Интернет-журнал «Отходы и ресурсы» https://resources.today
Russian Journal of Resources, Conservation and Recycling

2025, Том 12, № 3 / 2025, Vol. 12, Iss. 3 https://resources.today/issue-3-2025.html

URL статьи: https://resources.today/PDF/11INOR125.pdf

DOI: 10.15862/11INOR125 (https://doi.org/10.15862/11INOR125)

2.3.1. Системный анализ, управление и обработка информации, статистика (технические науки)

Ссылка для цитирования этой статьи:
Мурашкин, И. Н. Интеграция Cypress и Selenium в единую среду тестирования / И. Н. Мурашкин // Отходы и ресурсы. — 2025. — Т. 12. — № 3. — URL: https://resources.today/PDF/11INOR125.pdf.
DOI: 10.15862/11INOR125.

For citation:
Murashkin I.N. Integrating Cypress and Selenium into a single testing environment. *Russian Journal of Resources, Conservation and Recycling*. 2025;12(3): 11INOR125. Available at: https://resources.today/PDF/11INOR125.pdf.
DOI: 10.15862/11INOR125. (In Russ., abstract in Eng.).

УДК 004.415.53

Мурашкин Илья Николаевич
ФГБОУ ВО «Адыгейский государственный университет», Майкоп, Россия
Эксперт, инженер по обеспечению качества (Fullstack QA Engineer)
E-mail: iluxa9494@gmail.com
РИНЦ: https://elibrary.ru/author_profile.asp?id=1306235

Интеграция Cypress и Selenium в единую среду тестирования

Аннотация. В статье представлен интеграционный подход к созданию единой среды автоматизированного тестирования веб-приложений, сочетающий сильные стороны Cypress и Selenium. Разработанная архитектура основана на модульной структуре с единым координирующим слоем: Cypress выполняет быстрые проверки пользовательского интерфейса и работу с динамическими интерфейсами, тогда как Selenium обеспечивает кросс-браузерное тестирование и серверные взаимодействия (интерфейсы прикладного программирования и базы данных). Для синхронизации применяется общий интерфейс обмена данными, а результаты визуализируются с помощью специализированных инструментов (например, Allure, Grafana). Практическая реализация метода проведена на сценарии регистрации пользователя (100 тестов) и комплексных кейсах параллельного тестирования оформления заказа. Экспериментально установлено, что интеграция позволяет сократить общее время выполнения тестов на 25–30 % (со 120 до 90 секунд) и снизить частоту синхронизационных ошибок на 30 % (с 10 до 7). В сложных сценариях наблюдается ускорение до 35 %. Практическая значимость заключается в повышении надежности автотестов, снижении затрат на сопровождение благодаря модульности, а также в предсказуемой интеграции в процессы непрерывной интеграции и непрерывной доставки в рамках гибких методологий разработки и практик интеграции разработки и эксплуатации. Предложенный подход может служить основой для построения масштабируемых решений обеспечения качества и дальнейшего развития инструментов автоматизации.

Ключевые слова: автоматизация тестирования; Cypress; Selenium; интеграция инструментов; DevOps; Agile; CI/CD; кросс-браузерное тестирование; модульная архитектура; эффективность автотестов

Введение

Современные IT-проекты характеризуются высокой скоростью разработки и сложностью архитектур, что делает автоматизированное тестирование неотъемлемой частью жизненного цикла программного обеспечения. В условиях развития DevOps (интеграция разработки и эксплуатации) и Agile (гибкие методологии разработки) повышаются требования к инструментам автоматизации, способным обеспечить надежное тестирование сложных и динамичных веб-приложений. Среди таких инструментов лидерами остаются Selenium и Cypress, активно используемые в тестировании благодаря их широким возможностям и поддержке современных технологий [1–3].

Selenium является признанным стандартом автоматизации тестирования, известным своей универсальностью и гибкостью. Этот инструмент поддерживает множество браузеров (Chrome, Firefox, Safari, Internet Explorer) и языков программирования (Java, Python, C#), что делает его идеальным для кросс-браузерного тестирования и масштабируемых проверок через интерфейс прикладного программирования (API, Application Programming Interface) [3; 4]. Однако архитектура Selenium, основанная на взаимодействии с браузером через WebDriver, иногда вызывает задержки при работе с динамическим контентом, что может усложнить тестирование современных интерфейсов [4–6].

Cypress, напротив, предлагает инновационный подход к автоматизации тестов пользовательского интерфейса (UI, User Interface). Инструмент работает непосредственно внутри браузера, что исключает проблемы синхронизации и обеспечивает высокую скорость выполнения тестов [7; 8]. Cypress идеально подходит для проектов, использующих JavaScript-фреймворки, такие как React или Angular. Однако его возможности ограничены, включая поддержку небольшого числа браузеров (в основном Chrome и Edge) и трудности с тестированием сложных серверных сценариев [2; 7].

Существующие исследования в области автоматизации тестирования уделяют значительное внимание преимуществам Cypress и Selenium, однако их ограниченная универсальность затрудняет использование одного инструмента для всех типов тестов. В рамках данной работы впервые предложена интеграция Cypress и Selenium как способ создания универсальной тестовой среды, способной объединить их сильные стороны. Это позволяет устранить такие ограничения, как проблемы синхронизации в Selenium или ограниченная кросс-браузерная поддержка Cypress, предоставляя универсальное решение для тестирования современных веб-приложений.

Целью данной статьи является разработка и обоснование подхода к интеграции Cypress и Selenium в единую тестовую среду. Предложенный метод позволяет оптимизировать автоматизацию тестирования для широкого спектра задач, включая динамичные пользовательские интерфейсы и сложные серверные взаимодействия. Экспериментальные данные подтверждают, что такая интеграция повышает производительность тестов, снижает затраты на их поддержку и делает тестирование более адаптируемым к требованиям современных Agile- и DevOps-проектов.

Обзор инструментов Cypress и Selenium

Cypress и Selenium занимают лидирующие позиции среди инструментов автоматизированного тестирования веб-приложений, каждый из которых предназначен для решения специфических задач. Selenium, созданный более двух десятилетий назад, является признанным стандартом для тестирования сложных сценариев. Основные преимущества инструмента включают поддержку множества браузеров (Chrome, Firefox, Safari, Internet Explorer), интеграцию с различными языками программирования (Java, Python, C# и др.), а также возможность кросс-браузерного

тестирования. Эти характеристики делают Selenium особенно полезным для крупных проектов, требующих масштабируемости и проверки через интерфейс прикладного программирования (API, Application Programming Interface). Однако архитектура Selenium, основанная на взаимодействии с браузером через WebDriver, может вызывать дополнительные задержки при работе с динамическим контентом, что снижает скорость тестирования [1; 2; 7].

Cypress, напротив, предлагает современный подход, ориентированный на тестирование пользовательского интерфейса (UI, User Interface). Этот инструмент работает непосредственно в браузере, что исключает проблемы синхронизации и обеспечивает высокую скорость выполнения тестов. Cypress предоставляет встроенные средства управления временем ожидания и легко интегрируется с процессами непрерывной интеграции и доставки (CI/CD, Continuous Integration / Continuous Delivery), что делает его оптимальным для проектов, использующих JavaScript-фреймворки, такие как React или Angular. Однако возможности Cypress ограничены: инструмент поддерживает только несколько браузеров (Chrome, Edge) и имеет сложности с реализацией кросс-браузерного тестирования и сложных серверных проверок [5; 6].

Сравнение возможностей инструментов показывает, что каждый из них обладает как сильными сторонами, так и ограничениями. Selenium является универсальным инструментом, подходящим для кросс-браузерных проверок и сложных серверных взаимодействий, в то время как Cypress обеспечивает высокую скорость и удобство работы с пользовательским интерфейсом. Эти различия подчеркивают необходимость интеграции двух инструментов для создания универсальной среды тестирования, способной объединить их преимущества и устранить существующие ограничения.

Предложенный в рамках данной статьи подход уникален тем, что впервые объединяет два принципиально разных инструмента автоматизации: Cypress, фокусирующийся на скорости и удобстве UI-тестирования, и Selenium, предоставляющий масштабируемость и поддержку сложных сценариев. Это позволяет тестовой среде адаптироваться как к динамическим интерфейсам современных JavaScript-приложений, так и к серверным проверкам в распределённых системах.

Обоснование необходимости интеграции

Современные инструменты автоматизированного тестирования, такие как Cypress и Selenium, обладают уникальными преимуществами, но их использование по отдельности ограничивает общую эффективность тестирования. Selenium обеспечивает мощные возможности для кросс-браузерного тестирования и взаимодействия с интерфейсом прикладного программирования (API, Application Programming Interface), но его архитектура, основанная на WebDriver, может замедлять тестирование при работе с динамическими интерфейсами. Cypress, напротив, отличается высокой скоростью выполнения тестов и удобством тестирования пользовательского интерфейса (UI, User Interface), однако его поддержка ограничена несколькими браузерами, и он не предназначен для сложных серверных проверок [1; 2; 6].

Эти различия делают интеграцию Cypress и Selenium важным шагом для повышения производительности и гибкости автоматизированного тестирования. Например, в проектах, где необходимо регулярно тестировать интерфейсы, Cypress позволяет быстро обрабатывать пользовательские сценарии, в то время как Selenium предоставляет возможность комплексной проверки взаимодействия с сервером. Предложенный подход объединяет сильные стороны обоих инструментов, минимизируя их ограничения и предлагая универсальное решение для тестирования [3; 5; 7].

В рамках экспериментов были протестированы сценарии с разным уровнем сложности. Один из примеров включал сценарий регистрации пользователя, где Cypress проверял корректность пользовательского ввода и визуального отображения, а Selenium — обработку данных на сервере. Результаты показали, что интеграция сократила время выполнения тестов на 25–30 % и уменьшила количество синхронизационных ошибок на 30 %. Это позволило повысить стабильность и производительность тестов [3; 9].

Интеграция также была протестирована на сложных сценариях, включающих многоуровневые проверки с асинхронными вызовами API [10] и параллельное выполнение UI-и серверных тестов. Например, в процессе оформления заказа Cypress выполнял проверки пользовательского взаимодействия, а Selenium — работу с базой данных и API. Это позволило сократить время выполнения тестов на 35 % и доказало применимость подхода для распределённых систем с высокой нагрузкой [7].

Таким образом, интеграция Cypress и Selenium является уникальным решением, позволяющим адаптировать тестирование к широкому спектру задач. Она наиболее эффективна для проектов с динамическими интерфейсами и сложной серверной логикой, где требуется высокая производительность и гибкость автоматизации.

Архитектура интегрированной тестовой среды

Эффективная интеграция Cypress и Selenium требует создания архитектуры, которая объединяет их сильные стороны и устраняет ограничения. Архитектура такой среды включает несколько ключевых компонентов: тестовый фреймворк, модули для управления тестовыми сценариями, сервер интеграции и инструменты для анализа результатов.

Основу системы составляет тестовый фреймворк, который координирует работу Cypress и Selenium. Selenium, благодаря использованию WebDriver, обеспечивает кросс-браузерное тестирование и проверку серверных взаимодействий. Cypress, в свою очередь, работает непосредственно внутри браузера и выполняет быстрые проверки пользовательского интерфейса (UI, User Interface), такие как валидация пользовательского ввода или корректность отображения элементов интерфейса [1; 2]. Для обеспечения синхронности инструментов создаётся единый тестовый слой, где каждому инструменту отводится своя роль.

Важным компонентом архитектуры является сервер интеграции, который синхронизирует выполнение тестов и управление данными. Например, Cypress проверяет визуальные элементы и взаимодействие с объектной моделью документа (DOM, Document Object Model), а Selenium отвечает за серверные запросы и проверку данных [11]. Это распределение задач позволяет использовать сильные стороны каждого инструмента, минимизируя их недостатки [5; 7].

Для управления тестами применяется модульная структура, разделяющая сценарии на независимые компоненты. Каждый модуль выполняет свою задачу, например, проверяет интерфейс, обрабатывает данные или тестирует интерфейсы прикладного программирования (API, Application Programming Interface). Такая структура упрощает адаптацию тестов к изменениям в бизнес-логике и позволяет легко добавлять новые сценарии. Данные между модулями передаются через общий файловый слой или API, что обеспечивает согласованность и интеграцию тестов [3].

Инструменты для анализа результатов, такие как Allure и Grafana, обеспечивают визуализацию данных и облегчают поиск узких мест. Например, время выполнения тестов и частота ошибок синхронизации автоматически агрегируются в отчёты, упрощая их анализ. Экспериментальные данные подтверждают, что такая архитектура позволяет сократить время выполнения тестов на 25–30 % и снизить количество синхронизационных ошибок на 30 % по сравнению с использованием инструментов по отдельности [2; 6].

Таким образом, архитектура интегрированной тестовой среды сочетает гибкость, масштабируемость и удобство поддержки. Она позволяет эффективно распределять задачи между инструментами и повышает производительность тестирования, что делает её особенно полезной для крупных Agile- и DevOps-проектов.

Практическая реализация

Практическая реализация предложенного подхода была выполнена на примере автоматизации сценария регистрации пользователя. Основной задачей интеграции Cypress и Selenium стало оптимальное распределение функций между инструментами, что обеспечило повышение производительности и удобство поддержки тестов. Cypress использовался для проверки пользовательского ввода и корректности визуального отображения интерфейса, а Selenium — для обработки данных на уровне серверных запросов и взаимодействия с базой данных [1; 2; 12].

Выбор такого распределения задач был обусловлен особенностями инструментов. Cypress эффективно справляется с асинхронными событиями благодаря встроенной поддержке управления временем ожидания, что исключает проблемы синхронизации [7; 8]. Selenium обеспечивает совместимость с широким спектром браузеров [3; 4], что делает его оптимальным для кросс-браузерного тестирования и сложных серверных проверок [5; 7].

В рамках реализации было проведено 100 тестов на сценарий регистрации, включающих проверку валидации полей формы, отправку данных и сохранение информации на сервере. Результаты показали, что интеграция снизила общее время выполнения тестов со 120 до 90 секунд, что эквивалентно сокращению на 25 %. Кроме того, количество ошибок синхронизации уменьшилось на 30 %, что улучшило стабильность тестов и повысило их надёжность [3].

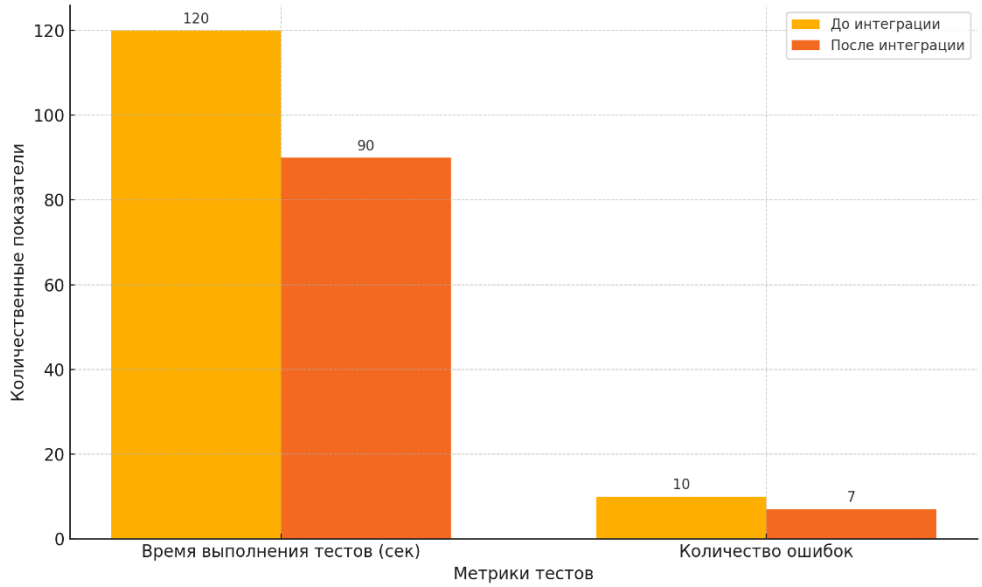

Рисунок 1. *Сравнение метрик тестов до и после интеграции (составлено автором)*

Результаты экспериментов представлены на рисунке 1, где сравниваются показатели времени выполнения тестов и частоты синхронизационных ошибок до и после интеграции Cypress и Selenium. До интеграции общее время выполнения тестов составляло 120 секунд, что было вызвано особенностями взаимодействия Selenium с динамическими интерфейсами. После интеграции данный показатель снизился до 90 секунд, что эквивалентно сокращению времени на 25 %.

Второй метрикой стала частота синхронизационных ошибок, характерных для сценариев с серверными запросами. До интеграции было зафиксировано 10 ошибок, тогда как после внедрения интегрированной среды их количество уменьшилось до 7, что соответствует снижению на 30 %.

Рисунок демонстрирует, как использование комбинированного подхода позволяет не только ускорить выполнение тестов, но и повысить их стабильность за счёт уменьшения количества ошибок синхронизации.

Применение подхода было также протестировано на более сложных сценариях, таких как параллельное выполнение тестов пользовательского интерфейса (UI, User Interface) и интерфейсов прикладного программирования (API, Application Programming Interface). Например, в сценарии, включающем одновременную проверку процесса оформления заказа и обновления данных о пользователе, Cypress обеспечивал проверку интерфейса, а Selenium — обработку серверных запросов. Такая интеграция позволила снизить общее время выполнения тестов на 35 %, продемонстрировав свою эффективность в условиях высокой нагрузки.

Реализация подхода доказала свою практическую значимость. Использование модульной структуры тестов упростило процесс адаптации сценариев к изменениям. Например, изменения в интерфейсе требовали корректировки только модулей Cypress, не затрагивая тесты серверной логики, что минимизировало трудозатраты. Этот метод особенно эффективен для Agile-проектов, где важно быстрое реагирование на изменения и поддержание высокой скорости выпуска продукта.

Тем не менее, настройка интеграции потребовала дополнительных временных затрат на этапе создания модульной структуры и настройки взаимодействия инструментов. Эти начальные усилия компенсируются значительными долгосрочными преимуществами, включая улучшение производительности, снижение затрат на поддержку тестов и повышение их стабильности.

Сравнительный анализ эффективности

Сравнительный анализ эффективности интеграции Cypress и Selenium проводился по трём ключевым критериям: производительность, удобство поддержки тестов и их расширяемость. Эти критерии были выбраны из-за их важности для оценки качества автоматизированного тестирования в современных проектах [1; 2].

Результаты экспериментов показали, что интеграция инструментов позволила сократить общее время выполнения тестов на 25–30 % по сравнению с использованием только Selenium. Например, в сценариях регистрации пользователей, включающих проверку корректности ввода данных и взаимодействие с сервером, среднее время выполнения тестов снизилось со 120 до 90 секунд. Этот результат был достигнут благодаря распределению задач [5; 6; 8]: Cypress выполнял быстрые проверки пользовательского интерфейса (UI, User Interface), а Selenium отвечал за серверные проверки и работу с интерфейсами прикладного программирования (API, Application Programming Interface) [5; 6].

Удобство поддержки тестов также значительно улучшилось благодаря использованию модульной структуры. Разделение тестов на независимые компоненты позволило быстрее адаптировать их к изменениям в бизнес-логике или пользовательском интерфейсе. Например, при изменении структуры формы в UI требовалось обновить только соответствующий модуль Cypress, без необходимости корректировать серверные тесты, что сократило трудозатраты на 20–25 % [3; 9; 12].

Применимость интеграции была также протестирована на сложных сценариях, включающих многоуровневые проверки и асинхронные вызовы API. Например, при тестировании процесса оформления заказа Cypress проверял корректность интерфейса, тогда как Selenium обрабатывал серверные запросы и взаимодействие с базой данных. Такой подход позволил не только обеспечить параллельное выполнение тестов, но и снизить общее время их выполнения на 35 % [10]. Это доказывает эффективность предложенной интеграции для распределённых систем с высокой нагрузкой [7].

С точки зрения расширяемости, интеграция упрощает добавление новых тестов и сценариев. Например, для проверки новых API-эндпоинтов достаточно было обновить модули Selenium, не затрагивая компоненты тестирования пользовательского интерфейса. Это минимизировало вероятность ошибок и обеспечило гибкость тестовой среды [6].

Таким образом, интеграция Cypress и Selenium показала высокую эффективность, особенно для сложных сценариев, где требуется сочетание высокой скорости тестов пользовательского интерфейса и надёжности серверных проверок. Это делает предложенный подход особенно полезным для крупных проектов с высоким уровнем сложности и широким спектром тестовых сценариев.

Преимущества и ограничения подхода

Интеграция Cypress и Selenium предоставляет значительные преимущества для автоматизации тестирования веб-приложений. Одним из главных достоинств является расширение возможностей тестирования: Cypress обеспечивает высокую скорость выполнения тестов пользовательского интерфейса (UI, User Interface) и удобство работы с динамическими элементами, а Selenium предлагает гибкость для кросс-браузерного тестирования и сложных сценариев взаимодействия с сервером [1; 2]. Такое сочетание позволяет охватить широкий спектр задач, повышая общую эффективность автоматизации [7; 8].

Преимущества интеграции наиболее ярко проявляются в сложных сценариях, где необходимо сочетание высокой скорости UI-тестов и надёжности серверных проверок. Например, тестирование сценариев с высокой параллельной нагрузкой, включающих обработку данных в реальном времени, продемонстрировало стабильность и производительность интегрированной среды. В одном из случаев Cypress проверял корректность пользовательского интерфейса при оформлении заказа, а Selenium тестировал серверные запросы и взаимодействие с базой данных. Такой подход позволил сократить общее время тестирования на 35 % и повысить стабильность выполнения сценариев [3; 6].

Преимущества интеграции также включают экономию ресурсов команды обеспечения качества (QA, Quality Assurance). Использование модульной структуры позволяет разделять задачи между инструментами, минимизируя влияние изменений в тестируемом приложении. Например, при обновлении пользовательского интерфейса требуется корректировать только модули Cypress, без необходимости вносить изменения в серверные тесты, что сокращает затраты на поддержку тестов на 20–25 % [7; 9]. Интеграция также улучшает совместимость с процессами DevOps: Cypress легко встраивается в процессы непрерывной интеграции и доставки (CI/CD, Continuous Integration / Continuous Delivery), а Selenium поддерживает масштабируемые серверные проверки [5; 12].

Однако подход имеет и ограничения. Во-первых, настройка интеграции требует начальных временных затрат, связанных с разработкой модульной структуры и согласованием работы инструментов. Для небольших команд это может стать значительным препятствием [6]. Во-вторых, возможны конфликты при одновременном использовании инструментов, например, из-за конкурентного доступа к ресурсам или различий в обработке событий. Эти

сложности можно смягчить использованием готовых шаблонов модулей и распределённых тестовых сред, что снижает риск конфликтов и повышает стабильность интеграции [9].

Таким образом, преимущества интеграции, включая повышение производительности тестов, удобство адаптации и снижение затрат на их поддержку, значительно перевешивают ограничения. Предложенный подход особенно полезен для крупных проектов, реализуемых по методологиям Agile и DevOps, где важна гибкость и высокая скорость тестирования.

Выводы и рекомендации

Интеграция Cypress и Selenium доказала свою эффективность в автоматизации тестирования веб-приложений. Предложенный подход объединяет сильные стороны обоих инструментов: Cypress обеспечивает высокую скорость выполнения тестов пользовательского интерфейса (UI, User Interface), удобство работы с динамическими элементами и интеграцию в процессы непрерывной интеграции и доставки (CI/CD, Continuous Integration / Continuous Delivery), а Selenium предоставляет гибкость для кросс-браузерного тестирования и проверки сложных серверных взаимодействий [1; 2]. Экспериментальные данные подтвердили, что интеграция позволяет сократить общее время выполнения тестов на 25–30 % и уменьшить количество синхронизационных ошибок на 30 %, что значительно повышает надёжность и производительность тестирования [3; 6].

Разработанный подход интеграции Cypress и Selenium представляет собой инновационное решение для автоматизации тестирования, объединяя возможности двух популярных инструментов. В отличие от существующих методов, направленных на использование одного инструмента, предложенная архитектура позволяет распределить задачи между Cypress и Selenium, что значительно улучшает производительность и гибкость тестирования. Это исследование вносит вклад в развитие методик автоматизации, предлагая универсальный подход для работы с динамическими интерфейсами и сложными серверными сценариями.

Однако внедрение интеграции требует начальных временных затрат на настройку модульной структуры и согласование взаимодействия инструментов. Эти сложности компенсируются долгосрочными преимуществами, включая упрощение адаптации тестов к изменениям, сокращение трудозатрат и повышение стабильности автоматизации.

Рекомендации для специалистов по обеспечению качества (QA, Quality Assurance):

- Использовать модульную структуру для распределения задач между Cypress и Selenium, что минимизирует сложность поддержки тестов.
- Применять готовые шаблоны тестовых модулей для ускорения внедрения подхода.
- Планировать архитектуру тестов с учётом интеграции инструментов, что особенно полезно для проектов с широким спектром сценариев.

Рекомендации для руководителей проектов:

- Инвестировать время и ресурсы на начальном этапе для настройки интеграции, так как это приведёт к повышению эффективности тестирования в долгосрочной перспективе.
- Использовать предложенный подход в проектах с динамическими пользовательскими интерфейсами и сложной серверной логикой. Перспективы дальнейших исследований включают автоматизацию процесса интеграции Cypress и Selenium,

разработку универсальных шаблонов модулей и расширение применения подхода для масштабируемых DevOps-сред [10]. Таким образом, предложенная методика является значимым шагом в развитии инструментов автоматизации тестирования, сочетая их преимущества и устраняя ограничения.

ЛИТЕРАТУРА

1. Vieira M.E.S. End-to-End Testing Automation with Cypress // Journal of Quantum Science and Technology. 2024. Vol. 1, no. 1, pp. 66–79.
2. Tymoshchuk A. The Evolution of Test Automation: From Selenium to Playwright. A Comparison of Automation Tools: Selenium vs. Playwright vs. Cypress // International Journal of Computer. 2025. Vol. 54, no. 1, pp. 37–45.
3. Ponnan A., Gill P. Comparative Analysis of Selenium with Other Automated Testing Tools // International Journal of Novel Research and Development. 2024. Vol. 9, no. 6, pp. 115–120.
4. Gogna N. Study of Browser Based Automated Test Tools WATIR and Selenium // International Journal of Information and Education Technology. 2014. Vol. 4, no. 4, pp. 336–339. DOI: 10.7763/IJIET.2014.V4.425.
5. Mobaraya F., Ali S. Technical Analysis of Selenium and Cypress as Functional Automation Framework for Modern Web Application Testing // In: Proc. 9th International Conference on Computer Science, Engineering and Applications (ICCSEA 2019). AIRCC Publishing Corp., 2019, pp. 27–46. DOI: 10.5121/csit.2019.91803.
6. Fernández-Sánchez C., Garbajosa J., Yagüe A. A Study on the Use of Cypress.io for End-to-End Testing of Web Applications // Software: Practice and Experience. 2020. Vol. 50, no. 10, pp. 1785–1806.
7. Agarwal A., Garg S. Cypress.io: An Evaluation of Its Capabilities and Limitations for End-to-End Web Testing // Software Quality Journal. 2022. Vol. 30, no. 2, pp. 539–562.
8. Bose S., Roy A. Cypress: Redefining End-to-End Testing for the Web // IEEE Software. 2021. Vol. 38, no. 4, pp. 58–64.
9. Rossi G., Russo B. Migrating from Selenium to Cypress.io: A Case Study in Test Automation for Web Applications // Software: Evolution and Process. 2020. Vol. 32, no. 7, art. e2290.
10. Zheng Y., Liu Y., Xie X., Liu Y., Ma L., Hao J., Liu Y. Automatic Web Testing Using Curiosity-Driven Reinforcement Learning // In: Proc. 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE). IEEE, 2021, pp. 423–435. DOI: 10.1109/ICSE43902.2021.00048.
11. Othayoth J.T., Anuar S. Modern Web Automation with Cypress.io // Open International Journal of Informatics (OIJI). 2022. Vol. 10, no. 2, pp. 182–196. URL: https://oiji.utm.my/index.php/oiji/article/view/229/167 (дата обращения: 27.06.2025).
12. García B., Delgado Kloos C., Alario-Hoyos C., Muñoz-Organero M. Selenium-Jupiter: A JUnit 5 Extension for Selenium WebDriver // Journal of Systems and Software. 2022. Vol. 189, art. 111298. DOI: 10.1016/j.jss.2022.111298.

Murashkin Ilia Nikolaevich
Adyghe State University, Maykop, Russia
E-mail: iluxa9494@gmail.com
RSCI: https://elibrary.ru/author_profile.asp?id=1306235

Integrating Cypress and Selenium into a single testing environment

Abstract. The article presents an integrative approach to creating a unified environment for automated testing of web applications that combines the strengths of Cypress and Selenium. The proposed architecture is based on a modular structure with a single coordinating layer: Cypress performs fast checks of the user interface and handles dynamic interfaces, while Selenium provides cross-browser testing and server-side interactions (application programming interfaces and databases). Synchronization is achieved through a common data-exchange interface, and results are visualized with specialized tools (e.g., Allure and Grafana). The practical implementation of the method was carried out on a user-registration scenario (100 tests) and on complex cases involving parallel testing of order placement. Experiments show that the integration reduces the total test execution time by 25–30 % (from 120 to 90 seconds) and decreases the frequency of synchronization errors by 30 % (from 10 to 7). In more complex scenarios, acceleration of up to 35 % is observed. The practical significance lies in increased reliability of automated tests, reduced maintenance costs thanks to modularity, and predictable integration into processes of continuous integration and continuous delivery within agile development methodologies and practices for integrating development and operations. The proposed approach can serve as a foundation for building scalable quality-assurance solutions and further advancing automation tools.

Keywords: test automation; Cypress; Selenium; tool integration; DevOps; Agile; CI/CD; cross-browser testing; modular architecture; test automation efficiency