ChatGPT

TestCloud Suite — Всё, что нужно QA — в одном облаке

Описание проблемы, которую решает продукт

Современные QA-команды часто вынуждены использовать разрозненные инструменты для управления тестированием: отдельные системы для планирования тест-кейсов, другие – для баг-трекинга, плюс вспомогательные утилиты для API и UI тестирования. Такой подход приводит к рассинхронизации данных, дублированию информации и усложнению коммуникации между командами. По сути, существуют **три распространённых подхода**: самый простой – вести тестовую документацию в таблицах, а дефекты — в отдельном баг-трекере; более продвинутый – использовать связку Test Management System (TMS) + корпоративный баг-трекер; и **идеальный** – внедрить единую платформу, которая покрывает все потребности тестирования [1]. TestCloud Suite реализует именно последний подход: он объединяет в одном облачном сервисе все процессы QA, устраняя разрозненность инструментов и обеспечивая **“единый источник истины”** для качества продукта [2]. Это решает ключевые проблемы: повышает прозрачность тестирования для всей команды, ускоряет обмен знаниями и уменьшает трудозатраты на поддержку множества интеграций.

Кроме интеграционных сложностей, отсутствие единой платформы затрудняет сбор **метрик качества**. Руководители часто не имеют количественного представления об общем качестве продукта, собирая разрозненные показатели вроде покрытия тестами, количества дефектов или скорости их исправления вручную. TestCloud Suite решает и эту проблему: платформа автоматически собирает и сводит ключевые метрики (прохождение тестов, дефекты по приоритетам, покрытие требований и др.) в общие **дашборды**, доступные всем заинтересованным сторонам. Это позволяет наглядно отслеживать прогресс тестирования и качество продукта в динамике [3] [4].

Обзор платформы и модули TestCloud Suite

TestCloud Suite – это облачная SaaS-платформа для QA-команд, объединяющая все необходимые инструменты тестирования в едином интерфейсе. Лозунг продукта – «*Всё, что нужно QA – в одном облаке*» – отражает его суть: TestCloud Suite предоставляет сквозное решение для управления качеством, от планирования тестов до отслеживания багов и автоматизации, устраняя необходимость переключаться между разными системами. Ниже приведены основные модули платформы и их возможности:

- **Test Manager (менеджер тестирования):** модуль для управления тест-кейсами и тестовыми планами. Здесь QA-инженеры создают и организуют тест-кейсы (как ручные, так и автоматизированные), составляют тест-наборы и планы тестирования, описывают шаги и ожидаемые результаты. Поддерживается версияция тест-кейсов и повторное использование сценариев в разных релизах. Test Manager предоставляет инструменты для планирования тестовых прогонов, назначения исполнителей и отслеживания статуса прохождения тестов в режиме реального времени. Благодаря интеграции с остальными модулями, обнаруженные в ходе тестов дефекты можно сразу регистрировать в баг-

трекере, а результаты автоматических тестов – автоматически подтягивать в систему. По сути, Test Manager охватывает весь цикл тестовой документации: от планирования до выполнения и учета результатов [5].

- **Bug Tracker (баг-трекер):** встроенная система отслеживания ошибок, позволяющая командам регистрировать, классифицировать и управлять дефектами напрямую внутри TestCloud Suite. Баг-трекер поддерживает все атрибуты дефектов: заголовок, описание шагов воспроизведения, приоритет, серьезность (severity), статус жизненного цикла (новый, в работе, исправлен, закрыт и т.д.), автор и назначенный исполнитель. Реализованы гибкие **рабочие процессы** для дефектов – настроенные статусы и переходы, уведомления ответственных, привязка багов к релизам или спринтам. Важной возможностью является привязка баг-репорта к конкретному тест-кейсу или тест-прогону в системе: например, тестировщик может завести баг напрямую из проваленного теста, и система автоматически свяжет дефект с шагом тест-кейса и версией приложения. Это обеспечивает полноценную трассируемость от теста к багу и обратно. Баг-трекер поддерживает добавление вложений (скриншоты, логи) и комментариев для упрощения воспроизведения ошибок. Благодаря тому, что баг-трекинг **встроен в платформу**, разработчики и тестировщики работают с дефектами в едином окне, без переключения на внешние системы [6].

- **API Test Workspace (рабочая среда API-тестирования):** модуль, предназначенный для проектирования, выполнения и автоматизации тестов API. Он предоставляет интерфейс, схожий с популярными инструментами типа Postman: QA-инженеры могут создавать коллекции API-запросов (GET/POST/PUT и др.), указывать параметры, тела запросов, заголовки, а затем выполнять эти запросы и проверять ответы. В TestCloud Suite API Workspace интегрирован с Test Manager, позволяя сохранять API-тесты как особый вид тест-кейсов. Возможна параметризация запросов и проверка ключевых полей ответа (статус-коды, JSON поля и пр.). Кроме ручного запуска запросов, модуль поддерживает автоматизацию API-тестов: например, настройку автозапуска коллекций по расписанию или при определенных событиях (по триггеру из CI/CD). Результаты исполнения API-тестов (успешно/неуспешно, время ответа, сравнение с эталонным ответом) сохраняются в систему и учитываются в общей статистике тест-прогонов. Такой инструмент позволяет тестировать бэкенд-сервисы и интеграционные сценарии без графического интерфейса, что особенно важно при тестировании микросервисов и проверке контрактов API.

- **UI Automation Hub (хаб UI-автоматизации):** модуль для управления и выполнения автоматизированных UI-тестов (например, Selenium/WebDriver сценариев) через облачную инфраструктуру. Он предоставляет централизованный **“хаб”** для подключения агентов выполнения тестов (тестовых узлов) и распределения на них тестовых задач. QA-команда может загрузить или зарегистрировать автоматические тесты UI (написанные, к примеру, на Selenium, Cypress, Playwright и др.) в системе – обычно указывается репозиторий или пакет с тестами. UI Automation Hub позволяет запускать эти тесты на разных окружениях и платформах: например, выполнить набор UI-тестов параллельно в разных браузерах (Chrome, Firefox, Safari) или на разных операционных системах. Внутри платформы настраиваются **пулы агентов** – физических или виртуальных машин (или контейнеров) с браузерами и нужными драйверами. Запуская тесты, система ставит задачи в очередь, агенты их выбирают и исполняют, а результаты (лог прохождения шагов, скриншоты экрана при ошибках, отчет о прохождении) возвращаются в TestCloud Suite. Таким образом, тестировщики могут инициировать и мониторить автотесты UI из единого интерфейса, а не вручную через IDE. Это существенно ускоряет прогоны регрессионных тестов и позволяет легко масштабировать автоматизацию под нагрузкой. UI Automation

Hub также интегрирован с Test Manager: статусы автотестов отображаются рядом с ручными, давая целостную картину покрытия. Таким образом, **ручное и автоматизированное тестирование объединены в едином окне** [7], что выгодно отличает TestCloud Suite от классических разрозненных решений.

- **Дополнительные модули:** помимо перечисленного, платформа включает вспомогательные компоненты, обеспечивающие бесперебойную работу и удобство использования. Среди них – **модуль отчетности и аналитики**, собирающий ключевые метрики (см. ниже) и позволяющий строить настраиваемые отчеты и графики по проектам. **Интеграционные адаптеры** позволяют связать TestCloud Suite с внешними инструментами при необходимости – например, синхронизировать баги с Jira или Azure DevOps, получать автотесты из GitHub, отправлять уведомления в Slack и т.п. Также присутствует **модуль уведомлений**, рассылающий email/SMS/Slack-оповещения о важных событиях (падение критического теста, приближающийся дедлайн тест-плана, изменение статуса бага). Все модули работают согласованно и доступны через единый веб-интерфейс и единый REST API. Платформа спроектирована модульной, поэтому новые компоненты можно легко подключать по мере развития продукта (см. будущие расширения).

В совокупности, эти модули обеспечивают полный цикл обеспечения качества в одном месте. Платформа покрывает потребности команд QA любого размера, позволяя им адаптировать рабочие процессы под себя. Например, система позволяет настроить собственные поля, статусы и права в каждом модуле под специфику проекта. Такой уровень кастомизации и интеграции всех инструментов в одном решении устраняет барьеры между тестировщиками, разработчиками и менеджментом, повышая эффективность и прозрачность QA-процесса [8].

Роли пользователей и разграничение прав

В TestCloud Suite реализована гибкая ролевая модель, позволяющая разграничить доступ и возможности пользователей в системе. Основные **роли пользователей** и их права следующие:

- **Администратор платформы:** обладает полными правами на уровне всей системы. Админ может создавать и удалять проекты, управлять пользователями (добавлять новых участников команды, назначать им роли), настраивать глобальные параметры системы (например, интеграции с внешними сервисами, пул автоматизационных агентов, тарифный план). Также администратор видит все проекты и данные, имеет право редактировать любые объекты (тест-кейсы, баги и т.д.) и осуществлять административные действия вроде резервного копирования данных, настройки очередей и пр. Как правило, роль админа назначается одному или нескольким наиболее ответственным лицам (например, QA-менеджеру или DevOps-инженеру проекта).

- **Менеджер QA / Тест-лид проекта:** имеет широкие права внутри отдельных проектов. Этот пользователь может создавать и редактировать тест-кейсы, объединять их в наборы, планировать прогоны, назначать тестировщиков на выполнение, а также просматривать и менять статус всех багов в своем проекте. QA-менеджер видит все метрики и отчеты по своему проекту, может настраивать рабочий процесс (статусы дефектов, дополнительные поля) в рамках проекта. Однако менеджер не управляет системными настройками вне проекта и не видит другие проекты, если не наделен правами в них. Таким образом, тест-лид отвечает за организацию тестирования в пределах конкретного продукта или команды.

- **Тестировщик (QA Engineer):** основная роль для членов QA-команды. Тестировщик может просматривать тест-кейсы в проекте, создавать новые и предлагать правки (в рамках своего доступа), выполнять назначенные ему тест-кейсы в прогоне, помечать результаты (прошел/не прошел, блокирован и т.д.), и заводить баг-репорты при обнаружении дефектов. Также тестировщик может комментировать объекты (тесты, баги), прикреплять файлы (например, скриншоты ошибок). Роль тестировщика, как правило, **не позволяет** редактировать или удалять чужие тест-кейсы, менять ключевые поля багов, если он не является их автором или исполнителем, и не дает доступа к административным функциям. Таким образом, у QA-инженера ограниченные права: в основном на создание новых объектов и изменение статусов своих заданий.

- **Разработчик:** данная роль предназначена для членов команды разработки, которым требуется доступ к системе для работы с дефектами и просмотра результатов тестов. Разработчик, как правило, имеет доступ **только к модулю Bug Tracker** в рамках своего проекта: он может просматривать баг-репорты, назначенные на него, менять статус дефекта (например, перевести в “Исправлено”), оставлять комментарии и прикладывать исправленный код (патчи) или ссылки на коммиты. При этом у разработчика обычно **нет прав** создавать или изменять тест-кейсы, либо отмечать результаты тестов (кроме случаев, когда он совмещает роли). Разработчики могут также просматривать дашборд метрик по своему проекту, чтобы понимать общую картину качества, но без возможности редактирования. Такая изоляция гарантирует, что разработчики вовлечены в процесс тестирования (получают фидбек о дефектах), но не мешают управлению тестовой документацией.

- **Гость/Наблюдатель (чтение):** при необходимости в системе может быть настроена роль с правами только на чтение. Это может быть полезно для внешних аудиторов, клиентов или менеджеров высшего уровня, которым нужно отслеживать прогресс без возможности вмешиваться. Гостевой пользователь может просматривать тест-планы, видеть отчёты и метрики, читать баг-репорты, но не создавать и не изменять их. Данная роль настраивается гибко – например, можно разрешить гостям видеть только определенные проекты.

TestCloud Suite позволяет настраивать **пользовательские роли** и тонко управлять правами. Например, можно создать роль “Automation Engineer”, которая будет иметь права на запуск/добавление автоматических тестов в UI Automation Hub, но без доступа к ручным тест-кейсам. Права разделяются по модулям (Test Manager, Bug Tracker, API Workspace, Automation Hub, Reports) и по типам действий (чтение, создание, редактирование, удаление, администрирование). Все это обеспечивает безопасное разграничение доступа: каждый пользователь видит и делает только то, что ему положено по роли. При необходимости права легко масштабируются – например, можно временно повысить привилегии пользователя для отпускающего релиза, а затем вернуть обратно. Вся активность пользователей логируется (audit log), что важно для контроля изменений и расследования инцидентов безопасности.

Ключевые метрики качества, отслеживаемые в платформе

Одним из важных преимуществ TestCloud Suite является встроенная система сбора и визуализации **метрик качества**. Платформа автоматически рассчитывает и отображает на интерактивных дашбордах ряд ключевых показателей, которые помогают оценить

эффективность процесса тестирования и качество продукта. Ниже перечислены основные метрики, доступные из коробки:

- **Покрытие тестами (Test Coverage):** доля функционала или требований, покрытых тест-кейсами. Выражается в процентах и может измеряться как отношение числа реализованных тест-кейсов к общему числу требований (coverage by requirements) либо количеством автоматических тестов к общему числу модулей кода (coverage by code). Дашборды показывают покрытие по модулям продукта, позволяя выявить области с недостаточным тестированием. Эта метрика отвечает на вопрос: *“Какой процент системы проверяется нашими тестами?”*

- **Статус выполнения тестов (Test Execution Status):** распределение результатов тестовых прогонов – сколько тестов **пройдено успешно**, сколько **провалено**, а сколько **заблокировано/пропущено**. Отображается обычно в виде диаграммы по последнему прогону или суммарно по релизу. Например, можно увидеть, что из 100 тестов 85 прошли (зелёный), 10 упали (красный), 5 заблокированы (серый). Эта метрика даёт мгновенное представление о текущем состоянии качества сборки.

- **Плотность дефектов (Defect Density):** количество найденных дефектов на единицу функциональности (например, на модуль или на 1000 строк кода). В системе она представлена в разрезе компонентов или требований – сколько багов зарегистрировано в каждом компоненте. Анализ дефектной плотности помогает выявить “проблемные” области приложения, где сосредоточено больше всего ошибок, и может указывать на необходимость дополнительного тестирования или рефакторинга в этих модулях.

- **Среднее время жизни дефекта (Defect Cycle Time):** скорость работы с багами – среднее время от обнаружения дефекта до его закрытия. Платформа отслеживает временные метки создания и закрытия каждого баг-репорта и выводит среднее значение, а также распределение (напр. X% багов закрываются за 1 день, Y% – за 3 дня, Z% – дольше недели). Также считается **среднее время реакции** разработчиков (time to first response) и **время на исправление**. Эти метрики важны для оценки эффективности процесса: насколько быстро команда устраняет обнаруженные проблемы.

- **Процент регрессионных дефектов:** доля багов, которые возникли в уже протестированных функциональностях (т.е. возобновились старые проблемы или что-то сломалось при исправлении другого). В системе помечаются дефекты, связанные с регрессиями, и выводится процент таких от общего числа. Высокое значение может указывать на проблемы с покрытием тестами или на то, что исправления нарушают существующий функционал.

- **Уровень автоматизации тестирования:** показатель, характеризующий долю автоматизированных тест-кейсов. Может измеряться как процент автоматизированных сценариев от общего числа тест-кейсов, либо доля тест-планов, выполняемых автоматикой. TestCloud Suite выводит этот KPI, позволяя отслеживать прогресс автоматизации по проектам. Например, команда может видеть, что из 500 регрессионных тестов автоматизировано 300 (60%). Рост этого показателя со временем демонстрирует повышение эффективности (больше рутины покрыто автотестами).

- **Баги, ушедшие в продакшн (Escaped Defects):** количество дефектов, обнаруженных пользователями или на этапе эксплуатации (то есть пропущенных QA-командой). Этот

показатель вводится вручную или через интеграцию с внешними системами поддержки. В отчётах платформы он сопоставляется с общим числом найденных багов в тестировании, давая **коэффициент утечки дефектов**. Низкое значение говорит о высоком качестве тестирования, а рост – сигнал пересмотреть стратегию тестирования.

- **Прогресс тестирования по спринту/релизу:** метрика, отображающая динамику выполнения тестовых задач. Например, сколько тест-кейсов запланировано и сколько из них уже выполнено на текущий день спринта. График Burn-down по тестированию позволяет менеджерам видеть, успевает ли команда протестировать функционал к дедлайну или есть отставания. Платформа автоматически обновляет эти графики на основе статусов тест-сьютов и прогонов.

Все метрики в TestCloud Suite интерактивны: можно проваливаться в подробности (drill-down) – например, кликнуть на сегмент “Проваленные тесты” и увидеть список конкретных упавших тест-кейсов и связанных багов. Также система поддерживает **кастомные метрики** – пользователь может настроить собственный показатель, комбинируя существующие данные (например, “скорость покрытия новых фич тестами” или “коэффициент переоткрытых дефектов”). Метрики обновляются в реальном времени или по расписанию (например, ежедневно), что даёт актуальную картину качества продукта на любой момент времени. Таким образом, TestCloud Suite не только помогает выполнять тестирование, но и **измерять его эффективность**, обеспечивая прозрачность для всех уровней – от инженеров до руководства.

Архитектура платформы TestCloud Suite

TestCloud Suite построен по современной многослойной архитектуре, обеспечивающей масштабируемость и отказоустойчивость системы. Ниже представлена диаграмма архитектуры платформы, включающая основные компоненты:

Рисунок 1. Архитектура TestCloud Suite: компоненты фронтенда, бэкенда и инфраструктуры.

Как показано на архитектурной схеме, система разделена на несколько слоёв:

- **Клиентский уровень (Frontend):** Взаимодействие пользователей происходит через веб-интерфейс (Single Page Application), доступный в браузере. Веб-клиент реализован с использованием современных фреймворков (например, React/Vue/Angular) и предоставляет удобный UI для всех модулей платформы. Весь функционал – от работы с тест-кейсами до просмотра отчетов – доступен через единый SPA, который обращается к backend по REST API. Безопасность обеспечивается через HTTPS и механизм JWT-токенов для аутентификации сессий.

- **Сервер приложений (Backend Services):** Серверная часть реализована как набор микросервисов (или модулей в пределах одного приложения), каждый из которых отвечает за свой участок функциональности. В частности, в backend входят сервисы: менеджера тестов, баг-трекера, API-тестирования, UI-автохаба, а также сервисы аутентификации/разграничения доступа и нотификаций. Модули бэкенда реализуют бизнес-логику: создание/редактирование объектов, проведение вычислений метрик, оркестрацию запуска автотестов и т.п. Между фронтендом и бэкендом происходит обмен по четко определенным API – это облегчает масштабирование, позволяя выносить тяжелые операции в фон. Например, при запуске набора автотестов UI фронтенд посылает

запрос в соответствующий сервис бэкенда, который ставит задачу в очередь и сразу возвращает управление пользователю, продолжая обработку асинхронно.

- **Инфраструктурный уровень:** включает базы данных и другие инфраструктурные компоненты. Основное хранилище данных – **реляционная БД** (например, PostgreSQL или MySQL), где в таблицах хранятся все сущности системы: проекты, пользователи, тест-кейсы, результаты прогонов, баги и прочее. Для ускорения некоторых операций могут применяться кэши (Redis) – например, кэширование часто запрашиваемых отчетов. **Файловое хранилище** (облачное S3-хранилище либо сервер в файловой системе) используется для сохранения вложений: скриншотов, логов, экспортированных отчетов. **Очередь сообщений** (например, RabbitMQ или Apache Kafka) применяется для ассинхронной обработки задач – таких как выполнение автоматических тестов, рассылка уведомлений, генерация отчетов. Бэкенд-сервисы помещают задания в очередь, благодаря чему тяжелые операции выполняются вне основного потока запросов и не тормозят работу UI.

- **Агенты выполнения тестов:** отдельный компонент архитектуры – это **агенты тестирования**, или исполнительные узлы, на которых запускаются автоматические тесты (особенно UI). Агенты представляют собой выделенные серверы или контейнеры, где установлены необходимые браузеры, эмуляторы устройств и т.п. для прогона тестов. Они подключены к системе через очередь: агент получает из очереди задачу (например, выполнить тестовый набор №5 на Chrome/Windows), выполняет её, и отправляет результаты обратно (либо напрямую в бэкенд-сервис, либо также через очередь). Такой подход позволяет горизонтально масштабировать нагрузку – можно добавлять сколь угодно много агентов, и очередь будет распределять между ними тестовые задания. Если агент выходит из строя, задачи перераспределяются на другие, обеспечивая отказоустойчивость автозапусков.

- **Внешние интеграции:** TestCloud Suite может взаимодействовать с внешними сервисами. На схеме это показано условным блоком “Email/SMS шлюз” – через который система отправляет почтовые уведомления или смски. Аналогично могут подключаться и другие интеграции: вебхуки в Slack, соединение с внешними баг-трекерами или CI/CD. Внешние системы взаимодействуют с платформой либо через API, либо через обмен сообщениями в очереди (для больших объемов данных). Все интеграционные точки защищены и настраиваются администратором.

На рисунке стрелками показаны основные потоки данных: пользователи через браузер обращаются к фронтенду (SPA), SPA отправляет REST-запросы к бэкенду. Бэкенд читает/записывает данные в базу, сохраняет файлы в хранилище. Для тяжёлых задач бэкенд ставит сообщение в очередь. Агенты-тестраннеры читают очередь, выполняют тесты, и затем результаты тестов возвращаются (помечено штриховой стрелкой) обратно в бэкенд, который сохраняет их в базу. Модуль уведомлений следит за событиями (через ту же очередь) и отправляет оповещения вовне (например, на email). Тестируемое приложение (SUT, System Under Test) показано справа пунктиром – оно не является частью TestCloud Suite, но на него направляются тестовые агенты для выполнения UI/API тестов.

Такая архитектура обеспечивает **масштабируемость** – можно независимо масштабировать фронтенд (например, запустить несколько экземпляров SPA на разных CDN), бэкенд-сервисы (в контейнерах Kubernetes), и кластер агентов тестирования. Также достигается **изоляция модулей** – сбой в одном сервисе (скажем, баг-трекере) не выведет из строя другие. Коммуникация через очередь повышает отказоустойчивость: если все агенты заняты или временно недоступны,

задачи будут ожидать в очереди, не теряясь. Использование облачных хранилищ позволяет гибко наращивать объемы данных (например, растёт число вложений – подключается дополнительный storage).

С точки зрения технологий, платформа построена на стеке, проверенном для высоких нагрузок: масштабируемый веб-сервер (например, Node.js, Java Spring Boot или Django) для API, реляционная база (PostgreSQL) для надежного хранения транзакционных данных, брокер сообщений (RabbitMQ) для очередей и контейнеризация (Docker/Kubernetes) для деплоя агентов и сервисов. Безопасность данных обеспечивается аутентификацией JWT, ролями (упомянутыми выше) и шифрованием трафика. Также реализованы механизмы резервного копирования базы и версионирования конфигураций.

В целом, архитектура TestCloud Suite спроектирована с учетом требований корпоративного уровня: она **масштабируется под большие QA-отделы**, надежна (нет единой точки отказа) и легко расширяема для подключения новых модулей и интеграций.

ER-диаграмма основных сущностей и связей

Для хранения данных TestCloud Suite использует реляционную базу данных, структура которой включает несколько основных сущностей, связанных между собой. Ниже представлена ER-диаграмма (упрощенная модель данных) с ключевыми таблицами, их полями и связями:

Рисунок 2. ER-диаграмма данных TestCloud Suite: сущности и связи между ними.

На диаграмме отражены следующие **сущности** (таблицы) и их атрибуты: - **Пользователь:** хранит информацию о пользователях системы. Поля: `user_id` (первичный ключ, UUID), имя, email, глобальная роль (например, ADMIN или просто базовая роль). Пользователь может состоять в нескольких проектах с разными ролями – эта связь реализована через отдельную таблицу *УчастникПроекта*. - **Проект:** представляет собой проект или продукт, качество которого обеспечивает команда. Поля: `project_id` (PK), название проекта, описание, владелец (owner_id – внеш. ключ на Пользователя, отвечающего за проект). В системе может быть несколько проектов (multi-tenant), и каждый проект имеет свою группу тестов, багов и команду. - **УчастникПроекта:** таблица для реализации связи многие-ко-многим между Пользователями и Проектами с указанием роли. Поля: составной первичный ключ из `user_id` + `project_id` (оба FK), и поле `роль` (Enum), определяющее права данного пользователя в этом проекте. Например, запись (user=Ivan, project=MobileApp, роль=QA Engineer) означает, что Иван – тестировщик в проекте MobileApp. Эта таблица отражает команду проекта и их роли. - **Тест-кейс:** основная сущность менеджера тестов, описывающая отдельный тестовый сценарий. Поля: `testcase_id` (PK), название/заголовок теста, тип (enum: Manual / Automated / API и т.д.), подробное описание может храниться в связанной таблице шагов (не показано для упрощения). Также FK `project_id` – указывает, к какому проекту относится тест-кейс. Поле `создал` (created_by, FK на Пользователя) фиксирует автора тест-кейса. Тест-кейсы могут быть объединены в тест-наборы (TestSuite), связь показана пунктиром – один набор включает много тест-кейсов. - **Тестовый набор (TestSuite):** сущность для группировки тест-кейсов. Поля: `suite_id` (PK), название набора, FK `project_id`. Связь: один тест-набор может содержать несколько тест-кейсов (1:N). (Замечание: в реальности возможна связь многие-ко-многим между наборами и кейсами, но в MVP можно ограничиться простым группированием). - **Прогон тестов (TestRun):** сущность, представляющая запуск набора тестов (например, регрессионный прогон перед релизом). Поля: `run_id` (PK), название прогона (или идентификатор сборки), дата запуска, FK `project_id` (к какому проекту относится прогон), и `запустил` (executed_by, FK на Пользователя, который

инициировал запуск). Прогон связан со множеством результатов тестов. - **Результат теста (TestResult):** сущность, фиксирующая результат выполнения конкретного тест-кейса в рамках конкретного прогона. Поля: `result_id` (PK), статус (enum: Passed, Failed, Blocked и т.п.), FK `testcase_id` (какой тест-кейс выполнялся), FK `run_id` (в рамках какого прогона) и опционально `bug_id` (FK на дефект, если по результату этого теста заведен баг). Связи: один TestRun имеет много TestResult (1:N), и каждый TestCase может иметь много результатов за разные прогоны (1:N). Эта таблица связывает тесты, прогоны и баги между собой. - **Дефект (Bug):** сущность баг-трекера, описывающая зарегистрированный дефект. Поля: `bug_id` (PK), заголовок, статус (enum: Open, In Progress, Resolved, Closed, Reopen и пр.), важность/приоритет (Severity/Priority, enum), FK `project_id` (привязка к проекту). Кроме того: `автор` (reported_by, FK на Пользователя, кто завёл баг) и `исполнитель` (assignee_id, FK на Пользователя, кому баг назначен на исправление). Связи: каждый проект имеет много багов (1:N). Также, один баг может быть связан с несколькими TestResult (если один дефект вызывает падение нескольких тестов) – на схеме это показано как связь 1 ко многим (один баг – множество результатов), пунктиром.

На ER-диаграмме серыми линиями показаны некоторые дополнительные связи (логические, не обязательно реализованные внешними ключами): например, Пользователь **создает** тест-кейс, запускает тест-ран, заводит баг и назначается на баг. Эти связи отражают ответственность, они могут не иметь отдельных таблиц (просто хранятся как поля FK, что мы и видим).

Типы данных в моделях выбраны с учетом требований: идентификаторы – UUID (глобально уникальные, удобно в распределенной системе), строки названий – varchar (обычно длиной 255), описания – text (для шагов тест-кейсов, описаний багов могут храниться большие тексты). Статусы и роли заданы через ENUM для удобства. Даты – формат datetime (например, ISO-формат в базе). Чувствительные данные (пароли пользователей, токены) в схеме не показаны, но хранятся в отдельной таблице пользователей, зашифрованными.

Эта ER-модель представляет упрощенный взгляд на базу данных TestCloud Suite. При реализации MVP фокус сделан на необходимых сущностях: проектах, тестах, результатах и багах, а также управлении пользователями и правами. По мере развития системы (см. Roadmap) модель будет расширяться – появятся сущности для требований, исторических данных, шаблонов тест-кейсов и др. Однако уже на этапе MVP такая структура данных позволяет обеспечить **целостность и связность** информации: тесты связываются с багами, баги – с пользователями, тесты – с проектами и пр. Это дает прочную основу для построения сквозной аналитики и удобного функционала для конечных пользователей.

Детальный Roadmap MVP (90 дней)

Запуск TestCloud Suite запланирован в виде MVP (минимально жизнеспособного продукта), реализованного за **90 дней** интенсивной разработки. Ниже представлен поэтапный план работ по неделям, показывающий какие функции будут готовы на каждом этапе трехмесячного цикла:

1. **Неделя 1:** Старт проекта, сбор и финализация требований. Подготовка архитектуры и проектирование базы данных. Настройка инфраструктуры (репозиторий, CI/CD, базовая облачная среда). Итог недели: готов подробный план разработки, утверждена архитектура и ER-модель, развернуто пустое веб-приложение с шаблонной страницей (проверка технологического стека).
2. **Неделя 2:** Реализация модуля аутентификации и управления пользователями. Добавление возможностей регистрации пользователей, входа в систему, восстановления пароля. Внедрение ролевой модели (базовые роли, привязка пользователя к проекту). Итог:

пользователи могут зарегистрироваться, админ – создавать проекты и добавлять членов команды с ролями.

3. **Неделя 3:** Разработка основного бэкенда для **Test Manager**. Создание API для тест-кейсов: добавление/редактирование/удаление тест-кейса, структура данных для шагов теста. Простая веб-страница для списка тест-кейсов проекта (пока без красивого UI). Итог: можно заносить тест-кейсы в систему и сохранять их в БД; отображается список тестов.
4. **Неделя 4:** Дополнение функционала Test Manager: поддержка **тестовых наборов** (Test Suite) и организации тест-кейсов по наборам/категориям. Реализация связанной логики на фронтенде (древовидный просмотр тест-сьютов и тестов). Добавление возможности планировать **тест-прогоны**: создается сущность TestRun с выбором набора и назначением исполнителя(ей). Итог: базовый функционал менеджера тестов завершён – тест-кейсы можно группировать и планировать к выполнению.
5. **Неделя 5:** Реализация модуля **Bug Tracker** (часть 1). Создание API для багов: добавление нового бага, изменение полей статуса, списка багов по проекту. Отображение списка багов на фронтенде, форма создания бага (в минимальном виде). Баги пока не связаны с тестами напрямую – это на следующей неделе. Итог: система позволяет создавать баг-репорты и просматривать их, обеспечивая базовый баг-трекинг внутри платформы.
6. **Неделя 6:** Интеграция Test Manager и Bug Tracker. Добавление возможности при провале теста сразу завести баг и связать его с тест-кейсом/прогоном (реализация поля bug_id в TestResult). На фронтенде – кнопка “Завести баг” в интерфейсе выполнения теста, предзаполнение шагов. Также реализуется смена статуса теста на “Failed” автоматически при создании бага. Итог: единый поток “нашёл баг при прохождении теста -> зарегистрировал дефект -> связал с результатом” работает внутри системы.
7. **Неделя 7:** Модуль **API Test Workspace** (MVP-версия). Интеграция сторонней библиотеки для отправки HTTP-запросов (например, Postman SDK или собственная реализация на основе fetch/axios). Реализация на фронтенде формы для выполнения одиночного API-запроса: ввод URL, метода, заголовков, тела и получение ответа. Пока без сохранения – просто “песочница”. Итог: пользователи могут через UI отправить запрос к API и увидеть ответ (JSON, XML) с подсветкой, что закладывает фундамент API Workspace.
8. **Неделя 8:** Развитие **API Test Workspace** – сохранение API-тестов и автоматизация. Добавление сущности “API тест-кейс” в базу, позволяющей сохранять набор запросов (коллекцию). В UI – возможность сохранить выполненный запрос как тест-кейс, указать ожидаемые проверки (например, статус 200). Настройка запуска таких тестов вручную и сохранения результата (Pass/Fail) в TestResult. Итог: API-тесты стали полноправной частью TestCloud Suite – их можно сохранять и учитывать при прогоне (например, включить API тесты в общий TestRun).
9. **Неделя 9:** Модуль **UI Automation Hub** (MVP-часть). Разработка очереди задач и простого агентского скрипта. Реализация возможности указывать для тест-кейса тип “Авто (UI)” и поле с путем к тест-скрипту (например, имя класса Selenium). Настройка тестового агента (отдельного процесса), который раз в N секунд опрашивает очередь задач. Пока агенты разворачиваются локально (для MVP – 1 агент). Итог: при пометке автотеста и запуске прогона, система ставит задачу в очередь, агент ее читает и возвращает фиктивный результат (например, всегда Pass). Этот каркас позволит подключить реальные тесты позже.
10. **Неделя 10:** Улучшение **UI Automation Hub** – выполнение реальных тестов. Интеграция с Selenium WebDriver: настройка запуска браузера на агенте, выполнение тестовых сценариев. Например, агент может запускать заранее написанные тесты (подключаемые как библиотека) или выполнять шаги, указанные в описании теста (MVP: запустить браузер, открыть URL, сделать скриншот). Сбор результатов: агент отправляет обратно лог (вывод консоли или отметка Pass/Fail). В платформе – отображение результатов автотестов

вместе с ручными. Итог: базовые авто-тесты UI могут реально запускаться и возвращать статус в систему.

11. **Неделя 11:** Разработка **модуля отчетности и метрик**. Создание страниц дашбордов: график состояния тестов (прошло/упало), количество найденных багов по приоритетам, прогресс спринта и т.п. Реализация агрегирующих запросов к БД для вычисления описанных выше метрик. Внедрение библиотеки для графиков (например, Chart.js) на фронтенде. Итог: первые версии отчетов готовы – QA-менеджеры могут видеть ключевые индикаторы качества проекта.
12. **Неделя 12:** Тестирование, багфиксы и улучшения UX. Последняя фаза перед релизом MVP посвящена тщательному **тестированию самой платформы**: команда проходит чек-листы на все реализованные функции, исправляет обнаруженные дефекты. Проводится нагрузочное тестирование ключевых модулей (несколько параллельных прогонов автотестов, большое число багов) и оптимизация запросов/кода при необходимости. Также вносятся мелкие улучшения интерфейса на основе отзывов первых пользователей (например, удобство формы создания теста, навигация между модулями). Итог: MVP-версия готова к деплою – все запланированные функции работают стабильно, система развернута в облаке и доступна пилотным пользователям.

По завершению этих 12 недель (90 дней) TestCloud Suite в версии MVP предоставит базовый, но функциональный набор возможностей: управление тестовой документацией, ведение багов, выполнение API и UI тестов (в ограниченном масштабе), сбор основных метрик. Такой поэтапный подход гарантирует, что к концу срока продукт покрывает минимальные потребности QA-команды и может быть продемонстрирован стейкхолдерам или ранним клиентам. Далее планируется фаза сбора обратной связи и доработка дополнительных возможностей (см. следующий раздел).

Модели монетизации и тарифные планы

TestCloud Suite предлагается клиентам по модели **Software-as-a-Service (SaaS)**, что предполагает регулярную оплату подписки за использование платформы. При разработке монетизации были учтены рыночные ориентиры и потребности различных сегментов пользователей. Ниже описаны ключевые модели и тарифы:

- **Freemium (бесплатный базовый план):** Для привлечения небольших команд и ознакомления с продуктом предусмотрен бесплатный тариф. Он включает ограниченный функционал: например, до 5 пользователей, 1-2 проекта, ограничение на количество тест-кейсов (скажем, 200) и на автоматизированные запуски (например, 50 запусков автотестов в месяц). В бесплатном плане может быть урезана мощность (ограничено число агентов или очередей). Однако базовые возможности (Test Manager, Bug Tracker, ручное тестирование) полностью доступны. Цель freemium – позволить маленьким командам стартапов или отдельных QA-специалистов начать работу без финансовых барьеров, с перспективой роста до платных планов.

- **Стандартная подписка (Cloud Team):** Основной платный тариф, рассчитанный на небольшие и средние команды QA. Предположительно, цена составит порядка **$10–20 за пользователя в месяц** при помесячной оплате (при годовой – скидка). Например, план *Team-10* за $100/мес включает до 10 пользователей, неограниченное число проектов, до 5000 тест-кейсов, 2000 автоматических тест-запусков в месяц. В этот тариф входят все модули (Test Manager, Bug Tracker, API, Automation) и базовая поддержка. Ценообразование конкурентоспособно: для сравнения, аналоги на рынке берут от ~$9 до $49 за

пользователя в месяц [9] [10], поэтому наш стандартный план расположен в нижней части этого диапазона, чтобы привлечь аудиторию.

- **Расширенный план (Business/Enterprise):** Для крупных организаций с большими командами и повышенными требованиями предлагаются расширенные тарифы. Это может быть план *Business* (например, до 50 пользователей, ~$15 за пользователя в месяц) и *Enterprise* (100+ пользователей, с индивидуальной ценой и скидками при большом объеме). В данных планах снимаются большинство ограничений: неограниченные тест-кейсы, расширенные ресурсы (приоритетное место в очереди агентов, больше параллельных запусков), а также добавляются **премиальные функции**. Например, интеграция с LDAP/Active Directory для единого входа, возможность развертывания выделенных агентов на стороне клиента, повышенный уровень поддержки (выделенный менеджер, SLA на реакции поддержки). Также Enterprise-клиентам доступны кастомизация под бренд, отдельный инстанс (если требуется изоляция данных), и ранний доступ к новым функциям.

- **Дополнительно – оплата за использование:** В дополнение к пользователя-ориентированным тарифам, планируется элемент usage-based оплаты для облакоёмких функций. Например, автотесты UI потребляют ресурсы (агенты, вычислительное время), поэтому сверх включенного лимита запусков в тарифе может взиматься плата **за минуту выполнения теста** или **за час работы агента**. Аналогично, хранение больших объемов вложений сверх квоты (скажем, более 50 ГБ) может тарифицироваться отдельно (несколько центов за гигабайт). Такая модель “pay-as-you-go” позволит командам, у которых нерегулярные пики нагрузки, гибко платить только за фактически использованные ресурсы, не переходя на более дорогой план постоянно.

- **Marketplace и расширения:** Хотя MVP-функциональность включена в тарифы, в будущем планируется собственный **Marketplace** расширений/плагинов. Партнеры или мы сами сможем предлагать дополнительные модули (например, модуль нагрузочного тестирования, интеграция с специфическими инструментами) за отдельную плату. Пользователь сможет докупать такие расширения “à la carte”. Это дополнительный канал монетизации без повышения базовой цены.

Для прозрачности, на сайте TestCloud Suite будет опубликована таблица тарифных планов, где перечислены ограничения каждого (кол-во пользователей, проектов, хранилище, лимиты API/UI запусков) и стоимость. Также будут указаны скидки: например, -20% при оплате за год вперёд, специальные условия для образовательных или open-source проектов (бесплатно или со скидкой). Такая тарифная сетка позволит охватить разные сегменты: от фрилансеров и стартапов (free/cheap) до enterprise-команд (premium).

Важно отметить, что выбранная SaaS-модель подразумевает постоянное обновление платформы для всех клиентов (мультиарендная среда). Мы берём на себя хостинг и поддержку – это обосновывает ежемесячную оплату и снимает головную боль у клиентов по установке и обновлению. При этом для некоторых крупных клиентов, предъявляющих требования по безопасности, может быть предложен **on-premise вариант** (развёртывание на их серверах) за отдельную цену (обычно значительно выше, с ежегодной оплатой поддержки). Но на старте фокус делается на облачной подписке.

Подводя итог: **монетизация TestCloud Suite комбинирует доступность и масштабируемость**. Бесплатный план снижает порог входа, стандартные подписки по конкурентной цене делают продукт привлекательным на фоне аналогов, а enterprise-предложения и add-on услуги

позволяют выстраивать долгосрочные отношения с крупными клиентами. Такой гибкий подход обеспечит приток пользователей на ранних этапах и рост выручки по мере расширения функционала платформы.

Будущие модули и расширения

После выпуска MVP и сбора первых отзывов, у TestCloud Suite запланировано активное развитие. Платформа задумана как **масштабируемый "комбайн" для QA**, поэтому в будущем будут добавляться новые модули и улучшаться существующие. Ниже перечислены наиболее вероятные и востребованные расширения функционала, запланированные в roadmap после MVP:

- **Модуль управления требованиями (Requirements Management):** Позволит связывать тест-кейсы с конкретными требованиями или user story. Пользователи смогут заводить в систему списки требований, user stories или спецификаций, а затем линковать к ним тесты. Это даст полноценную **трассируемость**: от требования – к тест-кейсу – к результату – к багу. Кроме того, появится метрика покрытия требований тестами. Такой модуль особенно актуален для больших проектов с формализованными требованиями (например, в банковской сфере или разработке по V-модели).

- **Нагрузочное и производительное тестирование:** Планируется интеграция с инструментами нагрузочного тестирования (JMeter, Gatling и др.) либо собственный модуль. QA-инженеры смогут задавать сценарии нагрузки (число виртуальных пользователей, скрипты) и запускать их из TestCloud Suite. Результаты (черезput, время отклика, количество ошибок под нагрузкой) будут визуализироваться в отчетах. Это расширение позволит командам выполнять **performance testing** в том же облаке, где ведутся функциональные тесты, и хранить результаты вместе с остальными тестовыми артефактами.

- **Модуль тестирования безопасности (Security Testing):** В перспективе платформа может обзавестись инструментами для статического анализа безопасности (SAST) и динамического тестирования безопасности (DAST). Например, интеграция с популярными сканерами уязвимостей (OWASP ZAP) для веб-приложений. Результаты сканирования уязвимостей могут оформляться как специальные типы дефектов в баг-трекере. Это позволит закрыть аспект качества, связанный с безопасностью, в общей экосистеме QA.

- **Библиотека тестовых случаев и переиспользование:** Добавление возможности создавать **шаблоны тест-кейсов** и шагов, а также их параметризация. Например, модуль "Library" где хранятся общие шаги (логин, навигация) или целые сценарии, которые можно включать в разные тест-кейсы (проявление принципа DRY – Don't Repeat Yourself в тест-дизайне). Также планируется внедрить **генератор тест-кейсов** на основе шаблонов: пользователь задает комбинации параметров – система автоматически создает серию тестов (техники эквивалентных классов и граничных значений можно будет автоматизировать).

- **AI-ассистенты и смарт-анализ:** В более отдаленной перспективе (с учетом трендов 2024-2025 гг) – интеграция элементов искусственного интеллекта. Например, AI-ассистент для **автоматического анализа упавших тестов**: на основании стека ошибок и шагов воспроизведения предлагать вероятную причину бага или соответствующий ранее найденный дефект. Либо генерировать черновики баг-репортов (описание) на основе

логов. Возможен AI-помощник для **генерации тест-кейсов** из описания требования (на естественном языке) – это ускорит процесс написания тестовой документации. Такие функции повысят интеллектуальность платформы и снизят рутинную нагрузку на инженеров.

- **Мобильное тестирование и интеграция с устройствами:** Добавление поддержки мобильных платформ: интеграция с облаками реальных устройств (BrowserStack, AWS Device Farm и др.) либо собственный модуль Mobile Hub. Это позволит запускать автоматические тесты мобильных приложений на различных устройствах прямо из TestCloud Suite и получать результаты аналогично веб-тестам. Также может быть реализовано управление ручным мобильным тестированием – например, распределение устройств, учёт тестового покрытия по моделям/OS и т.д.

- **Расширенные интеграции с CI/CD:** Хотя базовые интеграции присутствуют, будет развиваться глубокая связь с конвейерами CI/CD. Например, плагин для Jenkins/GitLab, который после билда автоматически создаёт TestRun в TestCloud Suite, запускает соответствующие тесты и возвращает статус. Таким образом, результаты тестирования отобразятся и в CI (для разработчиков) и в TestCloud (для истории и аналитики). Также возможна интеграция с инструментами отчетности типа Allure – импорт результатов автотестов, если команды уже имеют наработки.

- **Улучшения UX и collaboration:** Появится модуль коллективной работы – например, **комментарии в реальном времени**, упоминания коллег (@username) внутри системы для обсуждения тестов и багов. Интеграция с мессенджерами (Slack, Microsoft Teams) для отправки уведомлений о статусе тестирования прямо в каналы команд. Также планируется **борда задач для тестирования** – визуальное представление (канбан) прогресса тест-планов и устранения дефектов, чтобы QA-тимлиды могли управлять процессом как задачами.

- **Отчеты для руководства и кастомизация дашбордов:** Будут добавлены более гибкие инструменты аналитики – конструктор отчетов, где пользователь (например, QA-менеджер) сможет сам создавать отчеты, выбирая метрики, фильтры, группировки. Например, собрать отчет “Дефекты по компонентам за последний квартал” или “Тренд покрытия тестами по спринтам”. Это важное расширение для удовлетворения специфических потребностей разных компаний в отчетности.

- **Локализация и многоязычность:** После выхода на англоязычный рынок, добавим поддержку нескольких языков в интерфейсе (интернационализация), чтобы команды по всему миру могли использовать платформу на родном языке. Также, возможно, интеграция с переводческими сервисами для многоязычных проектов (перевод описаний тестов/дефектов).

Естественно, приоритезация этих модулей будет зависеть от обратной связи пользователей. Первочерёдно выглядят требования и нагрузочное тестирование, так как они логично дополняют функционал MVP. Далее – AI-функции, поскольку это конкурентное преимущество на рынке QA-инструментов.

TestCloud Suite изначально строится как **платформа**, готовая к расширению. Добавление нового модуля не нарушит работу других благодаря модульной архитектуре. Кроме того, будет открыт

API и SDK для партнеров, желающих разрабатывать плагины – это ускорит появление новых функций.

В итоге, видение продукта – стать **универсальной средой для обеспечения качества**, охватывающей все – от постановки требований до анализа впечатления пользователей. Шаг за шагом, через перечисленные расширения, TestCloud Suite будет приближаться к этому видению, оставаясь при этом удобным и производительным инструментом для каждой роли в QA-команде.

Ключевые отличия от существующих решений

Рынок инструментов для управления тестированием и баг-трекинга достаточно насыщен – существуют как узкоспециализированные решения, так и крупные комплексные платформы. Тем не менее, TestCloud Suite выгодно отличается от большинства из них по ряду параметров:

- **Полная интеграция “all-in-one”:** В отличие от сценария, когда компании вынуждены связывать между собой отдельный TMS, отдельный баг-трекер, инструмент для API и инфраструктуру для автотестов, TestCloud Suite предоставляет всё это из одного окна. Многие популярные связки требуют настройки интеграций (например, TestRail + Jira + Jenkins), что может быть сложно и ненадёжно. Наша же платформа изначально спроектирована целостно – модули работают на общей базе данных и имеют единый UX. Это устраняет проблемы синхронизации и “сводит” всю информацию о качестве продукта в единое хранилище [2]. Для команды это означает меньшие накладные расходы и риски – не нужно переключаться между системами, вручную дублировать данные или поддерживать плагины интеграции.

- **Облачность и готовность к масштабированию:** В отличие от старых решений, которые устанавливаются on-premise и требуют усилий по поддержке (например, HP ALM/QC или Jira Server с плагинами), TestCloud Suite – облачный SaaS. Это значит, что пользователи получают доступ из браузера в любое время, обновления устанавливаются автоматически, а масштабирование происходит прозрачно. Команда может начать с малого проекта и, вырастая, не беспокоиться о производительности – облачная архитектура (см. выше) позволит обслужить и 10, и 1000 пользователей без смены инструмента. Многие конкуренты либо вообще не предлагают облако, либо имеют ограничения по многопользовательской работе (например, открытые инструменты типа TestLink не так хорошо масштабируются). Наше решение изначально заточено под крупные распределённые команды.

- **Единый UI и удобство использования:** Платформа создавалась с акцентом на современный, интуитивно понятный интерфейс, объединяющий разные функции. В существующих решениях часто **разнородный опыт** – например, автотесты запускаются через CI с одними отчетами, ручные тесты ведутся в таблицах с другими отчетами, баги – вообще в стороннем интерфейсе. В TestCloud Suite всё оформлено в одном стиле: списки, фильтрация, панели управления унифицированы для тестов и багов. Есть глобальный поиск по системе (найти тест или баг по ключевым словам). Поддерживаются упоминания, комментарии – словно в социальных корпоративных сетях, чего нет во многих устаревших системах. Всё это снижает порог входа: новые пользователи быстрее освоят инструмент, а команда будет работать эффективнее, не теряя время на переключение контекста.

- **Объединение ручного и автоматизированного тестирования:** Многие инструменты на рынке фокусируются либо на ручном управлении тестами (как TestRail, PractiTest), либо на

автоматизации (как SauceLabs, BrowserStack для запуска тестов). TestCloud Suite изначально задуман объединить эти миры. В нашей платформе результаты ручных тест-кейсов и автотестов сводятся вместе: на дашборде видно общее состояние качества, независимо от того, как тест выполнялся. Автотесты можно запускать прямо из интерфейса (без необходимости лезть в Jenkins или запускать скрипты локально), и их результаты автоматически прикрепляются к соответствующим сценариям. Это большое преимущество: **“мануал” и “автоматизация” идут рука об руку**, повышая общую эффективность. Конкуренты только начинают двигаться в этом направлении – например, Azure DevOps и некоторые новые TMS заявляют о подобном подходе [7], но TestCloud Suite делает это более гибко и агностично к технологиям (мы не заставляем писать автотесты на конкретном инструменте, а поддерживаем разные).

- **Встроенный баг-трекинг:** В то время как многие QA-платформы по-прежнему полагаются на внешние баг-трекеры (ориентируясь на интеграцию с Jira и аналогами), мы предоставляем полноценный баг-трекер внутри. Это значит, что даже если компания не использует какой-либо отдельный инструмент управления разработкой, команда тестирования полностью самодостаточна с TestCloud Suite. Для тех же, кто использует, скажем, Jira – мы предложим двунаправленную синхронизацию, однако работа с дефектами может происходить в нашем удобном интерфейсе, специально заточенном под QA- процессы (сессии тестирования, воспроизведение багов и т.п.). Отсутствие необходимости постоянно переключаться в Jira – **большое преимущество по отзывам тестировщиков**, экономящее их время и нервные клетки. К тому же наш баг-трекер изначально увязан с тестами, что многие конкуренты дать не могут [6].

- **Гибкость и настройка под процессы:** TestCloud Suite сочетает **богатый функционал корпоративных систем** (как HP ALM/Octane, Zephyr Enterprise) с гибкостью легковесных современных приложений. Мы позволяем настроить все – от статусов и полей до интеграции с CI – но при этом интерфейс остаётся лёгким и не перегруженным лишним для тех, кому это не нужно. Существующие комплексные решения зачастую сложны в освоении и “неповоротливы” (долгое настройка, избыточность для малого проекта), а простые – напротив, не дотягивают по возможностям кастомизации. Наш продукт спроектирован так, чтобы быть полезным и стартапу, и enterprise: ненужные модули можно отключить, необходимые – настроить под свой workflow. Этот баланс выделяет TestCloud Suite на фоне монолитных конкурентов.

- **Стоимость владения и экономическая эффективность:** Наконец, немаловажное отличие – **цена**. Мы намеренно выбрали демократичную ценовую политику (см. предыдущий раздел). Многие аналогичные решения могут быть дорогими для небольших компаний или проектов. Например, отдельная лицензия TestRail или PractiTest обходится десяткам долларов на пользователя в месяц, плюс отдельная лицензия Jira, плюс затраты на инфраструктуру для автотестов. TestCloud Suite предлагает экономию за счёт объединения: один продукт закрывает все потребности по цене ниже суммарной стоимости всех отдельных. Кроме того, SaaS-модель означает отсутствие капитальных расходов на сервера и их поддержку. Таким образом, для руководства наш инструмент – способ сократить TCO (total cost of ownership) процесса обеспечения качества.

В заключение, **TestCloud Suite отличается комплексностью, интегрированностью и удобством**, оставаясь при этом доступным и гибким. Мы не изобретаем колесо заново в каждой области – а берём лучшее из практик тест-менеджмента, баг-трекинга, автоматизации – и объединяем на единой платформе. Это дает синергетический эффект: 1+1=3, когда целостное решение приносит больше пользы, чем набор разрозненных. Именно это ценят команды QA:

возможность сосредоточиться на качестве продукта, а не на администрировании инструментов. TestCloud Suite стремится стать для QA-инженеров тем же, чем полнофункциональные среды разработки стали для программистов – **универсальным рабочим пространством, значительно повышающим продуктивность**.

Источники: Основные идеи и преимущества, заложенные в TestCloud Suite, соответствуют современным трендам в QA-индустрии и подтверждаются обзорами инструментов [1] [7] [6], а также опытом успешных платформ, таких как PractiTest [2].

---

[1] [3] [4] [6] [7] Топ-10 лучших систем управления тестированием 2021
https://software-testing.ru/library/testing/test-management/3696-top-10-best-test-management-systems-2021

[2] [10] 20 Best Test Management Tools (2025 Updated)
https://www.practitest.com/test-management-tools/

[5] QATouch
https://help.qatouch.com/docs/v2/architectural-basics

[8] [9] 10 лучших инструментов управления тестированием в 2024 году / Хабр
https://habr.com/ru/companies/otus/articles/827670/