Статья поступила в редакцию: 22.09.2026
Статья принята к публикации: 02.10.2026
Статья опубликована: 28.10.2026

УДК 004.415.53

DOI: 10.32743/UniTech.2026.151.10.23504

Методологии автоматизации сквозного тестирования при интеграции платежных шлюзов в SAAS-платформах

Кузнецов Дмитрий
старший инженер по обеспечению качества (Senior QA Engineer), член-основатель отдела обеспечения качества, FunnelFox
Сербия, г. Белград
E-mail: kuznetsovdmitryv@gmail.com

Methodologies for automation of end-to-end testing when integrating payment gateways in SAAS platforms

Kuznetsov Dmitry
Senior Quality Assurance Engineer, Founding Member of the Quality Assurance Department, FunnelFox
Serbia, Belgrade

Аннотация

Предмет исследования составляют методологии автоматизации сквозного тестирования при интеграции платежных шлюзов в SaaS-платформах. Цель работы – разработать аналитическую модель выбора, проектирования и внедрения E2E-проверок, которые связывают пользовательский путь оплаты, API внешнего провайдера, внутренний биллинг и состояние доступа к продукту. Методология основана на сравнительном анализе уровней тестового покрытия, тематическом синтезе публикаций 2021–2026 гг., логико-структурном моделировании платежного сценария и обобщении отраслевой QA-практики. В результате разграничены API, интеграционные, регрессионные и сквозные проверки, определено место E2E-тестов в цепочке PR, dev, beta и production-контроля, предложена риск-ориентированная схема отбора платежных сценариев. В модель включены регуляторно обусловленные ветви EMV 3-D Secure, усиленной аутентификации клиента и ограничения PCI DSS, определяющие допустимые тестовые данные и границы платежного контура. Обоснована целесообразность автоматизации основных способов оплаты, сценариев подписки, отказа, возврата и повторной попытки. Полученная модель ориентирована на команды, которым требуется сократить ручной регресс, сохранить диагностичность падений, отделить дефекты интерфейса от ошибок биллинга и выстроить единые критерии готовности релиза.

Библиографическое описание: Кузнецов Д. Методологии автоматизации сквозного тестирования при интеграции платежных шлюзов в SAAS-платформах // Universum: технические науки : электрон. научн. журн. 2026. 10(151). URL: https://7universum.com/ru/tech/archive/item/23504

Практическая значимость результатов связана с построением устойчивого QA-процесса в SaaS-продуктах, где платежный шлюз сопряжен с биллингом, подписками, пользовательским интерфейсом и релизными решениями в условиях частых продуктовых обновлений SaaS.

Abstract

This study examines end-to-end testing automation methodologies for integrating payment gateways into SaaS platforms. The goal of the study is to develop an analytical model for selecting, designing, and implementing E2E checks that link user payment flows, external provider APIs, internal billing, and product access status. The methodology is based on a comparative analysis of test coverage levels, a thematic synthesis of publications from 2021–2026, logical-structural modeling of payment scenarios, and a summary of industry QA practices. As a result, the study distinguishes API, integration, regression, and end-to-end checks, defines the role of E2E tests across pull request, development, beta, and production stages, and proposes a risk-based framework for selecting payment scenarios. The model also incorporates EMV 3-D Secure branches, strong customer authentication conditions, and PCI DSS constraints on test data and payment-environment scope. It also identifies automation priorities for primary payment methods and for subscription, decline, refund, and retry flows. The resulting model is intended for teams seeking to reduce manual regression effort, preserve failure diagnosability, separate interface defects from billing errors, and establish consistent release-readiness criteria. The practical significance of the results lies in building a sustainable QA process in SaaS products, where the payment gateway is integrated with billing, subscriptions, user interface, and release solutions in the context of frequent SaaS product updates.

Ключевые слова: сквозное тестирование, E2E, платежный шлюз, SaaS, автоматизация тестирования, биллинг, CI/CD, PCI DSS, EMV 3-D Secure, усиленная аутентификация клиента.

Keywords: end-to-end testing, E2E, payment gateway, SaaS, test automation, billing, CI/CD, PCI DSS, EMV 3-D Secure, strong customer authentication.

Введение

Интеграция платежного шлюза в SaaS-платформу связывает интерфейс продукта, внутренний биллинг, API внешнего провайдера, callback-уведомления, статус подписки и доступ к платному функционалу. Проверка отдельного запроса дает ограниченную картину: API возвращает корректный ответ, при этом пользовательский путь завершается ошибкой из-за расхождения статусов, задержки webhook, неверной маршрутизации или сбоя обновления данных в биллинге.

Цель исследования – разработать аналитическую модель автоматизации E2E-тестирования платежных шлюзов в SaaS-платформах с учетом микросервисной архитектуры, регрессионного цикла и релизного пайплайна. Первая задача состоит в разграничении API, интеграционных, регрессионных и сквозных проверок в платежном процессе. Вторая задача связана с определением места E2E-тестов в цепочке PR, dev-среды, beta-релиза и production-контроля. Третья задача направлена на построение прикладной модели внедрения E2E-автоматизации с учетом бизнес-критичности способов оплаты, стоимости поддержки тестов

и стабильности окружения. Четвертая задача состоит в определении регуляторно обусловленных ветвей E2E-проверки, связанных с EMV 3-D Secure, усиленной аутентификацией клиента, или SCA, и границами PCI DSS-скоупа тестовой среды.

Новизна работы состоит в риск-ориентированной модели E2E-автоматизации платежных шлюзов, которая связывает пользовательское действие, API- и контрактные проверки, результат внешней транзакции, состояние биллинга, доступ к SaaS-продукту и релизное решение. В состав модели включены ветви 3-D Secure и SCA, а ограничения PCI DSS рассматриваются как условия выбора тестовых данных, формирования отчетных артефактов и разграничения SaaS-компонентов и среды обработки карточных данных, исходное предположение состоит в том, что практическая ценность E2E-набора определяется охватом критичных платежных переходов.

Материалы и методы

Корпус формировался посредством поиска в Google Scholar и Crossref Metadata Search, а также в издательских и институциональных каталогах IEEE Xplore, SpringerLink, MDPI, SciUp и WebThesis Politecnico di Torino. Поисковые запросы объединяли термины “end-to-end testing”, “E2E testing”, “payment gateway”, “digital payments”, “SaaS”, “API testing”, “integration testing”, “microservices”, “CI/CD” и “regression testing”, а также их русскоязычные соответствия. Отбор проводился последовательно по названию, аннотации и полному тексту.

В корпус включались работы 2021–2026 гг., непосредственно рассматривающие автоматизацию веб- и API-тестирования, взаимодействие микросервисов, E2E-покрытие, регрессию финансовых продуктов, CI/CD-пайплайны либо параметризацию платежных сценариев. Исключались дублирующиеся публикации, материалы, ограниченные модульным тестированием, обзорные тексты без описания метода и коммерческие страницы инструментов. Итоговый научно-технический корпус составили десять работ [1–6, 10–13]. Документы PCI SSC, EMVCo и Европейского союза [7–9] включались независимо от года публикации, поскольку они определяют действующие требования к данным платежных карт, процедурам аутентификации и исключениям из SCA.

Сравнительный анализ применялся для разграничения API-, интеграционных, регрессионных и сквозных проверок, тематический синтез связывал выводы работ по микросервисам, CI/CD и финансовому тестированию с платежным контуром SaaS-платформы, логико-структурное моделирование представляло E2E-сценарий как последовательность переходов между интерфейсом, платежным провайдером, механизмом аутентификации, биллингом и системой управления доступом, риск-ориентированная классификация использовалась при отборе способов оплаты, негативных ветвей и наборов, запускаемых на разных стадиях релиза.

Результаты

Автоматизация платежных интеграций начинается с разграничения уровней проверки. API-тест подтверждает корректность запроса, параметров, ответа и статуса. Интеграционный тест проверяет обмен между сервисами. Регрессионный набор снижает риск повторного дефекта после изменения кода. Сквозной тест проводит сценарий через цепочку действий,

близкую к пользовательскому пути, и фиксирует состояние продукта после оплаты. В работах по API-тестированию микросервисов надежность взаимодействия сервисов связывается с контролем контрактов и форматов обмена данными [1]. Для платежного шлюза этого слоя хватает только для части задачи, поскольку платеж проходит через выбор способа оплаты, внешнюю форму, возврат в продукт, webhook, биллинг и права доступа.

Автоматизированный тест веб-приложения ценен при повторяемой проверке пользовательских действий и результата [2]. В платежном сценарии автотест фиксирует цепочку после клика по кнопке оплаты: тариф передан в биллинг, шлюз получил ожидаемые параметры, транзакция завершилась заданным статусом, SaaS-платформа обновила подписку, пользователь получил доступ. E2E-сценарий связывает технический ответ сервиса с бизнес-результатом.

В микросервисной архитектуре платежный дефект часто возникает на стыке сервисов. Работы по интеграционному тестированию выделяют контрактные и API-проверки как средство согласования обмена между сервисами [3]. При интеграции платежного шлюза этот подход закрывает форматный риск, но ошибки завершения сценария остаются отдельной зоной контроля. Биллинг формирует корректный запрос, шлюз возвращает валидный ответ, затем пользователь получает неверный статус из-за ошибки redirect URL, повторного callback, гонки обновлений или неполного отображения результата.

DevOps-подходы связывают тестирование с быстрой обратной связью после изменения кода [4]. В платежных интеграциях команда QA соотносит тип изменения с глубиной проверки. На pull request запускаются быстрые API и smoke-проверки платежного контура. После слияния в dev-среду выполняется E2E-сценарий для основных способов оплаты. На beta-этапе проводится регрессия критичных платежных сценариев. Production-контроль ограничивается безопасными синтетическими сценариями и мониторингом без реального списания.

Регрессионный контроль в финтех-продуктах опирается на отбор сценариев по риску, частоте изменений и бизнес-критичности [5]. В SaaS-платформе это ведет к приоритету основных способов оплаты, подписки, первичной покупки, продления, отказа, возврата, смены тарифа и повторной попытки. Редкие варианты оплаты сохраняются в карте рисков и переносятся в периодический или ручной контроль, если автоматизация требует высокой поддержки при низкой частоте использования.

Исследователи E2E-покрытия микросервисных систем указывают, что пользовательский сценарий через интерфейс оставляет часть внутренних endpoint-ов без подтвержденного охвата [6]. Для платежных шлюзов это означает потребность в логах, трассировке, перехвате сетевых запросов, correlation ID, проверке статусов в биллинге и отчете по покрытым сервисным переходам. Зеленый happy path подтверждает покупку, но отказ банка, истечение сессии, повторный callback, рассинхронизация статусов и отложенная активация подписки требуют отдельных ветвей.

На рисунке 1 представлена обобщенная схема E2E-сценария оплаты в SaaS-платформе. Схема сохраняет абстрактный уровень описания конкретного продукта и фиксирует типовую логику проверки, где автотест проходит путь пользователя и сверяет состояние внутренних сервисов.

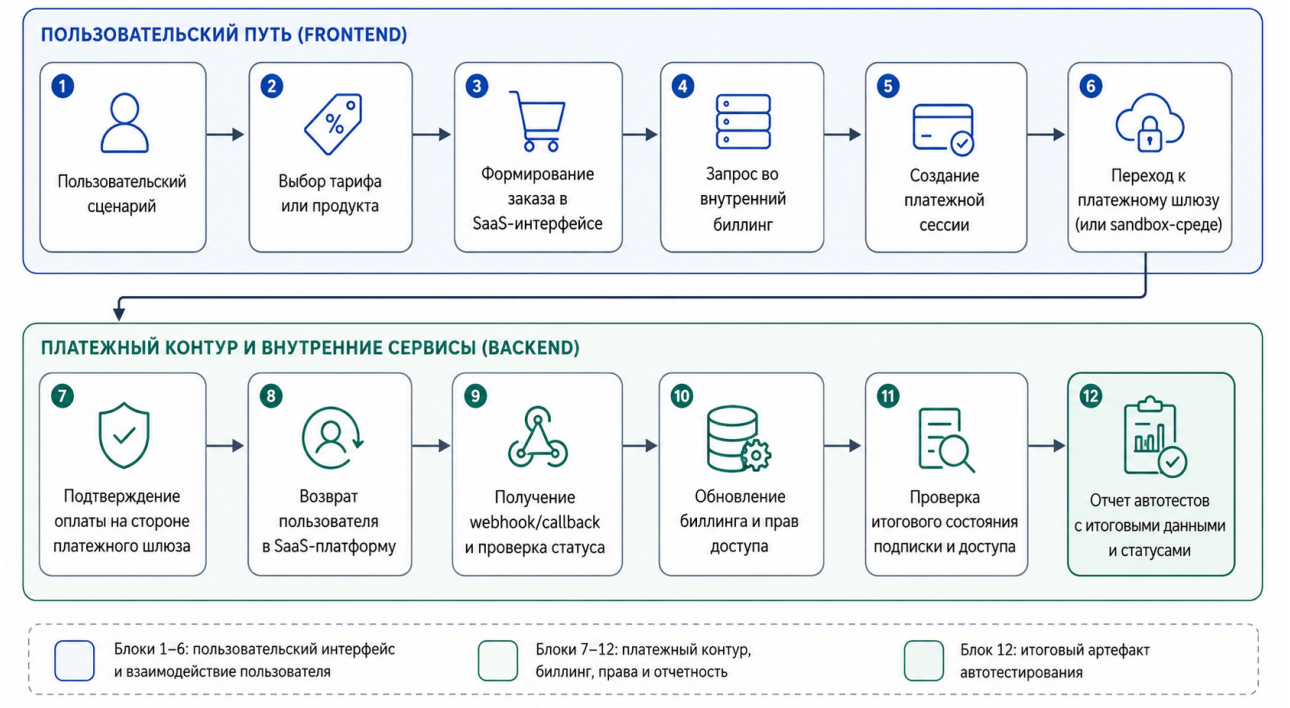

Рисунок 1. Логика автоматизированного E2E-сценария оплаты в SaaS-платформе (составлено автором)

Исследование интеграционных тестов в домене цифровых платежей обосновывает параметризуемый сценарий, где один платежный процесс проверяется с разными входными данными, типами платежей и ожидаемыми статусами [10]. Для SaaS-платформы это означает построение фреймворка через переиспользуемые сценарии. Команда задает базовый сценарий и варьирует способ оплаты, валюту, статус ответа, тип операции, сохраненное платежное средство, подписочный цикл, повторную попытку, refund или отмену.

Систематический обзор непрерывной интеграции связывает CI с качеством обратной связи, координацией команд, сборками и выявлением интеграционных проблем [11]. В платежных шлюзах E2E-тест нужен разработчикам, QA и владельцам продукта до продвижения сборки на следующий стенд. Пайплайн задает правило: какие платежные сценарии блокируют релиз, какие требуют ручного подтверждения, какие переходят в мониторинг после выпуска.

Исследование эволюции CI/CD-пайплайнов фиксирует регулярную перестройку конфигураций, многоинструментальность и изменение состава стадий [12]. Платежная интеграция меняется вместе с SDK, API-версиями, правилами 3-D Secure, webhook-событиями, интерфейсом платежной формы, токенизацией и обработкой отказов. Поэтому инженеры выносят UI-логику в page object или screen object, операции биллинга в API-клиент, работу со шлюзом в gateway adapter, подготовку данных в фабрики и результат прогона в единый отчет.

Работа по автоматическому веб-тестированию с curiosity-driven reinforcement learning рассматривает поиск сложных последовательностей действий, где дефекты возникают при нестандартных комбинациях поведения [13]. Для платежных интеграций это проявляется в отказах и повторах: пользователь закрывает платежную вкладку, возвращается назад,

меняет способ оплаты, получает pending-статус или пытается оплатить уже созданный заказ. Поэтому автоматизация платежей требует стабильных регрессионных сценариев и ограниченного набора исследовательских сценариев.

Сопоставление источников показывает, что платежный процесс требует многослойного теста: API-контракты и схемы данных [1, 3], охват внутренних переходов [6], параметризация статусов и платежных типов [10], проверка нестандартных пользовательских последовательностей [13]. Практический принцип состоит в построении матрицы сценариев. Первая ось описывает пользовательский путь: первая оплата, продление подписки, смена тарифа, отказ, возврат, повторная попытка. Вторая ось задает платежный механизм: банковская карта, сохраненное платежное средство, альтернативный метод, sandbox-статус. Третья ось фиксирует системный результат: статус транзакции, запись в биллинге, webhook, доступ пользователя, уведомление, отчетность.

Практическая ценность E2E-подхода раскрывается через контроль платежного поведения SaaS-продукта. Gateway обрабатывает транзакцию, затем платформа активирует подписку, применяет скидку, сохраняет идентификатор плательщика и переводит заказ в итоговый статус. Поэтому критерий успешности теста включает платежное событие и внутреннее состояние продукта. Методология E2E-автоматизации строится вокруг платежного риска. Выбор Playwright, Selenium, REST Assured или иного инструмента относится к реализации. Основу решения задают пользовательский путь, фиксируемые системные переходы, корректное состояние биллинга и правило блокировки релиза. Инженеры подбирают инструмент после описания сценариев.

Прикладная модель начинается с инвентаризации платежных сценариев: первичная покупка, подписка, продление, отмена, возврат, downgrade, upgrade, промокод, отказ банка, pending-статус, истечение сессии и повторная попытка. Затем команда отбирает приоритетные варианты по частоте использования, финансовым последствиям дефекта, влиянию на доступ пользователя, сложности воспроизведения и стабильности среды. Базовый E2E-набор формируется из малочисленных и надежных сценариев. Таблица 1 сопоставляет этапы внедрения, техническую реализацию и управленческий результат.

Таблица 1. Модель внедрения E2E-автоматизации платежных шлюзов в SaaS-платформе

Этап	Содержание решения	Результат для QA-процесса
Карта платежных сценариев	Описание покупки, подписки, отказа, возврата и продления	Отбор автотестов по риску
Приоритизация сценариев	Выбор главных способов оплаты и операций	Сокращение нагрузки на регрессию
Проектирование тестового слоя	Разделение UI, API, billing и gateway adapter	Снижение зависимости тестов от изменений интерфейса
Включение в CI/CD	Запуск разных наборов на PR, dev, beta и production	Контроль релизных переходов
Наблюдаемость и сопровождение	Логи, запросы, статусы, ревизия сценариев и данных	Быстрый разбор дефектов и устойчивость набора

Следующий блок связан с метриками. Без измеримых признаков E2E-набор превращается в перечень зеленых и красных прогонов, которые трудно сопоставить с релизным риском. Метрики нужны для решений: какой тест блокирует релиз, какой получает статус flaky, какой выводится из набора, какой сценарий команда переносит в ручную проверку.

Таблица 2. Метрики контроля E2E-тестирования платежных интеграций

Группа метрик	Показатель	Практическая интерпретация
Покрытие путей	Доля критичных сценариев в автоматизации	Зависимость релиза от ручного регресса
Стабильность	Доля flaky-тестов	Потребность в ревизии локаторов, ожиданий и данных
Скорость обратной связи	Время прохождения E2E-набора	Решение о разделении smoke и beta-регрессии
Диагностичность	Доля падений с полными артефактами	Скорость локализации дефекта
Биллинг-корректность	Совпадение статуса транзакции и состояния подписки	Выявление ошибок синхронизации и бизнес-правил
Регуляторное покрытие	Доля обязательных 3DS/SCA-ветвей, включенных в E2E-набор	Риск пропуска ошибок аутентификации и перехода к платежу
Безопасность артефактов	Доля прогонов без чувствительных карточных данных в логах, скриншотах и отчетах	Риск расширения платежного контура и утечки тестовых данных

Отдельную группу образуют сценарии, состав которых определяется правилами обращения с платежными данными и процедурой аутентификации. В этом блоке E2E-тест контролирует поведение интеграции в точках, где нормативное требование изменяет пользовательский путь, допустимые данные либо ожидаемый статус операции. PCI DSS v4.0.1 используется для определения границ платежной среды и состава тестовых артефактов. EMV 3-D Secure задает ветви frictionless и challenge, а для операций, подпадающих под PSD2, отдельно проверяется обработка SCA и применимых исключений [7–9].

Таблица 3. Контрольные точки E2E-тестирования платежной интеграции в SaaS-платформе

Условие	Проверяемая E2E-ветвь	Контрольные точки
Граница PCI DSS-скоупа	Переход на hosted payment page, iframe либо API провайдера	карточные данные не проходят через компоненты, объявленные вне платежного контура, в логах и отчетах отсутствуют чувствительные значения, используются sandbox-карты, токены и синтетические идентификаторы
3DS frictionless flow	Аутентификация завершается без действия пользователя	корректно получен результат аутентификации; платеж продолжен один раз; биллинг и подписка обновлены после итогового статуса
3DS challenge flow	Перенаправление либо встроенная форма challenge	проверяются успешное завершение, отмена, тайм-аут, ошибка ACS, возврат в SaaS-продукт и отсутствие дублирующей транзакции
Исключение из SCA	Провайдер принимает либо отклоняет запрос исключения	принятый exemption не нарушает завершение платежа, после отклонения выполняется переход к challenge, статусы аутентификации и транзакции не смешиваются

Условие	Проверяемая E2E-ветвь	Контрольные точки
Подписка и повторяющийся платеж	Первичное подтверждение платежного средства и последующая операция	проверяются токенизация, связь с исходным согласием, идемпотентность, изменение статуса подписки и обработка отказа
Отложенный callback	Webhook поступает после возврата пользователя	интерфейс не показывает окончательный успех до подтвержденного состояния, повторный callback не создает второй заказ или повторный доступ

Практический алгоритм внедрения состоит из семи шагов. Команда описывает платежный ландшафт SaaS-платформы, выделяет критичные сценарии, проектирует тестовый слой, создает стабильные тестовые данные, включает проверки в пайплайн, настраивает отчетность и пересматривает набор по данным релизов, дефектов и flaky-падений. Платежный домен задает ограничения автоматизации. Sandbox внешнего провайдера воспроизводит production-поведение частично. Chargeback, антифрод-блокировка и часть банковских отказов плохо подходят для стабильного автоматического прогона. Поэтому проверка строится по согласованным точкам: redirect, callback, статус операции, запись в биллинге, доступ пользователя и отчетные артефакты.

Стандартизация QA-процессов при росте команды требует документации, перечня обязательных релизных проверок и обмена сведениями между QA-инженерами. Для платежных интеграций нужен единый реестр сценариев с ограничениями sandbox, правилами тестовых данных, ожидаемыми статусами, известными блокерами и зонами ручного контроля. Параллельный запуск относится к этапу масштабирования. Если платежный E2E-набор мал, распределение по контейнерам усложняет инфраструктуру без заметного выигрыша. При росте набора параллелизация становится рациональной после стабилизации данных, изоляции сценариев и устранения зависимости тестов друг от друга.

Прикладное решение состоит в привязке E2E-теста к платежному обещанию продукта. Пользователь SaaS-платформы ожидает предсказуемый результат: оплата прошла, доступ открыт, статус понятен, повторная попытка проходит без дубля, отказ завершается понятным состоянием заказа. Автоматический тест проверяет этот результат через сетевые запросы, статус транзакции, запись биллинга, событие подписки и конечное состояние аккаунта.

Обсуждение

Предложенная модель отличается от подходов, в которых покрытие оценивается только по числу вызванных endpoint-ов. Для платежного процесса критичен не сам факт прохождения запроса, а согласованность нескольких состояний: внешней транзакции, webhook-события, записи биллинга, подписки и доступа пользователя. Метрики E2E- и API-покрытия [6] сохраняют диагностическую ценность, если один технически корректный переход сопровождается ошибочным состоянием продукта.

Риск-ориентированный отбор согласуется с подходами к регрессии финансовых продуктов [5], поскольку одинаковое расширение автоматизации не дает сопоставимого эффекта для всех платежных ветвей. Основные способы оплаты, первичная активация подписки, продление, отказ, возврат и повторная попытка воздействуют на выручку и доступ

к продукту непосредственно. Редкие методы оплаты либо нестабильные sandbox-сценарии могут сохраняться в периодическом контроле, если стоимость их сопровождения превышает диагностическую пользу каждого прогона.

Регуляторный слой меняет состав проверяемых ветвей. PCI DSS задает ограничения на границы среды и тестовые данные, EMV 3-D Secure разделяет аутентификацию на frictionless и challenge flows, а нормы SCA требуют различать применение аутентификации, запрос исключения и переход к challenge после отклонения исключения [7–9]. Поэтому успешный код платежной операции не считается достаточным результатом, пока не проверены исход аутентификации, возврат пользователя, идемпотентность callback и соответствие состояния подписки выполненной операции.

Расширение E2E-набора увеличивает продолжительность прогонов и число зависимостей от внешних систем. Компактный набор критичных сценариев сохраняет связь с пользовательским результатом, тогда как контрактные, API- и компонентные проверки локализуют дефекты на более низких уровнях.

Ограничения исследования

Предложенная модель построена на неоднородном корпусе, объединяющем журнальные статьи, конференционные материалы, отраслевые нормативно-технические документы, состав не дает единой базы для количественной оценки сокращения ручного регресса, длительности прогонов или числа предотвращенных дефектов.

Поведение sandbox-сред различается у платежных провайдеров. Часть банковских отказов, антифрод-решений, chargeback-событий и задержек callback воспроизводится только приближенно, поэтому состав автоматизируемых ветвей зависит от возможностей конкретного шлюза. Модель также требует адаптации к способу интеграции. Hosted payment page, iframe, прямое API-взаимодействие и токенизированные операции формируют разные границы платежного контура.

Положения о SCA относятся прежде всего к операциям, подпадающим под европейское платежное регулирование, для других юрисдикций матрица сценариев должна учитывать локальные правила аутентификации и требования платежных систем. PCI DSS определяет требования к защите платежных данных, но E2E-функциональный тест сам по себе не подтверждает соответствие организации стандарту.

Заключение

Разграничены API, интеграционные, регрессионные и E2E-проверки. API-тесты фиксируют корректность запросов и форматов, интеграционные проверки контролируют взаимодействия между сервисами, регрессия снижает риск повторных дефектов. E2E-тест связывает эти уровни с пользовательским результатом оплаты. Определено место E2E-тестов в релизном цикле SaaS-платформы. На PR-этапе применяются быстрые проверки и контракты, на dev-среде – основной платежный сценарий, на beta-среде – расширенный регресс критичных сценариев, на production-этапе – безопасный мониторинг и синтетические проверки без раскрытия внутренних механизмов. Предложена прикладная модель внедрения, объединяющая карту платежных сценариев, риск-ориентированный отбор, разделение UI-,

API-, billing- и gateway-слоев, интеграцию в CI/CD, наблюдаемость и регулярную ревизию набора. Ценность E2E-автоматизации определяется охватом критичных переходов между пользовательским действием, платежным шлюзом, аутентификацией, биллингом и доступом к продукту, для платежей с 3-D Secure и SCA к этим переходам относятся исход аутентификации, возврат пользователя, обработка исключения и согласование статусов аутентификации и транзакции. Практическая применимость модели зависит от возможностей sandbox-среды, наблюдаемости внутренних сервисов и регулярного пересмотра карты рисков.

Список литературы

1. Васильев Б.Я. Методика тестирования API в микросервисной архитектуре продукта // Международный журнал гуманитарных и естественных наук. – 2025. – № 4–1 (103). – С. 154–159. – DOI: 10.24412/2500-1000-2025-4-1-154-159.
2. Козмин В.С. Разработка и использование автоматизированных тестов для веб-приложения // Вестник науки. – 2025. – Т. 3, № 6 (87). – С. 1798–1803.
3. Мурашкин И.Н. Методологии интеграционного тестирования в микросервисной архитектуре // Научный журнал. – 2024. – № 3 (70). – С. 1–3.
4. Тюменцев Д.В. Автоматизация тестирования в DevOps: подходы и лучшие практики // Международный журнал гуманитарных и естественных наук. – 2024. – № 2–2 (89). – С. 156–159. – DOI: 10.24412/2500-1000-2024-2-2-156-159.
5. Хапанков А.С. Автоматизация регрессионного тестирования в быстро меняющихся средах разработки финансовых продуктов // Информатика. Экономика. Управление. – 2026. – Т. 5, № 1. – С. 1025–1032. – DOI: 10.47813/2782-5280-2026-5-1-1025-1032.
6. Abdelfattah A.S., Cerny T., Yero J., Song E., Taibi D. Test Coverage in Microservice Systems: An Automated Approach to E2E and API Test Coverage Metrics // Electronics. – 1913. – T. 13, № 10. – Art. 1913. – DOI: 10.3390/electronics13101913.
7. Commission Delegated Regulation (EU) 2018/389 of 27 November 2017 supplementing Directive (EU) 2015/2366 with regard to regulatory technical standards for strong customer authentication and common and secure open standards of communication // Official Journal of the European Union. – 2023. – № L 69. – P. 23–43.
8. EMVCo. EMV 3-D Secure Protocol and Core Functions Specification. Version 2.3.1.1. EMVCo. – EMVCo. – 2023.
9. PCI Security Standards Council. Payment Card Industry Data Security Standard: Requirements and Testing Procedures. Version 4.0.1. Wakefield, MA: PCI Security Standards Council. – Wakefield, MA: PCI Security Standards Council. – 2024.
10. Ruggirello V. A Groovy-based Domain-Specific Language for Digital Payment System End-to-End Testing: Master’s thesis. – Turin: Politecnico di Torino, 2025. – 92 P.
11. Soares E., Sizilio G., Santos J., Alencar D., Kulesza U. The Effects of Continuous Integration on Software Development: A Systematic Literature Review // Empirical Software Engineering. – 2022. – T. 27. – DOI: 10.1007/s10664-021-10114-1.

12. Zampetti F., Geremia S., Bavota G., Di Penta M. CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study // 2021 IEEE International Conference on Software Maintenance and Evolution. – 2021. – P. 471–482. – DOI: 10.1109/ICSME52107.2021.00048.
13. Zheng Y., Liu Y., Xie X., Liu Y., Ma L., Hao J., Liu Y. Automatic Web Testing Using Curiosity-Driven Reinforcement Learning // 2021 IEEE/ACM 43rd International Conference on Software Engineering. – 2021. – P. 423–435. – DOI: 10.1109/ICSE43902.2021.00048.

References

1. Vasilev B.Ya. [API testing methodology in product microservice architecture] // Mezhdunarodnyy zhurnal gumanitarnykh i estestvennykh nauk. – 2025. – No. 4–1 (103). – P. 154–159. – DOI: 10.24412/2500-1000-2025-4-1-154-159. (In Russ.).
2. Kozmin V.S. [Development and use of automated tests for a web application] // Vestnik nauki. – 2025. – Vol. 3, No. 6 (87). – P. 1798–1803. (In Russ.).
3. Murashkin I.N. [Methodologies of integration testing in microservice architecture] // Nauchnyy zhurnal. – 2024. – No. 3 (70). – P. 1–3. (In Russ.).
4. Tyumentsev D.V. [Automation of testing in DevOps: approaches and best practices] // Mezhdunarodnyy zhurnal gumanitarnykh i estestvennykh nauk. – 2024. – No. 2–2 (89). – P. 156–159. – DOI: 10.24412/2500-1000-2024-2-2-156-159. (In Russ.).
5. Khapankov A.S. [Automation of regression testing in rapidly changing environments for financial product development] // Informatika. Ekonomika. Upravlenie. – 2026. – Vol. 5, No. 1. – P. 1025–1032. – DOI: 10.47813/2782-5280-2026-5-1-1025-1032. (In Russ.).
6. Abdelfattah A.S., Cerny T., Yero J., Song E., Taibi D. Test Coverage in Microservice Systems: An Automated Approach to E2E and API Test Coverage Metrics // Electronics. – 1913. – Vol. 13, No. 10. – Art. 1913. – DOI: 10.3390/electronics13101913.
7. Commission Delegated Regulation (EU) 2018/389 of 27 November 2017 supplementing Directive (EU) 2015/2366 with regard to regulatory technical standards for strong customer authentication and common and secure open standards of communication // Official Journal of the European Union. – 2023. – P. 23–43.
8. EMVCo. EMV 3-D Secure Protocol and Core Functions Specification. Version 2.3.1.1. EMVCo. Vol. 2.3.1.1. – EMVCo, 2023.
9. PCI Security Standards Council. Payment Card Industry Data Security Standard: Requirements and Testing Procedures. Version 4.0.1. Wakefield, MA: PCI Security Standards Council. Vol. 4.0.1. – Wakefield, MA: PCI Security Standards Council, 2024.
10. Ruggirello V. A Groovy-based Domain-Specific Language for Digital Payment System End-to-End Testing: Master’s thesis. – Turin: Politecnico di Torino, 2025. – 92 P.
11. Soares E., Sizilio G., Santos J., Alencar D., Kulesza U. The Effects of Continuous Integration on Software Development: A Systematic Literature Review // Empirical Software Engineering. – 2022. – Vol. 27. – DOI: 10.1007/s10664-021-10114-1.

12. Zampetti F., Geremia S., Bavota G., Di Penta M. CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study // 2021 IEEE International Conference on Software Maintenance and Evolution. – 2021. – P. 471–482. – DOI: 10.1109/ICSME52107.2021.00048.
13. Zheng Y., Liu Y., Xie X., Liu Y., Ma L., Hao J., Liu Y. Automatic Web Testing Using Curiosity-Driven Reinforcement Learning // 2021 IEEE/ACM 43rd International Conference on Software Engineering. – 2021. – P. 423–435. – DOI: 10.1109/ICSE43902.2021.00048.