проверять: кому он ещё нужен, какие роли никто не использует, какие лицензии лежат без дела. Раздел «Безопасность» → «Пересмотр доступа» помогает в этом тремя инструментами: отчёты, кампании и гигиена ролей. Раздел виден тем, у кого есть право `access.manage`.

Обзор

На вкладке «Обзор» собраны сигналы: последняя кампания, срок следующего пересмотра по расписанию, пометка «просрочен» и короткие отчёты.

Отчёты

Отчёт	Что показывает	Что делать
Опасные права	Кто может очищать, выгружать, управлять доступом, менять провайдера ИИ и управлять CI	Убедиться, что круг не шире необходимого
Не использовались 90 дней	Роли, которыми человек не пользовался 90 дней: ни одного действия, которое роль разрешает на этом месте	Снять роль
Лицензии неактивных пользователей	Люди с лицензией, которые не входили 30 дней	Отозвать лицензию и отдать тому, кто ждёт
Доступ вне групп	Роли, выданные лично, а не через группу	Если один и тот же доступ нужен нескольким, перенести его в группу
Запреты и закрытые узлы	Каждый запрет и каждое закрытое место	Проверить, что они всё ещё нужны

В отчётах доступны массовые действия: «Снять роли», «Отозвать лицензии», «Снять запреты и открыть». Снятие применяется сразу, записывается в **журнал аудита**, человек получает

уведомление.

Кампании

Кампания передаёт пересмотр тем, кто лучше всех знает, нужен ли доступ.

Запуск

1. Откройте вкладку «Кампании» и нажмите «Запустить кампанию».
2. Задайте название. По умолчанию подставляется «Пересмотр доступа, N кв. ГГГГ».
3. Выберите, что пересматривать: весь доступ к проектам и организации, только роли с опасными правами или только выбранные проекты.
4. Выберите длительность: 1 неделя, 2 недели или 1 месяц.
5. Выберите, что делать со строками, которые не пересмотрели: оставить доступ и показать в отчёте либо снять доступ.
6. Выберите повтор: не повторять, каждый квартал или каждые полгода.
7. Запустите. Проверяющие получат уведомление и список строк.

Кто пересматривает

Что пересматривается	Кто проверяет
Доступ в проекте	Те, кто управляет в нём доступом (тест-лиды)
Доступ на всю организацию и роли администрирования	Администраторы организации
Членство в группах	Владелец группы или администратор организации. Роль группы тест-лид не снимает: пересматривается членство, а не роль группы

Решения проверяющего

Проверяющий видит раздел «Ваш пересмотр» со строками: человек, роль, где, как получена (лично, через группу, сервисный аккаунт), последний вход, последнее действие. По каждой

строке решение: «Оставить» или «Снять». Можно выбрать несколько строк. Снять нельзя ту строку, на которую у вас нет прав: она помечена.

Ход и закрытие

- «Напомнить проверяющим» отправляет напоминания тем, кто не закончил.
- Прогресс виден по проверяющим: пересмотрено N из M строк.
- «Закрыть кампанию» останавливает приём решений. Непросмотренные строки будут сохранены или сняты в зависимости от настройки кампании.
- Закрытые кампании остаются во вкладке «Прошлые кампании» с итогами: пересмотрено, снято, снято автоматически, не пересмотрено.
- Просроченные кампании закрываются автоматически, повтор по расписанию запускается сам.

Гигиена ролей

Сигнал	Что значит
Похожие роли	Совпадают на 90 % и больше: кандидаты на объединение. Кнопка «Объединить...» переносит выдачи в одну роль, дубли убираются, затронутые люди получают уведомление
Права, которыми никто не пользуется	Права своих ролей, которых нет ни в одном использовании. Можно «Убрать из роли...»
Роль выдана одному человеку	Возможно, хватит существующей роли или группы
Роль никому не выдавалась	Кандидат на удаление, её можно восстановить из журнала аудита в течение 30 дней
Слишком много своих ролей	Больше 30

Системные роли не учитываются: их нельзя изменить.

Проверка доступа

Раздел «Безопасность» → «Проверка доступа» отвечает на вопросы «почему у человека есть или нет доступа» и «кто может выполнить действие». Раздел виден тем, у кого есть право `access.view`. Обычный пользователь может проверить только свой доступ: чужой показывается при наличии `access.view` в нужном месте.

Режимы

Режим	Вопрос	Что выбрать
Человек	Может ли этот человек выполнить действие	Человека, действие (право) и место
Кто может	Кто в организации может выполнить действие	Действие и место

Параметры

Поле	Значение
Действие	Любое право из **справочника**
Где	Организация, проект или пространство
Объект	Необязательно: тест-кейс, чек-лист, прогон, плановый прогон, баг, автотест, общий шаг, а также проект и пространство. С объектом проверяются условия вроде «только свои», а место берётся из объекта. Введите тип объекта и ID

Что видит человек

Когда человек выбран, а действие ещё нет, показывается блок «Что видит {имя}»: для каждого проекта роли человека и откуда они (личная роль или группа). В строке проекта «Почему» подставляет проект в проверку, а «Убрать доступ...» предлагает убрать человека из группы, через которую пришёл доступ, или поставить запрет на проект. Сюда ведёт действие «Посмотреть как пользователь» из **карточки пользователя**.

Результат

Поле	Что значит
Результат	Разрешено или запрещено
Причина	Запрет, нет роли, место закрыто, условие не выполнено, нужна лицензия
Откуда право	Какая роль и где выдана, лично или через группу
Условие, лицензия	Как повлияли на ответ

В режиме «Кто может» результат - таблица людей, групп и сервисных аккаунтов с источником права. Её можно выгрузить в CSV. Если права нет ни у кого, подсказка предложит выдать роль, в которую оно входит, или выбрать другое место.

Из результата доступны переходы: «Открыть доступ пользователя» (карточка пользователя на разделе «Доступ») и «Кто ещё может это здесь».

Мой доступ

Страница показывает, что вы можете делать в DoQA и откуда это следует. Её видит каждый пользователь: свой доступ. Открывается из профиля, вкладка «Мой доступ».

Что на странице

Блок	Что показывает
Мои роли	Роли, выданные вам лично: какая и где (вся организация, проект, пространство). Для владельца организации - пометка «Владелец организации»
Мои группы	Группы, в которых вы состоите, и их роли
Лицензия	Есть ли у вас лицензия. Без неё все роли работают только на просмотр
Что я могу в проектах	Итог по доступным вам проектам
Мои запросы	Отправленные запросы доступа и их статус: ожидает, одобрен, отклонён, истёк

Если ролей ещё нет, страница предлагает запросить доступ к проекту.

Запросить доступ

Кнопка «Запросить доступ» открывает форму:

1. «Куда»: выберите проект или пространство из тех, что вы видите. Если нужного места нет в списке (например, пространство закрыто для вас), выберите «Другое (вставить ссылку)...» и вставьте ссылку на проект или пространство, которую прислал коллега.
2. «Зачем»: коротко объясните причину.
3. «Отправить запрос».

Дальнейший порядок описан на странице **Запросы доступа**.

Если что-то не работает

Если действие недоступно, причина отказа называется в сообщении:

Причина	Что делать
Доступ закрыт запретом	Обратитесь к администратору
Нет роли, которая это разрешает	Запросите доступ
Проект или пространство закрыто для доступа сверху	Запросите доступ именно в этом месте
Роль разрешает это только при определённых условиях	Условие, например «только свои», не выполнено
Для действия нужна лицензия	Попросите администратора выдать лицензию

Подробнее о проверке см. **Проверка доступа**.

Журнал аудита

Раздел «Безопасность» → «Журнал аудита» записывает каждое действие администратора: кто, когда, что это было и чем стало. Раздел виден тем, у кого есть право `audit.view`.

Что в таблице

Столбец	Содержание
Время	Когда произошло
Кто	Автор действия (пользователь или система)
Действие	Что сделано
Объект	Над чем: пользователь, группа, роль, сервисный аккаунт, проект, пространство, интеграция
Область	Организация, проект или пространство
Затронуто	Сколько людей коснулось изменение

Строку можно раскрыть: «Подробности» показывают код действия, затронутых людей и изменения по полям в виде «Было» и «Стало». Секреты в записях заменены.

Фильтры

Фильтр	Значения
Период	24 часа, 7 дней, 30 дней, 90 дней, за всё время
Тип действия	Доступ, ИИ, подключения, лицензии, люди, проекты, система
Кто	Автор действия
Объект	Тип объекта и его ID

Результат можно выгрузить в CSV («Экспорт в CSV»).

Журнал сущностей

Записи по конкретному объекту видны и на его странице, во вкладке «Аудит»: у пользователя, группы, сервисного аккаунта, проекта, подключения. Вкладка доступна при наличии `audit.view`. История изменений лицензии пользователя берётся из того же журнала.

Что попадает в журнал

- выдача и снятие ролей, запреты, закрытие и открытие мест;
- создание, изменение и удаление ролей и групп, изменение состава;
- приглашения, блокировка, сброс пароля, удаление пользователей, передача владения;
- выдача и отзыв лицензий, замена лицензионного ключа;
- токены и сервисные аккаунты;
- подключения, настройки ИИ;
- решения по запросам и пересмотру доступа;
- переход на новую систему прав.

Вход и SSO

Раздел «Безопасность» → «Вход и SSO» показывает, как пользователи входят в DoQA, и позволяет проверить подключение. Раздел виден тем, у кого есть право `sign_in.view`.

> **Только состояние**
>
> Настройки входа задаются переменными окружения установки. В интерфейсе они только для чтения: измените переменные и перезапустите приложение. Как это сделать: **Корпоративный вход (SSO)** и **Авторизация через LDAP**.

В облачной версии способы входа настраиваются в личном кабинете. Раздел доступен только в серверной (коробочной) версии.

Что показывает раздел

Блок	Содержание
Пароль	Доступен ли вход по паролю всем пользователям или только аварийным аккаунтам, пока включён вход только через SSO
SSO (OpenID Connect)	Включён ли, издатель, клиент, адрес перенаправления, задан ли Client secret, создаются ли пользователи при первом входе, режим «Только SSO», разрешённые домены
LDAP	Включён ли, сервер, база поиска, шифрование
Аварийные аккаунты	Кто может войти по паролю при включённом режиме «Только SSO»

Статусы подключения: включено, выключено, настроено неверно (например, SSO запрошен, но не заданы издатель или клиент). Кнопка «Проверить» проверяет подключение и показывает время последней проверки и ошибку, если она была.

Способ входа пользователя виден в его карточке, раздел «Вход и токены».

Группы из SSO и LDAP

Состав групп может следовать внешнему источнику.

Источник группы	Как определяется состав
SSO	Значение claim `groups` в токене единого входа
LDAP	DN группы в каталоге
По правилу	Домен почты, атрибут SSO или LDAP, способ входа

Состав пересчитывается при входе и по кнопке «Синхронизировать» в карточке группы. Группа по умолчанию добавляет новых людей при первом входе через SSO. Ошибки синхронизации видны в списке групп и в карточке группы.

Вход пользователей

В карточке пользователя, раздел «Вход и токены», показаны способ входа (почта и пароль, единый вход, каталог LDAP) и последний вход. Для SSO и LDAP пароль управляется во внешней системе.

Лицензии и оплата (облачная версия)

Организация покупает N лицензий с одним общим сроком. Лицензии распределяются сами: каждый добавленный пользователь сразу занимает свободную лицензию, вручную выдавать их не нужно. Пользователь с лицензией работает по своим ролям полностью. Если лицензий не хватило, следующие пользователи работают «Только просмотр»: роли и доступ у них сохраняются, но права на изменение не действуют. Когда вы докупаете лицензии, полный доступ включается сам. Сервисным аккаунтам лицензия не нужна.

Покупка, продление, счета и автопродление находятся прямо в DoQA, в разделе «Лицензии и оплата» меню администрирования.

Для серверной версии порядок другой, он описан в разделе **Лицензии (серверная версия)**.

Как распределяются лицензии

Пользователи стоят в очереди в порядке добавления. Владелец организации всегда первый. Первые N пользователей занимают лицензии, остальные ждут и работают «Только просмотр».

Событие	Что происходит
Добавили или пригласили пользователя	Он встаёт в конец очереди и сразу занимает свободную лицензию. Приглашённый занимает лицензию ещё до первого входа
Докупили лицензии	Ждущие получают их по порядку очереди
Сделали пользователя наблюдателем	Его лицензия освобождается и переходит первому ждущему
«Вернуть полный доступ» наблюдателю	Он встаёт в конец очереди и получает лицензию, если она свободна
Заблокировали пользователя	Лицензия освобождается сразу. После разблокировки пользователь встаёт в конец очереди
Удалили пользователя или отозвали приглашение	Лицензия освобождается

Событие	Что происходит
Часть лицензий закончилась или продлили меньше, чем куплено	Лицензии теряют последние в очереди

Владелец организации работает с полным доступом всегда, даже когда лицензий нет. Сделать наблюдателем владельца или самого себя нельзя.

Наблюдатель это пользователь, которого администратор сам оставил без лицензии. Для системы прав наблюдатель и ждущий одинаковы: оба работают только на просмотр. Разница в причине, она видна в подсказке у метки.

Как это выглядит в списке пользователей, см. **Пользователи**.

Кто что может

Раздел виден тем, у кого есть право `license.view`.

Право	Что даёт
`license.view`	Видеть лицензии, сроки и кто ждёт лицензию
`license.assign`	Делать наблюдателем и возвращать полный доступ
`billing.view`	Видеть «Платежи и счета», настройки оплаты, чеки, акты и акт сверки. Только просмотр. Подразумевает `license.view`
`billing.manage`	Покупать, платить, выставлять и отменять счета, привязывать и отключать автопродление, менять реквизиты. Подразумевает `billing.view`
`license.manage`	В облаке: заявка на продление пробного периода. Ключ и заявки на продление относятся к **серверной версии**

Роль «Администратор пользователей» включает все эти права. Роль «Аудитор» получает `billing.view`. Владелец организации и «Администратор организации» могут всё перечисленное. См. **Роли и права** и **Справочник прав**.

Права оплаты не требуют лицензии: человек с `billing.manage` может продлить истёкший срок, даже если сам работает только на просмотр. Тем, у кого нет `billing.manage`, вместо кнопок оплаты показана подсказка и кнопка «Запросить доступ».

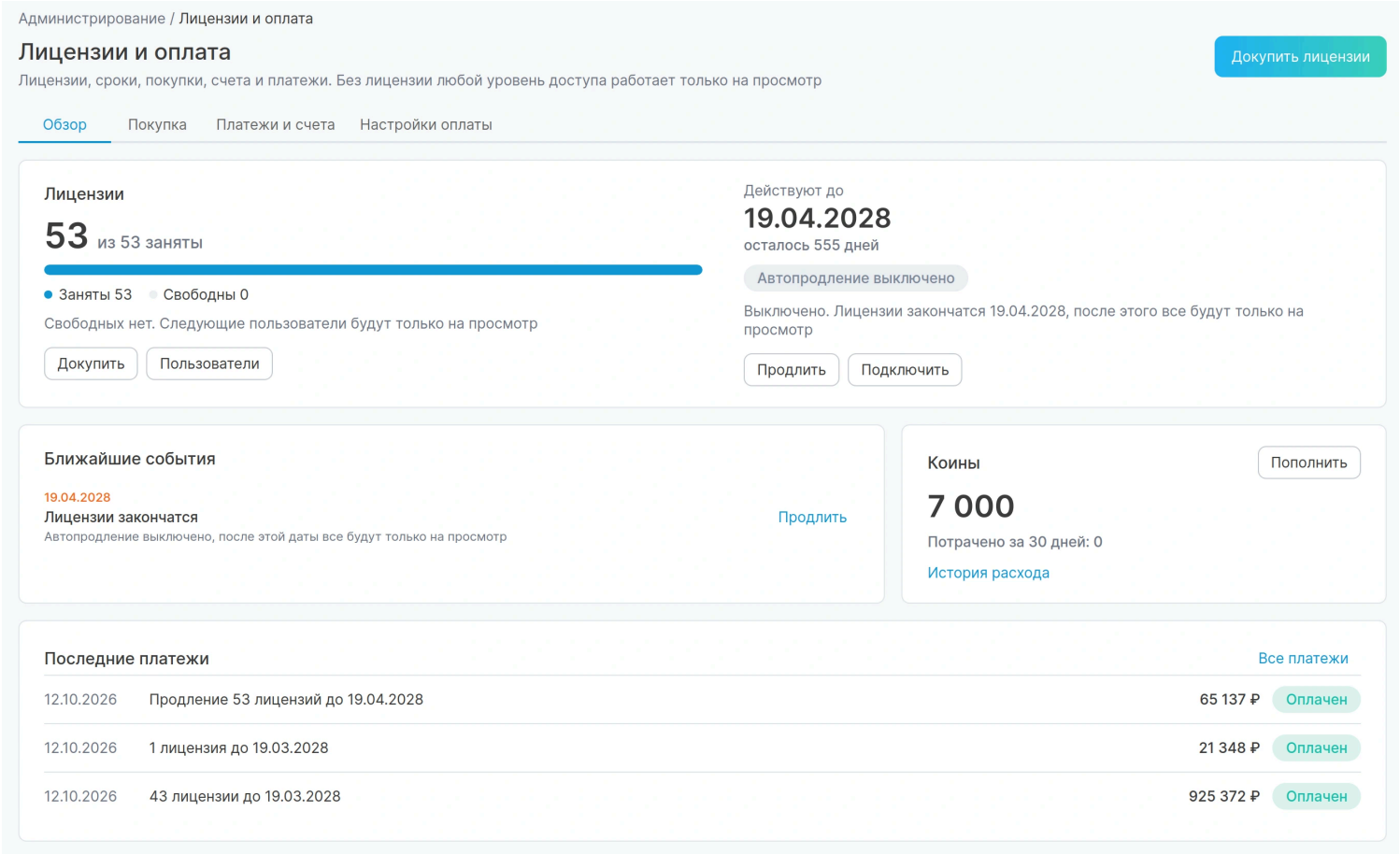
Вкладки раздела

В шапке раздела кнопка «Докупить лицензии» открывает «Покупку» с числом лицензий, которых не хватает.

Вкладка	Право	Что на ней
Обзор	`license.view`	Лицензии, срок, автопродление, ближайшие события, коины, последние платежи
Покупка	`billing.manage`	Новые лицензии, продление и пакеты коинов
Платежи и счета	`billing.view`	Оплаты и выставленные счета, чеки, акты, акт сверки. Бейдж на вкладке показывает число неоплаченных счетов
Настройки оплаты	`billing.view`	Автопродление и реквизиты для счетов

Вкладку, на которую нет права, вы не увидите. Если оплата временно недоступна, на её вкладках появится сообщение с кнопками «Повторить» и «Написать в поддержку». Лицензии и коины на «Обзоре» при этом продолжают показываться.

Обзор



Карточка «Лицензии»:

- слева «X из N заняты», полоса заполнения и легенда «Заняты», «Свободны» и «Ждут лицензию» (последняя только если кто-то ждёт, ссылка открывает этих пользователей). Под легендой подсказка: сколько не хватает, сколько свободно или что свободных нет. Кнопки «Докупить» и «Пользователи»;
- справа «Действуют до» с датой и числом оставшихся дней. Если часть лицензий закончится раньше общего срока, ниже строка «Из них M до даты». Метка «Автопродление включено» или «Автопродление выключено», сохранённый способ оплаты и строка о ближайшем списании («дата спишем ≈ сумма за N лицензий на 1 месяц»; автопродление продлевает лицензии основного срока, ранние блоки в N не входят) либо дата, когда лицензии закончатся. Кнопки «Продлить» и «Настроить» (или «Подключить», нужно `billing.manage`).

В пробный период вместо «Действуют до» показано «Пробный период до», кнопки «Купить» и «Продлить пробный период». Если срок закончился, сверху красная плашка с кнопкой «Продлить».

«Ближайшие события» собирают всё, что требует внимания, с датой и действием:

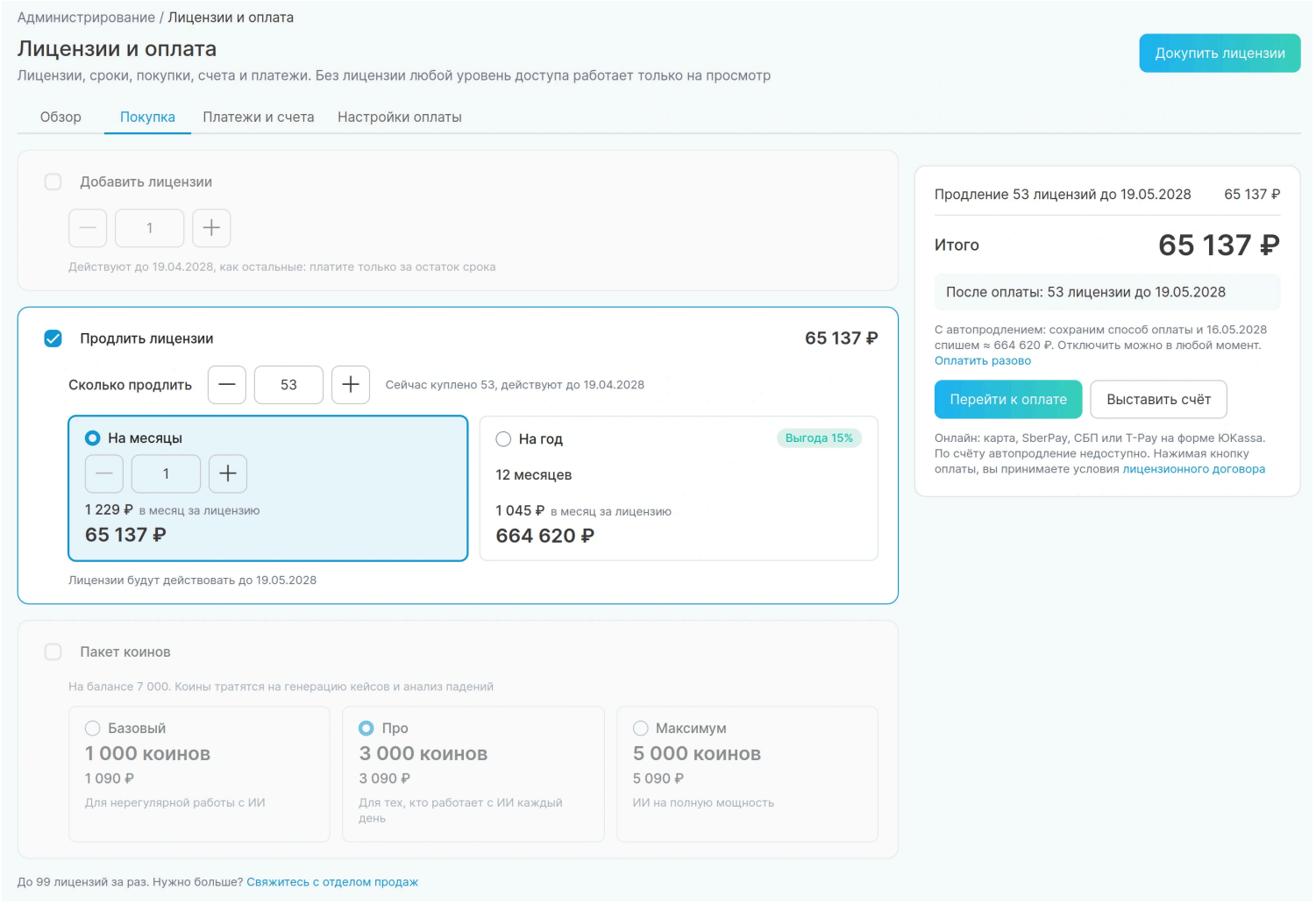
Событие	Действие
N пользователей без лицензии	«Докупить»
Счёт ждёт оплаты (оранжевая дата, у просроченного «Просрочен с»)	«Оплатить» открывает форму ЮKassa
Автопродление: дата и сумма списания, способ оплаты	«Настроить»
Лицензии закончатся (автопродление выключено)	«Продлить»
Закончатся M лицензий (часть лицензий с более ранним сроком)	«Продлить»
Списание не прошло: причина и дата следующей попытки	«Заменить способ»
Автопродление выключено: причина	«Подключить»
Пробный период закончится	«Купить»

Карточка «Коины»: баланс, сколько потрачено за 30 дней и на сколько месяцев примерно хватит. Если коинов мало, появится предупреждение. Кнопка «Пополнить» открывает «Покупку» с пакетом «Про», ссылка «История расхода» ведёт на страницу расхода ИИ.

«Последние платежи»: три последние операции и ссылка «Все платежи».

В меню администрирования рядом с пунктом «Лицензии и оплата» показан счётчик «занято из всего».

Покупка



Покупка состоит из трёх позиций. Позиция включается галочкой у заголовка, выключенная показана бледной и в заказ не попадает. Если изменить число или выбрать карточку внутри позиции, она включится сама. Справа от заголовка сумма позиции.

- **Добавить лицензии.** Счётчик от 1 до 99 и цена за лицензию. Новые лицензии действуют до общего срока, как остальные: вы платите только за остаток срока. В пробный период и после окончания срока добавить лицензии нельзя, сначала продлите.
- **Продлить лицензии.** «Сколько продлить» по умолчанию равно числу купленных лицензий вместе с добавляемыми в этом же заказе. Срок выбирается карточкой: «На месяцы» со счётчиком от 1 до 11 или «На год» с бейджем «Выгода 15%». В карточке цена в месяц за лицензию и сумма. Ниже дата, до которой лицензии будут действовать. Если продлить меньше, чем куплено, появится предупреждение: остальные лицензии закончатся в прежний срок, и последние в очереди перейдут в «Только просмотр».
- **Пакет коинов.** Баланс и три пакета: «Базовый», «Про» и «Максимум».

Цену считает сервер, суммы пересчитываются сразу. Самостоятельно можно купить до 99 лицензий за раз, для большего числа есть ссылка «Свяжитесь с отделом продаж».

Справа итог: строки позиций, «Итого» и что будет после оплаты (сколько лицензий и до какой даты, сколько коинов).

Под итогом строка автопродления. Если автопродление ещё не подключено, оплата картой по умолчанию сохраняет способ оплаты в ЮKassa и включает автопродление: в строке указаны дата будущего списания и сумма. Ссылка «Оплатить разово» отменяет сохранение, «Включить автопродление» возвращает его. Если автопродление уже подключено, в строке указан способ оплаты.

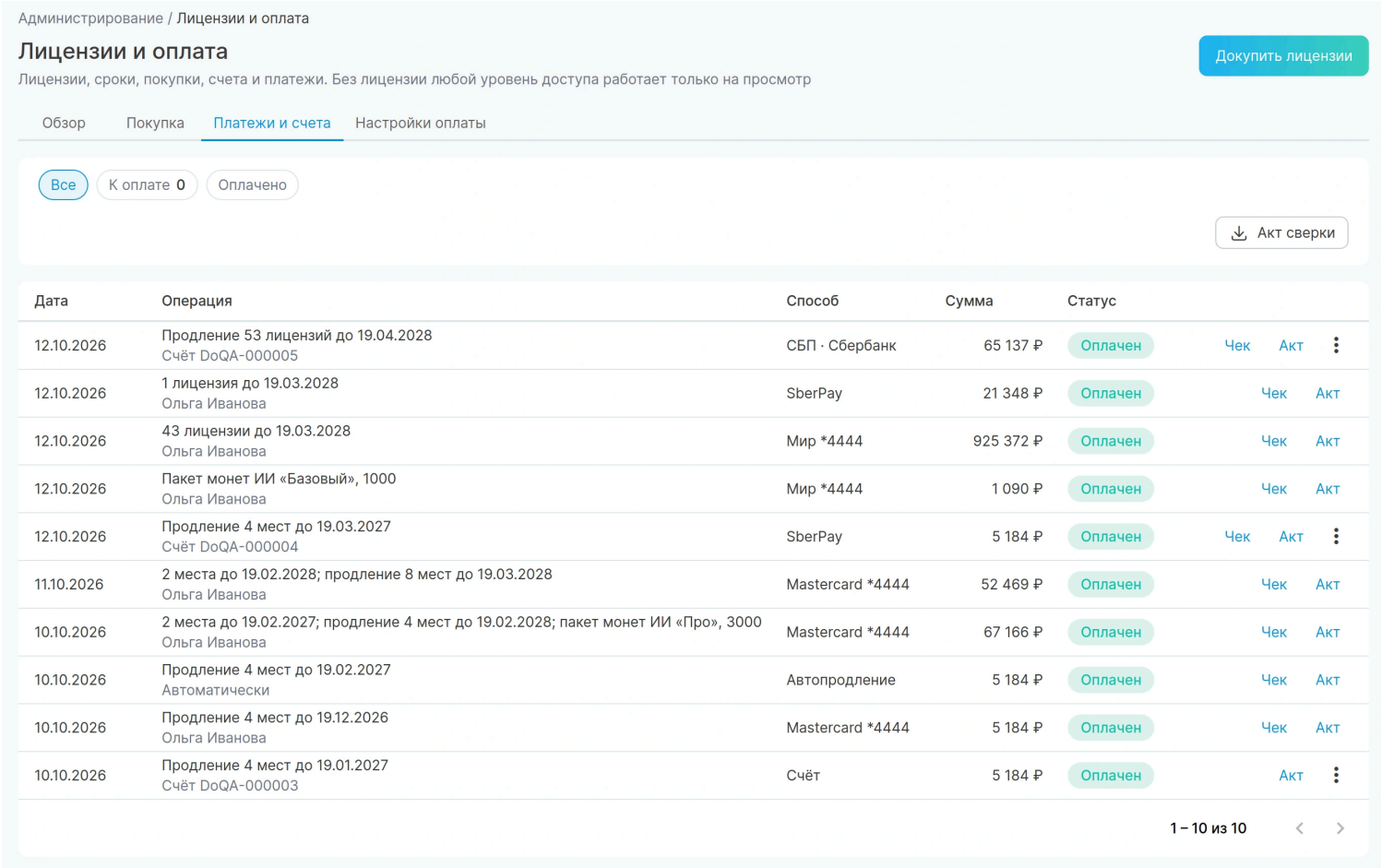
- «Перейти к оплате» сразу открывает форму ЮKassa: карта, SberPay, СБП или T-Pay. После оплаты вы вернётесь в раздел: на «Обзоре» появится сообщение «Оплачено» с итогом. Пока платёж подтверждается, открыта страница заказа со статусом.
- «Выставить счёт» выставляет счёт на основную организацию из реквизитов (если организаций несколько, предложит выбрать) и открывает «Платежи и счета». Счёт приходит на email для документов. Если реквизитов нет, откроются «Настройки оплаты» с формой добавления организации. По счёту автопродление недоступно.

Нажимая кнопку оплаты, вы принимаете условия лицензионного договора, ссылка на него под кнопками.

Входы в «Покупку» с заполненными позициями:

Откуда	Что выбрано
«Докупить N» в «Пользователях», «Докупить лицензии» в шапке	Добавить N лицензий
«Продлить» на «Обзоре»	Продление всех лицензий на месяц
«Пополнить» на «Обзоре», в шапке DoQA и на странице провайдера ИИ	Пакет коинов «Про»

Платежи и счета



Список оплат и счетов. Фильтр вверху: «Все», «К оплате N», «Оплачено». В строке дата (у счёта ниже срок оплаты), операция и кто её провёл (номер счёта, имя оплатившего, «Автоматически» для автопродления), способ оплаты («Мир *4444», «SberPay», «СБП · Сбербанк», «Счёт», «Автопродление»), сумма и статус: «Ожидает оплаты», «Просрочен», «Оплачен» или «Отменён».

Строка	Действия
Счёт ждёт оплаты	«Оплатить» открывает форму ЮKassa. В меню «Скачать счёт» и «Отменить счёт»
Оплачено	«Чек» (когда ЮKassa сформировала чек) и «Акт». Для счёта в меню «Скачать счёт»
Отменённый счёт	«Перевыставить»: новый счёт по тому же составу с текущими ценами

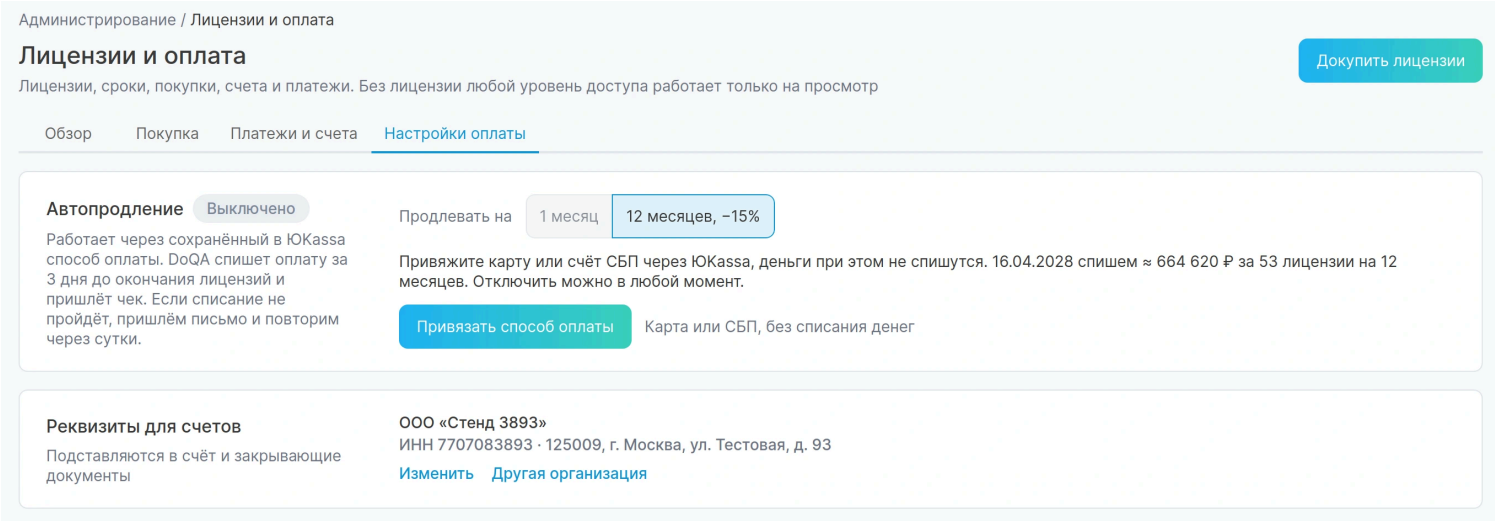
Для акта нужны реквизиты организации: без них DoQA предложит добавить их в «Настройках оплаты». Клик по строке открывает страницу заказа с составом, способом оплаты и документами.

«Акт сверки» скачивает PDF за период по выбранной организации: этот или прошлый квартал, этот год или свой период до года.

Счёт оплачивают в течение 7 дней, неоплаченный отменяется автоматически через 30 дней. Лицензии по счёту появятся после подтверждения оплаты.

Заказы с нулевой суммой (пробный период, начисления поддержки) в список не попадают, они видны в журнале аудита.

Настройки оплаты



Автопродление

Автопродление работает через сохранённый в ЮKassa способ оплаты. DoQA списывает оплату за 3 дня до окончания лицензий и присылает чек. Если списание не пройдёт, придёт письмо, и попытка повторится через сутки (всего четыре попытки, последняя в день окончания). После последней неудачной попытки автопродление выключается.

- «Продлевать на»: 1 месяц или 12 месяцев со скидкой.
- Если автопродление включено, показан способ оплаты (карта с последними цифрами и сроком, SberPay, СБП и другие) и ближайшее списание. «Заменить» привязывает другой способ, «Отключить» после подтверждения отвязывает способ оплаты от DoQA. Лицензии продолжат действовать до конца срока, а чтобы включить автопродление снова, способ нужно привязать заново.
- Если выключено, кнопка «Привязать способ оплаты» открывает форму ЮKassa: карта или СБП, деньги не списываются. После привязки автопродление включится, DoQA покажет

сообщение «Способ оплаты привязан, автопродление включено».

Переключателя у автопродления нет: включить его можно только привязкой способа или оплатой с автопродлением в «Покупке». В пробный период и после окончания срока привязка недоступна, сначала оплатите лицензии.

Реквизиты для счетов

Реквизиты подставляются в счёт и закрывающие документы. Показана основная организация: название, ИНН, адрес и email для документов и чеков.

- «Изменить» меняет email для документов.
- «Другая организация» добавляет организацию по ИНН (и КПП, если нужен) и делает её основной: название и адрес подставляются автоматически.
- «Все реквизиты (N)» раскрывает список организаций: назначить основной, изменить email, удалить.

Пробный период

Новая организация получает пробный период с несколькими лицензиями. На «Обзоре» показано, до какого числа он действует, а в «Ближайших событиях» его окончание.

Чтобы продлить пробный период, нажмите «Продлить пробный период» (нужно `license.manage`), выберите 7, 14 или 30 дней, при необходимости число лицензий и комментарий. Заявка уйдёт в поддержку, её статус виден на «Обзоре», решение придёт уведомлением.

Уведомления

Владелец организации и все, у кого есть `billing.manage`, получают уведомления в DoQA и на почту:

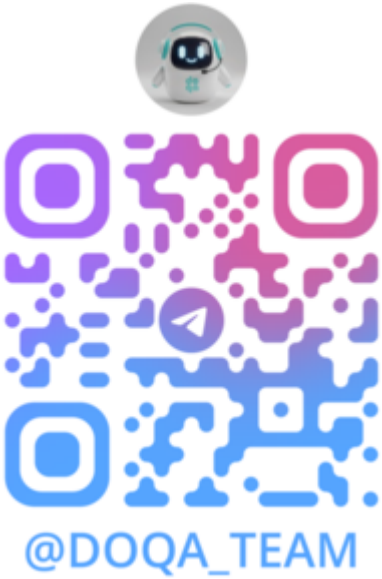
- оплата прошла и применена;
- лицензии заканчиваются: за 14, 7 и 1 день;
- счёт выставлен или отменён;

- через 7 дней пройдёт списание автопродления;
- списание не прошло, автопродление выключено;
- способ оплаты привязан или отвязан.

Уведомления об оплате приходят по умолчанию и не отключаются в настройках колокольчика.

Обратная связь

Если появятся вопросы по оплате, обратитесь в техподдержку по email support@doqa.app или в **telegram-канал поддержки** ↗.



Лицензии (серверная версия)

Лицензия включает права на изменение. Число лицензий задаёт ключ. Лицензии распределяются сами, как в облаке: пользователи стоят в очереди в порядке добавления, первые по очереди работают с полным доступом, остальные «Только просмотр» (см. **Как распределяются лицензии**). Онлайн-оплаты в серверной версии нет: ключ выдаёт DoQA, а продление запрашивают прямо из раздела.

Раздел «Лицензии»

Раздел виден тем, у кого есть право `license.view`. Делать пользователей наблюдателями и возвращать им полный доступ можно с `license.assign`, заменять ключ и запрашивать продление с `license.manage`. Права `billing.view` и `billing.manage` относятся к облаку и в серверной версии не используются. Владелец организации всегда работает с полным доступом. См. **Роли и права**.

В разделе одна вкладка «Ключ». Вверху карточка «Лицензии»: «X из N заняты», полоса, легенда «Заняты», «Свободны» и «Ждут лицензию» (ссылка открывает ждущих в списке пользователей), подсказка о свободных лицензиях и кнопка «Пользователи». Справа срок поддержки и обновлений.

Ниже показано, кому выдан ключ и когда его заменяли.

> **Важно**
>
> Когда поддержка закончилась, DoQA продолжает работать, но обновляться не будет: обновление версии после этой даты сделает систему неработоспособной.

Наблюдатели и очередь на лицензию работают так же, как в облаке: **Пользователи**.

Загрузка и замена ключа

Новый ключ приходит от DoQA файлом `key.txt` или текстом в письме.

1. Нажмите «Заменить ключ» и либо вставьте ключ в поле, либо нажмите «Загрузить файл» и выберите `key.txt`. Файл читается в браузере и сразу проверяется. Пустой файл и файл больше 64 КБ не принимаются.
2. Нажмите «Проверить ключ». Если ключ не подходит, DoQA назовёт причину:
   - текст повреждён или скопирован не полностью;
   - подпись ключа не сходится;
   - в ключе нет лицензий.
3. Блок «Что изменится» покажет «было» и «станет»: новые сроки и число лицензий. Если в новом ключе лицензий меньше, чем занято, последние в очереди перейдут в «Только просмотр»: будет указано, сколько именно. Если срок поддержки в новом ключе уже истёк, DoQA предупредит об этом.
4. Подтвердите замену.

Некорректный ключ не записывается: прежний ключ и лицензии остаются без изменений. После замены лицензии распределяются заново по очереди. Первичный ключ при установке проверяется так же.

История замен ключа (время, кто, что изменилось) видна в блоке «История замены ключа».

Запрос на продление или расширение

Кнопка «Запросить продление» открывает форму: сколько лицензий нужно, на какой срок (от 1 до 36 месяцев), контакт для ответа и комментарий. Нужно право `license.manage`.

Заявка уходит письмом на адрес поддержки DoQA (администратор сервера может заменить его переменной `ISSUE_MAIL`, см. **параметры среды**), с данными текущего ключа: кому выдан, сколько лицензий, до какой даты поддержка. Сама заявка и её отправка записываются в журнал аудита. Если почта сервера не настроена, письмо не уйдёт, и заявка получит статус «Не отправлена».

Статус последней заявки виден на вкладке «Ключ»:

Статус	Значение
Отправлена	Письмо ушло

Статус	Значение
Не отправлена	Адрес для заявок не задан или отправка письма не удалась. Напишите в поддержку напрямую
Ключ загружен	После заявки вы заменили ключ

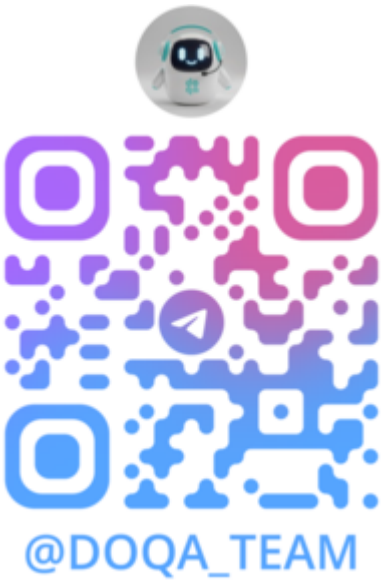
Новый ключ после согласования придёт тем же способом, что и прежний. Загрузите его, как описано выше.

Нехватка лицензий

Если лицензий меньше, чем пользователей, в списке пользователей появляется плашка «N пользователей работают только на просмотр: не хватило лицензий» с кнопкой «Запросить продление». Она открывает форму заявки, в которой уже указано нужное число лицензий (куплено плюс ждущие). Кнопку видят пользователи с `license.manage`.

Обратная связь

Если вам потребуется счёт на оплату или появятся дополнительные вопросы, вы можете обратиться в техподдержку по email support@doqa.app или в **telegram-канал поддержки** ↗.

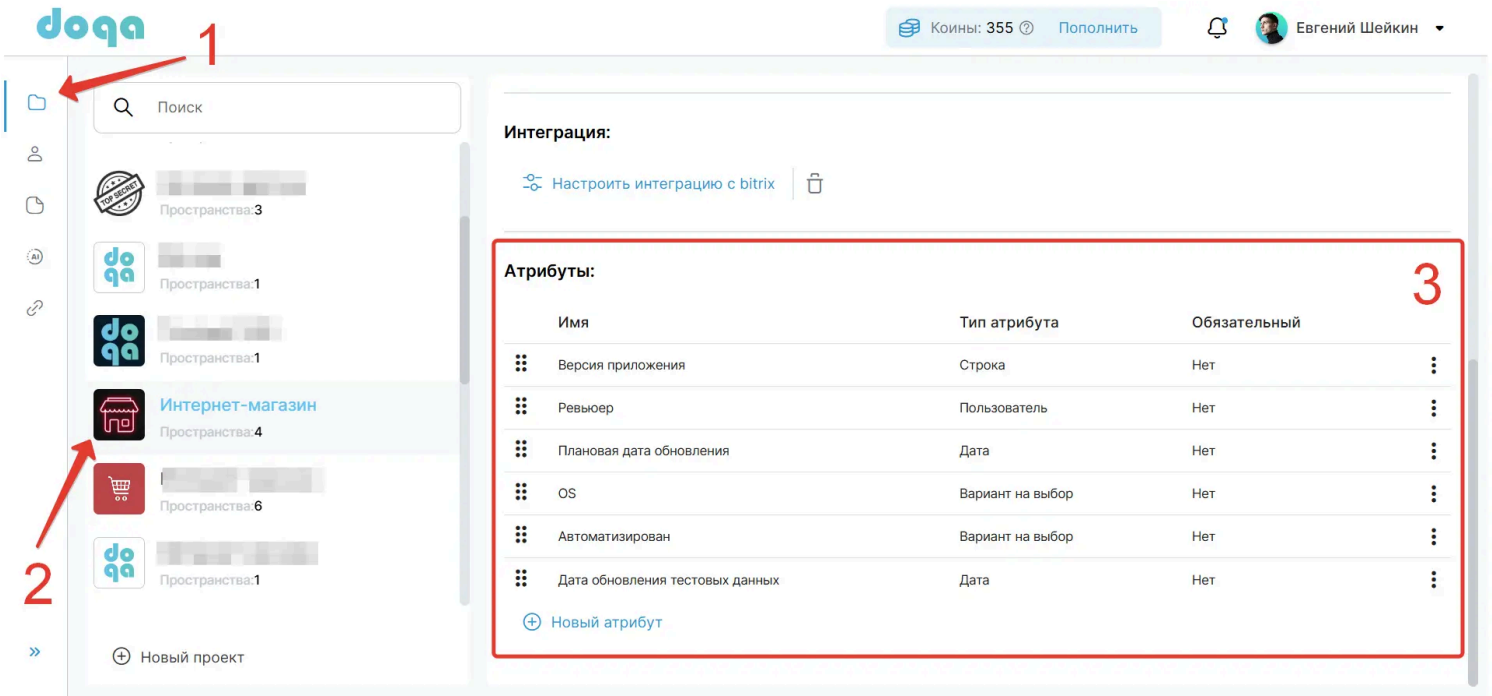


Атрибуты

Атрибут — дополнительное поле, которое вы сами заводите для проекта и заполняете в **тест-кейсах** и **чек-листах**. Набор атрибутов у каждого проекта свой. Как атрибуты выглядят и заполняются на стороне пользователя — см. **Атрибуты**.

Создавать и редактировать атрибуты могут те, у кого есть право `project_settings.manage` на проекте (роли «Тест-лид» и «Администратор проектов»).

Откройте раздел «Проекты» администрирования, выберите проект и перейдите на вкладку «Атрибуты»: там отображаются атрибуты, созданные для проекта.



Типы атрибутов

При создании атрибута можно выбрать один из 5 типов:

- **Пользователь**: позволяет выбрать пользователя DoQA из списка.
  - Пример: Атрибут «Ревьюер» для указания пользователя, который отвечает за ревью тест-кейса (цифра 1)
- **Строка**: принимает значение в виде строки.

  - Пример: Атрибут «Версия приложения», в котором для каждого тест-кейса или чек-листа можно указать совместимую версию приложения. Например, «3.0» (цифра 2).

- **Дата**: позволяет выбрать дату.
  - Пример: Атрибут «Плановая дата обновления» для явного указания даты следующего планового обновления теста (цифра 3).
- **Вариант на выбор**: содержит один или несколько заранее определенных вариантов. Этот тип удобно использовать, если возможные значения заранее известны и нужно ограничить выбор вариантов.
  - Пример: Атрибут «OS» с двумя возможными вариантами выбора «iOS» и «Android».
- **Множественный вариант на выбор**: содержит один или несколько заранее определенных вариантов. Этот тип удобно использовать, если возможные значения заранее известны и нужна возможность отметить несколько вариантов.
  - Пример: Атрибут «Совместимые версии продукта» с тремя возможными вариантами выбора «1.0», «1.3» и «1.7».

Атрибуты:

Имя	Тип атрибута	Обязательный
Ревьюер	Пользователь	Нет
Версия приложения	Строка	Нет
Плановая дата обновления	Дата	Нет
OS	Вариант на выбор	Нет
Совместимые версии продукта	Множественный вариант на выбор	Нет

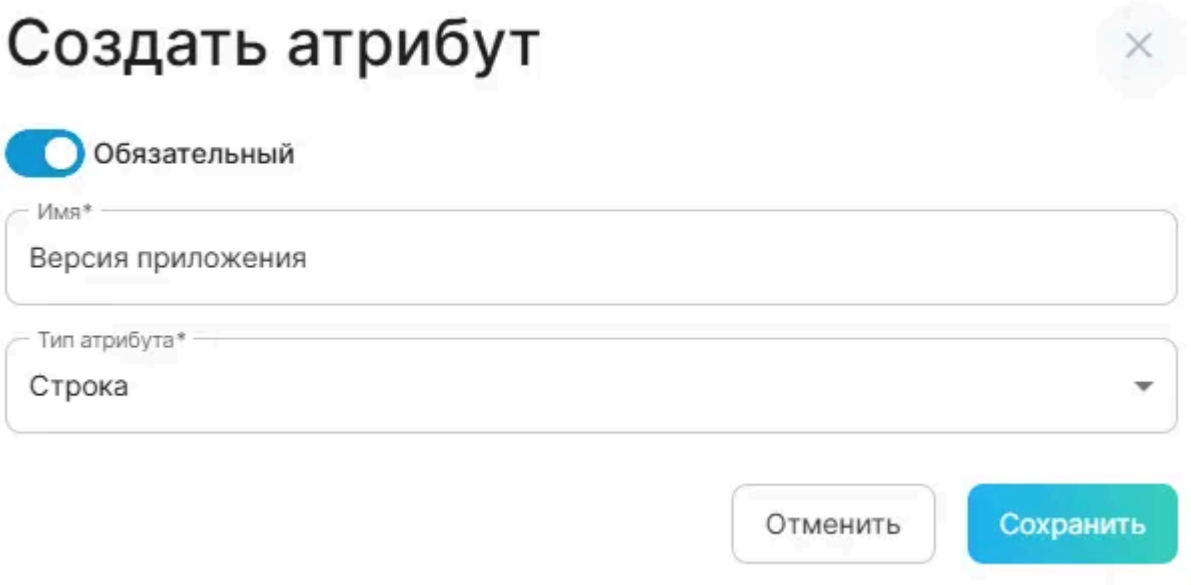
Создание атрибута

Нажмите на кнопку «**Новый атрибут**». Откроется окно «Создать атрибут»:

1. Тумблер «Обязательный» — если он включен, создать новый или сохранить существующий **тест-кейс** или **чек-лист** без заполнения этого атрибута не получится.
2. «Имя» — обязательное поле.
3. «Тип атрибута» — один из 5 типов, перечисленных выше.

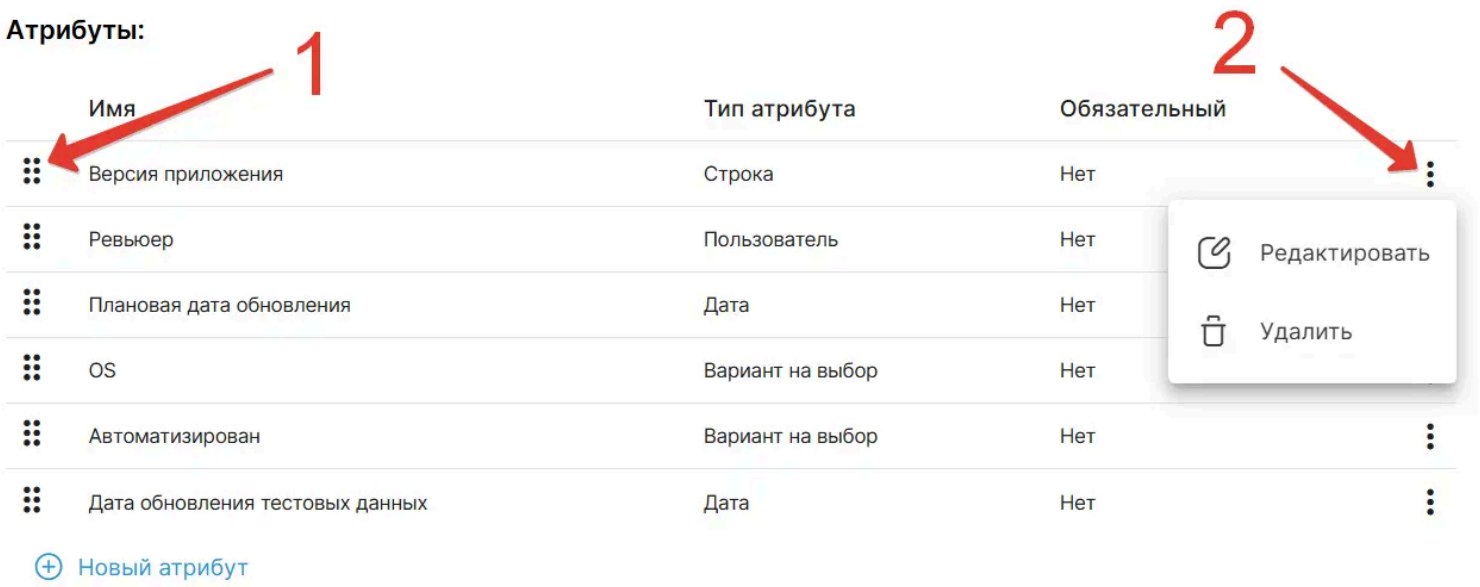
4. Для типов «Вариант на выбор» и «Множественный вариант на выбор» ниже появляется список полей «Вариант». Иконка «+» у последнего поля добавляет ещё один вариант, иконка корзины удаляет вариант. Последний оставшийся вариант удалить нельзя.

Нажмите «**Сохранить**».



Редактирование и удаление атрибута

Порядок отображения атрибутов можно изменить, перетащив их на нужную позицию (цифра 1). Для редактирования или удаления атрибута нажмите на иконку меню и выберите нужное действие (цифра 2).



Изменить можно имя атрибута, его обязательность, а у типов «Вариант на выбор» и «Множественный вариант на выбор» — список вариантов. Тип атрибута после создания не меняется: в окне «Редактировать атрибут» поле «Тип атрибута» заблокировано.

Важно

При удалении атрибута также будут удалены все его связи с тест-кейсами и чек-листами. Это действие нельзя отменить.

Интеграции с трекерами

DoQA позволяет установить связь с трекерами, чтобы заводить баг-репорты из интерфейса приложения. Подключённую интеграцию с Jira Server или YouTrack можно использовать ещё и как источник **требований**.

В разделе «Интеграции» собраны подключения к трекерам (описаны на этой странице) и к системам CI/CD (см. **Интеграции с CI/CD**). Одна и та же система может присутствовать в обоих видах и подключается для каждого отдельно. Сейчас так работает GitLab.

Чтобы заводить баг-репорты из интерфейса TMS, выполните следующие шаги:

- **Свяжите DoQA с вашим трекером**
- **Настройте связь проекта в DoQA с проектом в трекере**
- Чтобы статус и приоритет баг-репорта в DoQA обновлялись автоматически при изменении связанного баг-репорта в трекере, настройте обратную интеграцию (если используете **Jira Cloud**, **Jira Server**, **Bitrix24**, **YouTrack**, **GitLab**, **Kaiten** или **YandexTracker**). Redmine обратную интеграцию не поддерживает.

Создание связи между DoQA и трекером

Доступны интеграции с **Jira Server**, **Jira Cloud**, **YouTrack**, **Yandex Tracker**, **Redmine**, **Bitrix24**, **GitLab** и **Kaiten**.

Раздел «Подключения» → «Интеграции» виден тем, у кого есть право `integration.view`, подключать, менять, проверять и отключать системы можно с `integration.manage`. Без права управления параметры подключений доступны только для чтения. Привязывать проект к трекеру может тот, у кого есть право `tracker.manage` на проекте (оно входит в роль «Тест-лид»).

Каталог подключений

Подключения организации собраны в одном списке: проекты и пространства выбирают из него. Вкладки списка: «Все», «Трекеры», «CI», «Требования», «С ошибками». Для каждого

подключения видны тип, адрес, статус проверки («Успешно», «Ошибка», «Не проверялось») и где оно используется.

1. Нажмите «Подключить» и выберите систему в каталоге. Уже подключённая система помечена «Подключено».
2. Введите адрес и учётные данные. Набор полей зависит от трекера и описан на его странице.
3. Нажмите «Проверить подключение», чтобы сверить параметры до сохранения: пока не нажата «Подключить», ничего не сохраняется.
4. Нажмите «Подключить». Сервер проверяет учётные данные. Если проверка не прошла, подключение создаётся, но с ошибкой: откройте его, исправьте параметры и проверьте снова.

Страница подключения содержит вкладки «Параметры» (секреты не показываются, заменить значение можно кнопкой «Заменить»), «Использование» (проекты, пространства, прогоны, расписания, привязки репозиториев, токены проектов) и «Аудит». Для нового адреса нужен новый токен: сохранённый на другой хост не отправляется.

Отключение

Перед отключением DoQA показывает последствия: какие проекты перестанут создавать баг-репорты в трекере (их привязки будут удалены, ссылки созданных багов останутся), сколько расписаний перестанет работать, сколько привязок репозиториев и токенов проектов будет удалено. Прошлые прогоны остаются в истории.

Адрес вебхука и секрет

Обратная интеграция получает события трекера на адрес вебхука DoQA.

Адрес	Особенности
`https://domain.doqa.app/api/integrations/<трекер>/webhook`	Прежний адрес. Продолжает работать, настройки в трекерах менять не нужно. Для подключения должна

Адрес	Особенности
	существовать запись, иначе ответ 404
`https://domain.doqa.app/api/integrations/<трекер>/webhook/r/<секрет>`	Новый необязательный адрес с секретом подключения. Секрет у каждого подключения свой, запрос без верного секрета отклоняется

Для обоих адресов DoQA не доверяет содержимому запроса: данные задачи он заново читает из API трекера с сохранёнными учётными данными. Секрет получает администратор с правом `integration.manage` вместе с готовым адресом в данных подключения (поле `webhookUrl` в API списка подключений). Подставляйте новый адрес при настройке вебхука в трекере, если нужно закрыть адрес секретом. Значения `<трекер>`: `bitrix`, `gitlab`, `jira-cloud`, `jira-server`, `kaiten`, `yandex-tracker`, `youtrack`.

Настройки интеграции проекта

Для каждого проекта можно настроить свои параметры интеграции.

1. Откройте проект в разделе «Проекты» → вкладка «Трекер». Если подключённых трекеров нет, подключите его в разделе интеграций: вкладка предложит перейти туда или попросить администратора.
2. Выберите трекер. Чтобы использовать другой трекер, сначала отключите текущий.
3. Выберите проект в трекере и в поле «Тип» - тип создаваемых задач. Поле выбора проекта зависит от трекера: для Yandex Tracker вместо проекта выбирается «Очередь», для Kaiten - «Пространство / доска». Для Jira Server, если у проекта есть доски, добавляется поле «Доска».
4. Сопоставьте статусы DoQA («Открыт», «В работе», «Тестирование», «Закрыт») и приоритеты DoQA («Низкий», «Средний», «Высокий») со значениями трекера. Для Kaiten сопоставление приоритетов недоступно: сопоставляются только статусы, а приоритет DoQA проставляется через собственное свойство карточки (см. **Kaiten**).
5. Сохраните настройки.

После этого станет доступным создание баг-репортов в трекере задач через интерфейс DoQA. «Отключить трекер...» убирает привязку проекта, само подключение остаётся в разделе интеграций.

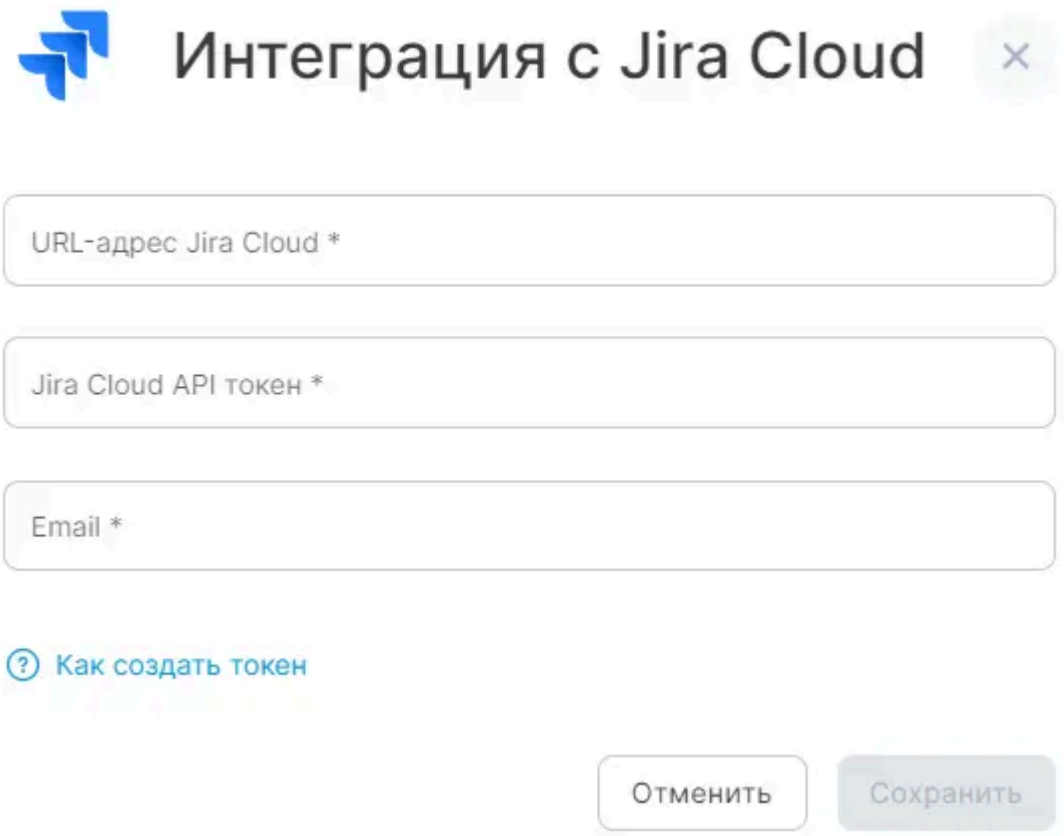
Требования из трекера

Связь с трекером используется не только для заведения баг-репортов: из трекера можно загружать требования. Требования загружаются только из Jira Server и YouTrack. Для остальных трекеров эта возможность недоступна. Настройка выполняется отдельно, на вкладке «Коннектор требований» в настройках пространства, см. **Коннектор требований**.

Jira Cloud

Для подключения интеграции с **Jira Cloud** выполните следующие шаги:

1. Введите URL-адрес Jira Cloud.
2. Введите API токен.
3. Введите email, связанный с вашим аккаунтом в Jira Cloud.



Обратная интеграция с Jira Cloud

DoQA позволяет настраивать обратную интеграцию с Jira Cloud, чтобы при изменении статуса или приоритета баг-репорта в трекере автоматически обновлялись соответствующие поля в связанном баг-репорте в DoQA.

Для настройки обратной интеграции выполните следующие шаги:

1. В Jira Cloud нажать «Настройки» → «Система»

2. В левом боковом меню выбрать пункт «Веб-хуки»

3. Нажать на кнопку «Создать веб-хук»

4. На экране создания вебхука заполнить:

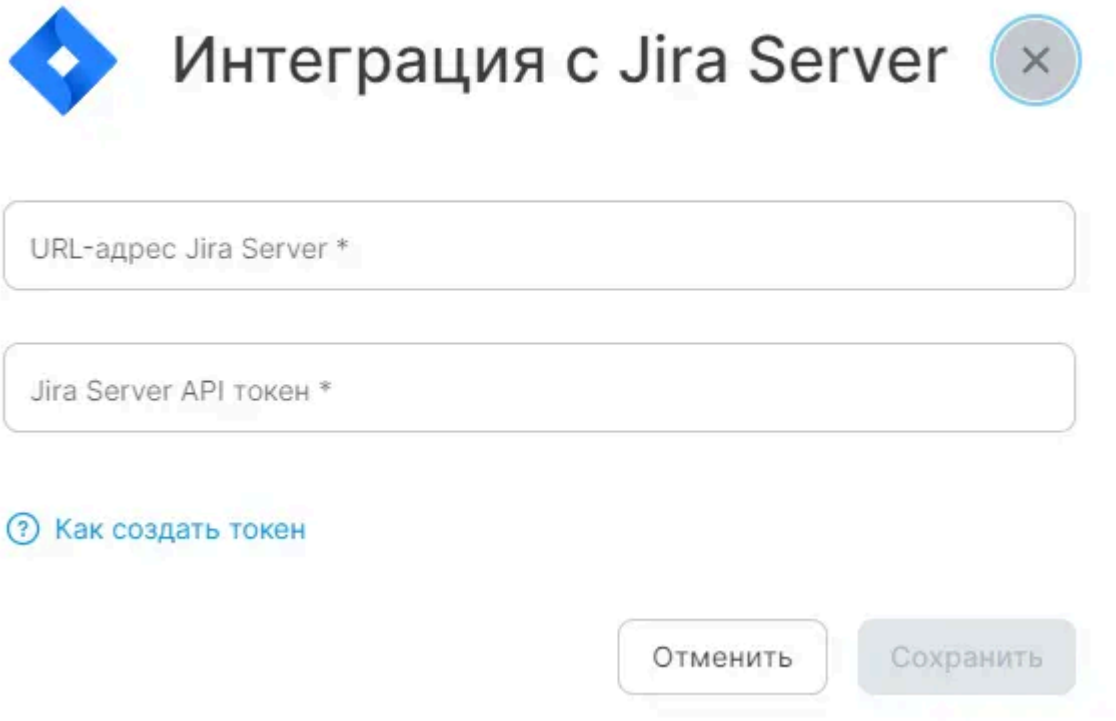
- Название вебхука
- URL по шаблону **https://domain.doqa.app/api/integrations/jira-cloud/webhook** ↗
- Отметить чек-бокс «Проблема» → «Обновление»

5. Подтвердить создание вебхука

Jira Server

Для подключения интеграции с **Jira Server** выполните следующие шаги:

1. Введите URL-адрес Jira Server и API токен.



> **Примечание:** Инструкция по получению токена доступна по ссылке в подсказке.

Обратная интеграция с Jira Server

DoQA позволяет настраивать обратную интеграцию с Jira Server, чтобы при изменении статуса или приоритета баг-репорта в трекере автоматически обновлялись соответствующие поля в связанном баг-репорте в DoQA.

Для настройки обратной интеграции выполните следующие шаги:

1. В Jira Server нажать «Настройки» → «Система»
2. В левом боковом меню выбрать пункт «Веб-хуки»
3. Нажать на кнопку «Создать веб-хук»

4. На экране создания вебхука заполнить:

- Название вебхука
- URL по шаблону **https://domain.doqa.app/api/integrations/jira-server/webhook** ↗
- Отметить чек-бокс «Проблема» → «Обновление»

5. Подтвердить создание вебхука

Jira Server как источник требований

Подключённую интеграцию с Jira Server можно использовать как источник **требований**. Настройка выполняется отдельно от обратной интеграции баг-репортов — на вкладке «Коннектор требований» в настройках пространства, см. **Коннектор требований**.

На стороне Jira до настройки коннектора заведите в проекте-источнике два кастомных поля: Administration → Issues → Custom fields → добавить поле.

- поле для ID связанных проверок — тип Text Field (multi-line), например `DoQA Tests`. В него DoQA списком пишет ID тест-кейсов и чек-листов, связанных с требованием.
- поле статуса покрытия — тип Select List (single choice), например `DoQA Coverage status`. Его значения (options) сопоставляются с пятью состояниями покрытия DoQA: «Нет тестов», «Все прошли», «Не все прошли», «Не прогонялись», «Требуется актуализация».

Имена полей и значений произвольные: нужные поля выбираются в настройках коннектора, а значения — в маппинге статусов покрытия. Если поля в проекте нет, коннектор покажет статус «Поле не найдено».

> **Внимание**
>
> После создания полей убедитесь, что они применены к нужному проекту в трекере и добавлены на экраны (screens) для отображения.

Yandex Tracker

Для подключения интеграции с **Yandex Tracker** выполните следующие шаги:

1. Создайте OAuth-приложение в Яндексе. Перейдите на **https://oauth.yandex.ru/client/new/** ↗ и заполните:

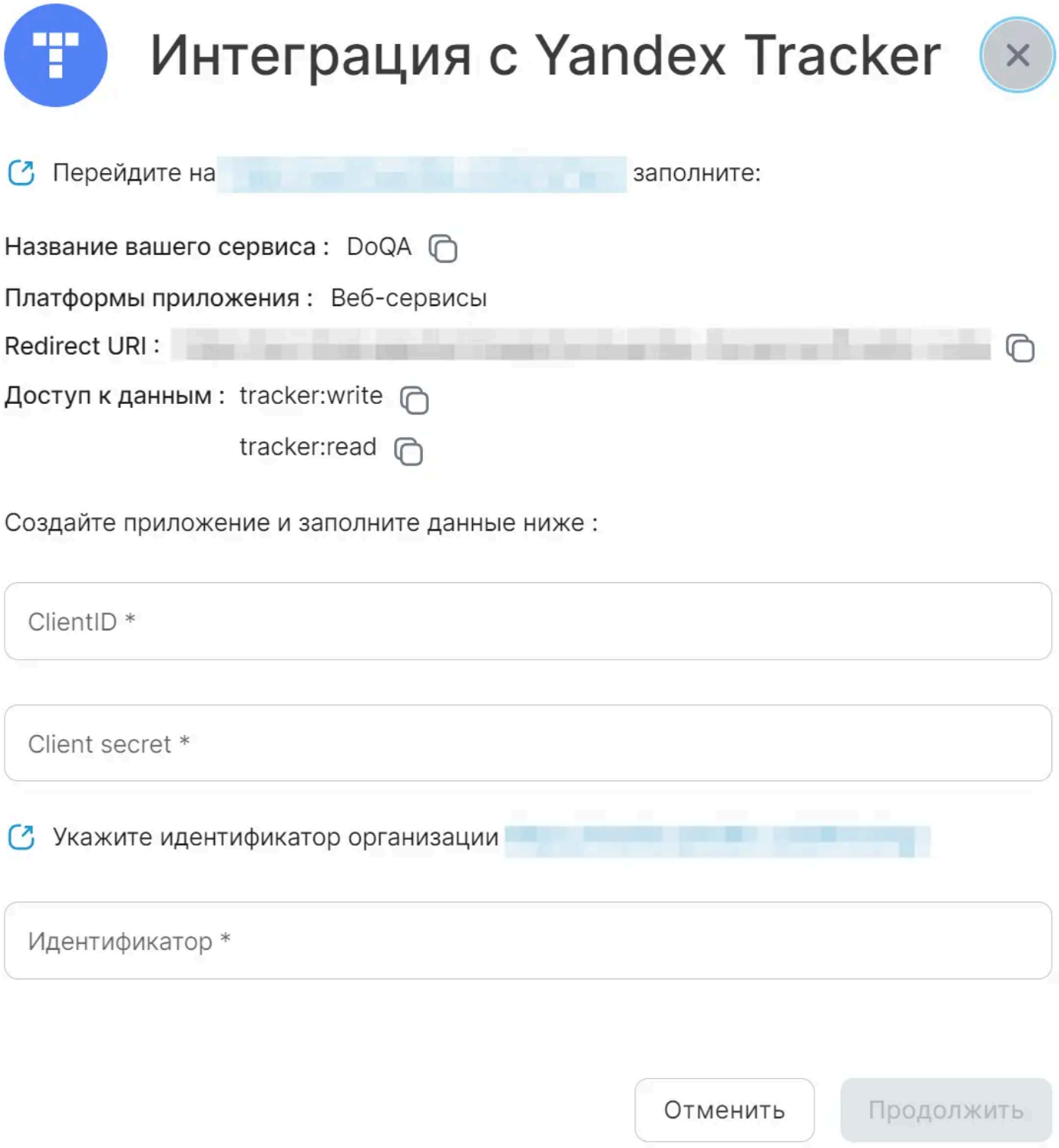
- Название вашего сервиса — `DoQA`
- Платформы приложения — «Веб-сервисы»
- Redirect URI — по шаблону `https://domain.doqa.app/api/integrations/yandex-tracker/verification-code`
- Доступ к данным — `tracker:write` и `tracker:read`

Эти же значения DoQA показывает прямо в форме подключения, с иконками копирования; Redirect URI подставляется под адрес вашего стенда автоматически.

2. Создайте приложение и скопируйте его Client ID и Client secret.

3. Введите в форме подключения следующие данные:

- «ClientID»
- «Client secret»
- «Заголовок организации» — X-Org-ID или X-Cloud-Org-ID
- «ID» — идентификатор организации, его можно посмотреть на **https://tracker.yandex.ru/admin/orgs** ↗



> **Примечание:** Если у вас только организация Yandex Cloud Organization, используйте заголовок X-Cloud-Org-ID, если только Яндекс 360 для бизнеса или оба типа организаций — X-Org-ID.

4. Когда вы заполните поля и подтвердите создание интеграции, откроется страница Яндекса. Там нужно авторизоваться под учётными данными пользователя, у которого есть доступ к проекту в Яндекс Трекере.

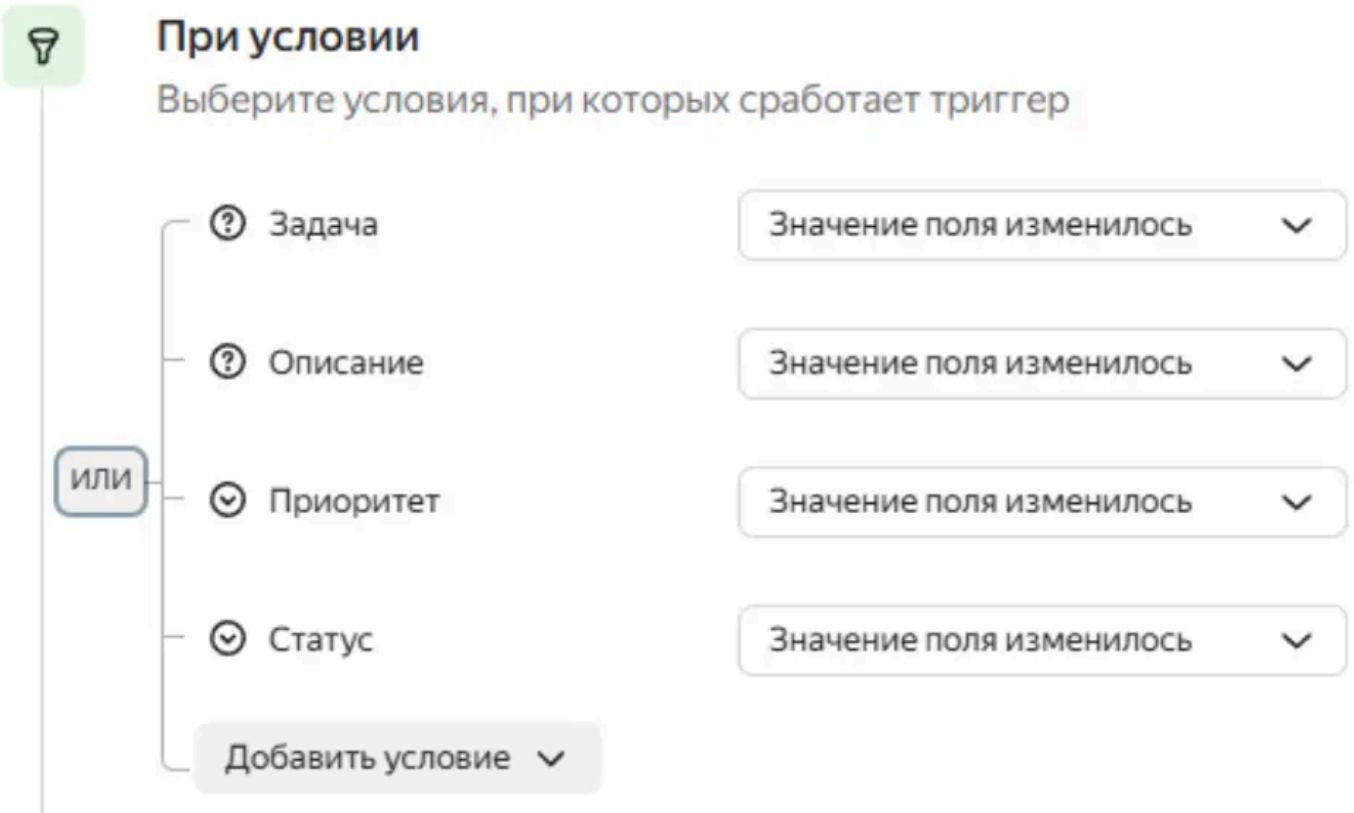
Обратная интеграция с Yandex Tracker

DoQA позволяет настраивать обратную интеграцию с Yandex Tracker, чтобы при изменении статуса или приоритета баг-репорта в трекере автоматически обновлялись соответствующие поля в связанном баг-репорте в DoQA.

Для настройки обратной интеграции выполните следующие шаги:

1. В Yandex Tracker перейти в раздел «Очереди»
2. Нажать на «Настройки очереди»
3. Слева выбрать «Автоматизация»
4. Нажать на кнопку «Создать» → «Триггер»
5. Заполнить условия.

- Выбрать: Задача, Описание, Приоритет и Статус
- Отметить значение «ИЛИ»



6. Заполнить раздел «Выполнить действие»:

- метод POST
- URL системы по шаблону **https://domain.doqa.app/api/integrations/yandex-tracker/webhook** ↗
- тип содержимого application/json
- email и password в теле запроса

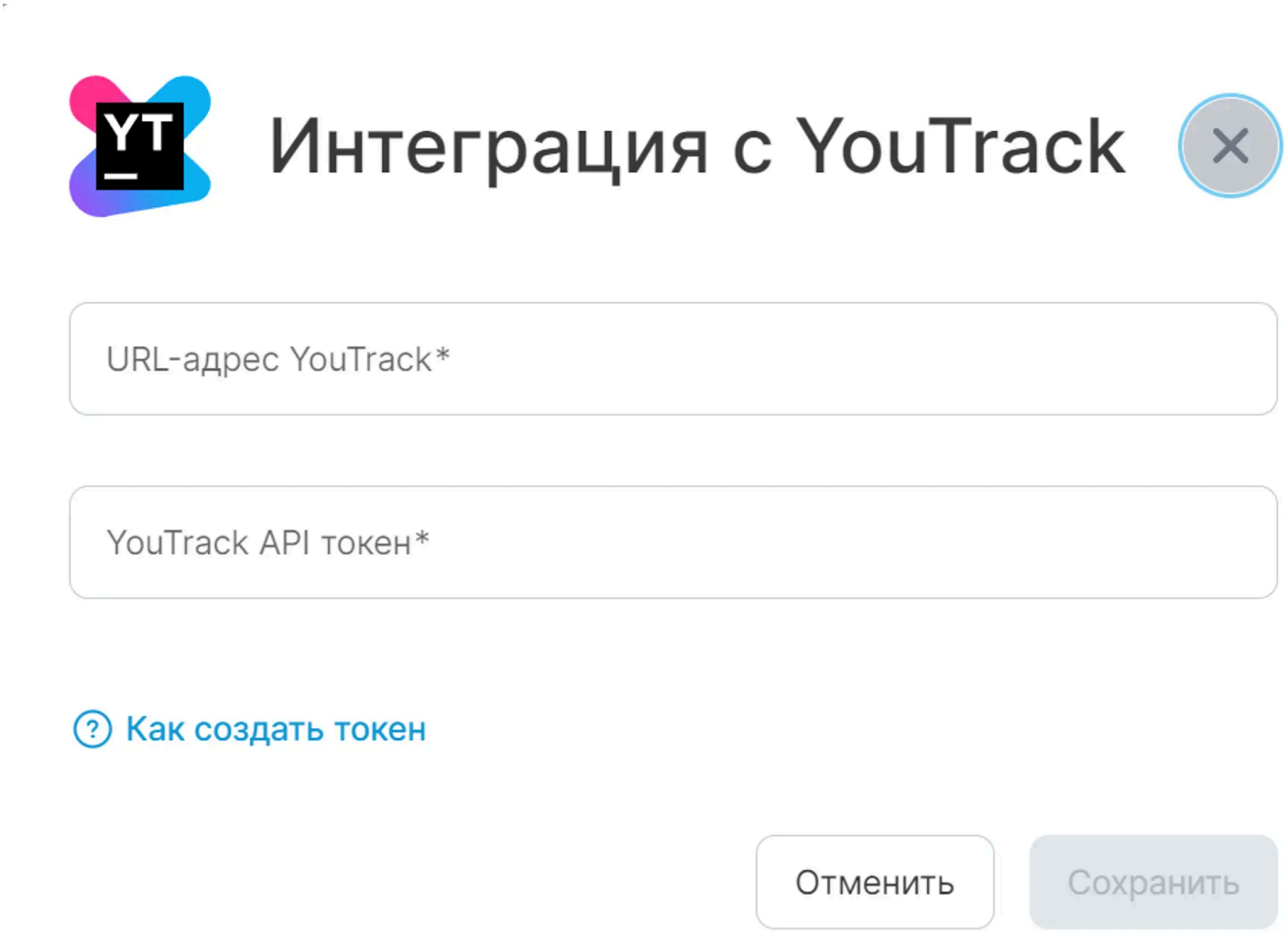
Тело запроса:

```
{
    "email": "user@example.com",
    "password": "password",
    "type": "yandexTracker",
    "issueId": "{{issue.key}}",
    "queue": "{{issue.queue.key}}",
    "updated": "{{currentDateTime}}",
    "priority": "{{issue.priority}}",
    "state": "{{issue.status}}",
    "summary": "{{issue.summary}}",
    "description": "{{issue.description}}"
}
```

YouTrack

Для подключения интеграции с **YouTrack** необходимо ввести:

1. URL-адрес YouTrack
2. YouTrack API токен

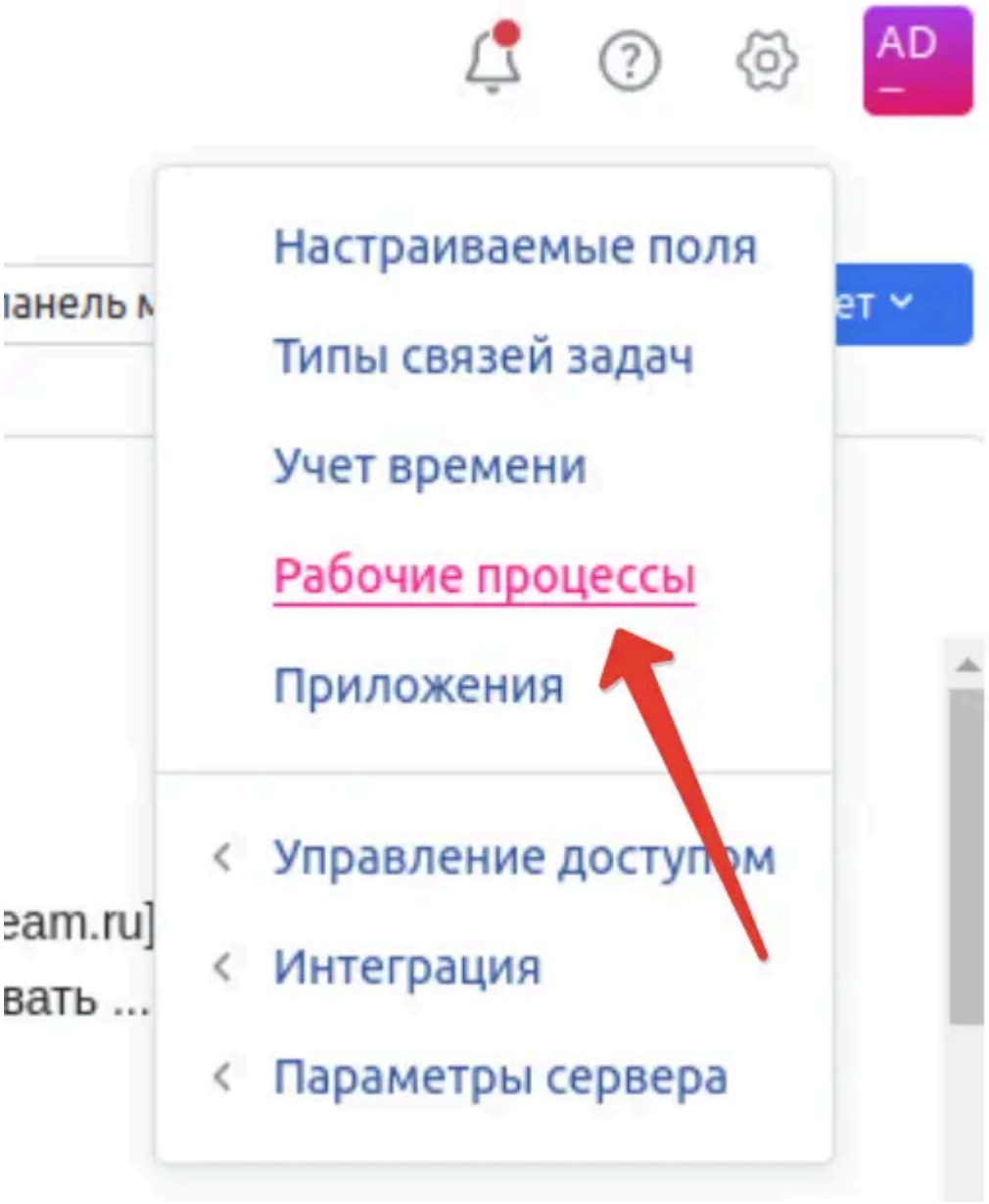


Обратная интеграция с YouTrack

DoQA позволяет настраивать обратную интеграцию с YouTrack, чтобы при изменении статуса или приоритета баг-репорта в трекере автоматически обновлялись соответствующие поля в связанном баг-репорте в DoQA.

Для настройки обратной интеграции выполните следующие шаги:

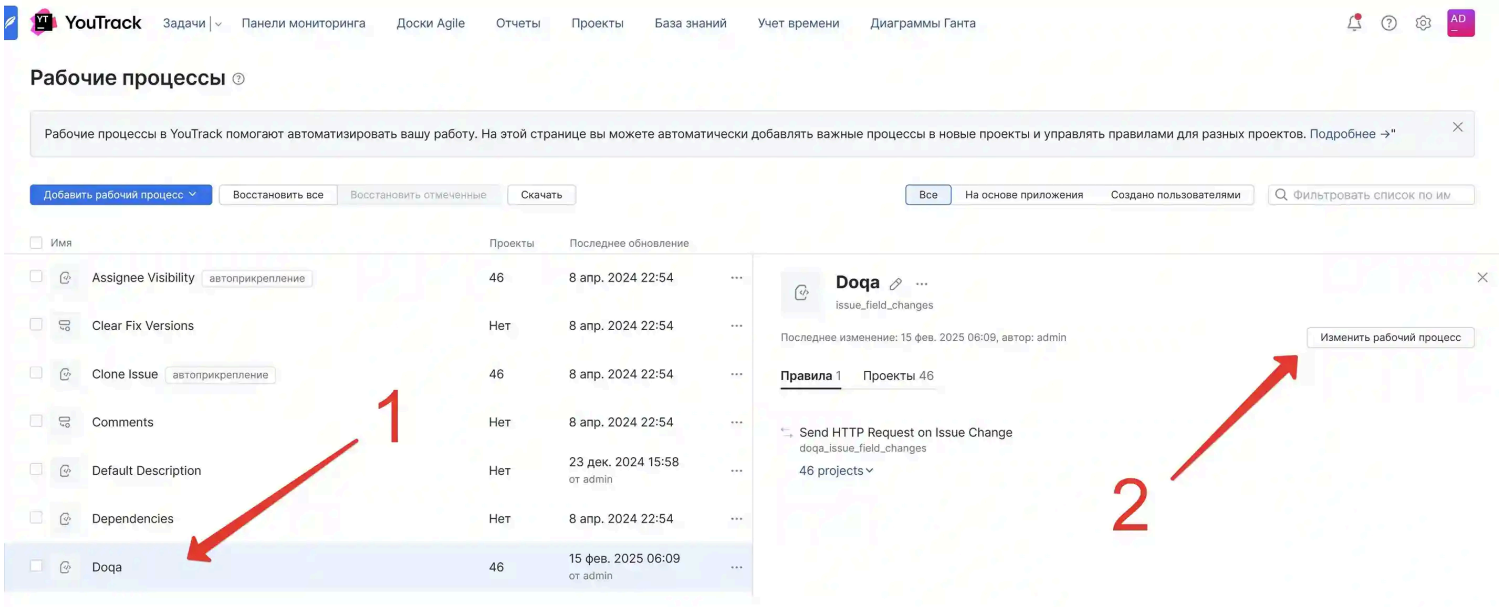
1. В настройках YouTrack зайти в рабочие процессы



2. Выбрать «Добавить рабочий процесс» → «Загрузить zip архив».

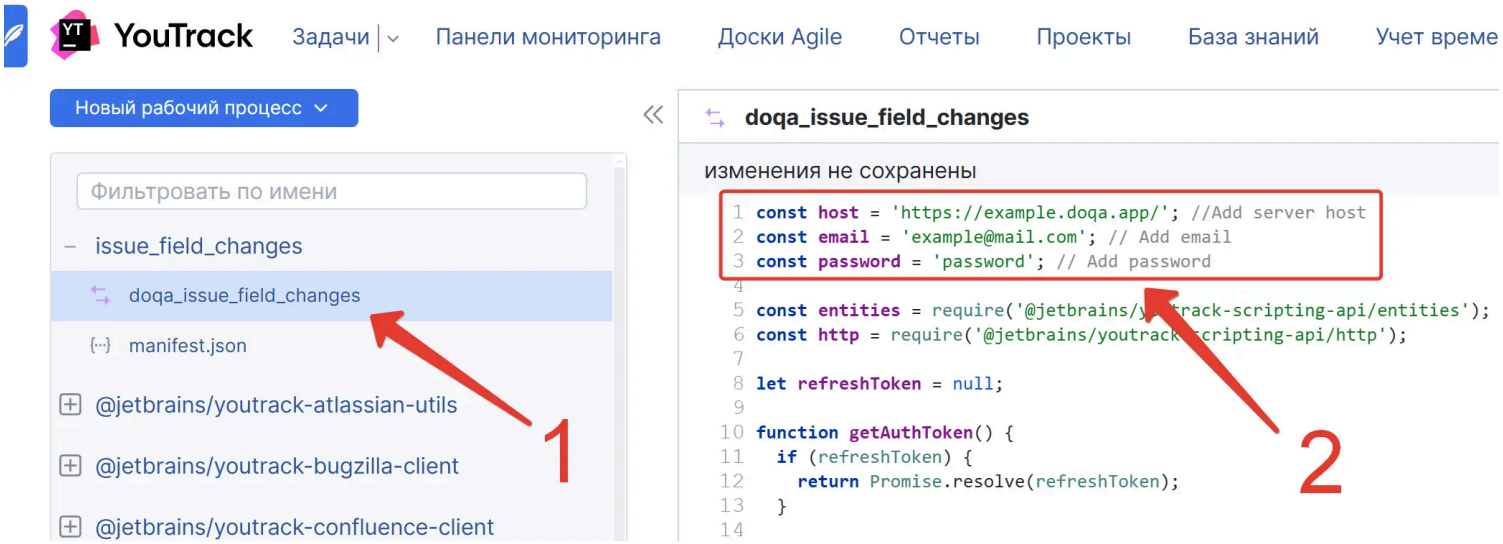
3. Прикрепить архив с настройками, который можно **скачать по ссылке:** ↗

4. Найти в списке рабочих процессов «Doqa» (цифра 1) и нажать «Изменить рабочий процесс» (цифра 2)



5. В списке найти doqa_issue_field_changes и заполнить поля:

- host — ссылка на вашу версию DoQA
- email — электронный адрес, используемый в качестве логина в DoQA. Используйте учетные данные пользователя DoQA с правами на администрирование
- password — пароль пользователя в DoQA



6. После необходимо привязать рабочий процесс к проекту:

- Зайти в рабочие процессы YouTrack
- Выбрать Doqa, затем «Изменить список проектов» и добавить нужные проекты

YouTrack как источник требований

Подключённую интеграцию с YouTrack можно использовать ещё и как источник **требований**. Настройка выполняется на вкладке «Коннектор требований» в настройках пространства, см. **Коннектор требований**.

На стороне YouTrack до настройки коннектора заведите в проекте-источнике два кастомных поля и примените их к этому проекту:

- поле для ID связанных проверок — тип «текст» (одиночное значение), например `DoQA Tests`. В него DoQA списком пишет ID тест-кейсов и чек-листов, связанных с требованием.
- поле статуса покрытия — тип «перечисление» (одиночное значение), например `DoQA Coverage status`. Его значения сопоставляются с пятью состояниями покрытия DoQA: «Нет тестов», «Все прошли», «Не все прошли», «Не прогонялись», «Требуется актуализация».

Имена полей произвольные: нужные поля выбираются в настройках коннектора. Если поля в проекте нет, коннектор покажет статус «Поле не найдено».

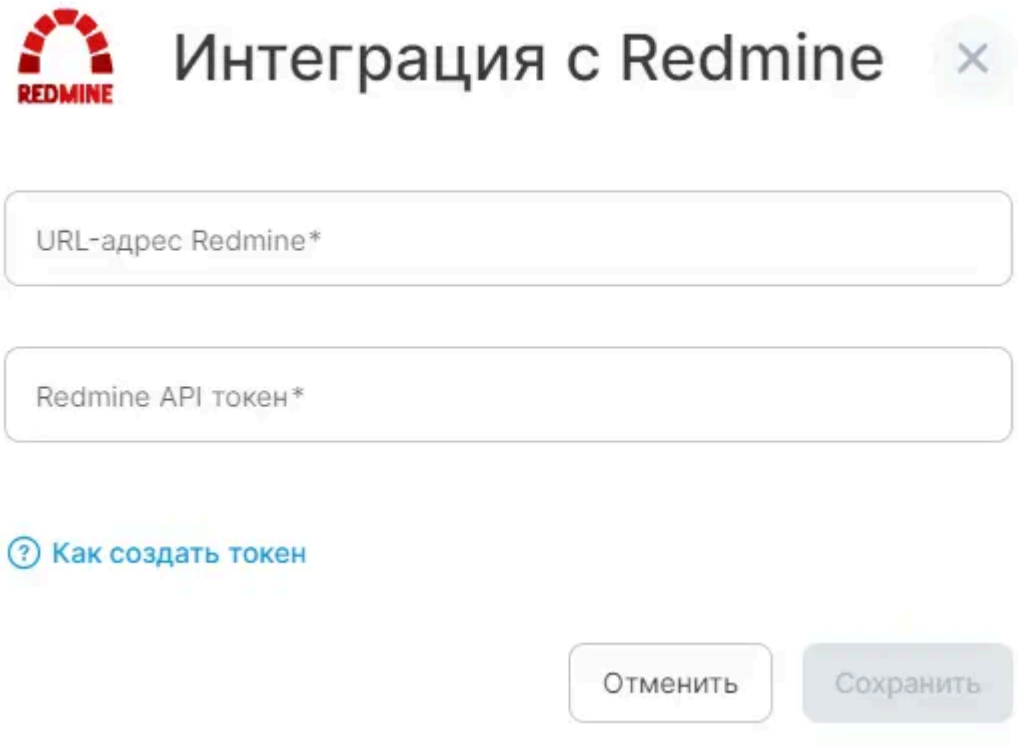
Внимание

Требованиям нужен свой workflow-скрипт в YouTrack, не тот, который описан выше в разделе «Обратная интеграция с YouTrack». Архив `youtrack_doqa_integration.zip` и правило `doqa_issue_field_changes` обслуживают только синхронизацию баг-репортов; загрузку требований они не включают.

Redmine

Для подключения интеграции с **Redmine** выполните следующие шаги:

1. Введите URL-адрес Redmine.
2. Введите API токен. Владелец API токена в Redmine должен иметь права администратора или быть владельцем проекта.

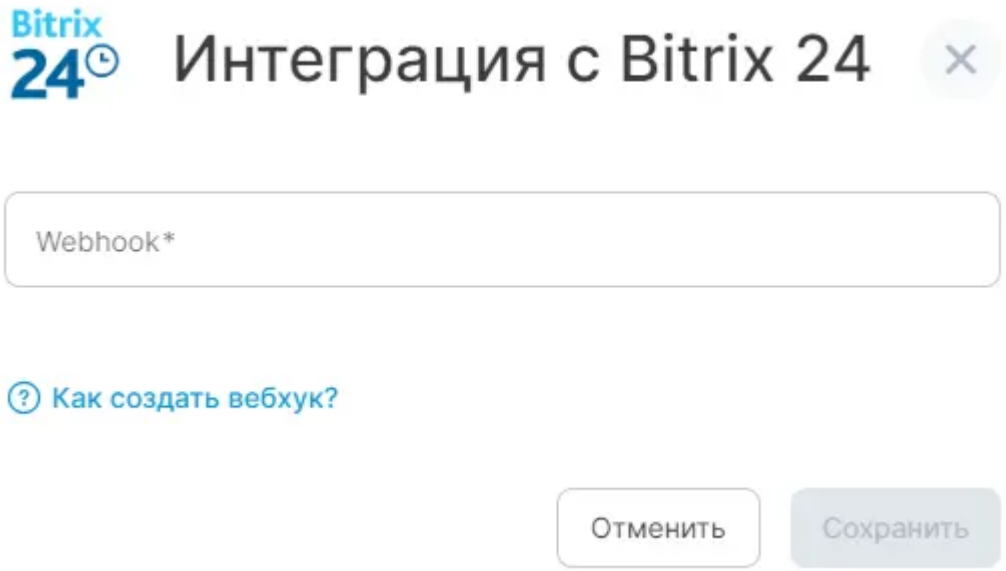


Обратная интеграция с Redmine

Обратная интеграция для Redmine не поддерживается: изменение статуса или приоритета задачи в Redmine не обновляет связанный баг-репорт в DoQA. Создание баг-репортов из DoQA в Redmine работает штатно.

Bitrix24

Для подключения интеграции с **Bitrix24** введите входящий вебхук Bitrix24 в поле «Webhook». В отличие от других трекеров, URL-адрес и API токен для Bitrix24 не запрашиваются. Инструкция по созданию вебхука доступна по ссылке «Как создать вебхук?» в форме подключения.



Для создания вебхука в Bitrix24:

- Перейдите в левое меню, нажмите «Карта сайта»
- Найдите пункт меню «Разработчикам»
- Нажмите на пункт меню «Другое»
- Нажмите на пункт меню «Входящий вебхук»
- Назначьте разрешения с помощью webhook:
  - Рабочие группы (sonet_group)
  - Диск (disk)
  - Задачи (task)
  - Пользователи (user)
  - Списки (lists)
  - Задачи (расширенные права) (tasks_extended)
- Нажмите «Сохранить»

Обратная интеграция с Bitrix24

DoQA позволяет настраивать обратную интеграцию с Bitrix24, чтобы при изменении статуса или приоритета баг-репорта в трекере автоматически обновлялись соответствующие поля в связанном баг-репорте в DoQA.

Внимание

В Bitrix24 нет низкого приоритета задачи. По умолчанию задача имеет нормальный приоритет. Если отметить иконку , то задача становится важной (приоритет «Высокий»).

Для настройки обратной интеграции выполните следующие шаги:

1. Откройте карту сайта и перейдите в раздел «Разработчикам» → «Готовые сценарии»
2. В открывшемся окне выберите «Другие»
3. Выберите «Исходящий вебхук»
4. Заполните ссылку на стенд DoQA по шаблону https://domain.doqa.app/api/integrations/bitrix/webhook ↗
5. Выберите событие «Обновление задачи (ONTASKUPDATE)»
6. Подтвердите создание вебхука.

Kaiten

Для подключения интеграции с **Kaiten** выполните следующие шаги:

1. Введите адрес вашей компании в Kaiten в поле «URL-адрес Kaiten». Адрес должен начинаться с `https://`.
2. Введите токен в поле «Kaiten API токен». Инструкция по получению токена доступна по ссылке «Как создать токен» в форме подключения.
3. Нажмите «**Сохранить**».

При сохранении DoQA проверяет соединение с Kaiten. Если адрес или токен неверны, интеграция не создаётся и форма остаётся открытой.

Особенности настройки интеграции проекта

Связь проекта в DoQA с доской в Kaiten настраивается так же, как для остальных трекеров (см. **Настройки интеграции проекта**), но форма отличается:

- Вместо поля «Проекты» выбирается «Пространство / доска». В списке — доски всех пространств Kaiten в виде «Пространство / Доска». Архивные пространства и доски в список не попадают.
- В поле «Тип» перечислены типы карточек Kaiten.
- Статусы DoQA сопоставляются с колонками и подколонками выбранной доски. Подколонка отображается как «Колонка / Подколонка».
- Сопоставление приоритетов для Kaiten не показывается. В Kaiten нет собственного поля приоритета, поэтому DoQA сам создаёт в компании кастомное свойство типа «список» с именем `Priority (DOQA)` и значениями `Low`, `Medium`, `High` и проставляет его в карточках.

Внимание

Если создать свойство `Priority (DOQA)` не удалось (например, у владельца токена нет прав или свойство недоступно на вашем тарифе), связь проекта всё равно сохранится. Но баг-репорты будут

уходить в Kaiten без приоритета, а обратная синхронизация приоритета работать не будет.

Обновление карточки из DoQA

Kaiten — единственный трекер, в котором DoQA обновляет уже созданную задачу: при редактировании баг-репорта в DoQA в связанной карточке Kaiten меняются заголовок, описание, колонка (по сопоставлению статусов) и приоритет. У остальных трекеров из DoQA только создаются задачи.

Если карточку перенесли на другую доску, заголовок и описание всё равно обновятся, а колонку DoQA менять не будет: иначе карточка вернулась бы на доску из настроек интеграции.

Обратная интеграция с Kaiten

DoQA позволяет настраивать обратную интеграцию с Kaiten, чтобы при изменении карточки в трекере автоматически обновлялись соответствующие поля в связанном баг-репорте в DoQA. Синхронизируются статус (по колонке карточки), приоритет (по свойству `Priority (DOQA)`), заголовок и описание.

Для настройки обратной интеграции выполните следующие шаги:

1. Скопируйте callback URL вида `https://domain.doqa.app/api/integrations/kaiten/webhook`. Готовый адрес для вашего стенда DoQA показан в форме подключения — в блоке «Вебхук обратной синхронизации».
2. В Kaiten откройте нужное пространство.
3. Перейдите в настройки пространства и выберите внешние вебхуки. Подробная инструкция — по ссылке «Открыть гайд Kaiten по настройке вебхука» в форме подключения.
4. Добавьте скопированный callback URL и включите отправку событий карточек. Изменения полей приходят в событии `card:update`.
5. Подтвердите создание вебхука.

Совет

Вебхук настраивается на пространство, а связь проекта в DoQA — на конкретную доску. События с других досок этого пространства DoQA получит, но пропустит: обновляются только баг-репорты, привязанные к доске из настроек интеграции проекта.

GitLab

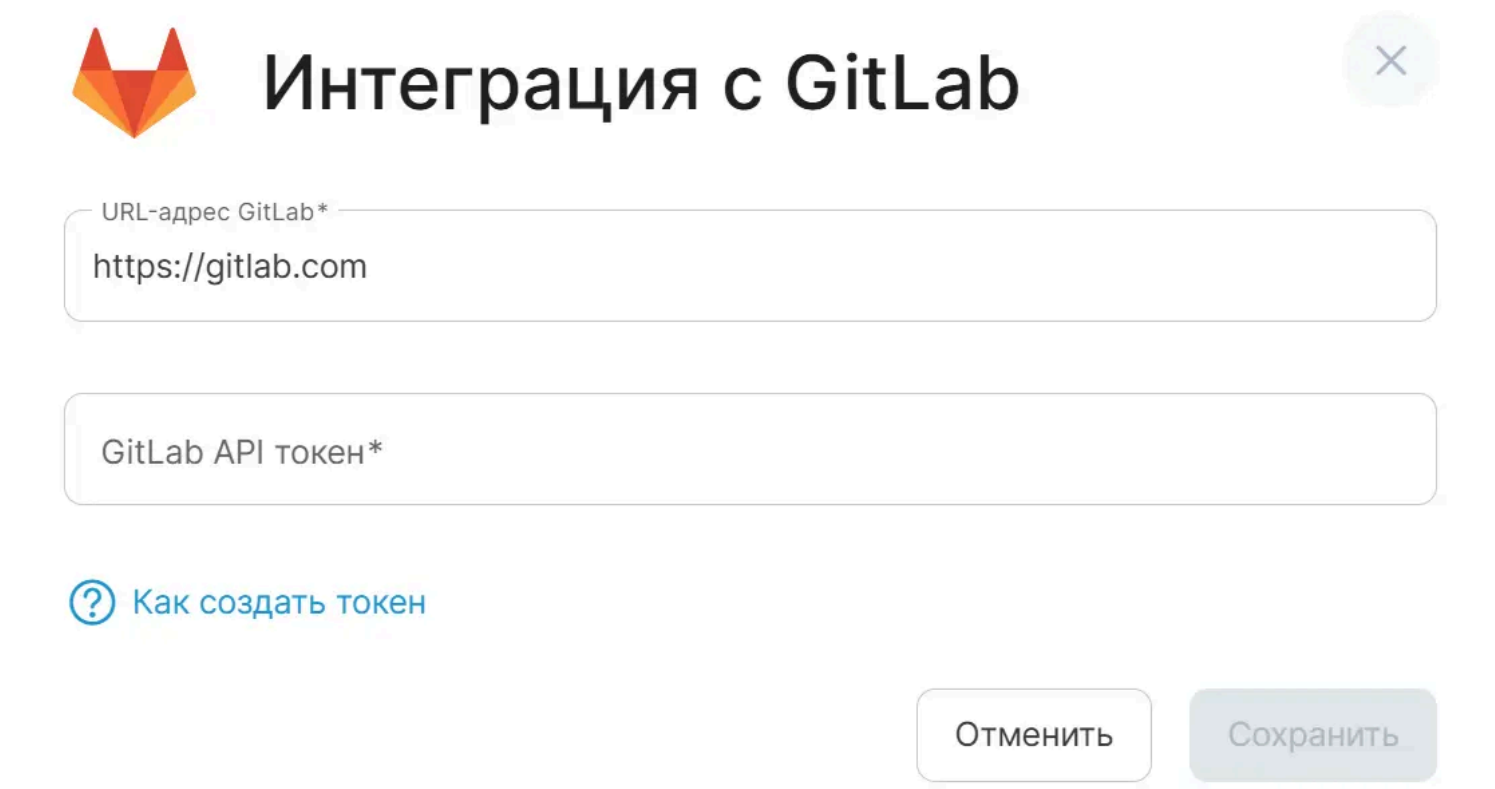
GitLab в DoQA используется в двух сценариях: как трекер задач — создание баг-репортов из DoQA и синхронизация их статусов (описано на этой странице); как CI/CD-платформа — запуск автотестов и загрузка результатов (см. **Интеграции с CI/CD**). Это две отдельные интеграции: карточка GitLab присутствует в обоих блоках раздела **Интеграции** и подключается в каждом из них отдельно.

В GitLab отсутствуют стандартные поля «Приоритет» и «Статус», поэтому для связи с трекером нужно использовать лейблы GitLab. Рекомендуется создать лейблы, соответствующие стандартным статусам и приоритетам DoQA:

- Приоритеты: высокий, средний, низкий.
- Статусы: открыт, в работе, тестирование, закрыт.

Для подключения интеграции с **GitLab** выполните следующие шаги:

1. Заполните ссылку на систему **https://gitlab.com** ↗
2. Введите токен
3. Сохраните изменения.



Обратная интеграция с GitLab

DoQA позволяет настраивать обратную интеграцию с GitLab, чтобы при присвоении баг-репорту в трекере лейблов, обозначающих статусы или приоритеты, автоматически обновлялись соответствующие поля в связанном баг-репорте в DoQA.

Кроме того, при закрытии issue в GitLab связанный баг-репорт в DoQA переходит в статус «Закрыт», независимо от того, какие на нём лейблы.

Для настройки обратной интеграции выполните следующие шаги:

1. В GitLab в настройках проекта выберите «Webhooks»
2. Нажмите на кнопку добавления нового вебхука
3. Заполните ссылку по шаблону **https://domain.doqa.app/api/integrations/gitlab/webhook** ↗
4. Отметьте чек-бокс Issue events
5. Подтвердите создание вебхука.

Интеграции с CI/CD

Интеграция с CI/CD нужна, чтобы запускать автотесты из интерфейса DoQA: целым прогоном, выборочно из плеера прогона или по одному тесту. Отправлять результаты в DoQA можно и без нее: утилитой doqa-cli или через API, см. **Автотесты**.

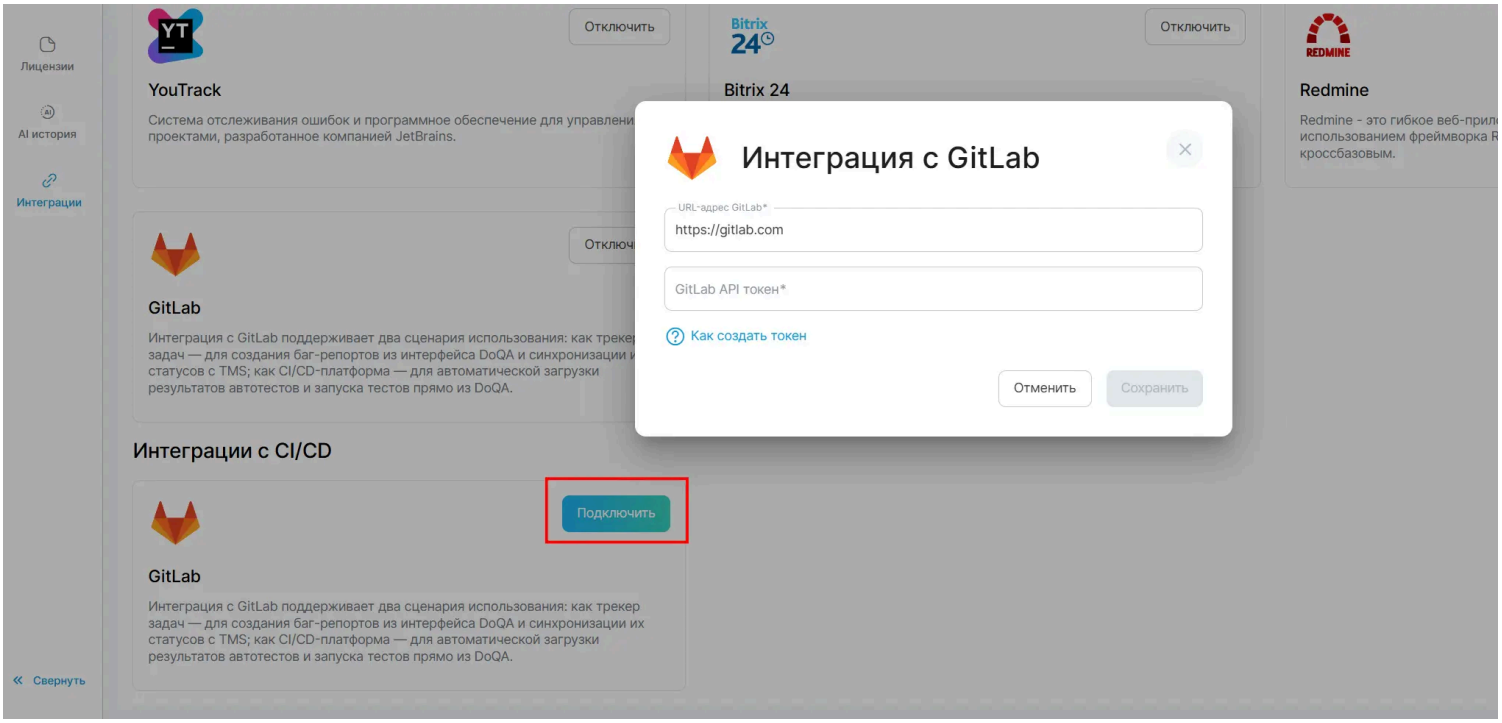
На уровне администратора подключаются GitLab, Jenkins и TeamCity; такое подключение видно всем пространствам тенанта. Остальные CI-системы (всего DoQA поддерживает девять) и несколько подключений одного типа настраиваются на уровне пространства — см. **Подключение CI-систем**.

> **GitLab подключается дважды**
>
> Раздел «Интеграции» состоит из двух блоков — «Интеграции с трекерами» и «Интеграции с CI/CD». Это две независимые интеграции с разными подключениями. Если GitLab уже подключен как трекер багов (см. **GitLab**), для запуска автотестов его всё равно нужно подключить отдельно — в блоке «Интеграции с CI/CD».

Подключение CI/CD-системы

Раздел «Интеграции» виден тем, у кого есть право `integration.view`, подключать системы можно с `integration.manage`. Подключать CI к пространству и менять привязки может тот, у кого есть право `ci.manage` на пространстве (роль «Тест-лид»).

1. Перейдите в настройки администратора, раздел «Интеграции».
2. В блоке «Интеграции с CI/CD» на карточке нужной системы — GitLab, Jenkins или TeamCity — нажмите «**Подключить**».
3. Заполните «URL-адрес» и «API токен»:
   - GitLab — токену нужны права (scopes) `api` и `read_api`;
   - Jenkins — токен в формате `логин:api-токен`;
   - TeamCity — токен доступа из профиля пользователя.
4. Нажмите «**Сохранить**».



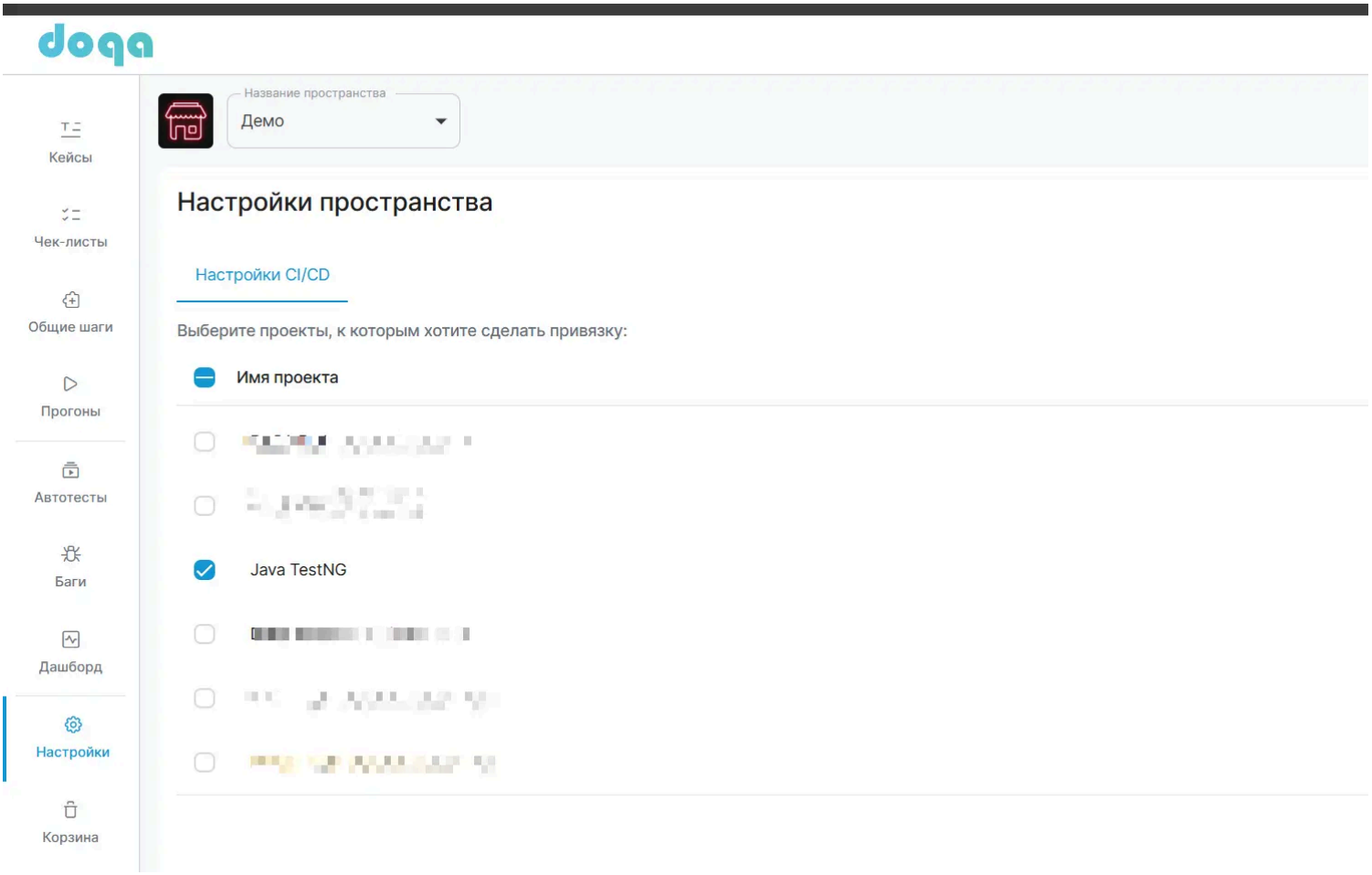
Если интеграция подключена, кнопка «Подключить» на карточке сменится на «**Отключить**».

Каждая система подключается один раз на весь тенант. Если нужно несколько подключений одного типа (например, два инстанса Jenkins), заводите их в настройках пространства, см. **Подключение CI-систем**.

Привязка проектов к пространству

Одного подключения интеграции недостаточно: нужно указать, какие проекты из GitLab доступны для запуска автотестов в конкретном пространстве.

1. Перейдите в пространство и в левом боковом меню нажмите на иконку настроек.
2. Откройте вкладку «Настройки CI/CD».
3. Под подсказкой «Выберите проекты, к которым хотите сделать привязку:» отметьте чек-боксами нужные проекты. Чек-бокс «Имя проекта» в шапке списка выбирает и снимает все сразу.
4. Нажмите «**Сохранить**». Кнопка неактивна, пока в списке ничего не изменилось.



Отмеченные проекты будут доступны для выбора при **создании прогона через запуск автотестов** и при **запуске автотестов из плеера прогона**.

Если проектов не видно

На вкладке «Настройки CI/CD» два пустых состояния, и означают они разное.

- «Интеграция CI/CD не настроена» и ссылка «Перейдите в раздел Интеграции» — GitLab не подключен в блоке «Интеграции с CI/CD». Подключите его, см. **Подключение CI/CD-системы**.
- «Список доступных проектов из CI/CD пуст. Проверьте настройки интеграции и валидность токена.» — интеграция подключена, но DoQA не получила от GitLab ни одного проекта. Проверьте URL-адрес, срок действия токена и его права (`api` , `read_api`), а также то, что у владельца токена есть доступ к нужным проектам.

Что должен уметь ваш пайплайн

При старте пайплайна DoQA передает в него набор переменных `DOQA_*` — полный список см. в **Переменных пайплайна**. Списка тестов среди переменных нет.

Чтобы запускались именно те тесты, которые выбрал пользователь, пайплайн должен сам запросить тест-план и отфильтровать тесты по полученным Allure ID:

```
GET {DOQA_URL}/api/integrations/ci/test-plan
    ?pipeline_id=$CI_PIPELINE_ID
    &ci_project_id=$CI_PROJECT_ID
    &space_id=$DOQA_SPACE_ID
    &token=$DOQA_TOKEN
```

В ответ приходят список Allure ID (`tests`), ID прогона в DoQA (`runId`) и признак `isNewRun`. Если пропустить этот шаг, прогонятся все тесты проекта.

Полное описание контракта, формат ответа и примеры для `.gitlab-ci.yml` — **Тест-план по API**.

Переменная DOQA_SPACE_ID

Заводить `DOQA_SPACE_ID` в настройках CI вручную нужно только для пайплайнов, которые стартуют сами — по коммиту или расписанию. При запуске из DoQA значение приходит в пайплайн автоматически и перекрывает заданное в настройках CI.

Смотрите также

- **Автотесты** — как результаты автотестов попадают в DoQA и как их запускать.
- **Прогоны** — где хранятся результаты.
- **GitLab** — подключение GitLab как трекера багов, другая интеграция.

Настройки DoQA AI

Здесь администратор выбирает LLM-провайдера, под которым работают **AI-функции**, правит промпты и смотрит расход коинов или токенов.

Откройте администрирование и выберите в боковом меню группу «ИИ». Провайдер и промпты доступны тем, у кого есть право `ai_settings.manage`, историю запросов и расход смотрят с `ai_usage.view` (роль «Аудитор» видит только расход).

Страница состоит из трёх частей, они переключаются в её собственном левом меню:

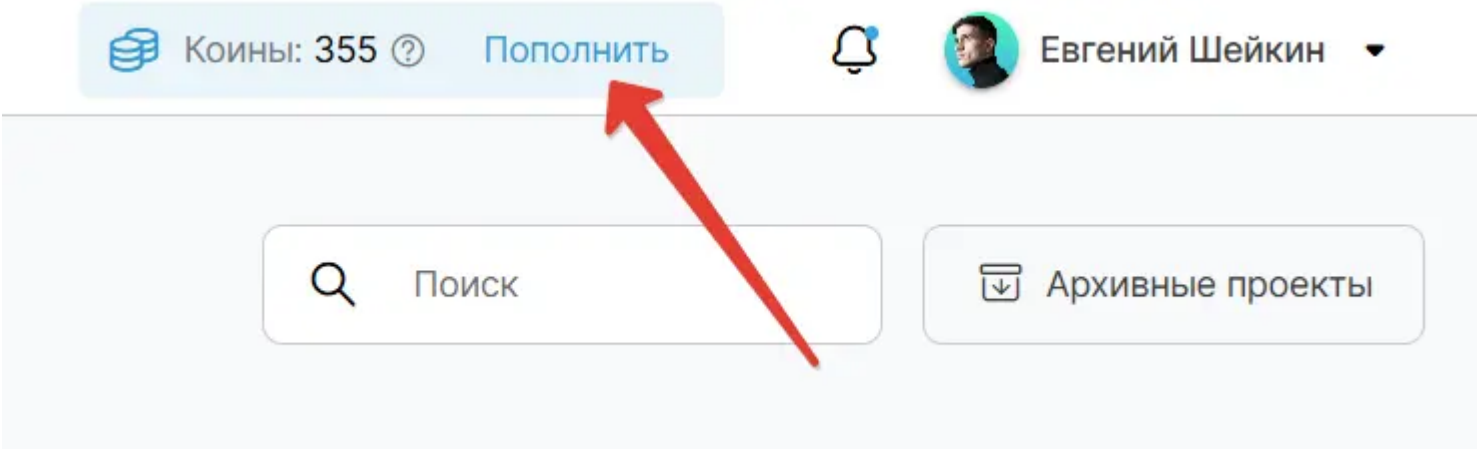
- «Провайдер» — какая модель и через какой API обслуживает AI-функции.
- «История операций» (в меню администрирования пункт «Использование») - что и кем было сгенерировано, во сколько это обошлось.
- «Промпты» — системные инструкции, которые DoQA отправляет модели. Сгруппированы по типам документации.

Использование встроенного провайдера

В облачной версии по умолчанию подключен встроенный провайдер DoQA. Он не требует настройки: ни токен, ни модель указывать не нужно. В серверной версии встроенного провайдера нет. Там AI-функции не заработают, пока вы не подключите своего провайдера.

Встроенный провайдер расходует коины — внутреннюю валюту DoQA. Новые клиенты получают 100 приветственных коинов.

Текущий баланс показан в шапке приложения; он обновляется после каждой AI-операции. Кнопка «**Пополнить**» рядом с балансом видна тем, у кого есть право `billing.manage`, и открывает покупку пакета коинов в разделе **«Лицензии и оплата»**.



Стоимость операций:

Операция	Цена
Генерация тест-кейса	6 коинов
Генерация набора тест-кейсов	30 коинов
Генерация чек-листа	6 коинов
AI анализ тест-кейса	13 коинов
Генерация названия баг-репорта	2 коина

Генерация тестов по требованию тарифицируется по виду результата: тест-кейс и чек-лист — 6 коинов, набор тест-кейсов — 30 коинов.

Если коинов на балансе не хватает, операция не выполняется и попадает в историю со статусом «Прервано». Коины за неё не списываются.

Подключение своего провайдера

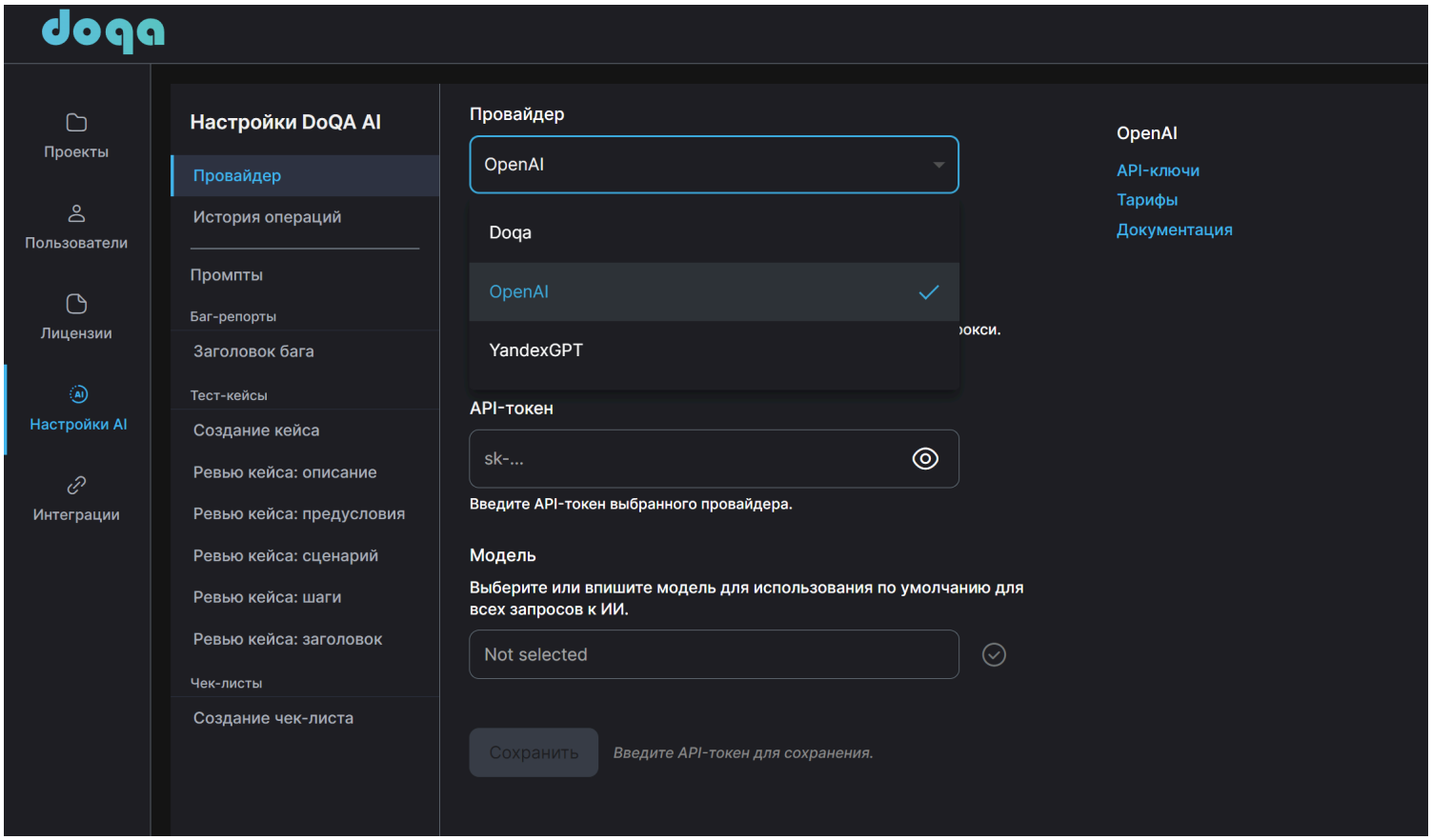
Своего провайдера, в том числе локальную модель, можно подключить и в облаке вместо встроенного, и в серверной версии. В этом случае коины не расходуются: вы платите своему LLM-провайдеру по его тарифу, а DoQA считает только токены.

В разделе «Провайдер» выберите провайдера в списке. Доступны:

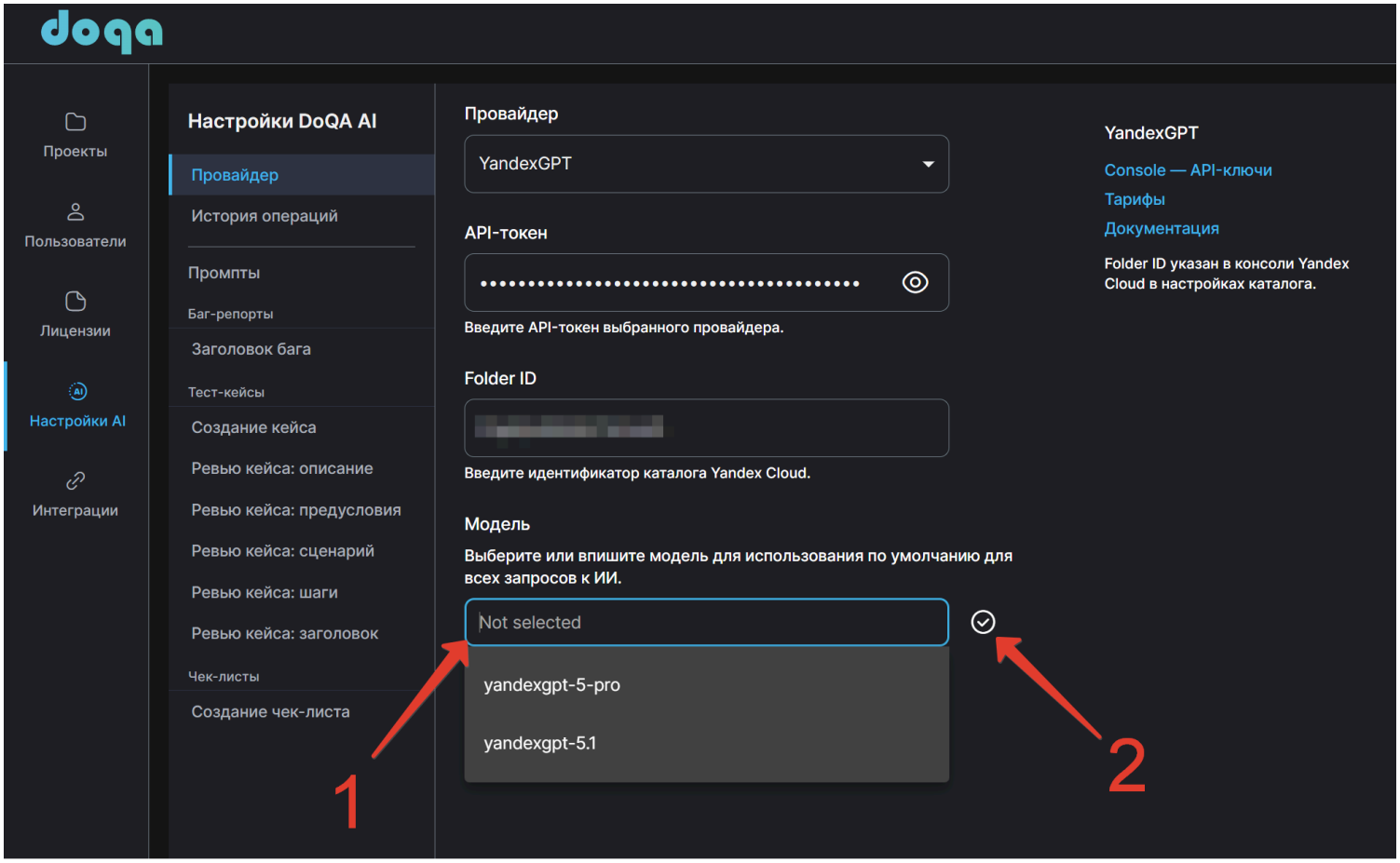
- **DoQA** — встроенный провайдер, только в облачной версии.

- **OpenAI** — нужны «API-токен» и «Модель». Поле «API-эндпоинт» необязательное: пустое — запросы идут на api.openai.com, заполненное — на указанный адрес. Так подключают прокси.
- **YandexGPT** — нужны «API-токен», «Folder ID» (идентификатор каталога Yandex Cloud) и «Модель».
- **Свой провайдер** — любой OpenAI-совместимый API. Обязательны «API-эндпоинт» (например, `http://localhost:11434/v1`) и «Модель». API-токен указывайте, только если ваш эндпоинт требует аутентификации.

Справа от формы DoQA показывает панель выбранного провайдера со ссылками «API-ключи», «Тарифы» и «Документация».



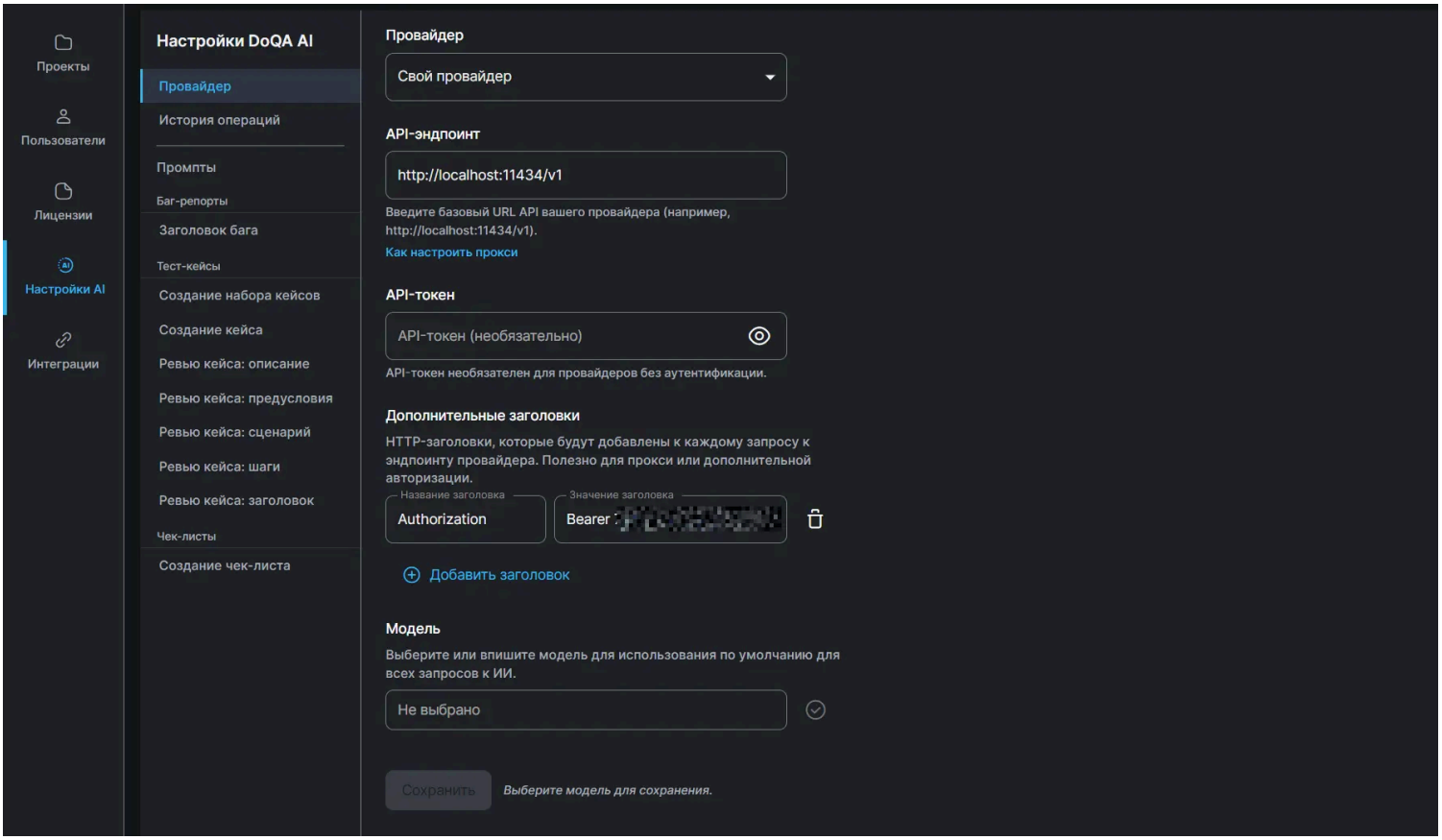
Модель можно выбрать из списка или вписать вручную (цифра 1). Чтобы проверить, что DoQA сможет подключиться к провайдеру с этими данными, нажмите кнопку справа от поля модели — «Проверить подключение» (цифра 2).



Кнопка «**Сохранить**» остаётся неактивной, пока не заполнены обязательные поля; рядом с ней написано, чего именно не хватает, например, «Введите API-токен для сохранения» или «Выберите модель для сохранения».

Совет

Настройки каждого провайдера хранятся отдельно. Переключившись между провайдерами в списке, вы не потеряете введённые ранее токен, эндпоинт и модель.



Дополнительные заголовки

Для «Своего провайдера» доступен блок «Дополнительные заголовки» — произвольные HTTP-заголовки, которые DoQA добавит к каждому запросу к вашему эндпоинту. Нужны, когда запросы идут через прокси или требуют дополнительной авторизации.

Нажмите «**Добавить заголовок**» и заполните пару «Название заголовка» / «Значение». Лишнюю пару можно удалить.

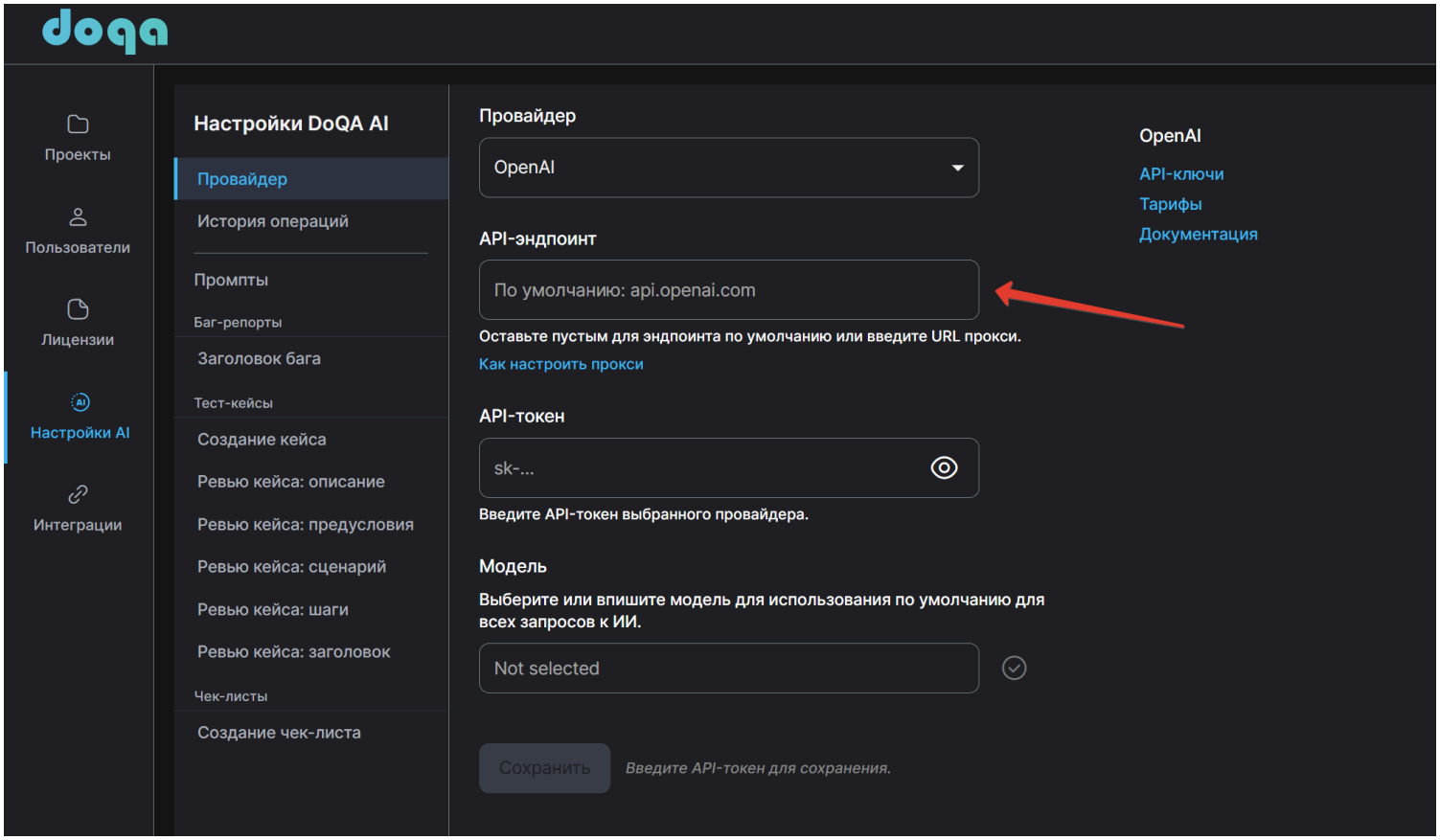
> **Внимание**
>
> Нестандартные модели могут не соблюсти формат ответа, и тогда DoQA не разберёт результат генерации. Используйте модели, совместимые с OpenAI.

Особенности серверной версии

В серверной версии встроенный провайдер DoQA из списка убран, по умолчанию выбран «Свой провайдер», а коины не используются. Пока провайдер не настроен, AI-функции в интерфейсе заблокированы с подсказкой «AI недоступен: не настроен провайдер в настройках администратора».

Настройка прокси для отдельных провайдеров

Для работы с LLM от OpenAI может понадобиться прокси: его адрес указывают в поле «API-эндпоинт». Подробнее о настройке прокси



Промпты

Промпт — это системная инструкция, которую DoQA отправляет модели вместе с вашим запросом. Правя промпт, вы меняете поведение AI-функции: формат результата, степень детализации, язык, принятые в команде формулировки.

Редактировать можно девять промптов. В левом меню они сгруппированы по типам документации:

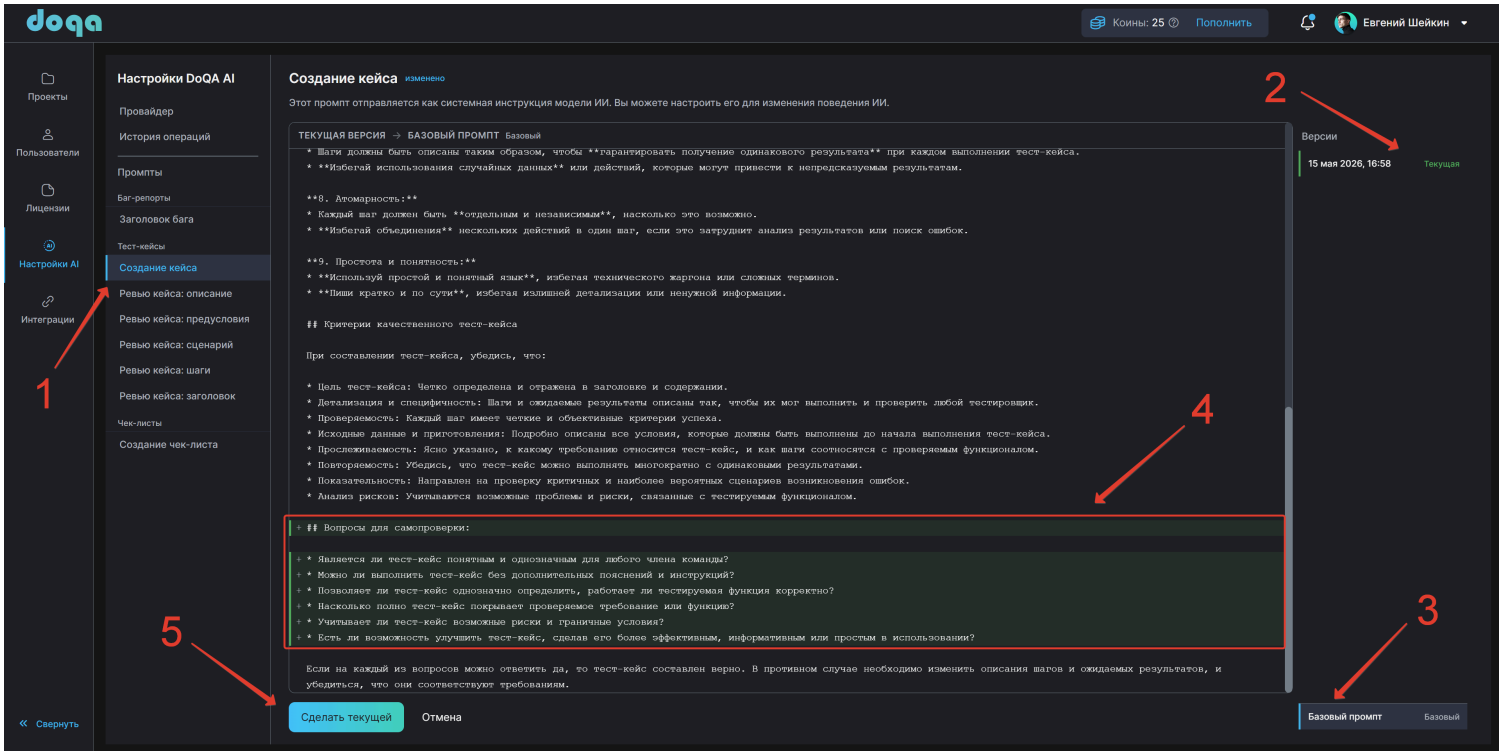
Группа	Промпт	Какая функция его использует
Баг-репорты	«Заголовок бага»	генерация названия баг-репорта

Группа	Промпт	Какая функция его использует
Тест-кейсы	«Создание кейса»	генерация тест-кейса
Тест-кейсы	«Создание набора кейсов»	генерация набора тест-кейсов
Тест-кейсы	«Ревью кейса: заголовок»	AI-анализ, блок «Анализ названия»
Тест-кейсы	«Ревью кейса: описание»	AI-анализ, блок «Анализ описания»
Тест-кейсы	«Ревью кейса: предусловия»	AI-анализ, блок «Анализ предусловия»
Тест-кейсы	«Ревью кейса: сценарий»	AI-анализ, блоки «Анализ цели тест-кейса» и «Проверка на множественность сценариев»
Тест-кейсы	«Ревью кейса: шаги»	AI-анализ, блок «Анализ шагов»
Чек-листы	«Создание чек-листа»	генерация чек-листа

Выберите промпт в списке (цифра 1) и отредактируйте текст. Длина промпта — не больше 10 000 символов; счётчик под полем показывает текущую длину. Кнопка «**Сохранить**» неактивна, если поле пустое или лимит превышен.

Каждое сохранение создаёт новую версию. Список версий справа показывает, какая из них «Текущая» (цифра 2). Исходный вариант всегда доступен как «Базовый промпт», закреплённый внизу списка (цифра 3). У изменённого промпта рядом с названием появляется бейдж «изменено».

Любую версию можно сравнить с текущей: различия подсвечиваются цветом (цифра 4). Чтобы применить просматриваемую версию, нажмите «**Сделать текущей**» (цифра 5). Так же откатываются к базовому промпту: выберите «Базовый промпт» и нажмите «**Сделать текущей**».



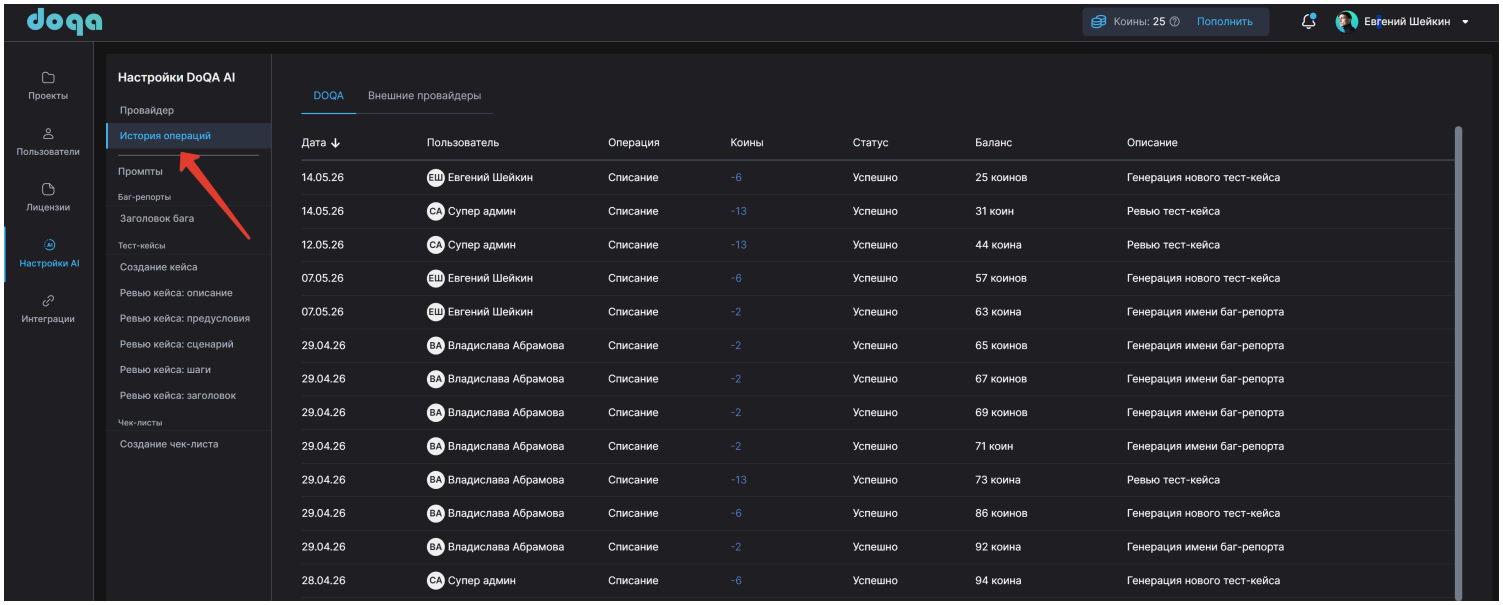
История операций

Раздел показывает, кто и когда запускал AI-функции и во сколько это обошлось. Данные разнесены по двум вкладкам.

Вкладка «DoQA» — расход коинов на встроенном провайдере. В серверной версии эта вкладка не показывается.

Колонка	Что в ней
«Дата»	дата операции, сортируется
«Пользователь»	кто запустил
«Операция»	«Списание» или «Пополнение»
«Коины»	сколько списано или начислено
«Статус»	«Успешно», «Ошибка» или «Прервано»
«Баланс»	баланс после операции
«Описание»	какая функция была вызвана

Статус «Ошибка» означает сбой при генерации, «Прервано» — что на балансе не хватило коинов. Начисление приветственных и подаренных коинов попадает в историю с описанием «Начисление бонусных коинов».



Вкладка «Внешние провайдеры» — расход токенов на своём провайдере. Коины здесь не списываются, поэтому колонок с ними нет: стоимость считайте по этим цифрам у своего LLM-провайдера.

Колонка	Что в ней
«Дата»	дата запроса, сортируется
«Пользователь»	кто запустил
«Провайдер»	через какого провайдера ушёл запрос
«Модель»	какая модель обработала запрос
«Токены (вх. / исх.)»	входные и выходные токены
«Статус»	«Успешно», «Ошибка» или «Отменено»
«Описание»	какая функция была вызвана

Иконка рядом с заголовком «Токены (вх. / исх.)» поясняет, что входящие токены — это объём данных (контекст, промпт, настройки), переданный из DoQA в модель, а исходящие — объём сгенерированного ответа.

Настройка прокси для LLM-провайдеров

Если ваша сеть блокирует прямой доступ к API LLM-провайдера, вы можете поднять обратный прокси nginx на своём VPS/VDS сервере и указать его адрес в поле «API-эндпоинт» настроек DoQA AI. На эту страницу ведёт ссылка «Как настроить прокси» прямо из формы настройки провайдера.

Прокси имеет смысл настраивать для двух провайдеров из списка: «OpenAI» и «Свой провайдер». Только у них есть поле «API-эндпоинт». У «DoQA» и «YandexGPT» адрес фиксированный, и поля для него нет.

«Свой провайдер» принимает любой OpenAI-совместимый API, поэтому через прокси можно вести и его. Anthropic и Google AI отдельными пунктами в списке провайдеров DoQA не представлены. См. **Настройки DoQA AI**.

Требования

- VPS/VDS сервер с публичным IP-адресом (рекомендуется Ubuntu 22.04+)
- Доменное имя с A-записью, указывающей на IP сервера
- Открытые порты 80 и 443 в файрволе
- Root или sudo доступ по SSH

---

1. Установите nginx и certbot

```bash
sudo apt update
sudo apt install -y nginx certbot python3-certbot-nginx
```

---

2. Создайте конфигурацию nginx

Создайте файл `/etc/nginx/sites-available/llm-proxy` со следующим содержимым. Замените `your-domain.example.com` на ваш реальный домен.

nginx

```nginx
server {
    listen 80;
    server_name your-domain.example.com;

    location /.well-known/acme-challenge/ {
        root /var/www/html;
    }

    location / {
        return 301 https://$host$request_uri;
    }
}

server {
    listen 443 ssl;
    server_name your-domain.example.com;

    ssl_protocols       TLSv1.2 TLSv1.3;
    ssl_ciphers         HIGH:!aNULL:!MD5;
    ssl_session_cache   shared:SSL:10m;
    ssl_session_timeout 10m;

    # --- OpenAI ---
    location /openai/ {
        proxy_pass              https://api.openai.com/v1/;
        proxy_ssl_server_name   on;
        proxy_set_header        Host api.openai.com;
        proxy_pass_request_headers on;
        proxy_read_timeout      120s;
        proxy_send_timeout      120s;
    }

    # --- Свой провайдер: подставьте адрес своего OpenAI-совместимого API ---
    location /custom/ {
        proxy_pass              https://api.example-llm.com/v1/;
        proxy_ssl_server_name   on;
        proxy_set_header        Host api.example-llm.com;
        proxy_pass_request_headers on;
        proxy_read_timeout      120s;
        proxy_send_timeout      120s;
```

```
    }
}
```

Блок `location /custom/` нужен, только если вы ведёте через прокси «Свой провайдер». Если используете один OpenAI, удалите его.

---

3. Включите сайт и получите SSL-сертификат

```bash
# Включить сайт
sudo ln -sf /etc/nginx/sites-available/llm-proxy \
            /etc/nginx/sites-enabled/

# Проверить конфигурацию и перезагрузить
sudo nginx -t && sudo systemctl reload nginx

# Получить сертификат Let's Encrypt
sudo certbot --nginx -d your-domain.example.com
```

Certbot автоматически обновит конфигурацию nginx с путями к SSL-сертификату и настроит автопродление.

Альтернатива: самоподписанный сертификат

Если Let's Encrypt недоступен, создайте самоподписанный сертификат:

```bash
sudo openssl req -x509 -nodes -days 365 \
  -newkey rsa:2048 \
  -keyout /etc/ssl/private/llm-proxy.key \
  -out /etc/ssl/certs/llm-proxy.crt \
  -subj "/CN=your-domain.example.com"
```

Затем добавьте эти строки в блок `server` (порт 443):

```nginx
ssl_certificate     /etc/ssl/certs/llm-proxy.crt;
ssl_certificate_key /etc/ssl/private/llm-proxy.key;
```

4. Настройте эндпоинт в DoQA

Откройте «Настройки администратора» → «AI Settings» → блок «Провайдер» и впишите адрес прокси в поле «API-эндпоинт» — см. **Настройки DoQA AI**.

Провайдер в DoQA	Что вписать в «API-эндпоинт»
OpenAI	`https://your-domain.example.com/openai/`
Свой провайдер	`https://your-domain.example.com/custom/`

Если оставить «API-эндпоинт» пустым, DoQA обращается к эндпоинту провайдера по умолчанию. Под полем об этом написано: «Оставьте пустым для эндпоинта по умолчанию или введите URL прокси». Для «Своего провайдера» поле обязательное.

5. Проверка

Проверьте прокси из командной строки (вы должны получить 401: это значит, что прокси работает):

```bash
curl -I https://your-domain.example.com/openai/chat/completions
```

Серверная версия

Содержание

- Требования
- Утилита управления сервером
- Описание параметров среды (.env)
- Настройка сервера почты
- Авторизация через LDAP
- Корпоративный вход (SSO)

Требования

Требования к аппаратному обеспечению

Для стабильной и эффективной работы **DoQA** рекомендуется следующая конфигурация сервера:

Минимальные требования

- **Процессор (CPU)**: 8 ядер
- **Оперативная память (RAM)**: 8 GB
- **Накопитель (SSD)**: 256 GB

Эта конфигурация подходит для небольших команд и ограниченного объема тестирования. Она обеспечивает базовую производительность, достаточную для тестирования и работы с небольшими проектами.

Рекомендуемые требования

- **Процессор (CPU)**: 12 ядер
- **Оперативная память (RAM)**: 32 GB
- **Накопитель (SSD)**: от 512 GB

Эта конфигурация рекомендуется для больших команд и интенсивного использования системы. Она обеспечивает высокую производительность и надежность для работы с большими объемами данных и множества одновременно выполняемых тестов.

Дополнительные рекомендации по аппаратному обеспечению

1. **Резервное копирование**
   - Настройте систему резервного копирования данных базы данных и хранилища MinIO для предотвращения потери данных.
2. **Производительность дисковой подсистемы**

- Использование SSD накопителей значительно увеличивает скорость доступа к данным и общую производительность системы по сравнению с HDD.

Программное обеспечение

Для установки и работы **DoQA** необходимы следующие компоненты программного обеспечения:

- **Docker Engine** 20.10.23 и выше. Эти версии предоставляют необходимые функции и улучшения для стабильной работы **DoQA**. **Инструкция по установке Docker** ↗
- **Docker Compose** v2.15.1 и выше. Эти версии включают поддержку необходимых функций для работы **DoQA**. **Инструкция по установке Docker Compose** ↗

Продакшн

В продакшене используйте DoQA только с SSL-сертификатом. Без SSL часть функций DoQA (например, веб-сокеты) будет недоступна, что негативно влияет на стабильность и безопасность.

Домен

Для публикации DoQA в интернете настоятельно рекомендуется использовать настоящий домен, указывающий A-записью на ваш сервер. Это упростит выпуск сертификата (команда `cert`) и доступ пользователей. Локальные имена `localhost` пригодны только для теста.

Внимание

Все действия по установке и настройке **DoQA** необходимо производить в консольном приложении. Пользовательский интерфейс настольных приложений или веб-интерфейс Docker в процессе установки не используется.

Дополнительные требования к программному обеспечению

1. **Интернет соединение**
   - Сервер должен иметь выход в сеть Internet для получения Docker образов с публичных репозиториев.

2. **Права доступа**

- Необходимы административные права доступа на сервере для установки и настройки Docker и Docker Compose.

3. **Среда выполнения**

- Операционная система Linux (рекомендуется) или Windows.

Утилита управления сервером

Утилита управления сервером DoQA — интерактивная CLI для управления серверным приложением. Ниже подробно описана её работа.

Дистрибутив

Скачайте последнюю версию

- Версия для Linux ↗
- Версия для Windows ↗

Вы также можете скачать дистрибутив из командной строки при наличии одной из утилит командной строки операционной системы:

Linux

curl wget

bash

```
curl -O https://doqa.app/downloads/doqa && chmod +x doqa
```

Windows

curl PowerShell

bash

```
curl -O https://doqa.app/downloads/doqa.exe
```

Интерактивный режим

Запустите утилиту DoQA из командной строки и следуйте интерактивным подсказкам.

Linux

```bash
./doqa
```

Быстрые команды

- Установка (install)
- Запуск приложения (start)
- Остановка приложения (stop)
- Обновление (update)
- Создать резервную копию данных приложения (backup)
- Восстановить данные приложения из резервной копии (restore)
- Выпуск/обновление SSL-сертификата (cert)
- Смена домена (domain)

Логирование

Утилита собирает логи в файл `doqa-cli.log`. Если возникли проблемы с установкой или запуском, пожалуйста, пришлите этот файл в техническую поддержку.

Установка (install)

Команда `install` устанавливает приложение DoQA.

Совет

Всегда используйте актуальную версию утилиты DoQA CLI. Скачайте свежую версию на странице **Утилита управления сервером**.

Linux

bash

```
./doqa install
```

Флаги

- `--version <x.y.z>` — указать версию конфигураций для установки. По умолчанию используется последняя версия.

Перед началом

- Убедитесь, что выполнены **требования** и установлены Docker и Docker Compose.
- Если планируете HTTPS/сертификаты, проверьте, что порты 80 и 443 свободны и доступны извне.
- Если используете доменное имя (не `localhost`), настройте DNS, чтобы домен указывал на ваш сервер.

Шаги установки (интерактивно)

После запуска команда задаст вопросы:

1. «Установить с базой данных?»:

- Да: будет развёрнут PostgreSQL в составе DoQA, пароль для БД сгенерируется автоматически.
- Нет: укажите параметры существующей БД (хост, порт, имя БД, пользователь, пароль).

2. «Использовать SSL сертификат?»:
   - Да: будет включена SSL-конфигурация Nginx. Сертификаты можно подключить вручную или выпустить позже командой `cert`.
   - Нет: будет использована HTTP-конфигурация.
3. «Введите домен для веб-приложения» — домен без протокола (например: `example.com` или `localhost`).

Рекомендация

Если нет острой необходимости использовать уже существующий сервер PostgreSQL, рекомендуется выбрать PostgreSQL, входящий в состав DoQA. Это позволит пользоваться встроенными функциями резервного копирования (бекапа) — командой `backup` — и упростит обслуживание.

Пример ответов:

```bash
Вы хотите установить связь с базой данных: y
Использовать SSL сертификат?: y
Введите домен для веб-приложения: example.doqabox.ru
```

Важно

Домен указывается без протокола. Для корректной работы рекомендуется HTTPS.

Продакшн

В продакшене используйте DoQA только с SSL-сертификатом. Без SSL часть функций DoQA будет недоступна (например, веб-сокеты), что повлияет на стабильность и удобство работы.

Что делает команда

- Создаёт служебные каталоги

- Загружает набор конфигураций и распаковывает их.
- Копирует `configs/.env.install` в `.env` и проставляет значения, генерирует секретные ключи.
- Подготавливает Nginx и Docker Compose файлы.
- Выполняет `docker compose build`.
- Запускает миграции БД.

Что дальше

- Для HTTPS используйте один из вариантов:
  - Выпустить сертификат автоматически: команда `cert`.
  - Подключить собственные сертификаты: поместите `fullchain.pem` и `privkey.pem` в `box/nginx/ssl/`.
- Настройте почту для приглашения пользователей. См. **Настройки почты**.
- Запустите приложение:

```bash
./doqa start
```

- Откройте в браузере URL, соответствующий `APP_URL` (например: `https://doqa.app/`), завершите первичную настройку и создайте супер-пользователя.
- Если меняли домен, выполните `domain`, затем перевыпустите сертификаты через `cert`.

Запуск приложения (start)

Команда `start` запускает серверное приложение DoQA. Её же выполняют после **остановки**, **обновления** и **восстановления из копии**.

Linux

bash

```
./doqa start
```

Что делает команда

1. Проверяет совместимость `docker-compose.yml` с версией утилиты.
2. Применяет миграции базы данных (`php artisan migrate --force`).
3. Поднимает контейнеры в фоновом режиме (`docker compose up -d --remove-orphans`).

Помимо запуска контейнеров, `start` меняет схему базы данных. Если миграции завершатся ошибкой, контейнеры не запустятся. В этом случае пришлите файл `doqa-cli.log` в техническую поддержку.

Перед запуском

- Установка должна быть завершена.
- Если меняли домен, выполните `domain`, затем выпустите сертификаты через `cert`.

Продакшн

В продакшене используйте DoQA только с SSL-сертификатом. Без SSL часть функций DoQA (например, веб-сокеты) будет недоступна, что повлияет на стабильность и удобство работы.

Запуск и проверка

1. Запустите приложение:

```bash
./doqa start
```

2. Проверьте статус контейнеров:

```bash
docker compose ps
```

3. Посмотрите логи при необходимости:

```bash
docker compose logs -f
```

Доступ к приложению

Откройте в браузере `https://<APP_URL>/`.

При первом запуске введите лицензионный ключ и создайте первого пользователя: он станет суперадмином системы и **владельцем системы**.

> **Важно**
>
> Без настройки почтового сервера вы не сможете приглашать новых пользователей. Настройте параметры почты в `.env`. См. **Настройка сервера почты**.

Полезные команды

- Остановить сервисы перед обслуживанием:

```bash
./doqa stop
```

- Перезапуск (после изменений в `.env`):

```bash
./doqa stop && ./doqa start
```

- Выпустить или обновить сертификаты для домена:

```bash
./doqa cert
```

Остановка приложения (stop)

Команда `stop` останавливает приложение DoQA. Обязательно останавливайте приложение при сервисных работах.

Использование

Linux

bash

```
./doqa stop
```

Описание

Команда выполняет `docker compose down --remove-orphans`: останавливает и удаляет контейнеры DoQA, в том числе оставшиеся от предыдущих версий конфигурации.

Данные при этом не удаляются: база данных и файлы лежат в каталогах `./postgresql` и `./minio` рядом с `docker-compose.yml`. Чтобы поднять приложение обратно, выполните `start`.

Обновление (update)

Команда `update` обновляет DoQA до актуальной версии.

Совет

Всегда используйте актуальную версию утилиты DoQA CLI. Ссылки и инструкции — на странице **Утилита управления сервером**.

Linux

bash

```
./doqa update
```

Обновление останавливает приложение

Последним шагом `update` гасит все контейнеры и завершается сообщением «Приложение успешно обновлено». После этого DoQA **не работает**, пока вы не выполните `./doqa start` вручную.

Флаги

- `--version <x_y_z>` — обновиться до конкретной версии конфигураций, например `--version 3_0_0`. По умолчанию выбирается последняя версия, доступная по вашей лицензии.
- `--skip-check` — пропустить проверку лицензии. Используйте только по указанию технической поддержки.
- `--bundle <каталог>` — обновиться из оффлайн-комплекта, собранного командой `doqa bundle`, без обращения к серверам загрузки. Версию задаёт манифест комплекта, поэтому `--version` с ним указывать не нужно.

Перед началом

- DoQA должен работать на **встроенной** PostgreSQL. Если в `.env` параметр `DB_HOST` указывает на внешнюю базу, обновление не начнётся: утилита ответит «Обновление

недоступно: DoQA подключён к внешней базе данных» и завершится. Поставляемый `docker-compose.yml` рассчитан только на встроенную БД; за переносом данных обратитесь в поддержку `support@doqa.app`.

- Приложение должно быть **запущено**: утилита обращается к его API за данными лицензии. Если ключевые контейнеры остановлены, она поднимет их сама.
- Лицензионный ключ должен быть введён в интерфейсе DoQA, иначе обновление прервётся сообщением «Введите лицензию в пользовательский интерфейс DoQa». См. **Лицензии (серверная версия)**.
- Выполняйте обновление в период минимальной нагрузки: приложение будет недоступно.
- Проверьте доступ в интернет и свободное место на диске.
- Перед обновлением утилита автоматически создаёт резервную копию базы данных и файлового хранилища.

> **Важно**
>
> Если были изменены базовые настройки DoQA, используется не поставляемое в комплекте ПО для хранения данных или ваш сервер не соответствует требованиям, резервная копия автоматически создана не будет. Сделайте её вручную. Способ зависит от вашей конфигурации.

Как проходит обновление

1. Проверяется лицензия и определяется целевая версия.
2. Создаётся резервная копия данных. Если она не удалась, обновление прерывается.
3. Приложение останавливается — до подмены `docker-compose.yml`, пока имена сервисов ещё совпадают с именами запущенных контейнеров.
4. Загружается и распаковывается архив конфигураций.
5. Обновляются `docker-compose.yml`, конфигурация Nginx и переменные образов в `.env`.
6. Пересобираются образы (`docker compose build`).
7. Применяются миграции базы данных.
8. Приложение останавливается снова. На этом команда завершается.

Обычно процесс занимает несколько минут.

После обновления

1. Запустите приложение. Без этого шага сайт не откроется:

```bash
./doqa start
```

2. Откройте в браузере адрес `https://<APP_URL>/` и проверьте, что система загружается корректно.
3. Если вы меняли домен, перевыпустите сертификаты: сначала `domain`, затем `cert`.
4. При необходимости посмотрите состояние контейнеров и логи:

```bash
docker compose ps

docker compose logs -f
```

Переход на новую систему прав

Если обновление приводит DoQA к версии с новой системой прав, при применении миграций права переносятся автоматически. Резервная копия, которую утилита создаёт перед обновлением, позволяет вернуться назад. Что переносится, как проверить результат и как откатиться: **Переход на новую систему прав**.

Обновление до конкретной версии

Версия указывается через подчёркивания так же, как называется архив конфигураций:

```bash
./doqa update --version 3_0_0
```

Ограничения:

- Если версия вышла после окончания срока поддержки по вашей лицензии, обновление прервётся сообщением «Обновление невозможно, так как версия X вышла после окончания срока действия лицензии ДД.ММ.ГГГГ». Ближайшую доступную версию утилита не подсказывает. Запустите `./doqa update` без `--version`, тогда она сама выберет последнюю разрешённую лицензией.
- Понижение версии запрещено: команда ответит «Обновление на понижение версии недоступно». Чтобы установить более раннюю версию, выполните установку заново.

Резервное копирование (backup)

Команда `backup` создаёт резервную копию базы данных PostgreSQL и загруженных файлов из хранилища MinIO.

Использование

Linux

bash

```
./doqa backup
```

Описание

Создаётся один архив `backup/backup_YYYYMMDD_HHMMSS.zip` в каталоге установки. Внутри него:

- `db-dumps/postgresql.sql` — дамп базы данных;
- `files.zip` — файлы из бакета MinIO (вложения, изображения).

Останавливать приложение для резервного копирования не нужно.

Резервная копия создаётся автоматически и перед **обновлением**.

Только встроенная PostgreSQL

Команда работает с базой данных, поставляемой в составе DoQA. Если при установке вы подключили внешний сервер PostgreSQL, дамп базы создан не будет. Настройте резервное копирование средствами своего сервера БД.

Восстановить данные из архива можно командой `restore`.

Восстановление из резервной копии (restore)

Команда `restore` восстанавливает базу данных PostgreSQL и файлы MinIO из архива, созданного командой `backup`.

Текущие данные будут удалены

Команда останавливает приложение и **полностью удаляет** текущие каталоги данных `./postgresql` и `./minio`, заменяя их содержимым архива. Дополнительных подтверждений она не спрашивает.

Всё, чего нет в архиве, восстановить будет нельзя. Перед восстановлением сделайте свежую копию текущего состояния: `./doqa backup`.

Использование

Файл резервной копии должен лежать в папке `backup` каталога установки: путь к файлу отсчитывается от неё. Если папки нет, создайте её и положите файл туда.

Linux

bash

```
./doqa restore
```

Утилита запросит имя файла. Имя можно передать и аргументом. Тогда вопрос не задаётся, что нужно для скриптов:

bash

```
./doqa restore backup_20260101_120000.zip
```

Что делает команда

1. Останавливает приложение (`docker compose down`).
2. Распаковывает архив во временную папку.
3. Удаляет каталог `./postgresql` и разворачивает дамп базы данных из архива.

4. Удаляет каталог `./minio` и разворачивает файлы из архива.
5. Удаляет временную папку.

После восстановления

Приложение остаётся остановленным. Запустите его:

```bash
./doqa start
```

Смена домена (domain)

Команда `domain` изменяет домен приложения в `.env`. Она обновляет значения `APP_URL`, `APP_FRONTEND_URL` и `MINIO_BUCKET_URL`.

Использование

Linux

bash

```
./doqa domain
```

Флаги

- `--set <домен>` — указать новый домен без вопросов (например: `example.com`).
- `--yes` — не задавать вопросов (требуется совместно с `--set`).

Описание

При запуске без параметров команда предложит ввести новый домен интерактивно. Домен указывается без протокола. Команда:

1. Определит, включен ли SSL (`USE_SSL=true`), чтобы выбрать `http` или `https`.
2. Сформирует `MINIO_BUCKET_URL` как `<protocol>://<домен>/doqa`.
3. Обновит в `.env` значения `APP_URL`, `APP_FRONTEND_URL` и `MINIO_BUCKET_URL`.

После смены домена необходимо установить корректные SSL-сертификаты для нового домена:

- Если у вас есть собственные сертификаты, поместите их в `box/nginx/ssl` с именами `fullchain.pem` и `privkey.pem`, затем перезапустите приложение.
- Либо автоматически выпустите бесплатные сертификаты Let's Encrypt командой `cert`.

Совет: сначала выполните `domain`, затем `cert`.

Примеры

- Сменить домен интерактивно:

Linux

```bash
./doqa domain
```

- Установить домен без вопросов:

```bash
./doqa domain --set example.com --yes
```

Выпуск/обновление SSL-сертификата (cert)

Команда `cert` выпускает или обновляет сертификат Let's Encrypt для домена из файла `.env` (`APP_URL`) и при необходимости автоматически включает SSL-конфигурацию Nginx.

Использование

Linux

bash

```
./doqa cert
```

Требования

- DoQA должна быть установлена.
- В `.env` должен быть указан домен без протокола: `APP_URL=example.com`. Домен устанавливается командой `doqa domain --set={ваш_домен_без_протокола}`.
- Домен должен указывать на текущий сервер (актуальная запись DNS A).
- Порт 80 (и 443) должен быть доступен извне. Во время выпуска команда временно остановит сервисы DoQA, чтобы освободить порты 80/443 для `certbot`.

Что делает команда

1. Останавливает сервисы DoQA, запускает `certbot` в контейнере Docker (standalone, с пробросом портов 80/443) для выпуска/обновления сертификата.
2. Копирует выданные файлы в `box/nginx/ssl`: `fullchain.pem` и `privkey.pem`.
3. Обновляет маркер времени последнего обновления в `box/letsencrypt/.last_renew`.
4. Запускает сервисы DoQA.

Флаги

- `--yes` — подтвердить без вопросов и использовать значения из `.env`.
- `--auto` — авто-режим (удобно для cron): обновлять только если прошло 75 дней с момента последнего успешного выпуска.
- `--schedule` — создать задачу автообновления (Linux cron). Добавляет задание, которое периодически запускает `cert --yes --auto`.
- `--unschedule` — удалить задачу автообновления (Linux cron). На других ОС настройте расписание вручную.

Примеры

- Первый выпуск сертификата (интерактивно):

Linux

```bash
./doqa cert
```

- Обновление без вопросов:

```bash
./doqa cert --yes
```

- Фоновый запуск в cron (Linux): создаст расписание автообновления.

```bash
./doqa cert --schedule
```

- Ручная проверка в крон-режиме (выполнит только если >= 75 дней):

```bash
./doqa cert --yes --auto
```

- Удалить автообновление (Linux):

```bash
./doqa cert --unschedule
```

Замечания

- Если используете внешний реверс-прокси, убедитесь, что порт 80 доступен для HTTP-01 проверки Let's Encrypt на этом сервере, либо выпустите/пробросьте сертификат на уровень Nginx DoQA самостоятельно.
- Если вы поменяли домен командой `domain`, после этого примените новые сертификаты вручную или выпустите их заново командой `cert`.

Описание параметров среды (.env)

Файл `.env` в каталоге установки задаёт настройки всех сервисов DoQA. Он создаётся командой `install` из поставляемого шаблона.

Значения `.env` попадают в контейнеры при их создании, поэтому правка применяется только после перезапуска приложения: `./doqa stop && ./doqa start`.

Почта (обязательно к изменению)

`MAIL_MAILER`, `MAIL_HOST`, `MAIL_PORT`, `MAIL_ENCRYPTION`, `MAIL_USERNAME`, `MAIL_PASSWORD`, `MAIL_FROM_ADDRESS`, `MAIL_FROM_NAME`.

Без настроенной почты нельзя приглашать пользователей и восстанавливать пароли. Подробно: Настройка сервера почты.

Домен и SSL (обязательно для продакшна)

Параметр	Описание и пример
`APP_URL`	Домен приложения без протокола. Пример: `example.com`. Менять: командой `domain`.
`APP_FRONTEND_URL`	Домен фронтенда (обычно равен `APP_URL`). Пример: `example.com`. Менять: командой `domain`.
`USE_SSL`	Включить HTTPS. Пример: `true` / `false`. Менять: командой `cert` или вручную при подключении своих сертификатов.
`MINIO_BUCKET_URL`	Публичный URL бакета. Пример: `https://example.com/doqa`. Менять: командами `domain`, `cert`.

Продакшн

Используйте реальный домен с A-записью на сервер и `USE_SSL=true`. Без SSL часть функций (веб-сокеты) недоступна.

Аутентификация

Параметр	Описание
`LDAP_ENABLED`	Вход по учётным данным из LDAP-каталога. Подробно: **Авторизация через LDAP**.
`SSO_ENABLED`	Корпоративный вход через провайдера OIDC. Подробно: **Корпоративный вход (SSO)**.

Ни один параметр `LDAP_*` и `SSO_*` в шаблон `.env` не входит, включая перечисленные выше `LDAP_ENABLED` и `SSO_ENABLED`. Все нужные строки добавляют в файл вручную.

Ключи и безопасность (менять не рекомендуется)

Значения генерируются при установке. Смена ключей разлогинит пользователей.

Параметр	Описание
`APP_KEY`	Ключ приложения. Пример: `base64:...`.
`JWT_SECRET`	Секрет для подписи токенов доступа.
`MINIO_KEY`	Ключ доступа к файловому хранилищу MinIO.
`MINIO_SECRET`	Секрет доступа к MinIO.
`PUSHER_APP_SECRET`	Секрет сервиса веб-сокетов.
`STATISTIC_API_KEY`	Ключ сервиса статистики.
`NOTIFICATION_API_KEY`	Ключ сервиса уведомлений.
`LLM_API_KEY`	Ключ сервиса ИИ-функций.

Параметр	Описание
`RABBITMQ_PASSWORD`	Пароль брокера сообщений.
`RABBITMQ_ERLANG_COOKIE`	Cookie кластера RabbitMQ.

База данных (менять для внешней БД)

Параметр	Описание и пример
`DB_HOST`	Хост БД. Пример: `postgres` (встроенная PostgreSQL) или `db.example.com`.
`DB_PORT`	Порт БД. Пример: `5432`.
`DB_DATABASE`	Имя БД. Пример: `doqa`.
`DB_USERNAME`	Пользователь БД. Пример: `root`.
`DB_PASSWORD`	Пароль БД.

Внешняя БД и обслуживание

Команды `backup` и `restore` работают только со встроенной PostgreSQL. При подключении к внешней БД настраивайте резервное копирование средствами своего сервера БД.

Redis (менять для внешнего Redis)

Параметр	Описание и пример
`REDIS_HOST`	Хост Redis. Пример: `redis`.
`REDIS_PASSWORD`	Пароль Redis. Пример: `null` или `s3cret`.
`REDIS_PORT`	Порт Redis. Пример: `6379`.

Параметр	Описание и пример
`REDIS_DB`	Индекс БД Redis для приложения. Пример: `2`. Базы `0` и `1` заняты брокером и результатами задач. Не занимайте их.
`REDIS_CACHE_DB`	Индекс БД Redis под кэш. Пример: `3`.

RabbitMQ (менять для внешнего брокера)

Брокер сообщений используется конвейером **автотестов**: через него результаты запусков попадают из парсера в приложение.

Параметр	Описание и пример
`RABBITMQ_HOST`	Хост брокера. Пример: `rabbitmq`.
`RABBITMQ_PORT`	Порт брокера. Пример: `5672`.
`RABBITMQ_USER`	Пользователь. Пример: `admin`.
`RABBITMQ_VIRTUAL_HOST`	Виртуальный хост. Пример: `/`.

Очереди

Параметр	Описание и пример
`QUEUE_WORKERS`	Число параллельных обработчиков фоновых задач. Пример: `3`. Через очереди уходят письма, импорт и экспорт. Увеличьте, если фоновые задачи копятся.

Веб-сокеты

Обновляют интерфейс без перезагрузки страницы. Требуют `USE_SSL=true`.

Параметр	Описание и пример
`BROADCAST_DRIVER`	Драйвер трансляции событий. Пример: `pusher`.
`PUSHER_APP_HOST`	Хост сервиса веб-сокетов. Пример: `websocket`.
`PUSHER_APP_ID`	Идентификатор приложения. Пример: `web-socket`.
`PUSHER_APP_KEY`	Ключ приложения. Пример: `web-socket`.
`PUSHER_APP_PORT`	Порт. Пример: `6001`. Менять: только при конфликте портов.
`PUSHER_SCHEME`	Протокол. Пример: `http`.

Порты

Параметр	Описание и пример
`SERVER_PORT`	HTTP порт снаружи. Пример: `80`. Менять: только при конфликте.
`SERVER_SSL_PORT`	HTTPS порт снаружи. Пример: `443`. Менять: только при конфликте.

Сервисы

Параметр	Описание и пример
`MINIO_ENDPOINT`	Внутренний адрес MinIO. Пример: `http://minio:9000`.
`MINIO_BUCKET`	Имя бакета. Пример: `doqa`.
`MINIO_REGION`	Регион MinIO. Пример: `ru-1`.
`STATISTIC_ENDPOINT`	Внутренний адрес сервиса статистики. Пример: `http://statistic:3000/`.
`NOTIFICATION_ENDPOINT`	Внутренний адрес сервиса уведомлений. Пример: `http://notification:3000/`.

Параметр	Описание и пример
`LLM_ENDPOINT`	Внутренний адрес сервиса ИИ-функций. Пример: `http://llm:3000/`. Подробно: **Настройки DoQA AI**.

Telegram-бот уведомлений

Заполняется, если пользователи получают **уведомления** в Telegram. Пока эти параметры не заданы, блок подключения бота в профиле пользователя не показывается, а переключатели в колонке «Telegram» остаются неактивными. Остальные каналы («E-mail», «DoQA») от этого не зависят.

Параметр	Описание
`BOT_TOKEN`	Токен бота, выданный BotFather.
`BOT_NAME`	Имя бота без символа `@`. Подставляется в ссылку на чат с ботом в профиле пользователя.

Оплата и лицензии

В серверной версии оплаты нет: число мест задаёт лицензионный ключ, а параметры `BILLING_*` не используются и в `.env` не нужны. Единственный параметр, который относится к лицензиям:

Параметр	Описание и пример
`ISSUE_MAIL`	Адрес, на который уходит заявка на продление из раздела «Лицензии». По умолчанию адрес поддержки DoQA. Для отправки письма нужна настроенная **почта**.

Параметры ниже нужны только в облачной версии и описаны для тех, кто разворачивает облачное окружение. Значения секретов в документации не приводятся.

Параметр	Где	Описание
`BILLING_API_URL`	продукт	Адрес API учётной системы. По умолчанию берётся `PERSONAL_CABINET_BACKEND_URL`
`BILLING_API_TIMEOUT`	продукт	Время ожидания ответа учётной системы, секунды. По умолчанию 10
`BILLING_RETURN_BASE_URL`	продукт	Адрес, на который платёжный сервис возвращает пользователя. По умолчанию домен организации
`PRODUCT_BILLING_EVENTS_URL`	учётная система	Адрес приёма событий в продукте. По умолчанию строится из основного домена
`BILLING_OFFER_VERSION`, `BILLING_OFFER_URL`	учётная система	Версия и ссылка оферты, которые показываются при оплате. По умолчанию действующая версия и ссылка на лицензионное соглашение DoQA Cloud
`BILLING_SELF_SERVICE_MAX_SEATS`	учётная система	Сколько мест можно купить без заявки в отдел продаж. По умолчанию 99
`BILLING_SALES_EMAIL`, `BILLING_SALES_URL`	учётная система	Контакт отдела продаж для заявок на большие объёмы

Отдельных ключей у оплаты нет. Продукт и учётная система защищают обмен теми же секретами, что и вход по ссылке из личного кабинета: `INTERNAL_JWT_TOKEN_SECRET` у продукта и `INTERNAL_JWT_SECRET` у учётной системы должны совпадать, как и `INTERNAL_SYSTEM_KEY` у продукта, учётной системы и консоли управления.

Docker-образы (служебные)

Значения обновляются автоматически при установке и обновлении. Редактировать вручную не требуется.

- `IMAGE_DOQA_API` — бэкенд, например `registry.control.doqa.app/doqa/doqa-backend:<версия>-box`.
- `IMAGE_DOQA_FRONTEND` — интерфейс.

- `IMAGE_DOQA_AUTOTEST_PARSER` — парсер автотестов.
- `IMAGE_DOQA_AUTOTEST_RESULT_PARSER` — парсер результатов автотестов.
- `IMAGE_DOQA_STATISTIC` — сервис статистики.
- `IMAGE_DOQA_NOTIFICATION` — сервис уведомлений.
- `IMAGE_DOQA_TG_BOT` — Telegram-бот.
- `IMAGE_DOQA_LLM` — сервис ИИ-функций.

Важно

- В продакшене используйте `USE_SSL=true` и реальный домен в `APP_URL`, указывающий A-записью на сервер.
- Параметры домена и SSL меняйте командами `domain` и `cert`, а не правкой файла.

Настройка сервера почты

Без настроенной почты нельзя приглашать пользователей и восстанавливать пароли. Параметры задаются в файле `.env` в каталоге установки.

Параметры

Параметр	Пример	Описание
`MAIL_MAILER`	`smtp`	Способ отправки. Обычный SMTP — `smtp`. Для сервера с самоподписанным сертификатом — `smtp_no_encryption`, см. ниже.
`MAIL_HOST`	`smtp.company.ru`	Хост почтового сервера.
`MAIL_PORT`	`587`	Порт. Обычно `587` для STARTTLS и `465` для SMTPS.
`MAIL_ENCRYPTION`	`tls`	Шифрование: `tls` для порта 587, `ssl` для порта 465, `null` — без шифрования.
`MAIL_USERNAME`	`no-reply@company.ru`	Пользователь SMTP.
`MAIL_PASSWORD`	`s3cret`	Пароль SMTP.
`MAIL_FROM_ADDRESS`	`no-reply@company.ru`	Адрес в поле «От кого».
`MAIL_FROM_NAME`	`DoQA`	Имя в поле «От кого».

`MAIL_PORT` и `MAIL_ENCRYPTION` задаются независимо друг от друга: порт `465` без `MAIL_ENCRYPTION=ssl` работать не будет.

Перезапустите приложение

Значения `.env` попадают в контейнеры при их создании. После правки почтовых параметров выполните `stop` и `start`, иначе приложение продолжит работать со старыми настройками:

```bash
./doqa stop && ./doqa start
```

SMTP без проверки сертификата

Для внутреннего почтового сервера с самоподписанным сертификатом переключите отправку на mailer, который сертификат не проверяет:

```
MAIL_MAILER=smtp_no_encryption
MAIL_VERIFY_PEER=false
MAIL_VERIFY_PEER_NAME=false
```

Параметр	Описание
`MAIL_VERIFY_PEER`	Проверять сертификат сервера. По умолчанию `false`.
`MAIL_VERIFY_PEER_NAME`	Проверять соответствие имени в сертификате. По умолчанию `false`.
`MAIL_AUTH_MODE`	Режим аутентификации SMTP: `plain`, `login`, `cram-md5`. По умолчанию не задан. Режим выбирается автоматически.

Остальные параметры (`MAIL_HOST`, `MAIL_PORT`, `MAIL_USERNAME`, `MAIL_PASSWORD`, `MAIL_ENCRYPTION`) этот mailer читает те же самые.

Проверка

1. Перезапустите приложение после правки `.env`.
2. Пригласите пользователя в настройках администратора и проверьте, что письмо пришло.
3. Если письма нет, посмотрите логи приложения, ошибки SMTP видны там:

```bash
docker compose logs -f api
```

Письма отправляются фоновым обработчиком очередей, а не в момент нажатия кнопки. Если контейнер очередей не поднят, письмо не уйдёт и ошибки в логах приложения не будет. Проверьте его состояние:

```bash
docker compose ps queue
```

Авторизация через LDAP

DoQA умеет проверять пароли пользователей в LDAP-каталоге, в том числе в Active Directory. Когда интеграция включена, локальные пароли DoQA не проверяются: все пользователи входят по учётным данным из каталога.

Параметры `LDAP_*` не входят в поставляемый шаблон `.env`. Добавьте нужные строки вручную. После правки перезапустите приложение: `./doqa stop && ./doqa start`.

Кто сможет войти

Зависит от того, включено ли автоматическое создание пользователей (JIT):

- **JIT выключен** (по умолчанию) — войти смогут только те, кто уже заведён в DoQA и завершил регистрацию. Email пользователя в DoQA должен совпадать с атрибутом `mail` его записи в каталоге. Пользователи, которых нет в каталоге, войти не смогут.
- **JIT включён** — заводить пользователей заранее не нужно, они создаются при первом входе. См. **Автоматическое создание пользователей**.

Изменения обратимы: чтобы вернуть обычный вход по паролю DoQA, выключите `LDAP_ENABLED` и перезапустите приложение.

Настройка

1. Добавьте в `.env` параметры подключения:

```
LDAP_ENABLED=true
LDAP_HOST=ldap.company.ru
LDAP_PORT=389
LDAP_USERNAME="cn=admin,dc=doqa,dc=app"
LDAP_PASSWORD="password"
LDAP_BASE_DN="dc=doqa,dc=app"
LDAP_ENCRYPTION=false
```

Параметр	Описание
`LDAP_ENABLED`	Включает проверку паролей в LDAP. По умолчанию `false`.
`LDAP_HOST`	Хост или IP-адрес LDAP-сервера.
`LDAP_PORT`	Порт сервера. Обычно `389`, для LDAPS — `636`.
`LDAP_USERNAME`	DN технической учётной записи. Ей нужны права на подключение, поиск записей и чтение атрибутов, перечисленных ниже.
`LDAP_PASSWORD`	Пароль технической учётной записи.
`LDAP_BASE_DN`	DN, внутри которого идёт поиск пользователей.
`LDAP_ENCRYPTION`	`false` — без шифрования (по умолчанию), `ssl` — подключение по LDAPS. Любое другое значение равносильно `false`.
`LDAP_TLS_CACERTFILE`	Путь к сертификату CA внутри контейнера. Нужен при `LDAP_ENCRYPTION=ssl`, если сертификат сервера выпущен корпоративным центром сертификации.
`LDAP_TLS_CERTFILE`	Путь к клиентскому сертификату внутри контейнера, если сервер его требует.

Шифрование не включается по умолчанию

LDAPS включается только при точном значении `LDAP_ENCRYPTION=ssl`. Пустое значение, `false` или опечатка означают подключение **без шифрования**. Пароли пойдут по сети открытым текстом, и никакой ошибки при этом не будет. Проверьте значение явно.

Файлы сертификатов должны быть видны изнутри контейнера приложения: положите их в каталог, который уже смонтирован в контейнер, и укажите путь так, как его видит контейнер — например, `/var/www/public/company-ca.crt`.

2. Убедитесь, что у учётных записей в каталоге заполнен атрибут `mail`: по нему DoQA находит пользователя, и его значение должно совпадать с email пользователя в DoQA.

3. На странице авторизации DoQA введите email и пароль из LDAP.

Маппинг атрибутов

Если в вашем каталоге атрибуты называются иначе, переопределите их. Все параметры необязательные, имена атрибутов указывайте строчными буквами.

Параметр	По умолчанию	Описание
`LDAP_ATTR_EMAIL`	`mail`	Атрибут с адресом почты. По нему же выполняется поиск пользователя в каталоге.
`LDAP_ATTR_FIRST_NAME`	`givenname`	Имя.
`LDAP_ATTR_LAST_NAME`	`sn`	Фамилия.
`LDAP_ATTR_UID`	пусто	Неизменяемый идентификатор записи: `entryuuid` в OpenLDAP, `objectguid` в Active Directory. Если не задан, вместо него используется DN. Тогда переименование или перенос записи в каталоге разорвёт связь с пользователем DoQA.

Автоматическое создание пользователей (JIT)

Пользователь, которого в DoQA ещё нет, может создаваться при первом входе. Управляют этим те же параметры, что и для **корпоративного входа (SSO)**. Отдельного флага для LDAP нет:

```
SSO_JIT_ENABLED=true
SSO_JIT_MODE=domain_allowlist
SSO_JIT_ALLOWED_DOMAINS=company.ru
```

Пользователь создаётся, только если пароль подтверждён LDAP-сервером **и** домен email проходит проверку `SSO_JIT_MODE`. Новый пользователь попадает в группы по умолчанию и в группы по правилам (в том числе по атрибутам LDAP), не занимает место в лицензии и не имеет локального пароля; остальные права выдаёт администратор внутри DoQA. Группу

можно связать с группой каталога по её DN: **Группы**. Подробно - в разделе **Автоматическое создание пользователей (JIT)**.

Диагностика

Все неудачные попытки входа возвращают пользователю одну и ту же ошибку авторизации: не удалось подключиться к серверу, не найдена запись, не подошёл пароль — сообщение одинаковое. Причину смотрите в логах приложения:

```bash
docker compose logs -f api
```

Корпоративный вход (SSO)

DoQA поддерживает вход через корпоративный провайдер идентификации по протоколу OpenID Connect (OIDC). Настраивается один провайдер на инсталляцию: Keycloak, AD FS, Azure AD (Entra), Blitz, GitLab self-managed — любой OIDC-совместимый.

Параметры `SSO_*` не входят в поставляемый шаблон `.env`. Добавьте нужные строки в `.env` вручную. После правки `.env` перезапустите приложение: `./doqa stop && ./doqa start`.

Регистрация клиента в провайдере

Перед правкой `.env` заведите DoQA как OIDC-клиент на стороне провайдера. Что нужно с обеих сторон:

- DoQA доступна по HTTPS на постоянном адресе.
- Провайдер доступен по сети из контейнера приложения: по его адресу идут чтение конфигурации, обмен кода на токен и загрузка ключей подписи.
- Провайдер отдаёт конфигурацию по `{issuer}/.well-known/openid-configuration` и подписывает ID-токен алгоритмом RS256. Токен, подписанный другим алгоритмом, DoQA отклоняет.

Клиент должен быть confidential (с секретом) и работать по authorization code flow. Разрешите scope `openid email profile` и укажите адрес возврата:

```
https://<APP_URL>/api/auth/sso/callback
```

Адрес возврата сверяется символ в символ: протокол, домен, порт, путь и завершающий слеш должны совпадать со значением `SSO_REDIRECT`, иначе провайдер откажет во входе.

Адрес провайдера должен быть одним и тем же для браузера и для контейнера приложения. Частая ошибка в Docker: в `SSO_ISSUER` записан внутренний адрес вроде `http://keycloak:8089`, к которому у браузера пользователя нет доступа.

Памятки по распространённым провайдерам:

- **Keycloak** — Clients → Create → OpenID Connect; «Client authentication» = On, секрет на вкладке «Credentials»; в «Valid redirect URIs» — адрес возврата; issuer — `https://<keycloak>/realms/<realm>`.
- **AD FS, Azure AD (Entra ID)** — App registration; Redirect URI типа Web — адрес возврата; выпустите секрет клиента; включите выдачу ID-токена; настройте выдачу claims `email` и `name`.
- **GitLab self-managed** — Admin/Group/User → Applications; Redirect URI — адрес возврата; scopes `openid email profile`; issuer — базовый адрес GitLab, например `https://gitlab.company.ru`.

Включение

Добавьте в `.env`:

```
SSO_ENABLED=true
SSO_ISSUER=https://idp.company.ru/realms/company
SSO_CLIENT_ID=doqa
SSO_CLIENT_SECRET=<секрет клиента из IdP>
SSO_REDIRECT=https://doqa.company.ru/api/auth/sso/callback
SSO_SCOPES="openid email profile"
SSO_LABEL="Corporate SSO"
```

Параметр	Описание
`SSO_ENABLED`	Включает корпоративный вход. По умолчанию `false`. Кнопка на странице входа появляется, только если заданы ещё и `SSO_ISSUER`, `SSO_CLIENT_ID`, `SSO_CLIENT_SECRET`.
`SSO_TYPE`	Тип коннектора. По умолчанию `oidc` — других значений нет.
`SSO_ISSUER`	Адрес провайдера. Конфигурация читается по стандартному пути `{issuer}/.well-known/openid-configuration`, отдельно указывать адреса authorize/token/jwks не нужно. С этим адресом сверяется значение `iss` в ID-токене. Синоним параметра — `SSO_BASE_URL`.
`SSO_CLIENT_ID`	Идентификатор клиента, заведённого в провайдере.

Параметр	Описание
`SSO_CLIENT_SECRET`	Секрет клиента. Хранится только в `.env`.
`SSO_REDIRECT`	Адрес возврата. Должен указывать на `https://<APP_URL>/api/auth/sso/callback` и быть добавлен в список разрешённых redirect URI на стороне провайдера.
`SSO_SCOPES`	Запрашиваемые scope, через пробел или запятую. По умолчанию `openid email profile`. Scope `openid` обязателен: без него провайдер не вернёт ID-токен.
`SSO_LABEL`	Название провайдера, которое видит пользователь на странице входа. По умолчанию `Corporate SSO`.

На странице входа появляется кнопка «**Корпоративный вход (SSO)**». Форма логина и пароля остаётся на месте: пока не включён `SSO_ENFORCE`, оба способа входа работают одновременно.

Тонкая настройка

Эти параметры менять обычно не нужно.

Параметр	Описание
`SSO_DISCOVERY_CACHE_TTL`	Время кэширования конфигурации провайдера, секунды. По умолчанию `3600`.
`SSO_JWKS_CACHE_TTL`	Время кэширования ключей подписи (JWKS), секунды. По умолчанию `3600`. Кэш нужен для ротации ключей на стороне провайдера.
`SSO_CLOCK_LEEWAY`	Допустимое расхождение часов сервера и провайдера при проверке срока действия токена, секунды. По умолчанию `60`.
`SSO_STATE_TTL`	Время жизни одноразового state/nonce/PKCE, секунды. По умолчанию `600`. Определяет, сколько у пользователя есть времени на аутентификацию в провайдере.

Как сопоставляются учётные записи

Когда пользователь входит через провайдера, DoQA ищет для него учётную запись по такому порядку:

1. Если внешняя учётная запись уже привязана, вход выполняется в связанного пользователя.
2. Если email подтверждён провайдером и такой пользователь в DoQA есть, внешняя учётная запись привязывается к нему автоматически, дальше вход идёт по привязке. Неподтверждённый email для поиска не используется, иначе чужую учётную запись можно было бы захватить.
3. Если пользователя нет, он создаётся, но только при включённом JIT-провижининге (ниже).
4. Иначе во входе отказано.

Роли из провайдера не читаются и не синхронизируются: права в DoQA выдаёт администратор внутри продукта через **роли и группы**. Состав **групп** может следовать провайдеру: группа с источником «SSO» заполняется из claim `groups` ID-токена, группа по правилу - из домена почты, атрибутов провайдера или способа входа. Членство пересчитывается при каждом входе.

Автоматическое создание пользователей (JIT)

JIT-провижининг создаёт пользователя при первом входе. Без него в DoQA смогут войти только те, кого администратор завёл заранее.

```
SSO_JIT_ENABLED=true
SSO_JIT_MODE=domain_allowlist
SSO_JIT_ALLOWED_DOMAINS=company.ru
```

Параметр	Описание
`SSO_JIT_ENABLED`	Включает автоматическое создание. По умолчанию `false`.
`SSO_JIT_MODE`	`domain_allowlist` — создавать только пользователей с доменом email из списка `SSO_JIT_ALLOWED_DOMAINS`. `open` — создавать любого, кого

Параметр	Описание
	пропустил провайдер. По умолчанию `domain_allowlist`. Отдельного значения «выключено» нет. За это отвечает `SSO_JIT_ENABLED`.
`SSO_JIT_ALLOWED_DOMAINS`	Домены через запятую или пробел. Учитываются только при `SSO_JIT_MODE=domain_allowlist`. Пустой список означает, что не создаётся никто.

Что происходит с созданным пользователем:

- Личных ролей у него нет. Он попадает в группы по умолчанию, в группы по правилам и в группы, соответствующие claim этого входа, и получает роли этих групп. Без групп с ролями он не видит ничего.
- Место в лицензии он не занимает: редактором его делает администратор вручную, см. **Лицензии (серверная версия)**.
- Локального пароля у него нет — только вход через провайдера.
- Дополнительные права на проекты и пространства выдаёт администратор внутри DoQA.

> **Внимание**
>
> Если `SSO_JIT_MODE` указан с опечаткой (любое значение, кроме `domain_allowlist` и `open`), пользователи не создаются вообще, а во входе новым сотрудникам отказывается без внятной причины. При `SSO_JIT_MODE=open` учётную запись в DoQA получит каждый, кто может войти в ваш провайдер. Проверьте, что круг пользователей провайдера действительно совпадает с кругом сотрудников, которым нужен DoQA.

JIT-политика общая для SSO и LDAP: те же три параметра включают автоматическое создание при первом входе по LDAP.

Обязательный вход через SSO (enforce)

`SSO_ENFORCE=true` запрещает вход по локальному паролю тем, кто считается управляемым через SSO: домен email входит в `SSO_JIT_ALLOWED_DOMAINS` либо к учётной записи привязана внешняя. Форма логина и пароля на странице входа при этом скрывается.

Исключение — только явный список:

```
SSO_ENFORCE=true
SSO_BREAKGLASS_EMAILS=admin@company.ru
```

Параметр	Описание
`SSO_ENFORCE`	Запрещает вход по паролю для SSO-управляемых пользователей. По умолчанию `false`. Проверка серверная. Вход по LDAP enforce не ограничивает: если включён `LDAP_ENABLED`, пароли по-прежнему проверяются в каталоге.
`SSO_BREAKGLASS_EMAILS`	Email через запятую или пробел, которым вход по паролю остаётся разрешён при любых условиях.

Заполните break-glass до включения enforce

Автоматических исключений нет. Владелец системы и суперадмин под `SSO_ENFORCE` попадают на общих основаниях.

Если включить `SSO_ENFORCE=true` с пустым `SSO_BREAKGLASS_EMAILS`, то при недоступном провайдере в DoQA не сможет войти никто: паролем вход запрещён, а через провайдера он не проходит. Восстановить доступ можно будет только правкой `.env` на сервере и перезапуском приложения.

Заполните `SSO_BREAKGLASS_EMAILS` **до** того, как включите `SSO_ENFORCE`, и проверьте, что у перечисленных учётных записей задан пароль.

Вход по паролю при недоступном провайдере

Форма пароля под `SSO_ENFORCE` скрыта, но не удалена. Чтобы её открыть, добавьте к адресу страницы входа параметр `?password`:

```
https://<APP_URL>/auth/login?password
```

Вход по этой форме пройдёт только для email из `SSO_BREAKGLASS_EMAILS`. Остальным SSO-управляемым пользователям сервер откажет, даже если пароль верный. Пользователей, чей домен не в списке `SSO_JIT_ALLOWED_DOMAINS` и у кого нет привязанной внешней учётной записи, enforce не касается: они входят паролем как обычно.

Задать пароль пользователю break-glass можно из консоли сервера. Пароль будет запрошен скрытно:

```bash
docker compose exec api php artisan users:set-password admin@company.ru
```

Проверка настройки

1. Откройте страницу входа. Если кнопки «**Корпоративный вход (SSO)**» нет, `SSO_ENABLED` не применился: проверьте, что приложение перезапущено после правки `.env`.
2. Нажмите кнопку. Браузер должен уйти на страницу аутентификации провайдера.
3. После входа вас вернёт в DoQA. Если вместо этого страница входа показывает «Не удалось войти через внешнего провайдера», причина указана в адресной строке параметром `error=sso_*` и в логах:

```bash
docker compose logs -f api
```

Значение `error`	Причина
`sso_callback`	Провайдер вернул ошибку или не передал код авторизации. Проверьте `SSO_REDIRECT` в `.env` и список разрешённых redirect URI в провайдере.
`sso_state`	Истекло время на аутентификацию (`SSO_STATE_TTL`) или запрос пришёл повторно.
`sso_not_allowed`	Пользователя нет в DoQA, а JIT выключен либо домен email не входит в `SSO_JIT_ALLOWED_DOMAINS`.

Переход на новую систему прав

Новая система прав (роли, группы, выдачи в проектах и пространствах, запреты) включается при обновлении DoQA. Права, выданные в прежней модели, переносятся автоматически, во время обычного обновления: отдельных команд запускать не нужно. Прежние возможности сохраняются; что меняется осознанно, описано в разделе **Что меняется для людей**.

Перед обновлением

1. Убедитесь, что обновление идёт по обычному порядку: **Обновление (update)**. Утилита сама создаёт резервную копию базы и файлов перед обновлением; если она не создаётся (изменённые настройки, внешнее хранилище, несоответствие требованиям), сделайте копию вручную: **Резервное копирование (backup)**.
2. Предупредите администраторов: после обновления раздел администрирования выглядит иначе, см. **Администрирование**.
3. Приложение недоступно на время обновления, обычно несколько минут.

Что происходит при обновлении

1. Создаются узлы дерева: организация, все проекты и пространства, включая архивные.
2. Создаются системные роли, см. **Роли и права**.
3. Прежние роли и доступы переводятся в выдачи по таблице ниже.
4. Для каждого человека сравниваются прежние и новые разрешённые действия. Если перевод какой-то прежней роли или доступа даёт больше, чем она давала, это ошибка перевода: обновление останавливается, ничего не переносится, прежняя система остаётся рабочей. Если у человека пропало прежнее действие, он попадает в отчёт с перечнем потерянных действий.
5. Итоговые права пересчитываются.
6. Таблицы прежней системы переименовываются в `legacy_bouncer_*` и остаются на 30 дней.

При каждом следующем обновлении системные роли сами приводятся к составу новой версии, права людей с этими ролями пересчитываются.

Отчёт о переходе записывается в **журнал аудита** записью «Переход на новую систему прав». В нём по людям перечислены расширения и сужения из раздела ниже, расхождения и потерянные действия.

Соответствие

Было	Стало
Владелец системы	Владелец организации
Роль «Супер админ»	«Администратор организации» и «Администратор проектов» на организации
Роль «Администратор пользователей»	«Администратор пользователей» (с правами лицензий и оплаты, так что покупка и продление остаются доступны)
Роль «Создатель проектов»	«Создатель проектов»
«Администрирование» для всех проектов	«Тест-лид» на организации
«Редактирование» для всех проектов	«Тестировщик» на организации
«Просмотр» для всех проектов	«Наблюдатель» на организации
«Администрирование» одного проекта	«Тест-лид» на проекте
«Редактирование» одного проекта	«Тестировщик» на проекте
«Просмотр» одного проекта	«Наблюдатель» на проекте
Все три раздела пространства (кейсы, чек-листы, прогоны)	«Тестировщик» на пространстве
Часть разделов пространства	Своя роль вида «Тестировщик (только кейсы)», «Тестировщик (кейсы и прогоны)» и подобные, на пространстве. Создаются только для встречающихся сочетаний
«Скрыть» проект	**Запрет** «весь доступ» на проекте

Было	Стало
Лицензия	Без изменений
Токен проекта	Сервисный аккаунт с ролью «Агент CI» на проекте, секрет тот же: переменные CI менять не нужно
Личный API-токен	Личный токен с правами владельца на всех его местах, срок действия не меняется: токен без срока остаётся бессрочным

Что меняется для людей

Права в новой системе складываются: всё, что выдано человеку на организации, проекте и пространстве, действует вместе. Поэтому есть два осознанных изменения, оба попадают в отчёт перехода по людям.

Расширение: было погашено ролью, стало действует. В прежней модели общая роль «Просмотр» (`view_role`) отключала доступы «Администрирование» и «Редактирование» проекта и правку кейсов, прогонов и чек-листов в пространствах, даже если они были выданы человеку отдельно. Роль «Редактор пространств» (`spaces_editor_role`) так же отключала «Администрирование» проекта. Теперь такие доступы работают: человек с «Наблюдателем» на организации и «Тест-лидом» на проекте может править этот проект. В отчёте перехода это класс «было погашено ролью, стало действует»: кто, какой ролью было погашено, какие доступы и где. Если расширение не нужно, уберите лишнюю выдачу в карточке человека.

Сужение: интеграции и настройки ИИ организации. Подключение и отключение интеграций и трекеров и настройки ИИ на уровне организации теперь доступны только администраторам организации. Раньше интеграциями могли управлять также люди с «Администрированием» любого проекта, с ролями «Редактор проектов» и «Создатель проектов», а настройками ИИ также «Администратор пользователей», «Редактор проектов» и люди с «Администрированием» проекта. Эти люди перечислены в отчёте перехода как сужения. Если кому-то из них нужно управлять интеграциями или ИИ, выдайте роль с правами `integration.manage` или `ai_settings.manage` на организации, см. **Роли и права**.

Остальные изменения закрывают дефекты прежних прав:

- держатели прежних общих уровней «Просмотр» и «Редактирование» больше не могут выдавать права в проектах: управление доступом даёт право `access.manage`, которого нет у ролей «Наблюдатель» и «Тестировщик»;
- администратор пользователей не может выдать себе или другим полный доступ в проектах: выдача роли требует, чтобы все её права были у выдающего.

Подробно о правах для выдачи ролей: **Роли и права**. Прежнюю схему с уровнями замещают **роли**, **доступ на всю организацию** и **доступ к проектам**.

Команды

При обновлении ничего запускать не нужно: перенос, системные роли и пересчёт прав выполняет само обновление. Команды ниже нужны для диагностики и выполняются на сервере приложения (серверная версия).

Команда	Назначение
`access:migrate --dry-run`	Пробный перенос: показать отчёт, ничего не сохраняя. Флаг `--report=<файл>` сохраняет отчёт в JSON, включая расширения и сужения по людям
`access:migrate`	Повторный перенос. Ничего не дублирует
`access:sync-system-roles`	Принудительно привести системные роли к составу текущей версии
`access:reconcile`	Пересчитать итоговые права всех пользователей
`access:legacy-restore`	Вернуть таблицам прежней системы прежние имена (откат в окне 30 дней)
`access:purge-legacy`	Удалить таблицы `legacy_bouncer_*`, когда прошло 30 дней после перехода
`access:grant-org-admin <почта>`	Выдать человеку «Администратора организации» и «Администратора проектов» на организации

После обновления

1. Откройте администрирование и проверьте раздел «Пользователи» и «Роли»: у администраторов должны быть те же возможности.
2. Просмотрите отчёт «Переход на новую систему прав» в журнале аудита: расширения (было погашено ролью), сужения (интеграции и настройки ИИ) и людей с потерянными действиями.
3. Настройте **группы** и переведите повторяющиеся личные роли в них: отчёт «Доступ вне групп» в **пересмотре доступа** покажет кандидатов.
4. Проверьте токены проектов и интеграции CI.

Откат

В течение 30 дней таблицы прежней системы сохраняются. Откат на предыдущую версию приложения возможен командой `access:legacy-restore` и восстановлением из резервной копии. Выдачи и роли, созданные в новой системе после перехода, в прежней версии не действуют. Токены проектов, в том числе выпущенные после перехода, в прежней версии продолжают работать. При повторном обновлении перенос выполняется заново: права, выданные в прежней версии за время отката, тоже переносятся. По истечении 30 дней `access:purge-legacy` удаляет таблицы окончательно.

Релизы облачной версии DoQA

DoQA 4.3 Titanium от 17.09.2026

Версия 4.3 целиком посвящена автоматизированному тестированию. В DoQA появился полноценный раздел **«Автотесты»** с собственным каталогом, запуск пайплайнов в трёх CI-системах, критерии приёмки прогона Quality Gate, работа с нестабильными тестами и автоматический разбор причин падений. Ручные и автоматизированные проверки по-прежнему находятся в одном прогоне. Теперь у автотеста в нём есть история попыток, шаги, вложения и логи. Не обошли и ручное тестирование: в продукте появился глобальный поиск по всей документации, авторaспределение исполнителей, прогноз времени прохождения прогона и выгрузка отчётов в Excel и CSV.

Раздел «Автотесты»

- Все автотесты пространства собраны в одном каталоге с деревом структуры, фильтрами по ветке, тегам, источнику и карантину.
- У каждого автотеста есть состояние: здоровый, нестабильный, сломанный, в карантине, отключённый, восстановленный.
- Боковая панель автотеста показывает историю прохождений, стабильность, связанные тест-кейсы.
- Смотрите каталог «по источнику»: из какого репозитория и пайплайна приехал каждый автотест.
- Запускайте выбранные автотесты прямо из каталога.
- Настраивайте состав и ширину столбцов каталога, назначайте автотестам собственные теги.
- Создавайте ручной тест-кейс на основе автотеста, у которого ещё нет связей.

Подробнее о каталоге автотестов

Запуск автотестов в CI/CD

- Подключайте три CI-системы: GitLab, Jenkins, TeamCity.

- Заводите CI-подключения на уровне пространства и держите несколько подключений одного типа, например два сервера GitLab.
- Запускайте только выбранные автотесты: DoQA собирает нативный фильтр раннера. Если набор фильтром не выражается, прогон идёт целиком, а не молча урезается.
- Один прогон может держать несколько пайплайнов сразу: разнородный набор разбивается на группы «интеграция, проект, ветка», а прогон закрывается, когда завершился последний из них.
- Формы запуска автотеста предзаполняются последней использованной связкой «CI-система, проект, ветка».
- Задавайте порядок прохождения автотестов в наборе перед стартом при запуске автотестов из каталога.

Подробнее о запуске из CI/CD

Расписание запусков

- Запускайте автотесты по расписанию, без изменений в конфигурации CI.
- Задавайте название будущих прогонов, часовой пояс, окружение и поведение на случай, если предыдущий прогон расписания ещё не завершился.
- Переопределяйте для запусков по расписанию критерии Quality Gate и политику авто-повторов.
- Журнал срабатываний показывает каждый запуск: созданный прогон или причину, по которой он не состоялся. После серии сбоев расписание отключается само.

Подробнее о расписаниях

Quality Gate — критерии приёмки прогона

- Опишите, при каких условиях прогон считается пройденным: минимальная доля успешных тестов, предельное число падений, минимальное число тестов, максимальная длительность, обязательные окружения.
- Запретите «позеленение» прогона за счёт повторов и включите ранний останов.
- Критерии задаются в настройках пространства и переопределяются для отдельного прогона или расписания.
- Вердикт виден прямо в отчёте о прогоне, а команда `doqactl gate` возвращает его в пайплайн кодом выхода.

Подробнее о Quality Gate

Нестабильные тесты, карантин и авто-повтор

- DoQA сама находит нестабильные автотесты по истории прогонов: доля падений в окне, разный исход на одном коммите, «прошёл со второй попытки».
- Отправляйте нестабильный тест в карантин: он продолжает выполняться, но не роняет Quality Gate и пайплайн. Карантин снимается автоматически, когда тест успокоился.
- Настраивайте пороги нестабильности и срок снятия пометки.
- Настраивайте авто-повтор упавших автотестов: число попыток, задержка, теги для повтора и исключения, поведение пайплайна, если все попытки упали.

Кластеры ошибок и известные проблемы

- Падения с одинаковой причиной автоматически собираются в один кластер по сигнатуре ошибки.
- Раскладывайте падения по классам «Продукт», «Автотест» и «Окружение» собственными правилами. Шаблон проверяется прямо в форме.
- Заводите баг из кластера или привязывайте существующий. Кластер можно отметить известной проблемой: она перестаёт шуметь в следующих прогонах и открывается заново при регрессе.

Подробнее о результатах и разборе падений

Прогон и его плеер

- Перепроходите упавший автотест: автоматически новым запуском или вручную по шагам последнего автоматического прохождения. Каждая попытка сохраняется отдельно, прежний результат не затирается.
- Перезапускайте выбранные строки завершённого прогона пакетом.
- Вкладка «Пайплайны и джобы»: список шагов пайплайна со статусами, остановка и перезапуск отдельной джобы, номер запуска в CI-системе и ссылка на пайплайн.
- Результат автотеста показывает дерево шагов и фикстур, вложения (скриншоты, видео, трассировки и логи) и параметры запуска. Вложения доступны по каждой попытке.
- Бейдж «Не применено» в шапке прогона показывает, сколько результатов не были засчитаны и почему: устаревший результат, тест вне плана, тест не выполнялся, вложения

не приняты.

- Смотрите, куда ушло время одного запуска, от команды в CI до приезда отчёта.

Дашборд «Автотесты» и аналитика

- Новая вкладка «Автотесты» в дашбордах пространства: доля автоматизации, здоровье парка тестов, доля успешных прохождений, нестабильность, длительность, топ падающих тестов, разбивка по веткам и окружениям, экономия времени CI и топ кластеров ошибок.
- Карта покрытия и лента прогонов во времени показывают, что шло параллельно и чем закончилось.
- Фильтруйте дашборд по ветке, тегу и периоду.

Адаптеры, CLI и плагины IDE

- Адаптеры для JUnit 5, JUnit 4, TestNG и PyTest отправляют результаты в DoQA прямо из прогона тестов, без файлового отчёта.
- Закрепляйте за тестом постоянный идентификатор аннотацией `@DoqaId`, а меткой `@DoqaCreateManualCase` просите DoQA завести для теста ручной кейс.
- Утилита командной строки умеет больше: `upload` — загрузка готового отчёта JUnit XML или Allure, `run` — прогон тестов с отправкой результата одной командой, `watch` — слежение за прогоном, `gate` — вердикт Quality Gate для шага пайплайна, `sync-ids` — простановка идентификаторов в исходниках тестов, `login` — сохранение адреса, токена и пространства.
- Плагины для IntelliJ IDEA и VS Code: рядом с каждым тестом видно, привязан ли он к кейсу, кейс можно найти и привязать, проставить идентификатор для теста или всего файла и посмотреть каталог автотестов пространства, не выходя из редактора.
- Параметризованные тесты больше не схлопываются в одну строку. У каждого набора данных свой результат. Теги теста доезжают в DoQA как метки.

Подробнее об **адаптерах** и **плагинах**

Глобальный поиск

- Строка поиска в шапке приложения ищет по тест-кейсам, чек-листам и общим шагам всего продукта.

- Отдельная страница результатов с фильтрами и подсветкой совпадений в найденном тексте.

Прогоны и ручное тестирование

- Авторaспределение исполнителей: тесты распределяются между выбранными участниками равномерно по среднему, медианному или плановому времени прохождения.
- Прогноз времени прохождения прогона по накопленной статистике.
- Умное копирование прогона: можно скопировать в новый прогон, например, только упавшие тесты.
- Выгрузка отчёта о прогоне в Excel и CSV.
- Единый диалог добавления тестов в прогон: кейсы, чек-листы и автотесты в одном окне с общим поиском, деревом папок и фильтрами по тегам.
- Ссылка на прогон запоминает открытую вкладку. Можно переслать коллеге ссылку ровно на то, что вы смотрите.

Подробнее о прогонах

Кейсы, чек-листы и требования

- Настраивайте состав выгрузки тест-кейсов и чек-листов и выбирайте формат файла.
- Связывайте требование напрямую с автотестом, без тест-кейса-посредника. Автотест участвует в расчёте покрытия требования, а в карточке требования появилась панель связанных автотестов.
- У тест-кейса появился статус автоматизации и признак происхождения: видно, что кейс создан из автотеста.
- Синхронизация требований из Confluence: можно взять пространство целиком, а не только поддерево от выбранной страницы. Повторная синхронизация большого пространства больше не обрывается на середине.

Уведомления

Новые события в настройках уведомлений: автотест признан нестабильным, сбой запуска автотестов по расписанию.

Подробнее об уведомлениях

Интерфейс

Настройки пространства собраны на одной странице с разделами. Обновлены страница «Проекты» и боковое меню, футер переехал в выпадающее меню. В меню пользователя появился пункт «О DoQA».

DoQA 4.2 Niccolum от 16.07.2026

Модуль управления требованиями

- Настройте синхронизацию между DoQA и трекером, чтобы получать требования в TMS.
- Связывайте требования с тест-кейсами и чек-листами для контроля покрытия требований тестами.
- Генерируйте тест-кейсы и чек-листы с помощью ИИ на основе содержания требования.
- Получайте уведомления об изменении требований и необходимости актуализировать связанные тесты.
- Используйте один из двух режимов отображения:
  - дерево требований;
  - матрица покрытия.
- Экспортируйте матрицу покрытия в удобном формате.

Улучшения в плеере прогона

- Перезапускайте автотест прямо из результатов прохождения: «Пройти заново → Автоматически».
- Для каждого теста в прогоне отображаются:
  - метрика надежности;
  - среднее время выполнения за последние 10 запусков;
  - история трех последних результатов.

ИИ

- Подключайте локальные LLM и любые модели с OpenAI-совместимым API.
- Генерируйте наборы тест-кейсов на основе требования из трекера.
- Настраивайте промпт генерации набора тест-кейсов, чтобы адаптировать результаты под процессы вашей команды.

Редизайн главной страницы

- Выберите один из двух вариантов отображения проектов и пространств:
  - классическое дерево;
  - альтернативный просторный режим.
- Настройте порядок отображения проектов и пространств. Администраторы могут задать порядок по умолчанию для всех пользователей.
- Для пространств добавлено поле «Описание».
- Поиск теперь работает не только по названиям проектов, но и по названиям пространств и описаниям.

Кастомизация таблиц

- Настройте отображение таблиц:
  - во вкладках «Кейсы», «Чек-листы» и «Прогоны»;
  - в плеере прогона.
- Изменяйте:
  - набор столбцов;
  - порядок столбцов;
  - ширину столбцов.
- Сохраняйте настройки в пресеты и быстро переключайтесь между ними.

Множественные параметры в тестах

- Добавляйте в тест-кейсы и чек-листы несколько параметров: локальных и глобальных.
- В отдельной вкладке в редакторе ТК или ЧЛ можно выбрать способ формирования комбинаций параметров: все комбинации, каждый выбор, линейное сопоставление, попарное тестирование (pairwise).

Подробнее о параметризации

Плановое время прохождения тестов

- Указывайте плановое время выполнения тест-кейса или чек-листа.

Комментарии из прогонов в редакторах ТК и ЧЛ

- Комментарии, оставленные при выполнении тестов в прогонах, теперь доступны в редакторах тест-кейсов и чек-листов на отдельной вкладке.

Обновленный Public API

- Расширен список методов Public API.
- В профиле пользователя появилась возможность создать постоянный персональный API-токен.

Подробнее о Public API

DoQA 4.1 Cuprum от 13.05.2026

Комментирование тест-кейсов и чек-листов

- Панель комментирования открывается в редакторе и позволяет оставлять одиночные комментарии или открывать обсуждения, а также упоминать других пользователей DoQA. Автор получит уведомление, когда на его комментарий ответят.
- Отработанный комментарий можно отметить как «решенный» и оставить для истории, либо удалить.
- Функционал комментирования доступен также пользователям без лицензий. Теперь бесплатные «наблюдатели» становятся полноценными участниками коммуникации в DoQA. Приглашайте аналитиков, разработчиков и дизайнеров вашей команды в DoQA для совместного ревью тестовой документации.

Подробнее о комментариях

Параметризация тестов

- Добавьте параметр с его возможными значениями в тест-кейс или чек-лист, а также укажите в тексте вхождения этого параметра. При добавлении теста в прогон под каждое значение параметра будет создан отдельный слепок теста.

Подробнее о параметризации

Развитие DoQA AI

- Помимо использования ИИ-функций через встроенную интеграцию с LLM-сервисом и оплатой через DoQA, добавлен альтернативный вариант: **подключение альтернативных провайдеров** OpenAI и Яндекс с возможностью выбора любых доступных моделей.
- **Пользовательская настройка ИИ-промптов.** Промпты для всех ИИ функций DoQA теперь отображаются в настройках администратора и могут быть изменены пользователями под свои процессы и стандарты тестирования.
- Потоковая обработка ИИ-запросов. Мы повысили скорость и надежность генерации информации с помощью ИИ. Теперь ожидание результатов генерации составляет от пары секунд, в зависимости от размера запроса.

Улучшение пользовательского опыта

Мгновенные сообщения об изменении списка тестов в прогоне теперь можно отключить в профиле пользователя, если получение данных уведомлений не требуется.

DoQA 4.0 — Carboneum от 19.03.2026

Версия DoQA 4.0 открывает новый этап развития системы. В этом релизе появилась поддержка автоматизированного тестирования и интеграция с CI/CD-процессами.

Теперь ручные и автоматизированные проверки можно объединить в одном прогоне, запускать автотесты напрямую из интерфейса DoQA, связывать их с тест-кейсами и анализировать результаты в едином отчете.

Кроме того, обновлён интерфейс плеера прогонов и добавлены новые возможности при прохождении тестов в прогоне.

Поддержка автоматизированного тестирования

DoQA теперь позволяет работать с автотестами как с полноценной частью тестового процесса.

- **Связка автотестов с тест-кейсами**. Автотесты могут автоматически связываться с тест-кейсами по Allure ID. Это позволяет объединить ручную тестовую документацию и автоматизированные проверки в единую структуру.
- **Отображение автотестов в прогоне**. Результаты автотестов теперь отображаются непосредственно в прогоне и могут быть объединены в единый список с ручными тест-кейсами и чек-листами. К пройденным автотестам можно оставлять баг-репорты и комментарии, а также перепроходить их вручную.

Ранее загруженные в DoQA отчеты о прохождении автотестов мигрируют в «Прогоны» и будут помещены в отдельную папку.

Интеграция с GitLab CI/CD

DoQA теперь может напрямую взаимодействовать с пайплайнами CI/CD. Создайте интеграцию между DoQA и GitLab CI/CD, после чего выберите проекты из репозитория, в которых планируете запускать автотесты.

- **Запуск пайплайнов из DoQA**. Вы можете создать новый прогон через запуск пайплайна в CI/CD непосредственно из интерфейса DoQA. По мере выполнения автотестов результаты будут загружаться в созданный прогон.

При запуске пайплайна из DoQA можно выбрать один из доступных проектов из CI/CD, а также ветку.

- **Выборочный запуск автотестов**. Автотесты, связанные с тест-кейсами через Allure ID, можно запускать прямо из прогона, отметив нужные элементы.

Импорт результатов автотестов

В интерфейс добавлен функционал создания прогона через загрузку отчётов о прогонах автотестов в форматах JUnit XML и Allure. Также результаты прохождения автотестов можно

загружать в DoQA через утилиту doqa-cli.

Обновлённый интерфейс для работы с прогонами

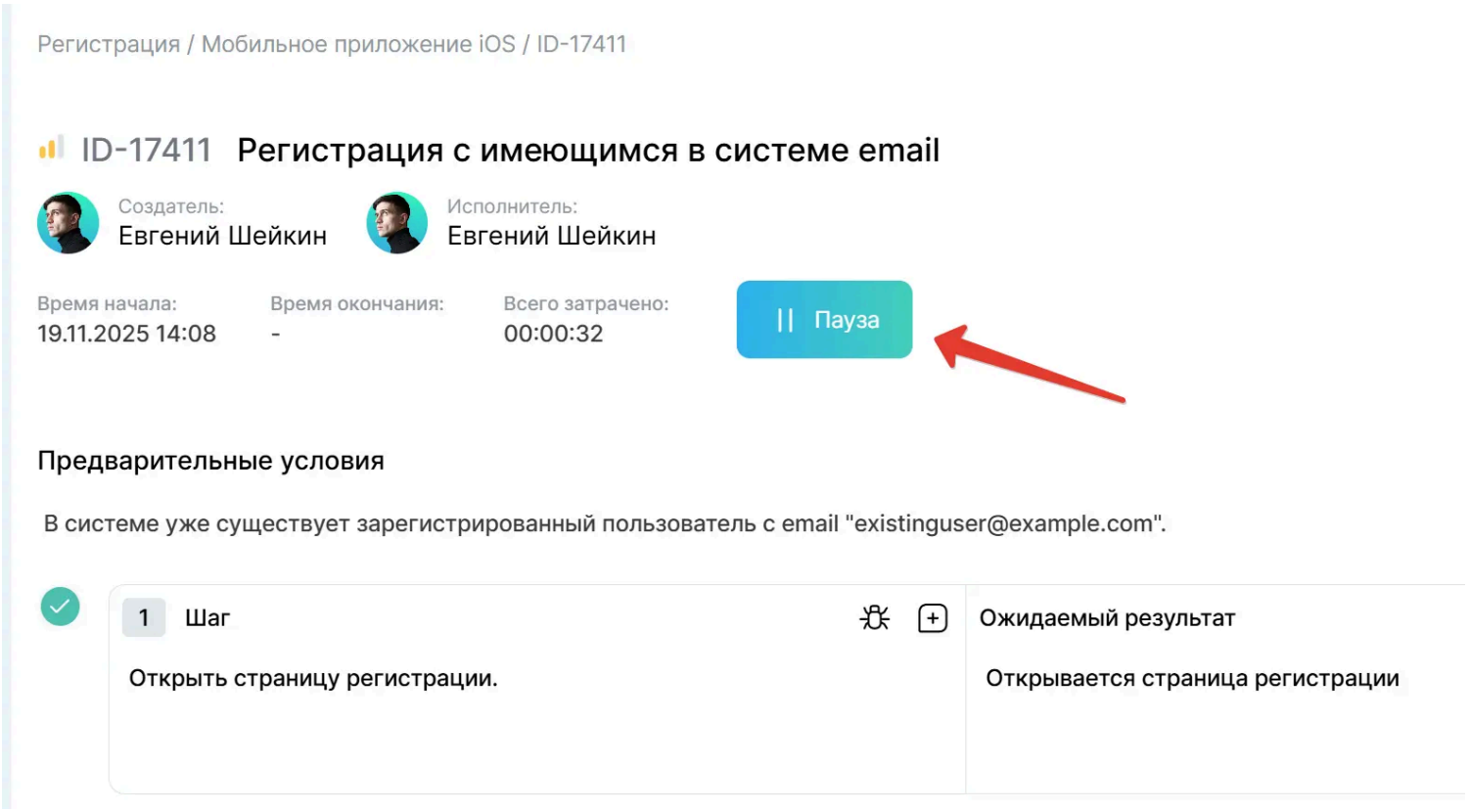
В версии 4.0 переработан интерфейс работы с прогоном, чтобы ускорить выполнение тестов и упростить навигацию.

- **Новое окно создания прогона**. Создание прогона теперь выполняется через компактное модальное окно без перехода в отдельный редактор. Помимо стандартного создания прогона можно выбрать альтернативные варианты: загрузку отчетов о прогонах автотестов и запуск автотестов в CI/CD.
- **Редактирование прогона во время выполнения**. Добавлять тест-кейсы и менять их порядок можно прямо в плеере прогона, не переходя в отдельный редактор. Назначать исполнителей на тесты в прогоне теперь можно массово на основе выбранных фильтров.
- **Обновлённый экран прохождения теста**. Мы переработали экран прохождения тестов в прогоне, сделав его более структурированным и увеличив полезное пространство.
- **История прохождений**. Интерфейс выполнения тестов теперь разделён на две вкладки: прохождение и история прохождения. Вы можете увидеть все попытки перепрохождения теста в прогоне.
- **Папки в прогонах**. На основе обратной связи от наших пользователей мы реализовали папки в прогонах для более удобного хранения документации.

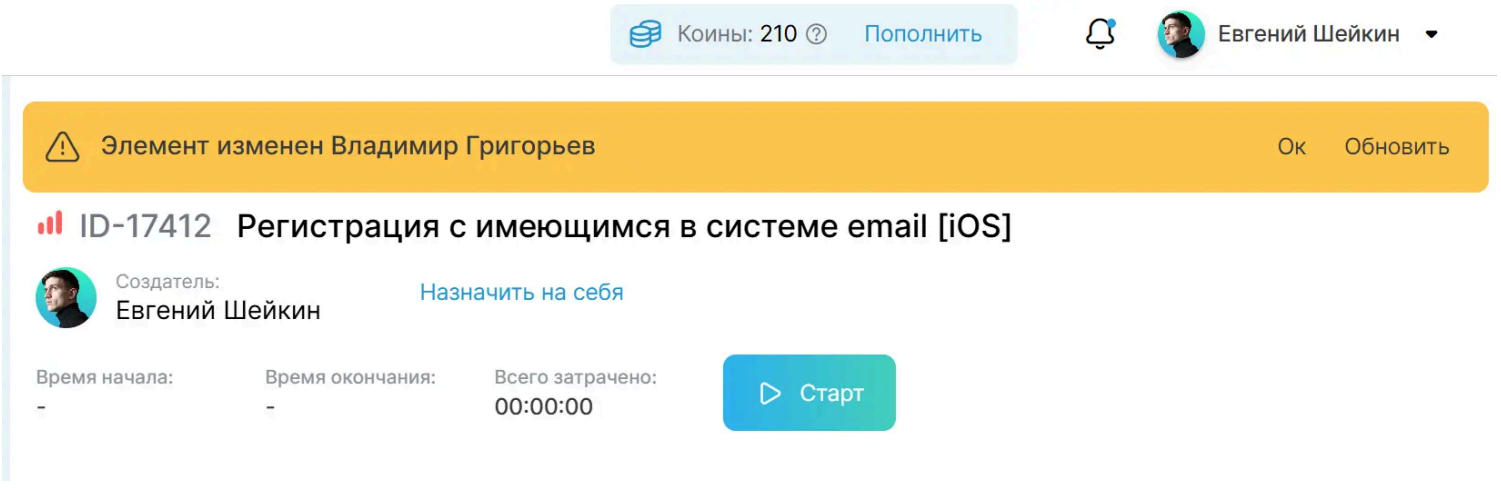
3.7.0 от 18.11.2025

Текущий релиз был посвящен доработкам функционала на основе обратной связи от наших пользователей и улучшению пользовательского опыта.

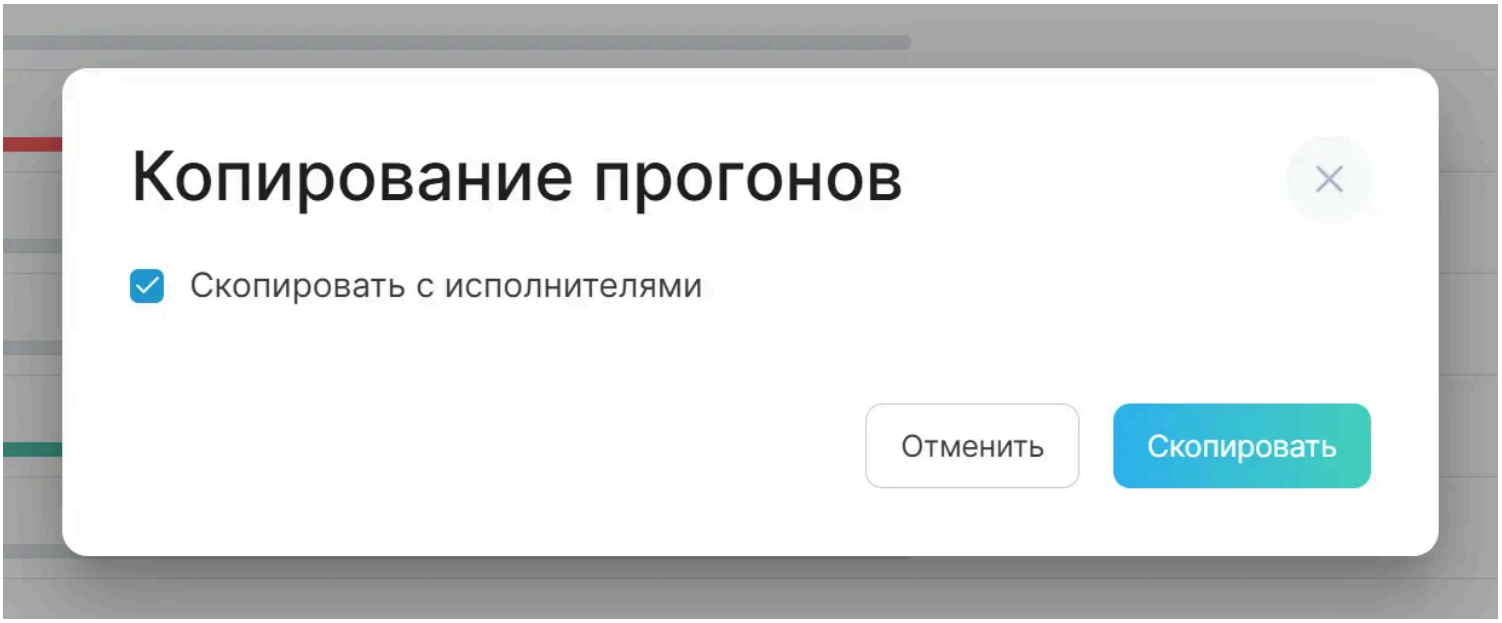
- Кнопки старта и паузы в окне прохождения тестов в прогоне. В плеере прохождения кейсов и чек-листов появились кнопки старт/пауза. Это полезно при переключении между задачами, когда нужно поставить прохождение на паузу, чтобы не искажать статистику.



- Уведомления в плеере прогона. Если другой пользователь изменил статус прогона или открытого вами для прохождения теста, вы сразу же увидите соответствующее уведомление.

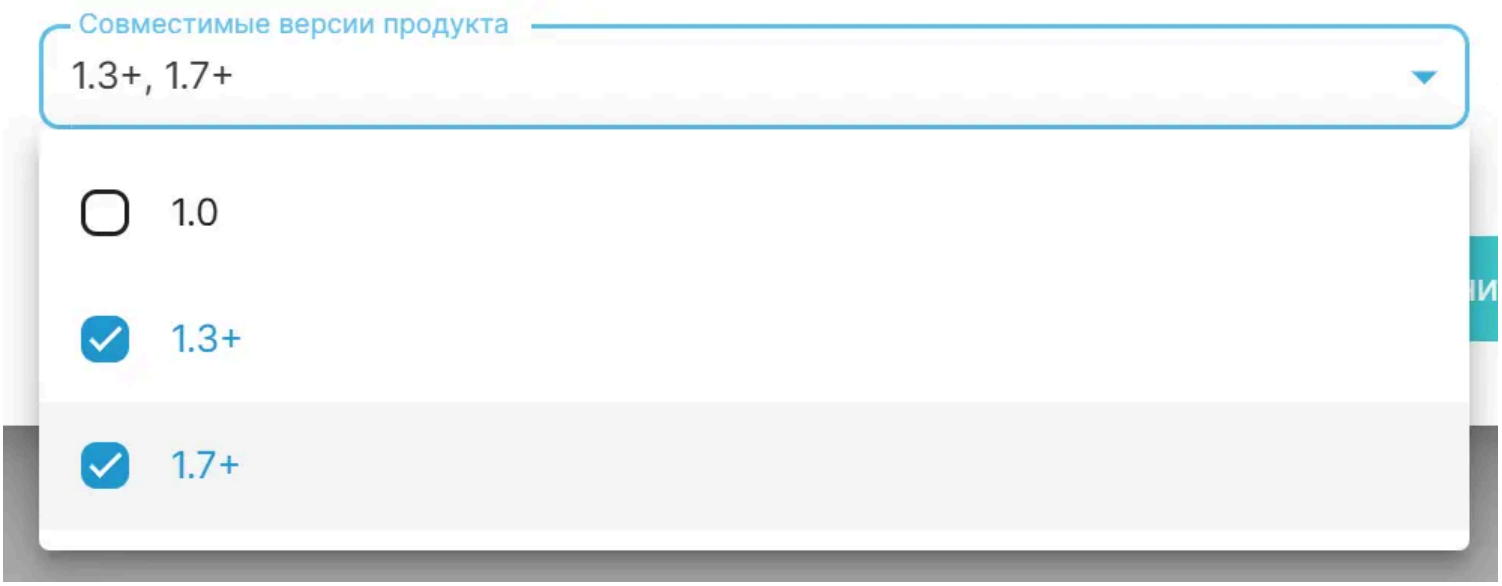


- Копирование прогона с исполнителями или без. Это позволяет сохранить прежнее распределение исполнителей в новом прогоне, не назначая их заново.



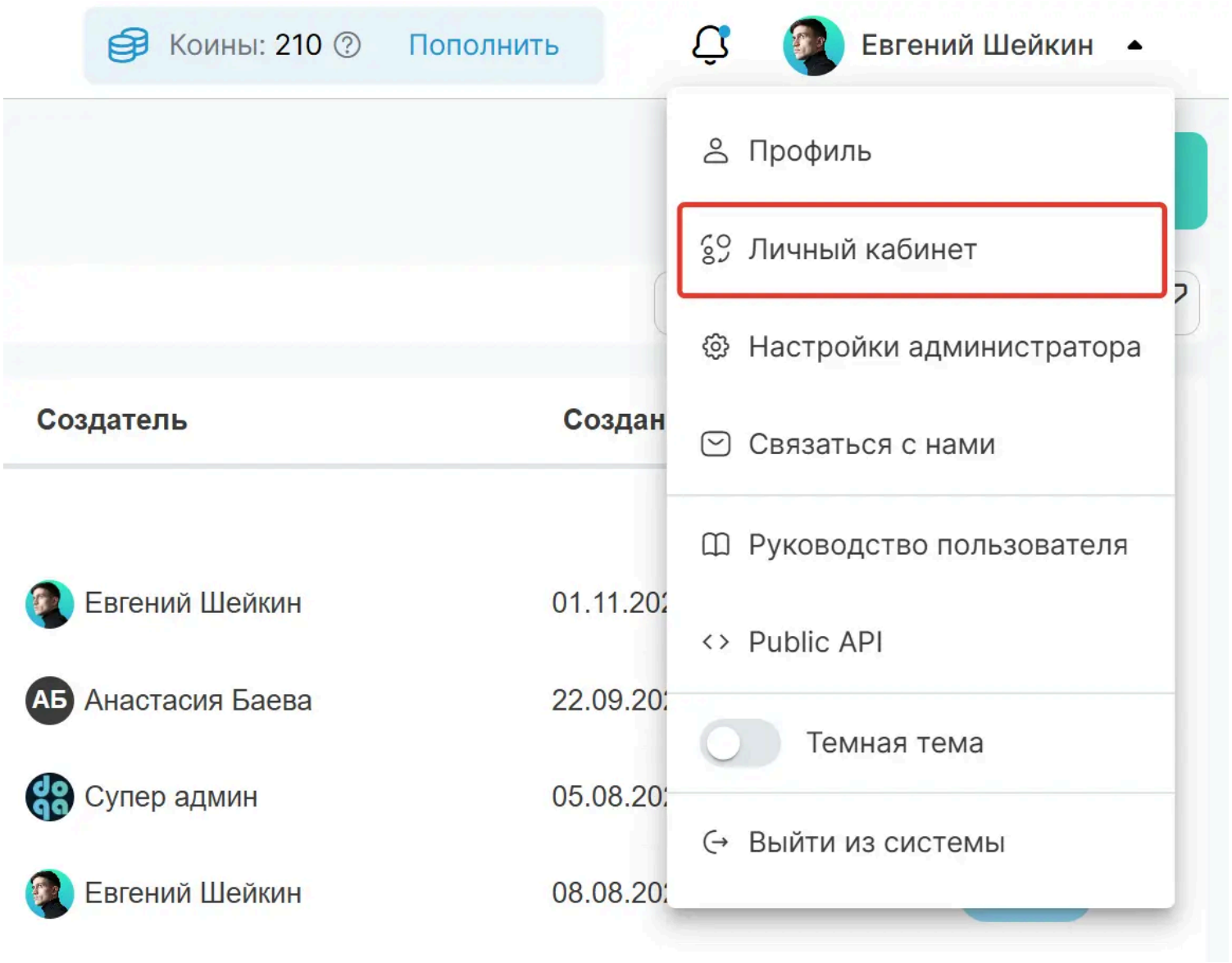
- Новый атрибут с множественным выбором. Добавлен новый тип атрибута с множественным выбором.

Подробнее об атрибутах



- Доработки импорта проектов. При импорте проекта из TMS Test IT теперь сохраняется порядок тест-кейсов в папках (секциях).

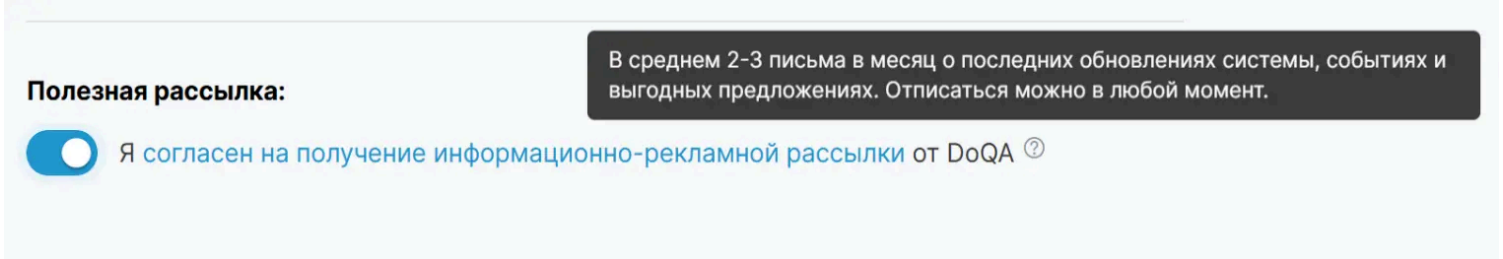
- Переход в Личный кабинет через меню пользователя. Теперь перейти в ЛК можно напрямую из пользовательского меню.



- Обновленный парсер отчетов автотестов. При загрузке отчетов автотестов теперь можно указывать его имя через параметр. Это помогает однозначно идентифицировать отчеты и облегчает навигацию.

Подробнее о загрузке отчетов о прогонах автотестов

- Подписка на рекламную рассылку из профиля. В профиле стала доступна подписка на рассылку от команды DoQA.



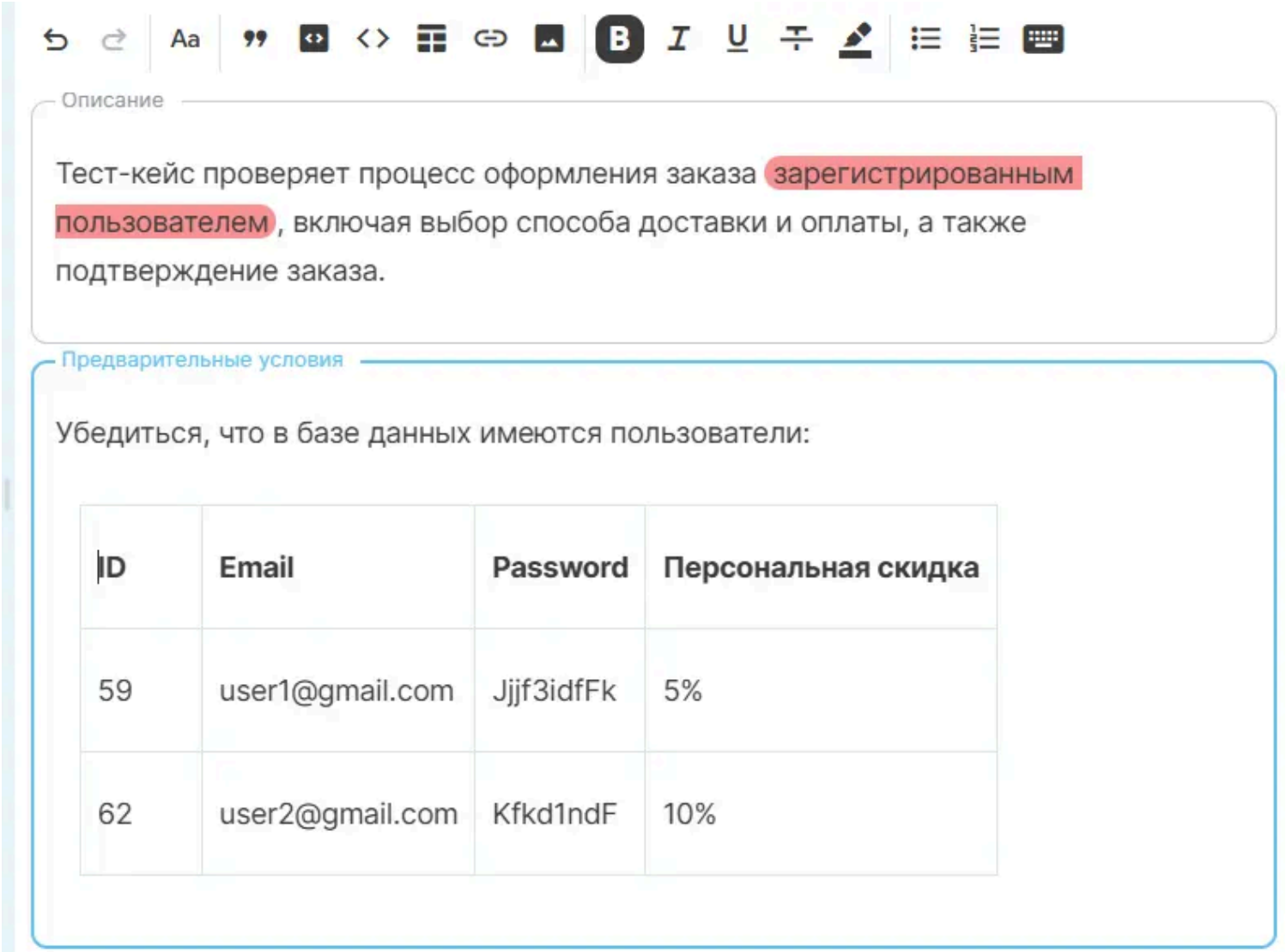
Мы делимся: новостями о платформе, актуальными предложениями, анонсами релизов и улучшений.

Рассылку можно отменить в любой момент.

3.6.0 от 29.09.2025

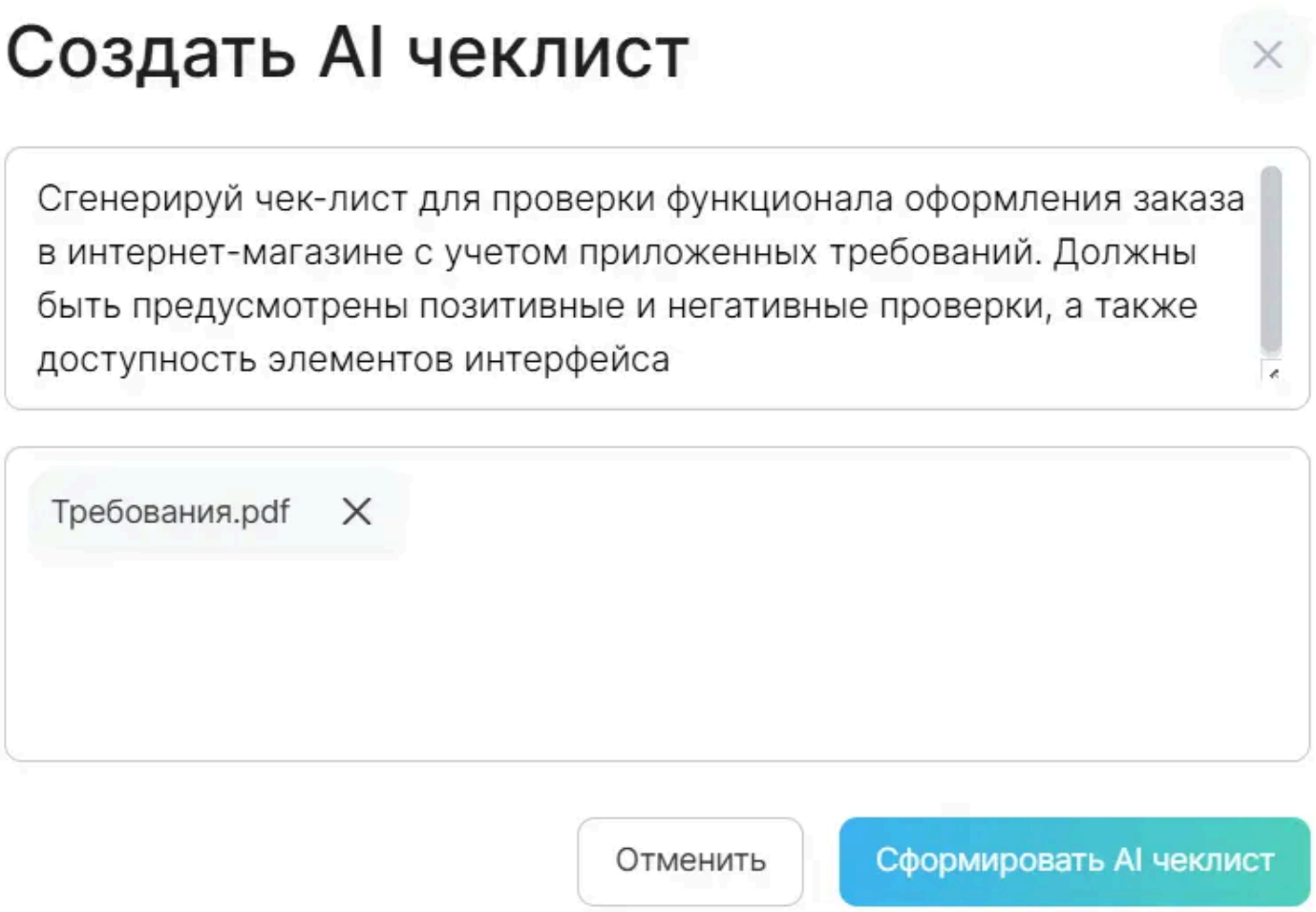
В этом релизе мы обновили текстовый редактор, добавили новые AI-возможности, расширили функционал уведомлений и реализовали несколько запросов пользователей.

- Новый текстовый редактор. В новом текстовом редакторе, помимо привычных функций, доступны: настройка размера изображений, добавление таблиц, использование **горячих клавиш** для ускорения работы.

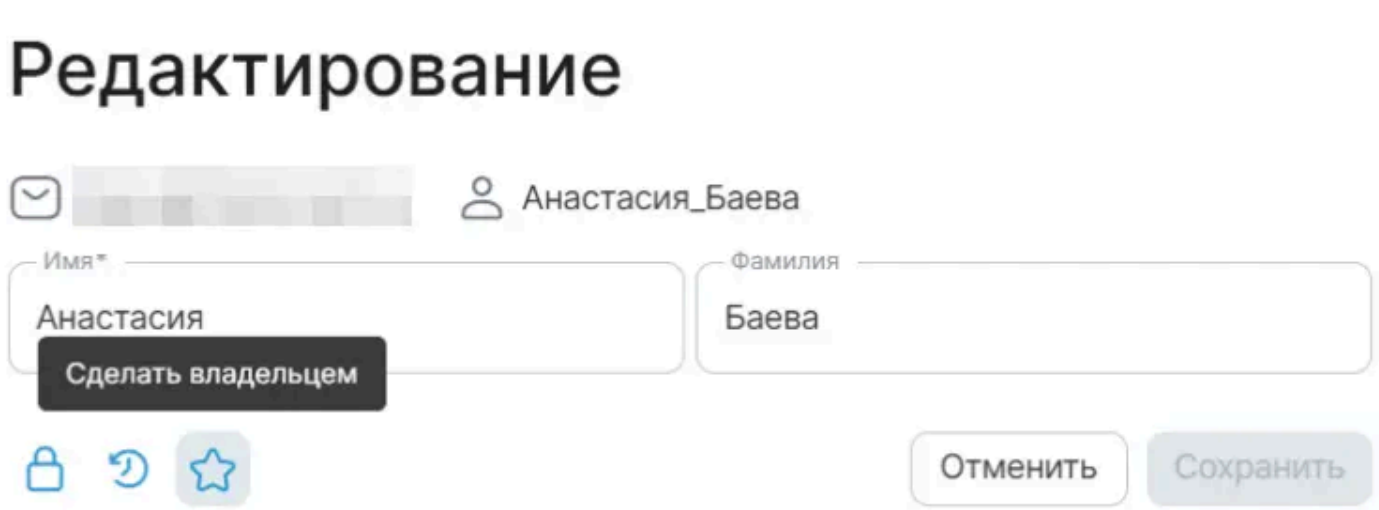


- Генерация чек-листов. Больше не нужно составлять проверки вручную: опишите сценарий или скопируйте фрагмент требований к фиче — AI предложит готовый чек-лист с многоуровневыми проверками.

Подробнее об AI-функциях

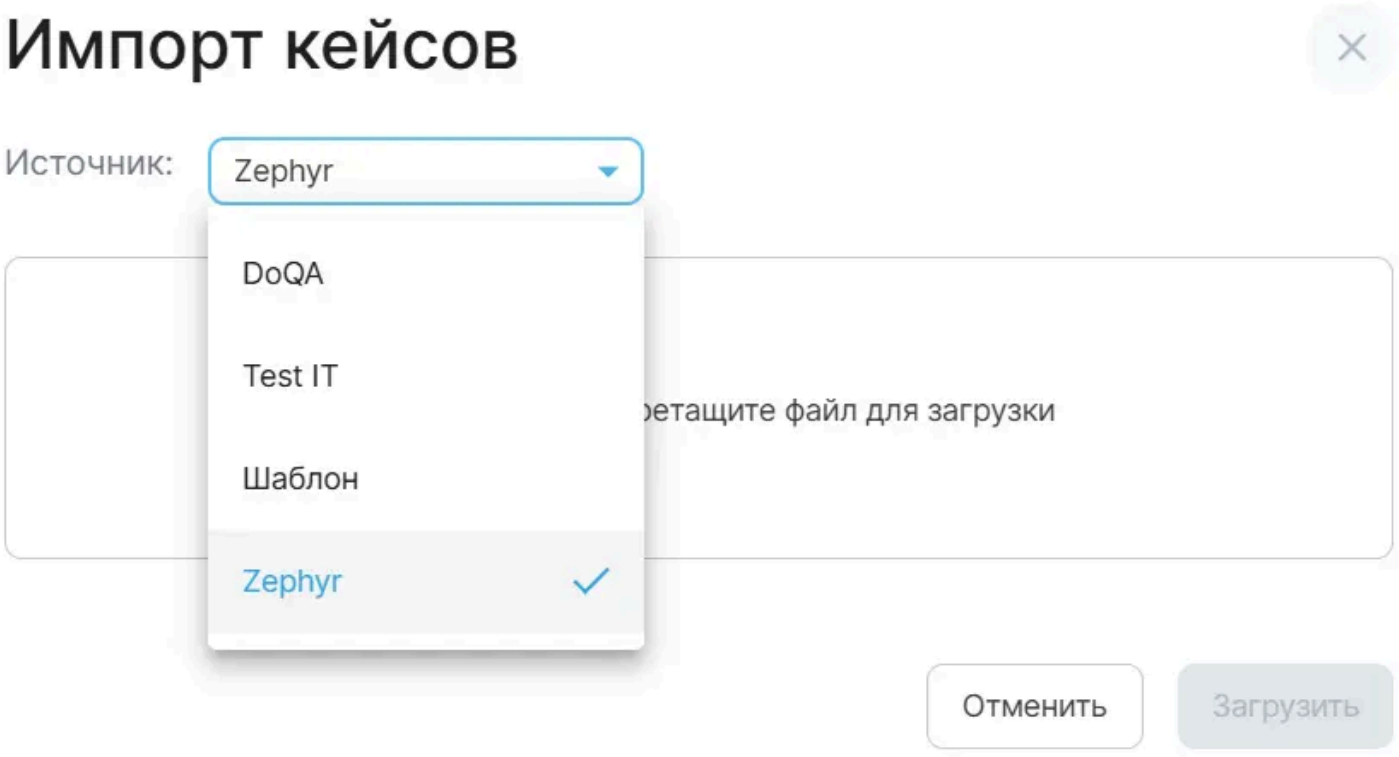


- Новые возможности для администраторов. Больше не нужно обращаться в поддержку, чтобы передать права владельца системы. Теперь администраторы могут изменить владельца системы прямо в карточке пользователя. Кроме того, помимо стандартного приглашения в систему через email, появилась возможность скопировать прямую ссылку-приглашение из карточки пользователя.



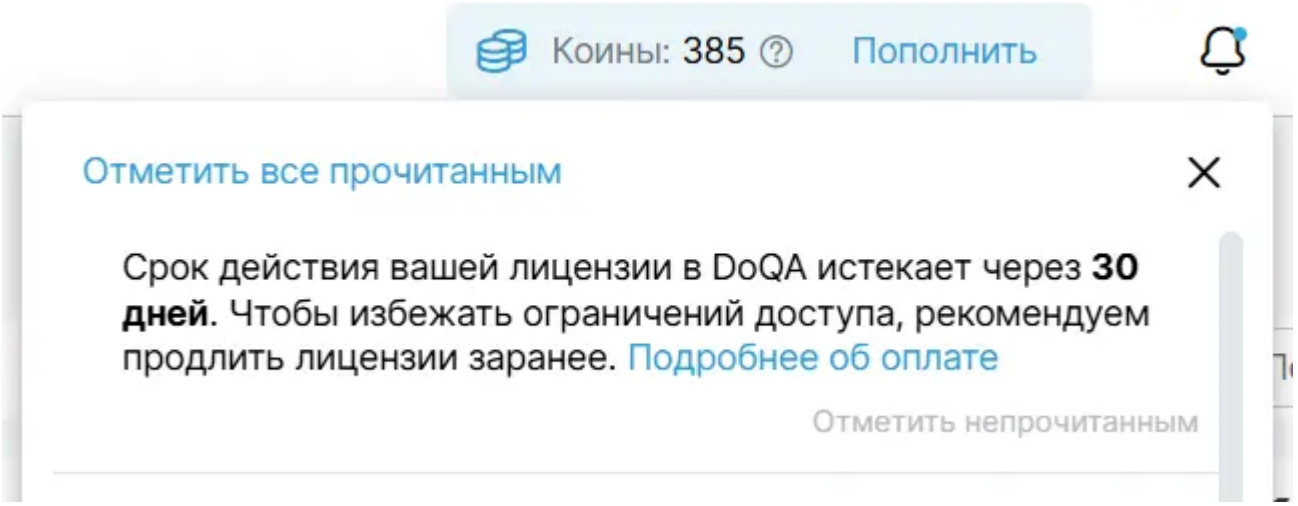
- Кастомная интеграция с Jira Server. Мы добавили универсальный конструктор полей для интеграций. Это значит, что вы видите не только стандартные, но и кастомные поля баг-трекера. Набор подстраивается под каждый проект.

- Импорт тест-кейсов в формате Zephyr. Добавили **новый формат импорта** тест-кейсов из TMS Zephyr, чтобы упростить процесс переноса тестовой документации при переходе в DoQA.



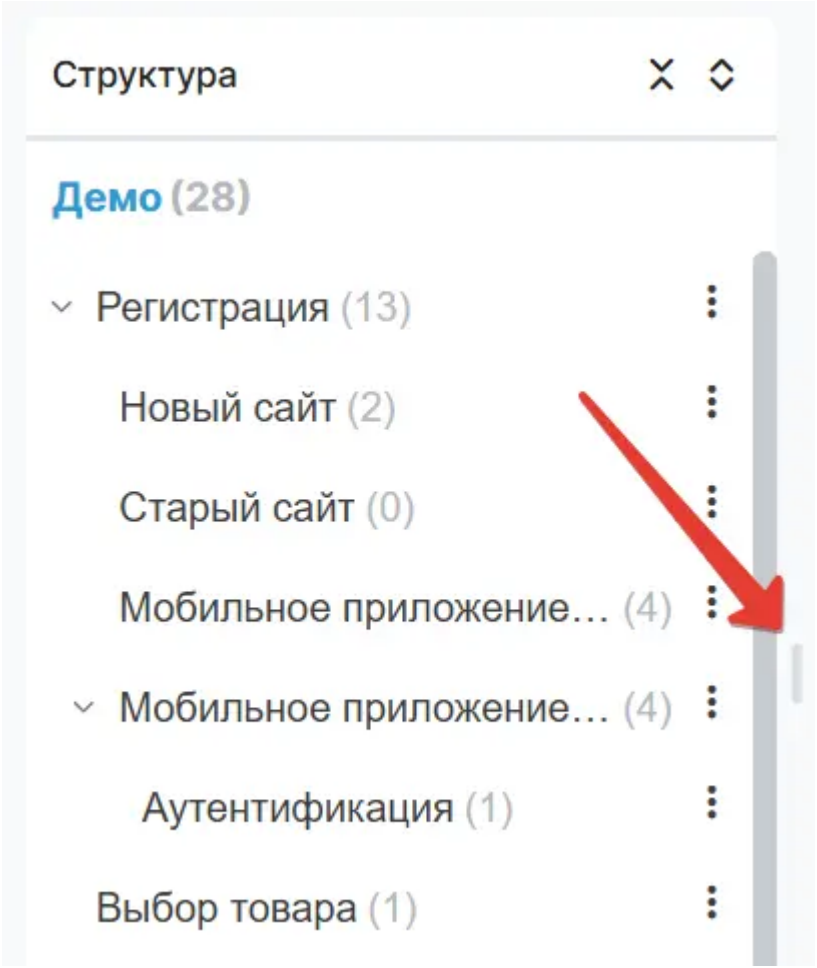
- Развитие функционала уведомлений. В DoQA появилось два новых канала получения уведомлений: внутренние уведомления с колокольчиком и нотификации через Telegram-бота.

Подробнее о настройке уведомлений



Улучшения на основе обратной связи от наших пользователей:

- В списке прогонов теперь отображается число баг-репортов по каждому прогону.
- Усовершенствовали структуру папок. Теперь можно настроить ее ширину, а у каждой папки в структуре появилось меню.



- Порядок отображения кастомных атрибутов в системе теперь можно установить в настройках проекта.

Атрибуты:

Имя	Тип атрибута	Обязательный
Ревьюер	Пользователь	Нет
Версия приложения	Строка	Нет
Плановая дата обновления	Дата	Нет
OS	Вариант на выбор	Нет
Автоматизирован	Вариант на выбор	Нет
Дата обновления тестовых данных	Дата	Нет

Ревьюер

Новый атрибут

- В полный отчет о прогоне выводится описание прогона.

3.5.0 от 31.07.2025

Мы расширили функции DoQA AI, упростили документооборот и сделали интерфейс ещё интуитивнее.

- AI-ревью тест-кейсов. Мы добавили новую функцию в DoQA AI: система проверяет тест-кейсы на полноту и точность, анализирует их и предлагает улучшения и аргументированные правки. Результат можно утвердить или точечно изменить по подсказкам AI.
- Лаконичные e-mail нотификации и новый вид уведомлений. Получайте новое уведомление «Ежедневная сводка по прогону» по состоянию всех незавершенных прогонов и другие нотификации о важных событиях в одном e-mail вместо цепочки писем.
- Мультивыбор чек-боксов. По запросам пользователей мы внедрили мультивыбор чек-боксов. Теперь вы можете выбрать сразу несколько чек-боксов, привычно зажав Shift, чтобы ускорить массовую работу с элементами.
- Учет таймзоны пользователя в разделе с дашбордами. Теперь часовой пояс пользователя учитывается и при отображении событий в аналитических дашбордах.

Улучшения интерфейса:

- Добавлен чек-бокс для массового выбора прогонов.
- Окно с названием баг-репорта теперь адаптируется под длину текста.
- Добавили несколько технических улучшений для более плавной и удобной работы системы.
- Расширили возможности фильтра в редакторе прогона: теперь легче добавить в созданный прогон нужные тест-кейсы или чек-листы.

3.4.0 от 04.07.2025

В этом релизе в DoQA появился AI.

- DoQA AI: умный помощник прямо в интерфейсе. Искусственный интеллект помогает создавать и улучшать тестовую документацию.
- Генерация тест-кейсов. Опишите сценарий и требования — и получите готовый кейс. AI сформирует шаги и структуру и заполнит поля с учётом специфики проекта.

- Генерация названия баг-репорта на основе его содержания. Опишите фактический результат прохождения теста — AI сформирует ёмкий заголовок, который передаёт суть дефекта команде.
- 100 бесплатных коинов на использование DoQA AI. Коины AI — внутренняя валюта DoQA для ИИ-функций. Они списываются за каждый запрос, например, генерацию тест-кейса или имени баг репорта. У каждого своя стоимость. Пользователи триальной версии DoQA, а также платной облачной версии получат по 100 бесплатных коинов на команду, которые можно использовать в течение 14 дней.
- Счета в личном кабинете. Теперь вы можете оформить счёт на оплату лицензий и коинов DoQA AI прямо из интерфейса DoQA, без писем, ожиданий и ручного взаимодействия.
- Новый фильтр для тестовой документации. Добавили возможность фильтрации тест-кейсов и чек-листов по дате создания и последнего редактирования. Это упрощает навигацию в документации и контроль актуальности.
- Email-уведомления о ключевых действиях в системе. Оставайтесь в курсе изменений без ручного мониторинга. Теперь DoQA предупредит вас по e-mail, если: вас назначили исполнителем теста в прогоне или ответственным за тест-кейс или чек-лист; завершился созданный вами прогон; до окончания срока лицензий осталось 30 дней.
- Расширение форматов импорта. Добавили новый формат импорта проектов.
- Новое массовое действие с элементами. Теперь тест-кейсы и чек-листы можно массово перетаскивать в папки вместо того, чтобы сортировать каждый элемент по отдельности. Помимо этого появился новый способ их перемещения через диалоговое окно, в котором можно отметить целевую папку в дереве папок.
- Темная тема в личном кабинете. Теперь вы можете выбрать подходящее оформление не только в основном интерфейсе DoQA, но и в личном кабинете. Переключайтесь между темами в один клик.
- Учет часового пояса пользователя. Просматривайте актуальное время обновления тест-кейсов и чек-листов в соответствии с вашей тайм-зоной.
- Обновления интерфейса. Стало проще отслеживать статусы пользователей системы: «активный», «заблокированный», «приглашенный». Теперь они отображаются прямо в списке пользователей в настройках администратора. У владельца тенанта также появилось свое обозначение.

3.3.0 от 17.05.2025

В DoQA добавлена тёмная тема.

Новый стиль охватывает все элементы интерфейса. Переключайтесь между темами в один клик в настройках профиля.

3.2.0 от 24.04.2025

Мы добавили в DoQA аналитические дашборды, расширили интеграции и улучшили управление тестами.

- Аналитический дашборд в рабочем пространстве. Теперь в каждом пространстве есть собственный аналитический центр. Отслеживайте динамику прохождения тестов, контролируйте активность команды и находите узкие места в процессах.
- Общие шаги в тестах. Создайте библиотеку повторяющихся шагов и вставляйте их в тест-кейсы. Изменения таких шагов автоматически применяются ко всем тестам, в которых они задействованы.
- Новый способ взять тест в работу. В прогонах теперь можно нажать кнопку «Назначить на меня», чтобы быстро стать исполнителем тест-кейса или чек-листа, если он был свободен.
- Прямая и обратная интеграция с GitLab. Теперь данные из DoQA можно отправлять напрямую в GitLab, чтобы команда работала в едином потоке без переключения между инструментами.
- Обратная интеграция с Yandex Tracker, Jira Cloud/Server и Bitrix24. Изменения статуса и приоритета баг-репорта в трекерах автоматически подтягиваются в DoQA и моментально отражаются в системе.
- Продолжаем улучшать интерфейс приложения. Теперь боковое меню с вкладками можно свернуть до иконок, чтобы расширить рабочее пространство.

3.1.0 от 03.03.2025

В этом обновлении мы сделали управление лицензиями удобнее, добавили историю платежей и улучшили контроль за временем выполнения тестов.

- Личный кабинет для управления лицензиями в облачной версии приложения. Теперь все операции с лицензиями собраны в одном месте: оплачивайте, продлевайте и деактивируйте лицензии прямо в личном кабинете. Также доступна история всех платежей, чтобы вы всегда могли отследить финансовые операции.

- Таймер выполнения тест-кейсов и чек-листов. Анализируйте, сколько времени уходит на прохождение тестов. Отсчет времени начнется автоматически при проставлении статуса первому шагу или чеку.
- Возможность повторного прохождения теста в прогоне. На экране прохождения теста в прогоне появилась кнопка «Пройти заново», при нажатии на которую сбрасывается прогресс прохождения выбранного теста и обнуляется время его прохождения.

3.0.0 от 17.02.2025

В новом релизе мы расширили управление документацией: добавили инструменты для гибкой настройки, распределения задач и контроля изменений, а также усилили интеграцию с YouTrack.

Теперь в DoQA доступны:

- Настройка уникальных пользовательских атрибутов для тест-кейсов и чек-листов. Сделайте документацию более детализированной, добавив к ней новые поля с информацией: выберите формат (дата, пользователь, строка или атрибут с вариантами на выбор) и внесите необходимые тестовые данные.
- Назначение ответственного в тест-кейсах и чек-листах. Укажите пользователя, ответственного за тест-кейс или чек-лист. Это позволит команде видеть, кто отвечает за актуализацию тестовых артефактов.
- Обновленная фильтрация и окно массового редактирования документации. Теперь фильтр и окно массового редактирования можно настраивать: вы сами выбираете, какие поля отображать. Также появилась возможность фильтрации и массовых действий по атрибутам и ответственным.
- Версионность тест-кейсов и чек-листов. Теперь, если нужно откатить какие-то изменения в документации, выберите одну из ранних версий и сделайте ее текущей.
- Обратная интеграция с YouTrack. Теперь при изменении статуса или приоритета баг-репорта в YouTrack, соответствующие поля связанного баг-репорта в DoQA обновятся автоматически.
- Уведомления об истечении лицензий. Будьте уверены в непрерывности процессов тестирования: за неделю до истечения срока действия лицензий и в последний день вы получите напоминание о продлении.

2.9.0 от 01.02.2025

Пока команда разработки готовит для вас первое масштабное обновление этого года, мы рады сообщить, что теперь в DoQA появился функционал быстрой оплаты лицензий. В разделе «Лицензии» можно выбрать период продления лицензий, ознакомиться с актуальными ценами и провести оплату по банковской карте.

2.8.0 от 25.12.2024

Рады представить новые возможности для удобной работы с системой в финальном релизе 2024 года.

- Архивирование проектов — теперь вы можете отправлять неактуальные проекты в архив. Такие проекты не будут отображаться на главной странице, но их всегда можно будет восстановить при необходимости.
- Корзина удаленных элементов для защиты от случайного удаления тестовых артефактов. Помещенные в корзину элементы можно восстановить в течение 30 дней.
- Оптимизация прогонов — сортируйте тесты по количеству баг-репортов и комментариев и просматривайте список исполнителей в таблице прогонов.
- Расширение фильтрации элементов по тегам — теперь кроме значения «ИЛИ» вы можете применять фильтр «И» для более точной выборки.
- Импорт проектов из Qase перенесен из публичного API в интерфейс системы.
- Стала доступна обработка отчетов о прогонах автотестов в формате Allure.

2.7.0 от 05.12.2024

Упрощаем управление дефектами, улучшаем плеер прогонов и расширяем возможности тест-кейсов и чек-листов. В свежем релизе:

- отдельная вкладка «Дефекты», в которой отображаются все баг-репорты со ссылками на прогоны — теперь отчеты о дефектах со всего рабочего пространства собраны в одном месте;
- обновленный редактор тест-кейсов и чек-листов — увеличено количество шагов и чеков до 300 штук, появился счетчик чеков и шагов, а работа в редакторе стала более

интуитивно понятной;

- обновленный вид комментариев — интерфейс стал более приятным и информативным, а в самих комментариях появились перекрестные ссылки на соответствующие им тест-кейсы, чек-листы и прогоны;
- прогресс-бар и счетчики заведенных баг-репортов и комментариев у каждого теста в плеере прогона;
- возможность открывать страницу с баг-репортом по ссылке в отдельной вкладке — делиться данными стало еще удобней.

2.6.0 от 17.10.2024

Расширяем возможности интеграции и аналитики. Встраивать DoQA в процессы управления тестированием стало еще удобней.

- Интеграция с Битрикс24 и Redmine — данные из DoQA теперь можно отправлять напрямую в эти системы.
- Полный отчет о прогоне — теперь отчет содержит не только информацию по результатам тестов, но и сами тесты.
- Новые метрики по исполнителям в отчетах.
- Настраиваемые конфигурации для прогонов, которые позволяют указать, на каком окружении и в какой среде необходимо проводить тестирование.
- Возможность удаления прогонов из автотестов.

2.5.0 от 05.09.2024

Это обновление оптимизирует работу с плеером, расширяет форматы импорта документации и не только.

- Плеер прогонов стал универсальным. Теперь в одном прогоне можно запускать и тест-кейсы, и чек-листы.
- Плеер изменился внешне, а механика работы и редактирования стала более комфортной и понятной.
- В прогонах теперь можно назначать исполнителей для тестов. Если исполнитель не назначен, то тест может пройти любой тестировщик.

- У шагов и чеков появились статусы для конкретизации результатов тестирования, а также возможность указывать баги и комментарии.
- В кейсах появилось поле «Тестовые данные» для внесения сопутствующей информации. Так тест-кейсы станут более подробными.
- Стало возможно импортировать проекты в DoQA из Qase в формате JSON с помощью публичного API. Нам неважно, в какой стране вы находитесь. Пользоваться DoQA можно из любой локации.
- Поиск необходимой информации о DoQA стал проще: руководство пользователя и инструкции переехали на новый портал документации.

2.4.0 от 04.07.2024

Система начала движение к статусу универсального инструмента по работе с любыми видами тестов, и это обновление — первый шаг к будущим масштабным изменениям.

- Автотесты — теперь DoQA не только упрощает и оптимизирует ручное тестирование, но и позволяет интегрировать результаты автотестов, чтобы вся отчетность удобно просматривалась в одном инструменте. Отправляйте новые отчеты по автотестированию прямо в DoQA или загрузите в TMS уже существующие.
- Интеграция с YouTrack — одна из самых запрашиваемых фич. Теперь данные из DoQA можно отправлять напрямую в систему управления проектами YouTrack, упрощая работу с этим инструментом.

2.3.5 от 03.06.2024

Мы продолжаем расширять интеграции DoQA с внешними сервисами, пока в разработке находится глобальное обновление раздела прогонов. В новой версии появилась одна из самых запрашиваемых фич — интеграция с системой управления проектами Yandex Tracker.

- Баги теперь можно отправлять в Yandex Tracker напрямую из DoQA.
- Изменилась работа с тегами внутри редакторов тестов. К тестам можно добавлять любой из уже созданных тегов, а логика редактирования и добавления тегов переехала в обновленный редактор.
- Редактор тегов стал более компактный, наглядный и понятный.

- Таблицы кейсов и чек-листов больше не схлопываются при изменениях и удалении элементов.
- Работать с диалоговыми окнами стало удобнее. Теперь их можно перемещать, чтобы лучше видеть нужную информацию.

2.3.0 от 25.04.2024

В DoQA появилась первая интеграция с внешним сервисом — Jira. Теперь баг-репорты можно отправлять в Jira напрямую из интерфейса DoQA с помощью обновленной формы.

- Появился раздел интеграций, в котором можно подключить Jira Cloud и Jira Server.
- Для каждого проекта можно настроить свои параметры интеграции.
- Форма баг-репорта обновилась, и оформление бага стало проще. DoQA сама собирает данные из кейсов и чек-листов.
- Стало возможным массовое редактирование кейсов и чек-листов: смена статуса, приоритета, работа с тегами.
- Расширен функционал работы с тегами. Теперь их можно удалять и очищать при массовом редактировании.
- При создании кейсов, чек-листов и папок таблица остается в развернутом виде.

2.2.7 от 19.03.2024

В этом релизе мы расширили возможности редактора и добавили другие обновления, которые делают работу в DoQA еще более удобной:

- В кейсах и чек-листах появился раздел вложений. Теперь к тестам можно прикреплять файлы;
- В редакторе теперь можно работать с кодом и выделять текст цветом;
- Редактор сохраняет заданную ширину после обновления страницы;
- Таблица элементов остается активной при использовании редактора и позволяет быстро переключаться между кейсами и чек-листами;
- В плеере прогона появилось обозначение пути до кейса/чек-листа;
- Стало возможным возобновлять прохождение прогона даже после его завершения.

2.2.0 от 31.01.2024

Встречайте долгожданное обновление структуры таблиц и системы работы с папками.

- Мы отказались от пагинации. Теперь работа с элементами доступна в едином структурированном списке.
- Дерево папок переехало из бокового меню в таблицу для удобного переключения вкладок.
- Появились функции «свернуть» и «развернуть» папки.
- В дереве папок сохраняются состояния после перезагрузки и обновления страницы: если папки были открытыми, то такими и останутся.
- Таблицы стали более наглядными и теперь отображают вложенность.
- Добавлено окно подтверждения удаления кейсов и чек-листов.
- Порядок создания кейсов и чек-листов стал удобней. Теперь их можно создавать прямо внутри нужной папки.
- Увеличена длина имени папки.
- Добавлена возможность использовать спецсимволы в пароле.

2.1.7 от 21.12.2023

Пока в DoQA готовится масштабное обновление структуры таблиц, мы выпустили несколько улучшений, которые делают работу с TMS более удобной:

- Изображения в документации разворачиваются по клику на весь экран;
- К чек-листам и кейсам можно добавлять статус «Ревью»;
- Дата обновления кейса/чек-листа теперь видна в таблице. Стало проще отслеживать обновление документации;
- Доработан импорт для получения дерева папок из секций.

2.1.0 от 26.10.2023

Встречайте первый релиз этой осени: мы доработали функционал системы и немного затронули дизайн в преддверии будущего масштабного обновления. Что нового в DoQA:

- обновился внешний вид таблиц в разделах;
- обновился дизайн и механика работы фильтра. Теперь кейсы, чек-листы и прогоны можно фильтровать по большему количеству параметров;
- добавлена возможность фильтрации по папкам;
- стало возможно удаление прогонов;
- стало возможно добавление элементов в прогон напрямую из раздела или папки;
- появились уведомления о необходимости наличия лицензии при редактировании прав пользователя.

2.0.0 от 31.08.2023

DoQA обновилась. Теперь наша TMS доступна и в серверной версии.

Также в этом релизе DoQA дополнилась:

- возможностью импортировать тест-кейсы и чек-листы в нескольких форматах;
- защитой от случайного переключения и закрытия редактора с уведомлением о несохраненных изменениях;
- сохранением тестовой документации по горячим клавишам для более продуктивной и быстрой работы;
- обновлением дизайна интерфейса... ...и другими инфраструктурными улучшениями.

1.7.4 от 26.07.2023

Пока готовится новое серверное решение DoQA, делимся промежуточным релизом, в котором мы обновили дизайн экранов, а также провели фикс и оптимизацию работы с тестовой документацией. В новом дизайне:

- главная страница;
- профиль пользователя;
- страница авторизации и регистрации;
- форма обратной связи, восстановления пароля и т.д. Теперь работа в DoQA стала удобнее и стабильнее.

1.7.1 от 20.06.2023

DoQA продолжается меняться и обрастать актуальным функционалом. Мы подготовили новый релиз, который делает работу с TMS более быстрой и комфортной.

- Изменен дизайн интерфейса. Мы запустили масштабную работу по редизайну DoQA, и первые результаты вы можете заметить уже сейчас: обновилась общая стилистика сервиса, хедер, футер, кнопки и множество более мелких деталей, которые влияют на положительный опыт взаимодействия с DoQA. Систему ждут большие перемены. Скоро TMS станет стильной и современной.
- Обновлены настройки администратора. Стало удобнее настраивать доступы, приглашать новых пользователей и управлять текущими.
- Разделены роли администратора пользователей и администратора проекта. Теперь можно назначать их по отдельности или дать обе роли одному пользователю.
- Добавлена возможность редактировать доступы внутри проекта. Администратор проекта сам набирает команду и может управлять доступами без необходимости обращаться к администратору пользователей.
- Обновлено меню пользователя. Мы добавили в шапку пункты с руководством и нашими контактами, а также свитчер.
- Оптимизирована UX-составляющая интерфейса.

1.6.0 от 19.01.2023

Функционал, которым вы могли пользоваться только в кейсах, теперь доступен в чек-листах и прогонах. Посвящаем этот релиз равномерности. Вот какие обновления вас ждут:

- В чек-листах и прогонах появились история изменений, теги и фильтр элементов в таблицах. Ориентироваться в проектах стало удобнее.
- Расширен функционал работы с папками. Теперь можно создавать вложенные папки и удалять из них ненужные элементы.
- Оптимизированы запросы для ускорения загрузки страниц.
- Много инфраструктурных и UI/UX улучшений.

1.5.0 от 15.12.2022

В этом обновлении появилось больше инструментов для просмотра статистики выполненной работы и отслеживания процессов.

В новую версию добавлены:

- Возможность создать отчет о прохождении тест-ранов — с полной статистикой и диаграммами для удобного восприятия.
- Гистограмма прохождения чек-листа — прогресс теперь можно отслеживать визуально.
- Вкладка для комментариев и найденных багов в кейсах/чек-листах и прогонах — все важные моменты можно будет зафиксировать в специальной вкладке, чтобы не возвращаться отдельно к каждому кейсу.
- История запуска чек-листа в составе прогона.
- Подтверждение окончания прохождения рана — для возможности перепроверить свою работу до того, как в нее больше нельзя будет вносить изменения.
- История изменений в редакторе кейсов — видеть кто, когда и какие правки вносил в кейсы.
- Гиперссылки по ID прогона из истории запуска — для того, чтобы можно было сразу вернуться в нужный прогон с помощью одного клика.
- Переключение локализации Ru/En — для тех, кто работает на английском языке.

1.4.1 от 10.11.2022

Новая порция улучшений в нашей TMS. Оценить их можно уже с сегодняшнего дня.

- Добавление картинок к кейсам и чек-листам.
- Отображение картинок в тест-ранах. Изображения подстраиваются под размер плеера.
- Повторное приглашение пользователя, не прошедшего регистрацию — на случай, если первое письмо попало в спам или стало неактуальным.
- Стиль редактора чек-листов, единый с кейсами и тест-ранами.
- Много инфраструктурных и UI/UX улучшений.

1.4.0 от 17.10.2022

- Плеер чек-листов.
- Возможность прикрепить ссылку на баг в плеере чек-листов.
- Создание тест-рана из чек-листов.
- Импорт чек-листов из xlsx. Актуальный функционал для тех, кто привык вести тестовую документацию в Excel или Google Docs.
- Удаление приглашенных пользователей, еще не прошедших регистрацию в DoQA.
- Кнопка «Очистить строку поиска».
- Гиперссылки из тест-рана на редактор тест-кейса — чтобы оперативно вносить необходимые изменения.
- История изменений версий DoQA — теперь вы можете следить за тем, как меняется и развивается наша TMS.

1.3.0 от 31.08.2022

В новом релизе DoQA мы доработали интерфейс и добавили несколько изменений в функционал.

- Система тегов в тест-кейсах — для ускорения поиска нужной информации.
- Фильтрация тест-кейсов по статусу, создателю и тегам.
- Создание тест-ранов из папок с кейсами.
- Индексация скопированных элементов — теперь вы можете отличать оригинальный элемент (тест-кейс, чек-лист и т.д.) от его копии по индексу, который появится после названия элемента.
- Локализация электронных писем — все системные письма от DoQA переведены на русский язык.
- Изменение ширины редактора. Вы можете регулировать размер рабочего пространства в TMS под себя.
- Подсказки к полям в формах регистрации и авторизации.
- Форма обратной связи — можно поделиться мыслями, предложениями и замечаниями.

1.2.0 от 25.07.2022

Тестировщики нашей команды регулярно используют DoQA на своих проектах, благодаря чему мы всегда знаем, каких фич им не хватает для увеличения эффективности работы. И в этом месяце по мотивам обратной связи пользователей мы добавили в сервис:

- Вложенность чеков до 8 уровня.
- Права и доступ к чек-листам. Теперь вы сможете определять, кто из пользователей может или не может редактировать определенные чек-листы.
- Копирование и удаление чек-листов.
- Закрытие модальных окон по Esc&Overlay.
- Подтверждение для сброса пароля.

1.1.0 от 06.06.2022

Это первое обновление DoQA. Мы начинаем реализовывать наши планы и внедрять востребованный функционал в TMS.

Что появилось нового?

- История прохождения кейса в составе тест-рана.
- Фильтрация истории прохождения кейса по исполнителю и результатам. Удобно в условиях, когда данных слишком много, чтобы не искать информацию вручную.
- Настройка прав и доступов на просмотр, редактирование и администрирование. Теперь каждому члену команды можно присвоить свою роль в TMS. Один человек может иметь разные типы прав на разных проектах.
- Блокировка пользователей — для случаев, когда пользователю необходимо ограничить доступ к проектам по какой-либо причине, но без его полного удаления из системы.
- Вложение элементов в папки — для структурирования кейсов и поддержания порядка в рабочем пространстве.
- Чек-листы.
- Аватарки проектов и пользователей.
- Русская локализация — теперь пользоваться нашей TMS можно не только на английском языке.

Релизы серверной версии DoQA

DoQA 4.3-box Titanium от 25.09.2026

Версия 4.3 целиком посвящена автоматизированному тестированию. В DoQA появился полноценный раздел **«Автотесты»** с собственным каталогом, запуск пайплайнов в трёх CI-системах, критерии приёмки прогона Quality Gate, работа с нестабильными тестами и автоматический разбор причин падений. Ручные и автоматизированные проверки по-прежнему находятся в одном прогоне. Теперь у автотеста в нём есть история попыток, шаги, вложения и логи. Не обошли и ручное тестирование: в продукте появился глобальный поиск по всей документации, авторaспределение исполнителей, прогноз времени прохождения прогона и выгрузка отчётов в Excel и CSV.

Раздел «Автотесты»

- Все автотесты пространства собраны в одном каталоге с деревом структуры, фильтрами по ветке, тегам, источнику и карантину.
- У каждого автотеста есть состояние: здоровый, нестабильный, сломанный, в карантине, отключённый, восстановленный.
- Боковая панель автотеста показывает историю прохождений, стабильность, связанные тест-кейсы.
- Смотрите каталог «по источнику»: из какого репозитория и пайплайна приехал каждый автотест.
- Запускайте выбранные автотесты прямо из каталога.
- Настраивайте состав и ширину столбцов каталога, назначайте автотестам собственные теги.
- Создавайте ручной тест-кейс на основе автотеста, у которого ещё нет связей.

Подробнее о каталоге автотестов

Запуск автотестов в CI/CD

- Подключайте три CI-системы: GitLab, Jenkins, TeamCity.

- Заводите CI-подключения на уровне пространства и держите несколько подключений одного типа, например два сервера GitLab.
- Запускайте только выбранные автотесты: DoQA собирает нативный фильтр раннера. Если набор фильтром не выражается, прогон идёт целиком, а не молча урезается.
- Один прогон может держать несколько пайплайнов сразу: разнородный набор разбивается на группы «интеграция, проект, ветка», а прогон закрывается, когда завершился последний из них.
- Формы запуска автотеста предзаполняются последней использованной связкой «CI-система, проект, ветка».
- Задавайте порядок прохождения автотестов в наборе перед стартом при запуске автотестов из каталога.

Подробнее о запуске из CI/CD

Расписание запусков

- Запускайте автотесты по расписанию, без изменений в конфигурации CI.
- Задавайте название будущих прогонов, часовой пояс, окружение и поведение на случай, если предыдущий прогон расписания ещё не завершился.
- Переопределяйте для запусков по расписанию критерии Quality Gate и политику авто-повторов.
- Журнал срабатываний показывает каждый запуск: созданный прогон или причину, по которой он не состоялся. После серии сбоев расписание отключается само.

Подробнее о расписаниях

Quality Gate — критерии приёмки прогона

- Опишите, при каких условиях прогон считается пройденным: минимальная доля успешных тестов, предельное число падений, минимальное число тестов, максимальная длительность, обязательные окружения.
- Запретите «позеленение» прогона за счёт повторов и включите ранний останов.
- Критерии задаются в настройках пространства и переопределяются для отдельного прогона или расписания.
- Вердикт виден прямо в отчёте о прогоне, а команда `doqactl gate` возвращает его в пайплайн кодом выхода.

Подробнее о Quality Gate

Нестабильные тесты, карантин и авто-повтор

- DoQA сама находит нестабильные автотесты по истории прогонов: доля падений в окне, разный исход на одном коммите, «прошёл со второй попытки».
- Отправляйте нестабильный тест в карантин: он продолжает выполняться, но не роняет Quality Gate и пайплайн. Карантин снимается автоматически, когда тест успокоился.
- Настраивайте пороги нестабильности и срок снятия пометки.
- Настраивайте авто-повтор упавших автотестов: число попыток, задержка, теги для повтора и исключения, поведение пайплайна, если все попытки упали.

Кластеры ошибок и известные проблемы

- Падения с одинаковой причиной автоматически собираются в один кластер по сигнатуре ошибки.
- Раскладывайте падения по классам «Продукт», «Автотест» и «Окружение» собственными правилами. Шаблон проверяется прямо в форме.
- Заводите баг из кластера или привязывайте существующий. Кластер можно отметить известной проблемой: она перестаёт шуметь в следующих прогонах и открывается заново при регрессе.

Подробнее о результатах и разборе падений

Прогон и его плеер

- Перепроходите упавший автотест: автоматически новым запуском или вручную по шагам последнего автоматического прохождения. Каждая попытка сохраняется отдельно, прежний результат не затирается.
- Перезапускайте выбранные строки завершённого прогона пакетом.
- Вкладка «Пайплайны и джобы»: список шагов пайплайна со статусами, остановка и перезапуск отдельной джобы, номер запуска в CI-системе и ссылка на пайплайн.
- Результат автотеста показывает дерево шагов и фикстур, вложения (скриншоты, видео, трассировки и логи) и параметры запуска. Вложения доступны по каждой попытке.
- Бейдж «Не применено» в шапке прогона показывает, сколько результатов не были засчитаны и почему: устаревший результат, тест вне плана, тест не выполнялся, вложения

не приняты.

- Смотрите, куда ушло время одного запуска, от команды в CI до приезда отчёта.

Дашборд «Автотесты» и аналитика

- Новая вкладка «Автотесты» в дашбордах пространства: доля автоматизации, здоровье парка тестов, доля успешных прохождений, нестабильность, длительность, топ падающих тестов, разбивка по веткам и окружениям, экономия времени CI и топ кластеров ошибок.
- Карта покрытия и лента прогонов во времени показывают, что шло параллельно и чем закончилось.
- Фильтруйте дашборд по ветке, тегу и периоду.

Адаптеры, CLI и плагины IDE

- Адаптеры для JUnit 5, JUnit 4, TestNG и PyTest отправляют результаты в DoQA прямо из прогона тестов, без файлового отчёта.
- Закрепляйте за тестом постоянный идентификатор аннотацией `@DoqaId`, а меткой `@DoqaCreateManualCase` просите DoQA завести для теста ручной кейс.
- Утилита командной строки умеет больше: `upload` — загрузка готового отчёта JUnit XML или Allure, `run` — прогон тестов с отправкой результата одной командой, `watch` — слежение за прогоном, `gate` — вердикт Quality Gate для шага пайплайна, `sync-ids` — простановка идентификаторов в исходниках тестов, `login` — сохранение адреса, токена и пространства.
- Плагины для IntelliJ IDEA и VS Code: рядом с каждым тестом видно, привязан ли он к кейсу, кейс можно найти и привязать, проставить идентификатор для теста или всего файла и посмотреть каталог автотестов пространства, не выходя из редактора.
- Параметризованные тесты больше не схлопываются в одну строку. У каждого набора данных свой результат. Теги теста доезжают в DoQA как метки.

Подробнее об **адаптерах** и **плагинах**

Глобальный поиск

- Строка поиска в шапке приложения ищет по тест-кейсам, чек-листам и общим шагам всего продукта.

- Отдельная страница результатов с фильтрами и подсветкой совпадений в найденном тексте.

Прогоны и ручное тестирование

- Авторaспределение исполнителей: тесты распределяются между выбранными участниками равномерно по среднему, медианному или плановому времени прохождения.
- Прогноз времени прохождения прогона по накопленной статистике.
- Умное копирование прогона: можно скопировать в новый прогон, например, только упавшие тесты.
- Выгрузка отчёта о прогоне в Excel и CSV.
- Единый диалог добавления тестов в прогон: кейсы, чек-листы и автотесты в одном окне с общим поиском, деревом папок и фильтрами по тегам.
- Ссылка на прогон запоминает открытую вкладку. Можно переслать коллеге ссылку ровно на то, что вы смотрите.

Подробнее о прогонах

Кейсы, чек-листы и требования

- Настраивайте состав выгрузки тест-кейсов и чек-листов и выбирайте формат файла.
- Связывайте требование напрямую с автотестом, без тест-кейса-посредника. Автотест участвует в расчёте покрытия требования, а в карточке требования появилась панель связанных автотестов.
- У тест-кейса появился статус автоматизации и признак происхождения: видно, что кейс создан из автотеста.
- Синхронизация требований из Confluence: можно взять пространство целиком, а не только поддерево от выбранной страницы. Повторная синхронизация большого пространства больше не обрывается на середине.

Уведомления

Новые события в настройках уведомлений: автотест признан нестабильным, сбой запуска автотестов по расписанию.

Подробнее об уведомлениях

Интерфейс

Настройки пространства собраны на одной странице с разделами. Обновлены страница «Проекты» и боковое меню, футер переехал в выпадающее меню. В меню пользователя появился пункт «О DoQA».

DoQA 4.2-box Niccolum от 14.08.2026

Модуль управления требованиями

- Настройте синхронизацию между DoQA и трекером, чтобы получать требования в TMS.
- Связывайте требования с тест-кейсами и чек-листами для контроля покрытия требований тестами.
- Генерируйте тест-кейсы и чек-листы с помощью ИИ на основе содержания требования.
- Получайте уведомления об изменении требований и необходимости актуализировать связанные тесты.
- Используйте один из двух режимов отображения:
  - дерево требований;
  - матрица покрытия.
- Экспортируйте матрицу покрытия в удобном формате.

Улучшения в плеере прогона

- Перезапускайте автотест прямо из результатов прохождения: «Пройти заново → Автоматически».
- Для каждого теста в прогоне отображаются:
  - метрика надежности;
  - среднее время выполнения за последние 10 запусков;
  - история трех последних результатов.

ИИ

- Подключайте локальные LLM и любые модели с OpenAI-совместимым API.
- Генерируйте наборы тест-кейсов на основе требования из трекера.
- Настраивайте промпт генерации набора тест-кейсов, чтобы адаптировать результаты под процессы вашей команды.

Редизайн главной страницы

- Выберите один из двух вариантов отображения проектов и пространств:
  - классическое дерево;
  - альтернативный просторный режим.
- Настройте порядок отображения проектов и пространств. Администраторы могут задать порядок по умолчанию для всех пользователей.
- Для пространств добавлено поле «Описание».
- Поиск теперь работает не только по названиям проектов, но и по названиям пространств и описаниям.

Кастомизация таблиц

- Настройте отображение таблиц:
  - во вкладках «Кейсы», «Чек-листы» и «Прогоны»;
  - в плеере прогона.
- Изменяйте:
  - набор столбцов;
  - порядок столбцов;
  - ширину столбцов.
- Сохраняйте настройки в пресеты и быстро переключайтесь между ними.

Множественные параметры в тестах

- Добавляйте в тест-кейсы и чек-листы несколько параметров: локальных и глобальных.
- В отдельной вкладке в редакторе ТК или ЧЛ можно выбрать способ формирования комбинаций параметров: все комбинации, каждый выбор, линейное сопоставление, попарное тестирование (pairwise).

Подробнее о параметризации

Плановое время прохождения тестов

- Указывайте плановое время выполнения тест-кейса или чек-листа.

Комментарии из прогонов в редакторах ТК и ЧЛ

- Комментарии, оставленные при выполнении тестов в прогонах, теперь доступны в редакторах тест-кейсов и чек-листов на отдельной вкладке.

Обновленный Public API

- Расширен список методов Public API.
- В профиле пользователя появилась возможность создать постоянный персональный API-токен.

Подробнее о Public API

Корпоративная авторизация через SSO/LDAP

- Единая форма входа. При включении SSO экран авторизации показывает корпоративный вход. В строгом режиме (enforce) доступен только SSO, а локальный пароль отключён.
- Привязка и связывание. Новые пользователи из корпоративного каталога создаются автоматически (JIT). Существующие учётки связываются через совпадение почты.

Подробнее о корпоративной авторизации

DoQA 4.1-box Cuprum от 03.06.2026

DoQA AI

- Теперь в серверной версии DoQA доступны ИИ-функции: генерация тест-кейсов и чек-листов, заголовков баг-репортов, а также ревью тест-кейсов.

- Пользователи могут подключить провайдеры OpenAI и Яндекс, а также локальные провайдеры и выбрать любые доступные модели для генерации контента.
- Пользовательская настройка ИИ-промптов. Промпты для всех ИИ функций DoQA отображаются в настройках администратора и могут быть изменены пользователями под свои процессы и стандарты тестирования.
- Потоковая обработка ИИ запросов. Ожидание результатов генерации составляет от пары секунд, в зависимости от размера запроса.

Комментирование тест-кейсов и чек-листов

- Панель комментирования открывается в редакторе и позволяет оставлять одиночные комментарии или открывать обсуждения, а также упоминать других пользователей DoQA. Автор получит уведомление, когда на его комментарий ответят.
- Отработанный комментарий можно отметить как «решенный» и оставить для истории, либо удалить.
- Функционал комментирования доступен также пользователям без лицензий. Теперь бесплатные «наблюдатели» становятся полноценными участниками коммуникации в DoQA. Приглашайте аналитиков, разработчиков и дизайнеров вашей команды в DoQA для совместного ревью тестовой документации.

Подробнее о комментариях

Параметризация тестов

- Добавьте параметр с его возможными значениями в тест-кейс или чек-лист, а также укажите в тексте вхождения этого параметра. При добавлении теста в прогон под каждое значение параметра будет создан отдельный слепок теста.

Подробнее о параметризации

Улучшение пользовательского опыта

Мгновенные сообщения об изменении списка тестов в прогоне теперь можно отключить в профиле пользователя, если получение данных уведомлений не требуется.

DoQA 4.0-box Carboneum от 23.04.2026

Версия DoQA 4.0 открывает новый этап развития системы. В этом релизе появилась поддержка автоматизированного тестирования и интеграция с CI/CD-процессами.

Теперь ручные и автоматизированные проверки можно объединить в одном прогоне, запускать автотесты напрямую из интерфейса DoQA, связывать их с тест-кейсами и анализировать результаты в едином отчете.

Кроме того, обновлён интерфейс плеера прогонов и добавлены новые возможности при прохождении тестов в прогоне.

Поддержка автоматизированного тестирования

DoQA теперь позволяет работать с автотестами как с полноценной частью тестового процесса.

- **Связка автотестов с тест-кейсами**. Автотесты могут автоматически связываться с тест-кейсами по Allure ID. Это позволяет объединить ручную тестовую документацию и автоматизированные проверки в единую структуру.
- **Отображение автотестов в прогоне**. Результаты автотестов теперь отображаются непосредственно в прогоне и могут быть объединены в единый список с ручными тест-кейсами и чек-листами. К пройденным автотестам можно оставлять баг-репорты и комментарии, а также перепроходить их вручную.

Ранее загруженные в DoQA отчеты о прохождении автотестов мигрируют в «Прогоны» и будут помещены в отдельную папку.

Интеграция с GitLab CI/CD

DoQA теперь может напрямую взаимодействовать с пайплайнами CI/CD. Создайте интеграцию между DoQA и GitLab CI/CD, после чего выберите проекты из репозитория, в которых планируете запускать автотесты.

- **Запуск пайплайнов из DoQA**. Вы можете создать новый прогон через запуск пайплайна в CI/CD непосредственно из интерфейса DoQA. По мере выполнения автотестов результаты будут загружаться в созданный прогон.

При запуске пайплайна из DoQA можно выбрать один из доступных проектов из CI/CD, а также ветку.

- **Выборочный запуск автотестов**. Автотесты, связанные с тест-кейсами через Allure ID, можно запускать прямо из прогона, отметив нужные элементы.

Импорт результатов автотестов

В интерфейс добавлен функционал создания прогона через загрузку отчётов о прогонах автотестов в форматах JUnit XML и Allure. Также результаты прохождения автотестов можно загружать в DoQA через утилиту doqa-cli.

Обновлённый интерфейс для работы с прогонами

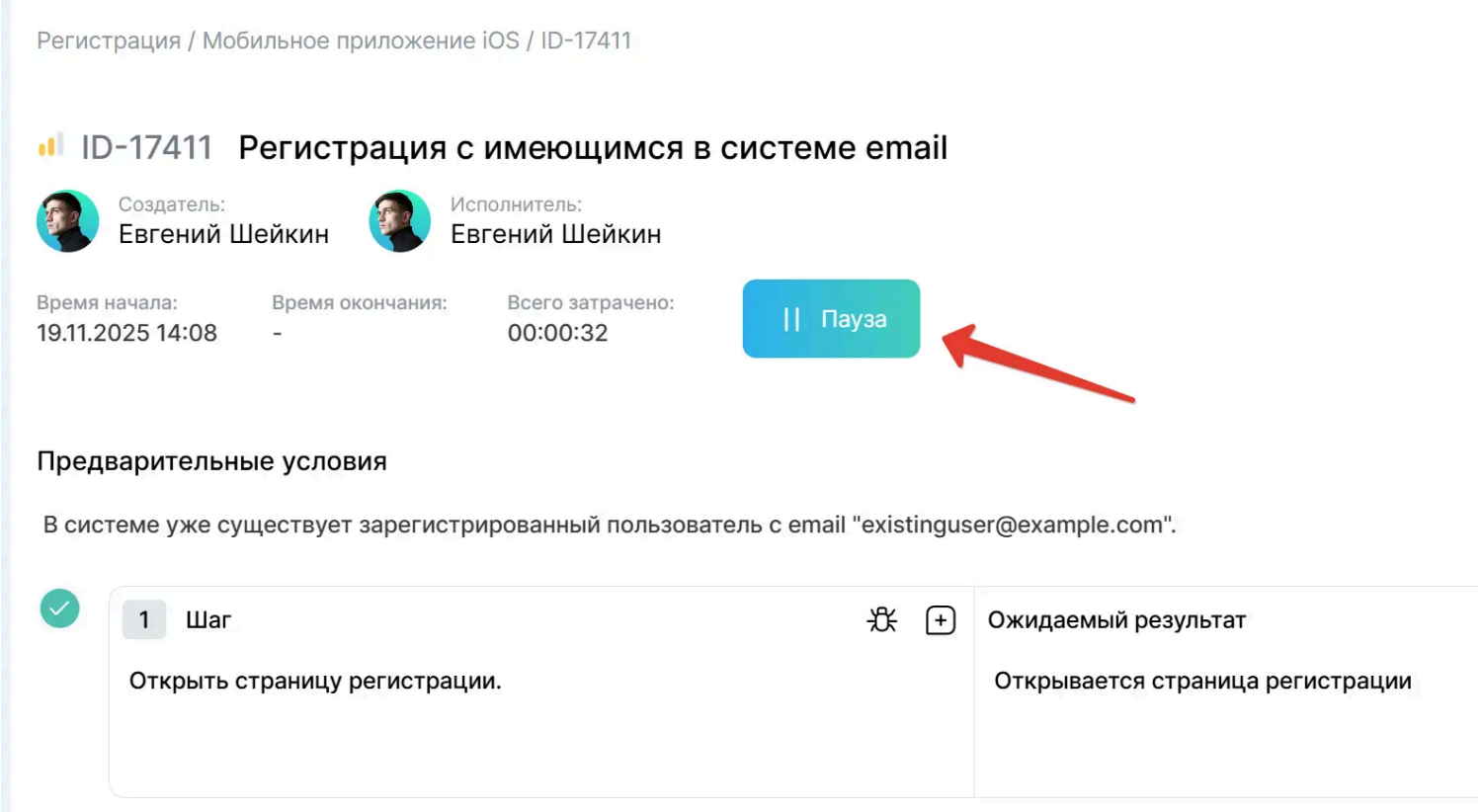
В версии 4.0 переработан интерфейс работы с прогоном, чтобы ускорить выполнение тестов и упростить навигацию.

- **Новое окно создания прогона**. Создание прогона теперь выполняется через компактное модальное окно без перехода в отдельный редактор. Помимо стандартного создания прогона можно выбрать альтернативные варианты: загрузку отчетов о прогонах автотестов и запуск автотестов в CI/CD.
- **Редактирование прогона во время выполнения**. Добавлять тест-кейсы и менять их порядок можно прямо в плеере прогона, не переходя в отдельный редактор. Назначать исполнителей на тесты в прогоне теперь можно массово на основе выбранных фильтров.
- **Обновлённый экран прохождения теста**. Мы переработали экран прохождения тестов в прогоне, сделав его более структурированным и увеличив полезное пространство.
- **История прохождений**. Интерфейс выполнения тестов теперь разделён на две вкладки: прохождение и история прохождения. Вы можете увидеть все попытки перепрохождения теста в прогоне.
- **Папки в прогонах**. На основе обратной связи от наших пользователей мы реализовали папки в прогонах для более удобного хранения документации.

3.7.0-box от 17.12.2025

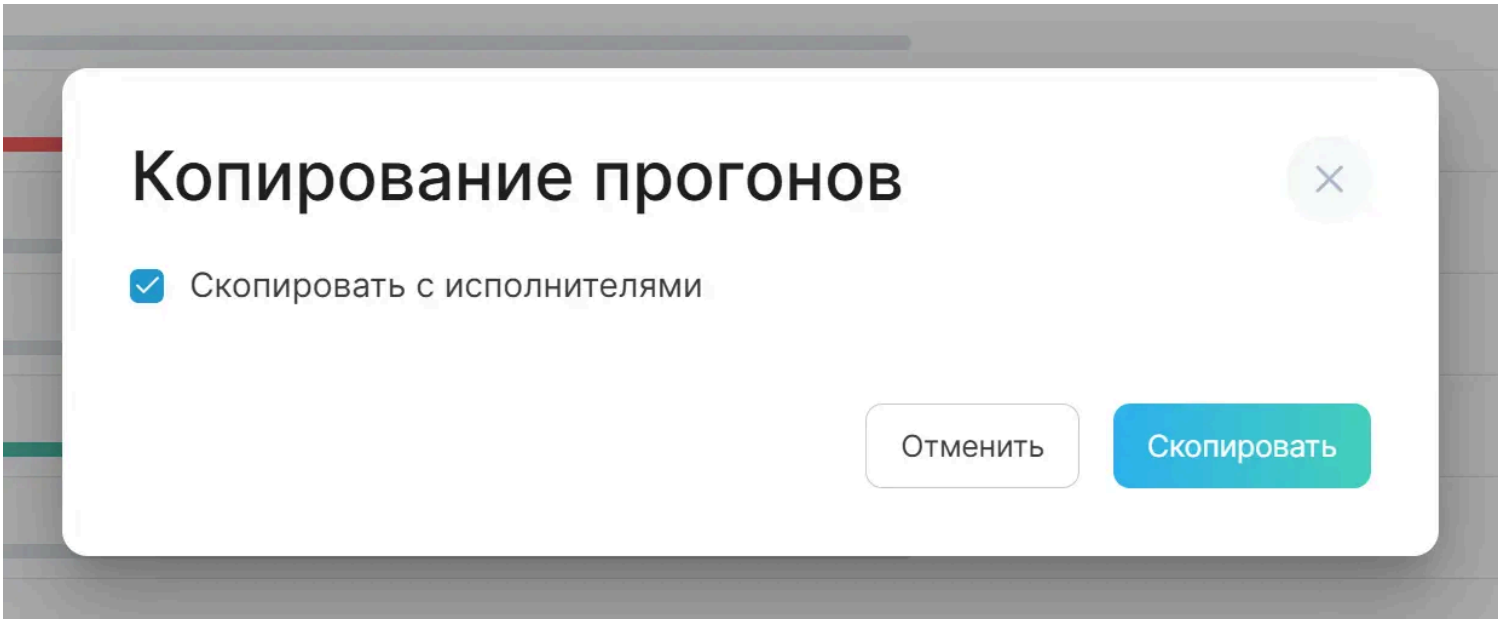
Текущий релиз был посвящен доработкам функционала на основе обратной связи от наших пользователей и улучшению пользовательского опыта.

- Кнопки старта и паузы в окне прохождения тестов в прогоне. В плеере прохождения кейсов и чек-листов появились кнопки старт/пауза. Это полезно при переключении между задачами, когда нужно поставить прохождение на паузу, чтобы не искажать статистику.



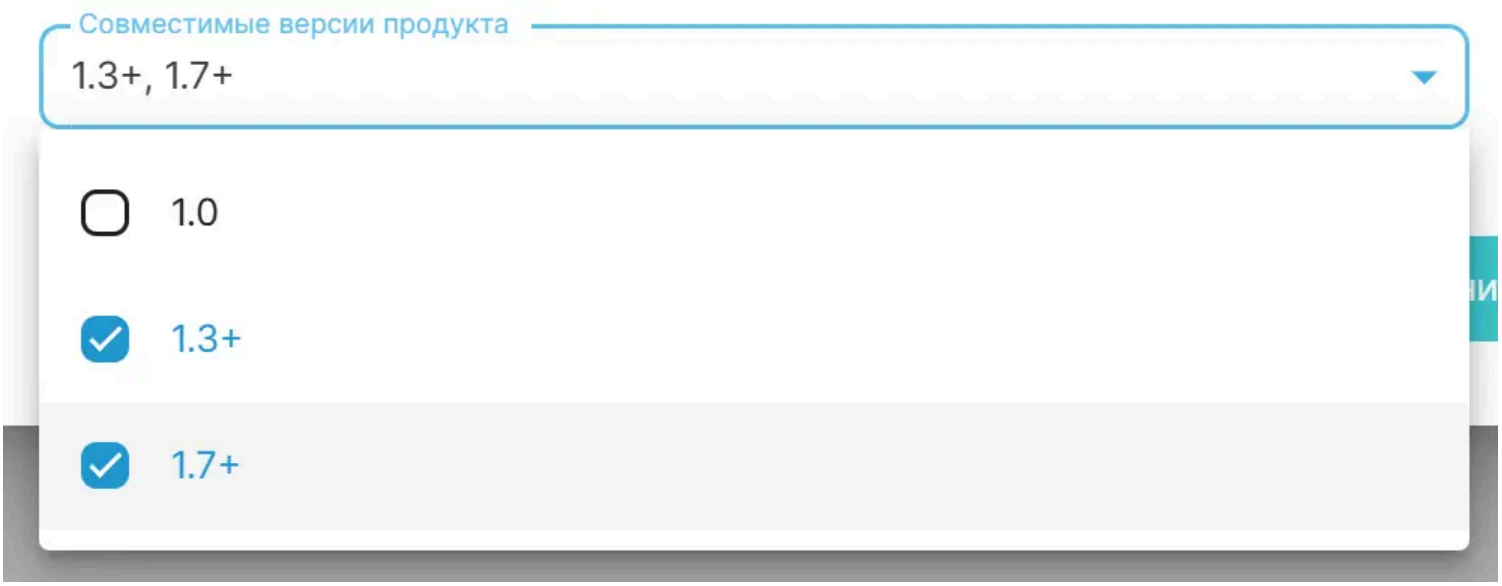
- Уведомления в плеере прогона. Если другой пользователь изменил статус прогона, вы сразу же увидите соответствующее уведомление.

- Копирование прогона с исполнителями или без. Это позволяет сохранить прежнее распределение исполнителей в новом прогоне, не назначая их заново.



- Новый атрибут с множественным выбором. Добавлен новый тип атрибута с множественным выбором.

Подробнее об атрибутах



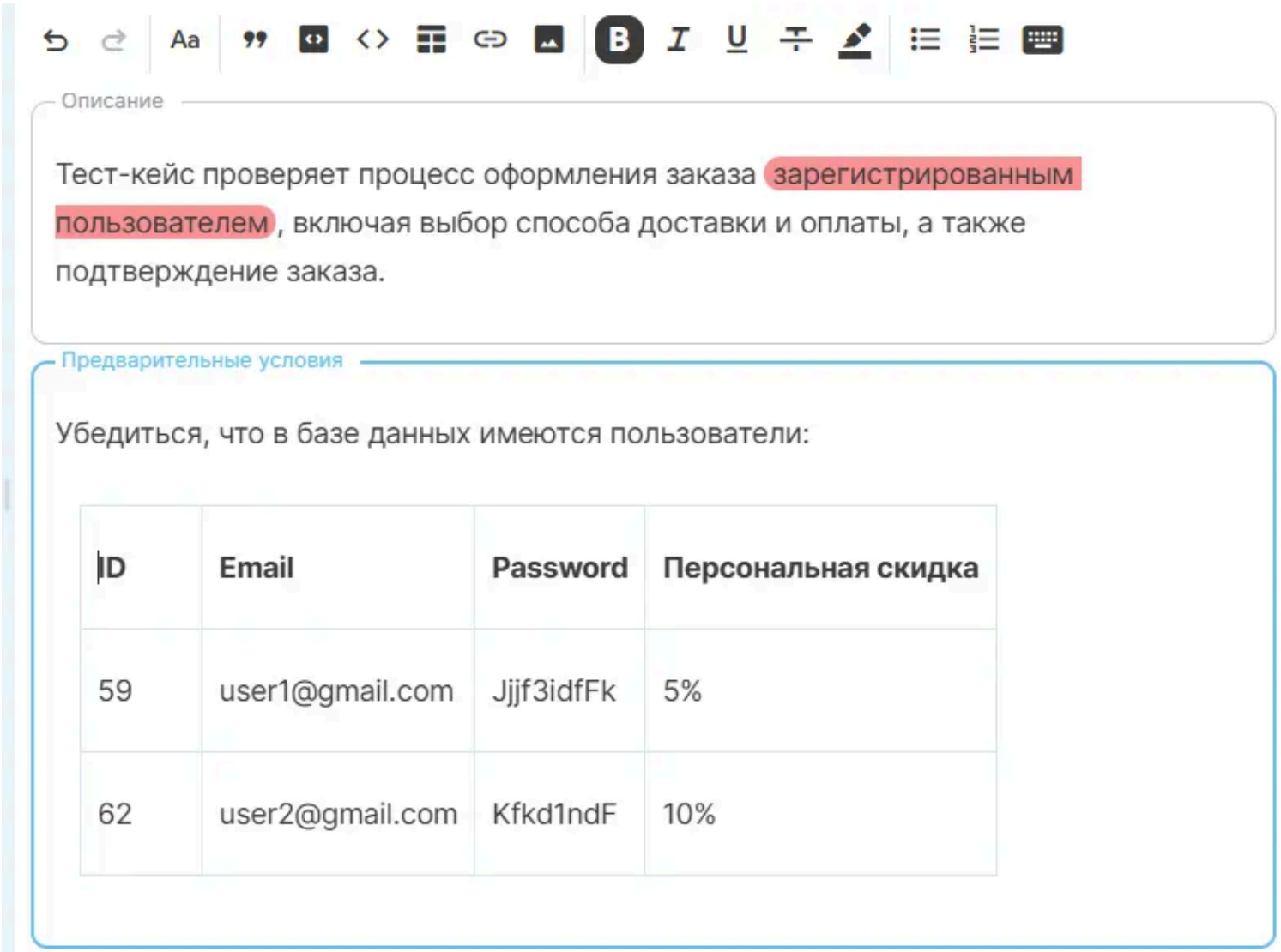
- Доработки импорта проектов. При импорте проекта из TMS Test IT теперь сохраняется порядок тест-кейсов в папках (секциях).

- Обновленный парсер отчетов автотестов. При загрузке отчетов автотестов теперь можно указывать его имя через параметр. Это помогает однозначно идентифицировать отчеты и облегчает навигацию.

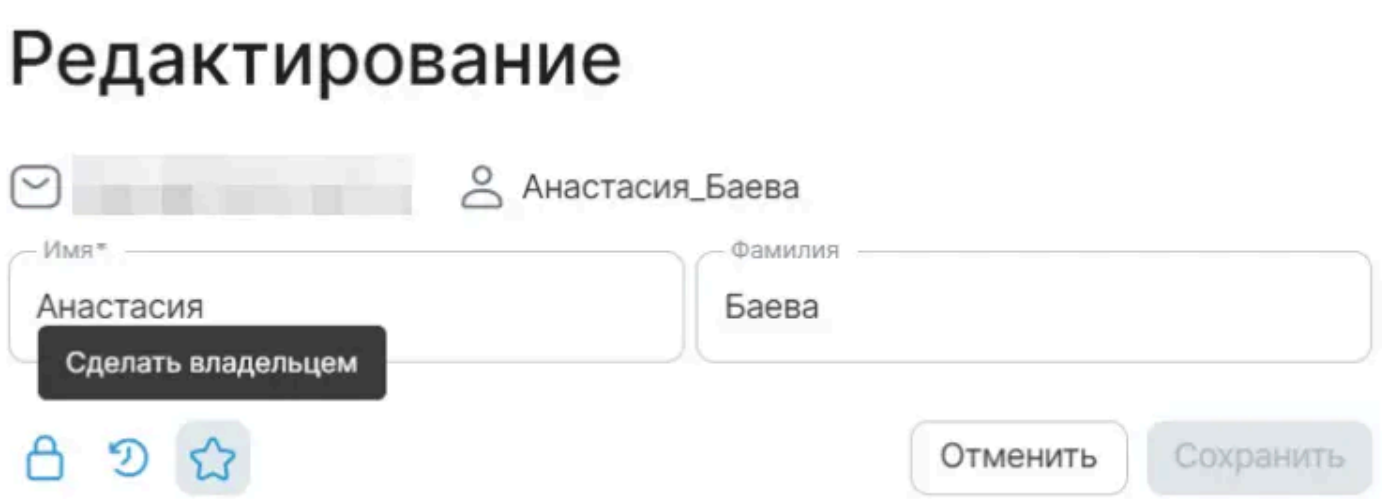
3.6.0-box от 11.11.2025

В этом релизе мы обновили текстовый редактор, расширили функционал уведомлений и реализовали несколько запросов пользователей.

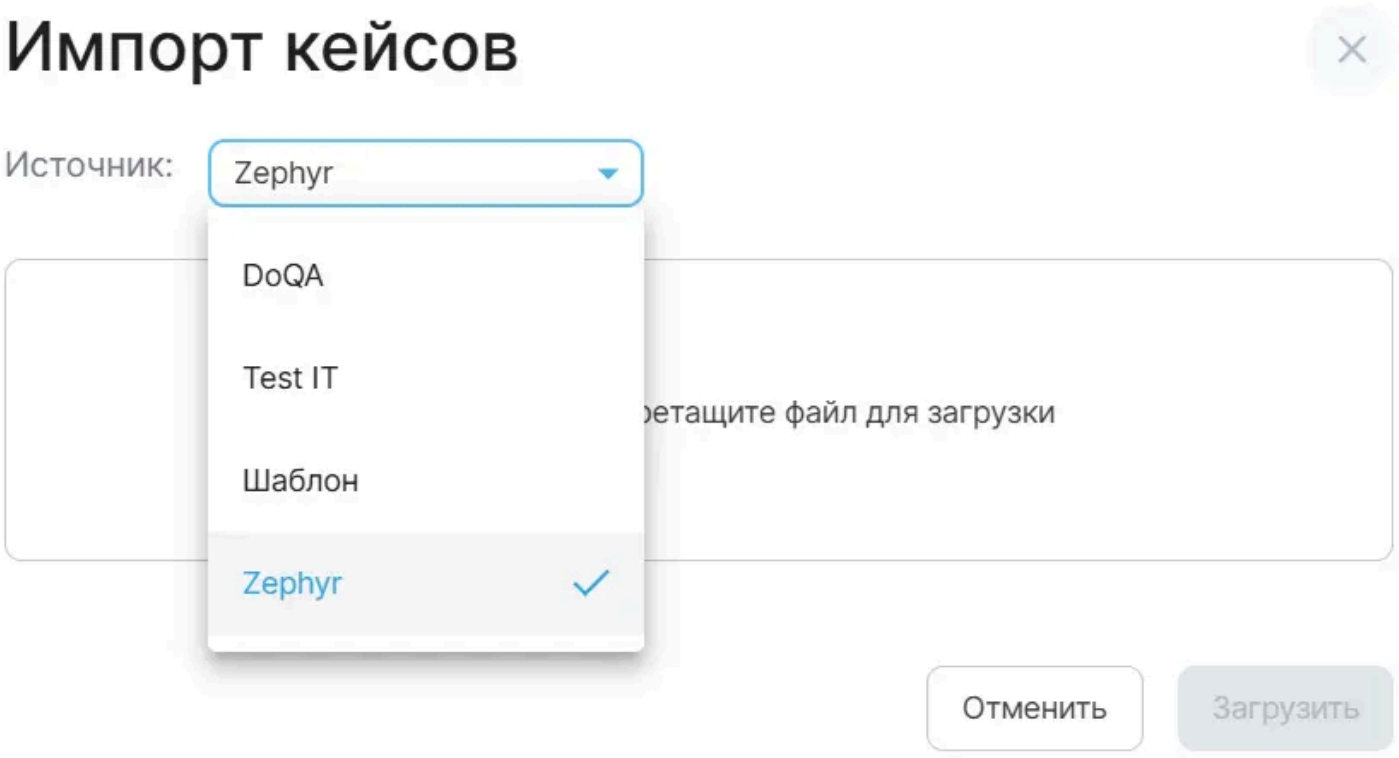
- Новый текстовый редактор. В новом текстовом редакторе, помимо привычных функций, доступны: настройка размера изображений, добавление таблиц, использование **горячих клавиш** для ускорения работы.



- Новые возможности для администраторов. Больше не нужно обращаться в поддержку, чтобы передать права владельца системы. Теперь администраторы могут изменить владельца системы прямо в карточке пользователя. Кроме того, помимо стандартного приглашения в систему через email, появилась возможность скопировать прямую ссылку-приглашение из карточки пользователя.

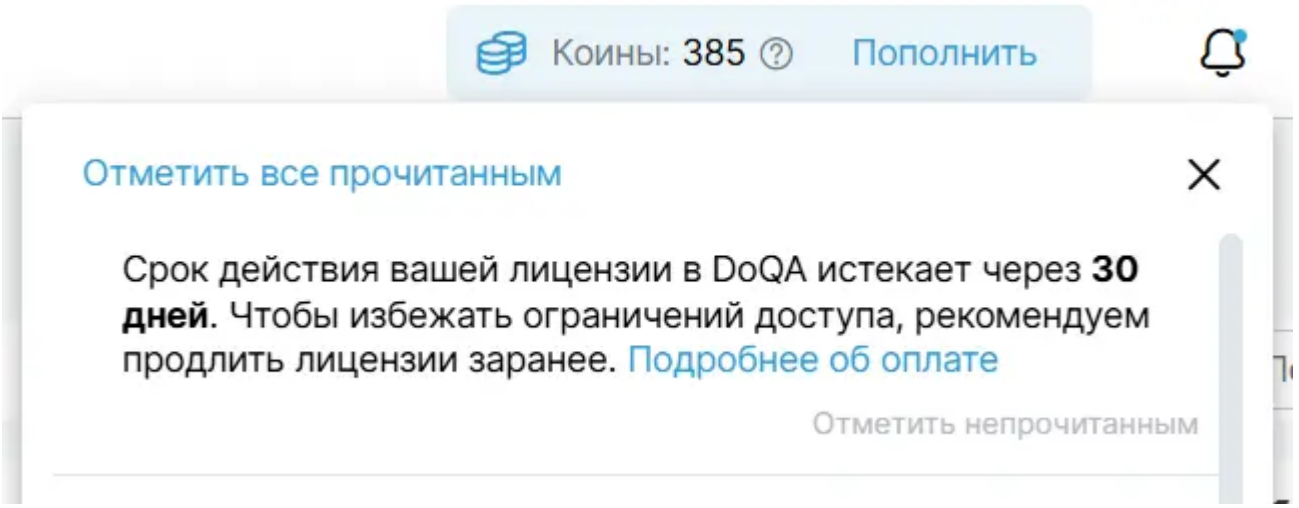


- Кастомная интеграция с Jira Server. Мы добавили универсальный конструктор полей для интеграций. Это значит, что вы видите не только стандартные, но и кастомные поля баг-трекера. Набор подстраивается под каждый проект.
- Импорт тест-кейсов в формате Zephyr. Добавили **новый формат импорта** тест-кейсов из TMS Zephyr, чтобы упростить процесс переноса тестовой документации при переходе в DoQA.



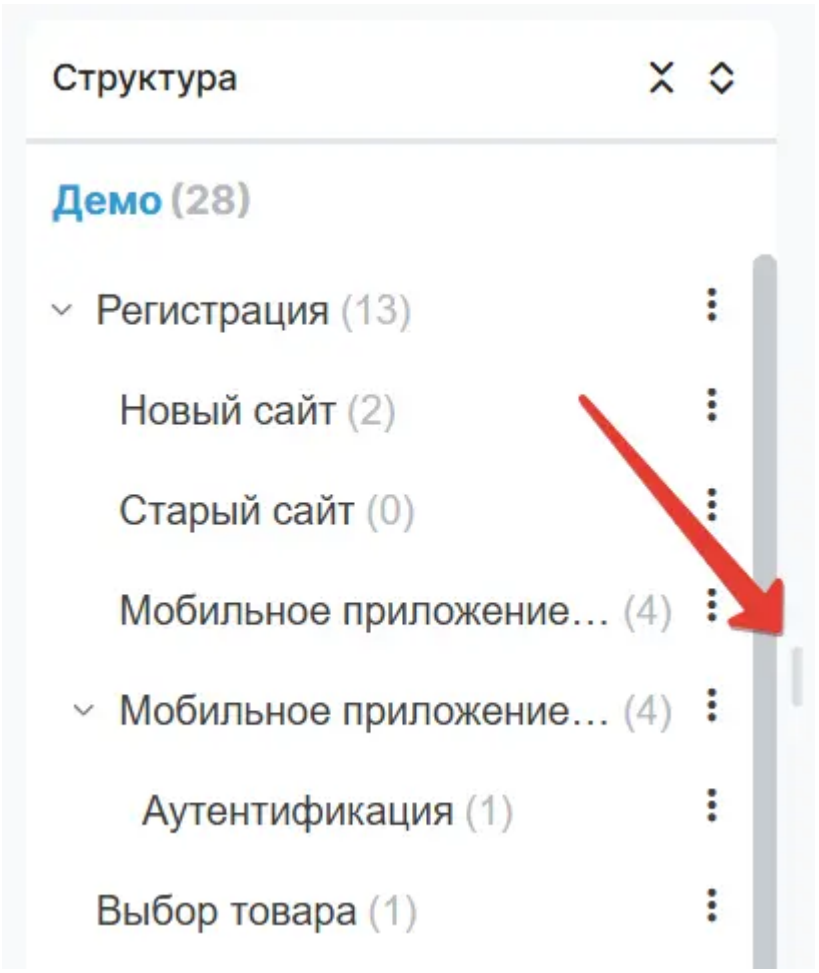
- Развитие функционала уведомлений. В DoQA появилось два новых канала получения уведомлений: внутренние уведомления с колокольчиком и нотификации через Telegram-бота.

Подробнее о настройке уведомлений



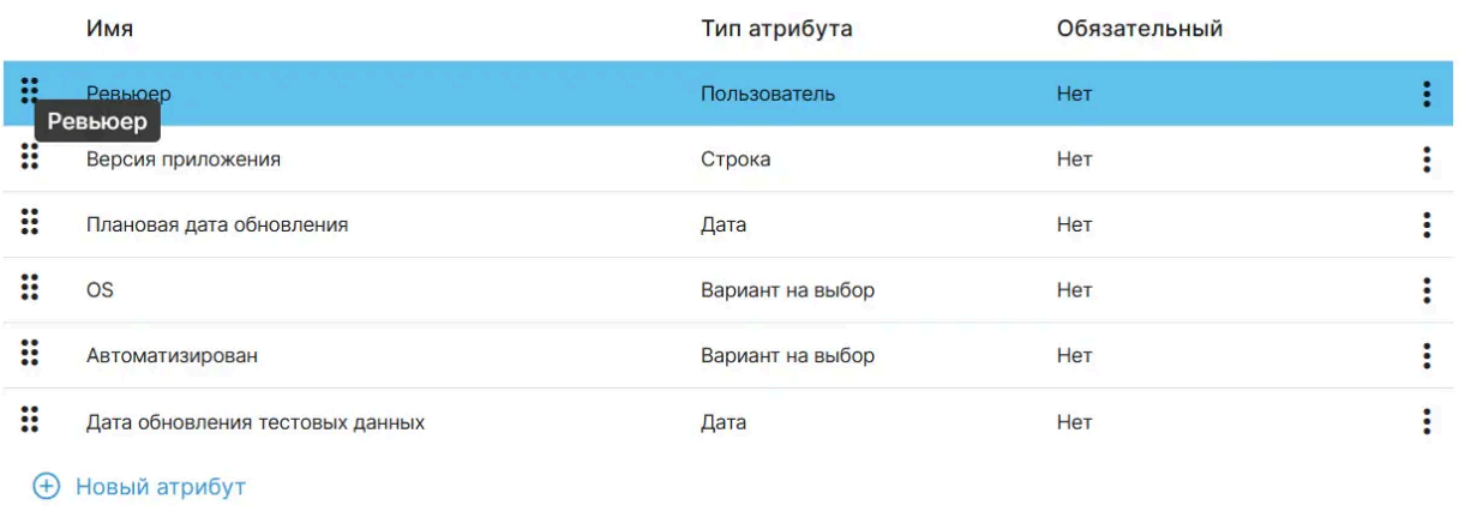
Улучшения на основе обратной связи от наших пользователей:

- В списке прогонов теперь отображается число баг-репортов по каждому прогону.
- Усовершенствовали структуру папок. Теперь можно настроить ее ширину, а у каждой папки в структуре появилось меню.



- Порядок отображения кастомных атрибутов в системе теперь можно установить в настройках проекта.

Атрибуты:



- В полный отчет о прогоне выводится описание прогона.

3.5.0-box от 05.09.2025

Мы упростили документооборот и сделали интерфейс ещё интуитивнее, а также облегчили настройку приложения.

- Обновленная утилита управления приложением. Теперь утилита позволяет создавать сертификаты для настройки безопасного соединения по https и менять домен через консольную команду. Для обновления на версию 3.5-box обязательно используйте новую утилиту.
- Email-уведомления о ключевых действиях в системе. Оставайтесь в курсе изменений без ручного мониторинга. Теперь DoQA сообщит вам по e-mail:

1. О назначении исполнителем теста в прогоне или ответственным за тест-кейс или чек-лист;
2. О завершении созданного прогона;
3. О предстоящем окончании срока лицензий;
4. Сводную информацию по активным прогонам.

- Мультивыбор чек-боксов. По запросам пользователей мы внедрили мультивыбор чек-боксов. Теперь вы можете выбрать сразу несколько чек-боксов, привычно зажав Shift, чтобы ускорить массовую работу с элементами.
- Учет таймзоны пользователя в разделе с дашбордами. Теперь часовой пояс пользователя учитывается и при отображении событий в аналитических дашбордах.

Улучшения интерфейса:

- Добавлен чек-бокс для массового выбора прогонов.
- Окно с названием баг-репорта теперь адаптируется под длину текста.
- Добавили несколько технических улучшений для более плавной и удобной работы системы.
- Расширили возможности фильтра в редакторе прогона: теперь легче добавить в созданный прогон нужные тест-кейсы или чек-листы.

3.4.0-box от 06.08.2025

- Новый фильтр для тестовой документации. Добавили возможность фильтрации тест-кейсов и чек-листов по дате создания и последнего редактирования. Это упрощает навигацию в документации и контроль актуальности.
- Расширение форматов импорта. Добавили новый формат импорта проектов.
- Новое массовое действие с элементами. Теперь тест-кейсы и чек-листы можно массово перетаскивать в папки вместо того, чтобы сортировать каждый элемент по отдельности. Помимо этого появился новый способ их перемещения через диалоговое окно, в котором можно отметить целевую папку в дереве папок.
- Учет часового пояса пользователя. Просматривайте актуальное время обновления тест-кейсов и чек-листов в соответствии с вашей тайм-зоной.
- Обновления интерфейса. Стало проще отслеживать статусы пользователей системы: «активный», «заблокированный», «приглашенный». Теперь они отображаются прямо в списке пользователей в настройках администратора. У владельца тенанта также появилось свое обозначение.

3.3.0-box от 18.06.2025

В серверной версии DoQA теперь доступны обновления, которые уже зарекомендовали себя в облаке: расширенные интеграции, новые инструменты аналитики и улучшенное управление тестовой документацией. Помимо этого, мы обновили утилиту управления сервером.

- Новая утилита управления приложением. В обновление включены важные технические исправления, при этом команды управления остались прежними. Настоятельно

рекомендуем обновить утилиту (ссылка размещена в руководстве пользователя), особенно при активном использовании функции резервного копирования.

- Аналитический дашборд в рабочем пространстве. Теперь в каждом пространстве есть собственный аналитический центр. Отслеживайте динамику прохождения тестов, контролируйте активность команды и находите узкие места в процессах.
- Общие шаги в тестах. Создайте библиотеку повторяющихся шагов и вставляйте их в тест-кейсы. Изменения таких шагов автоматически применяются ко всем тестам, в которых они задействованы.
- Новый способ взять тест в работу. В прогонах теперь можно нажать кнопку «Назначить на меня», чтобы быстро стать исполнителем тест-кейса или чек-листа, если он был свободен.
- Прямая и обратная интеграция с GitLab. Теперь данные из DoQA можно отправлять напрямую в GitLab, чтобы команда работала в едином потоке без переключения между инструментами.
- Обратная интеграция с Yandex Tracker, Jira Cloud/Server и Bitrix24. Изменения статуса и приоритета баг-репорта в трекерах автоматически подтягиваются в DoQA и моментально отражаются в системе.
- Темная тема. Добавили поддержку тёмной темы.
- Продолжаем улучшать интерфейс приложения. Теперь боковое меню с вкладками можно свернуть до иконок, чтобы расширить рабочее пространство.

3.1.0-box от 03.03.2025

В этом обновлении мы улучшили контроль за временем выполнения тестов.

- Таймер выполнения тест-кейсов и чек-листов. Анализируйте, сколько времени уходит на прохождение тестов. Отсчет времени начнется автоматически при проставлении статуса первому шагу или чеку.
- Возможность повторного прохождения теста в прогоне. На экране прохождения теста в прогоне появилась кнопка «Пройти заново», при нажатии на которую сбрасывается прогресс прохождения выбранного теста и обнуляется время его прохождения.

3.0.0-box от 17.02.2025

В новом релизе мы расширили управление документацией: добавили инструменты для гибкой настройки, распределения задач и контроля изменений, а также усилили интеграцию с YouTrack.

Теперь в DoQA доступны:

- Настройка уникальных пользовательских атрибутов для тест-кейсов и чек-листов. Сделайте документацию более детализированной, добавив к ней новые поля с информацией: выберите формат (дата, пользователь, строка или атрибут с вариантами на выбор) и внесите необходимые тестовые данные.
- Назначение ответственного в тест-кейсах и чек-листах. Укажите пользователя, ответственного за тест-кейс или чек-лист. Это позволит команде видеть, кто отвечает за актуализацию тестовых артефактов.
- Обновленная фильтрация и окно массового редактирования документации. Теперь фильтр и окно массового редактирования можно настраивать: вы сами выбираете, какие поля отображать. Также появилась возможность фильтрации и массовых действий по атрибутам и ответственным.
- Версионность тест-кейсов и чек-листов. Теперь, если нужно откатить какие-то изменения в документации, выберите одну из ранних версий и сделайте ее текущей.
- Обратная интеграция с YouTrack. Теперь при изменении статуса или приоритета баг-репорта в YouTrack, соответствующие поля связанного баг-репорта в DoQA обновятся автоматически.
- Уведомления об истечении лицензий. Будьте уверены в непрерывности процессов тестирования: за неделю до истечения срока действия лицензий и в последний день вы получите напоминание о продлении.

2.8.0-box от 25.12.2024

Рады представить новые возможности для удобной работы с системой в финальном релизе 2024 года.

- Архивирование проектов — теперь вы можете отправлять неактуальные проекты в архив. Такие проекты не будут отображаться на главной странице, но их всегда можно будет восстановить при необходимости.
- Корзина удаленных элементов для защиты от случайного удаления тестовых артефактов. Помещенные в корзину элементы можно восстановить в течение 30 дней.

- Оптимизация прогонов — сортируйте тесты по количеству баг-репортов и комментариев и просматривайте список исполнителей в таблице прогонов.
- Расширение фильтрации элементов по тегам — теперь кроме значения «ИЛИ» вы можете применять фильтр «И» для более точной выборки.
- Импорт проектов из Qase перенесен из публичного API в интерфейс системы.
- Стала доступна обработка отчетов о прогонах автотестов в формате Allure.

2.7.0-box от 05.12.2024

Упрощаем управление дефектами, улучшаем плеер прогонов и расширяем возможности тест-кейсов и чек-листов. В свежем релизе:

- отдельная вкладка «Дефекты», в которой отображаются все баг-репорты со ссылками на прогоны — теперь отчеты о дефектах со всего рабочего пространства собраны в одном месте;
- обновленный редактор тест-кейсов и чек-листов — увеличено количество шагов и чеков до 300 штук, появился счетчик чеков и шагов, а работа в редакторе стала более интуитивно понятной;
- обновленный вид комментариев — интерфейс стал более приятным и информативным, а в самих комментариях появились перекрестные ссылки на соответствующие им тест-кейсы, чек-листы и прогоны;
- прогресс-бар и счетчики заведенных баг-репортов и комментариев у каждого теста в плеере прогона;
- возможность открывать страницу с баг-репортом по ссылке в отдельной вкладке — делиться данными стало еще удобней.

2.6.0-box от 17.10.2024

Расширяем возможности интеграции и аналитики. Встраивать DoQA в процессы управления тестированием стало еще удобней.

- Интеграция с Битрикс24 и Redmine — данные из DoQA теперь можно отправлять напрямую в эти системы.
- Полный отчет о прогоне — теперь отчет содержит не только информацию по результатам тестов, но и сами тесты.

- Новые метрики по исполнителям в отчетах.
- Настраиваемые конфигурации для прогонов, которые позволяют указать, на каком окружении и в какой среде необходимо проводить тестирование.
- Возможность удаления прогонов из автотестов.

2.5.0-box от 05.09.2024

Это обновление оптимизирует работу с плеером, расширяет форматы импорта документации и не только.

- Плеер прогонов стал универсальным. Теперь в одном прогоне можно запускать и тест-кейсы, и чек-листы.
- Плеер изменился внешне, а механика работы и редактирования стала более комфортной и понятной.
- В прогонах теперь можно назначать исполнителей для тестов. Если исполнитель не назначен, то тест может пройти любой тестировщик.
- У шагов и чеков появились статусы для конкретизации результатов тестирования, а также возможность указывать баги и комментарии.
- В кейсах появилось поле «Тестовые данные» для внесения сопутствующей информации. Так тест-кейсы станут более подробными.
- Стало возможно импортировать проекты в DoQA из Qase в формате JSON с помощью публичного API. Нам неважно, в какой стране вы находитесь. Пользоваться DoQA можно из любой локации.
- Поиск необходимой информации о DoQA стал проще: руководство пользователя и инструкции переехали на новый портал документации.

2.4.0-box от 04.07.2024

Система начала движение к статусу универсального инструмента по работе с любыми видами тестов, и это обновление — первый шаг к будущим масштабным изменениям.

- Автотесты — теперь DoQA не только упрощает и оптимизирует ручное тестирование, но и позволяет интегрировать результаты автотестов, чтобы вся отчетность удобно просматривалась в одном инструменте. Отправляйте новые отчеты по автотестированию прямо в DoQA или загрузите в TMS уже существующие.

- Интеграция с YouTrack — одна из самых запрашиваемых фич. Теперь данные из DoQA можно отправлять напрямую в систему управления проектами YouTrack, упрощая работу с этим инструментом.

2.3.5-box от 03.06.2024

Мы продолжаем расширять интеграции DoQA с внешними сервисами, пока в разработке находится глобальное обновление раздела прогонов. В новой версии появилась одна из самых запрашиваемых фич — интеграция с системой управления проектами Yandex Tracker.

- Баги теперь можно отправлять в Yandex Tracker напрямую из DoQA.
- Изменилась работа с тегами внутри редакторов тестов. К тестам можно добавлять любой из уже созданных тегов, а логика редактирования и добавления тегов переехала в обновленный редактор.
- Редактор тегов стал более компактный, наглядный и понятный.
- Таблицы кейсов и чек-листов больше не схлопываются при изменениях и удалении элементов.
- Работать с диалоговыми окнами стало удобнее. Теперь их можно перемещать, чтобы лучше видеть нужную информацию.

2.3.0-box от 25.04.2024

В DoQA появилась первая интеграция с внешним сервисом — Jira. Теперь баг-репорты можно отправлять в Jira напрямую из интерфейса DoQA с помощью обновленной формы.

- Появился раздел интеграций, в котором можно подключить Jira Cloud и Jira Server.
- Для каждого проекта можно настроить свои параметры интеграции.
- Форма баг-репорта обновилась, и оформление бага стало проще. DoQA сама собирает данные из кейсов и чек-листов.
- Стало возможным массовое редактирование кейсов и чек-листов: смена статуса, приоритета, работа с тегами.
- Расширен функционал работы с тегами. Теперь их можно удалять и очищать при массовом редактировании.
- При создании кейсов, чек-листов и папок таблица остается в развернутом виде.

2.2.7-box от 19.03.2024

В этом релизе мы расширили возможности редактора и добавили другие обновления, которые делают работу в DoQA еще более удобной:

- В кейсах и чек-листах появился раздел вложений. Теперь к тестам можно прикреплять файлы;
- В редакторе теперь можно работать с кодом и выделять текст цветом;
- Редактор сохраняет заданную ширину после обновления страницы;
- Таблица элементов остается активной при использовании редактора и позволяет быстро переключаться между кейсами и чек-листами;
- В плеере прогона появилось обозначение пути до кейса/чек-листа;
- Стало возможным возобновлять прохождение прогона даже после его завершения.

2.2.0-box от 31.01.2024

Встречайте долгожданное обновление структуры таблиц и системы работы с папками.

- Мы отказались от пагинации. Теперь работа с элементами доступна в едином структурированном списке.
- Дерево папок переехало из бокового меню в таблицу для удобного переключения вкладок.
- Появились функции «свернуть» и «развернуть» папки.
- В дереве папок сохраняются состояния после перезагрузки и обновления страницы: если папки были открытыми, то такими и останутся.
- Таблицы стали более наглядными и теперь отображают вложенность.
- Добавлено окно подтверждения удаления кейсов и чек-листов.
- Порядок создания кейсов и чек-листов стал удобней. Теперь их можно создавать прямо внутри нужной папки.
- Увеличена длина имени папки.
- Добавлена возможность использовать спецсимволы в пароле.

2.1.7-box от 21.12.2023

Пока в DoQA готовится масштабное обновление структуры таблиц, мы выпустили несколько улучшений, которые делают работу с TMS более удобной:

- Изображения в документации разворачиваются по клику на весь экран;
- К чек-листам и кейсам можно добавлять статус «Ревью»;
- Дата обновления кейса/чек-листа теперь видна в таблице. Стало проще отслеживать обновление документации;
- Доработан импорт для получения дерева папок из секций.

2.1.0-box от 26.10.2023

Встречайте первый релиз этой осени: мы доработали функционал системы и немного затронули дизайн в преддверии будущего масштабного обновления. Что нового в DoQA:

- обновился внешний вид таблиц в разделах;
- обновился дизайн и механика работы фильтра. Теперь кейсы, чек-листы и прогоны можно фильтровать по большему количеству параметров;
- добавлена возможность фильтрации по папкам;
- стало возможно удаление прогонов;
- стало возможно добавление элементов в прогон напрямую из раздела или папки;
- появились уведомления о необходимости наличия лицензии при редактировании прав пользователя.

2.0.0-box от 31.08.2023

DoQA обновилась. Теперь наша TMS доступна и в серверной версии.

Также в этом релизе DoQA дополнилась:

- возможностью импортировать тест-кейсы и чек-листы в нескольких форматах;
- защитой от случайного переключения и закрытия редактора с уведомлением о несохраненных изменениях;
- сохранением тестовой документации по горячим клавишам для более продуктивной и быстрой работы;

- обновлением дизайна интерфейса... ...и другими инфраструктурными улучшениями.