Программный комплекс распределенного тестирования веб-приложений

П.А. Редкин, А.С. Алёшкин

***Аннотация* — Статья посвящена тестированию веб-приложений, которое играет ключевую роль в создании качественных, надежных и безопасных продуктов. Для современной адаптации к интернет-технологиям требуется внедрение современных методик тестирования, а также глубоких знаний и постоянного самосовершенствования процессов создания решений. В статье рассмотрены основные этапы такого совершенствования: внедрение автоматизации тестирования, использование передовых инструментов и обеспечение общего высокого уровня безопасности. Ключевыми аспектами тестирования рассматриваются: нагрузочное тестирование, тестирование безопасности и юзабилити-тестирование, а также подходы к выполнению и анализу результатов выполнения тестов.**

Статья подчеркивает важность комплексного подхода к обеспечению качества веб-приложений и исследует роль автоматизации в упрощении и оптимизации процессов тестирования. В статье описываются различные инструменты, используемые для тестирования, включая Apache JMeter, LoadRunner, Gatling для нагрузочного тестирования, а также OWASP ZAP и Burp Suite для тестирования на безопасность. Авторы также обсуждают методы оценки юзабилити и представляют современные инструменты для юзабилити-тестирования, такие как UserTesting, Optimal Workshop и Lookback.io.

Рассмотрена интеграция тестирования в цикл "непрерывной интеграции и доставки" (CI/CD), что позволяет сократить время разработки, и, при этом гарантировать высокое качество продукта на всех этапах его создания.

Результатом проведенного технологического обзора становится система распределенного тестирования, которая представляет собой эффективный подход к реализации задач тестирования, позволяя командам разработки быстро идентифицировать и устранять даже потенциальные проблемы.

***Ключевые слова* — автоматизация тестирования, нагрузочное тестирование, тестирование веб-приложений, тестирование на безопасность.**

I. Введение

В современном мире цифровые технологии проникают во все сферы жизни, а качество и безопасность веб-приложений выходят на первый план. Развитие интернет-технологий с Web 1.0 до Web 3.0 и появление новых подходов в программировании и тестировании требуют от специалистов не только глубоких технических знаний, но и понимания актуальных трендов в области качества программного обеспечения. В связи с этим, комплексный подход к тестированию веб-приложений, охватывающий такие аспекты как, производительность, безопасность и удобство использования, становится бесспорно важным. Данная статья посвящена анализу современных методик и инструментов тестирования, которые позволяют разработчикам и тестировщикам повысить качество веб-приложений, обеспечить их надежность и безопасность, а также сделать их максимально удобными и понятными для конечных пользователей.

Основываясь на обширном обзоре литературы и анализе передовых практик в области разработки программного обеспечения, в статье рассматриваются ключевые аспекты и инструменты нагрузочного тестирования, тестирования безопасности и юзабилити-тестирования. Автоматизация тестирования выделяется как ключевой элемент, позволяющий сократить время на тестирование и повысить общую эффективность процессов.

Целью данного исследования является не только предоставление обзора существующих инструментов и методик, но и демонстрация важности интегрированного подхода к тестированию, который позволит разработчикам создавать высококачественные и безопасные веб-приложения. Представленный анализ подкреплен примером разработки комплексного программного решения для распределённого тестирования веб-приложений.

II. Эволюция интернет-технологий

Интернет радикально изменил все аспекты жизни общества, от бизнеса до личной жизни, предоставив ключевые преимущества в разработке программных продуктов, такие как экономия на установке решений и автоматические обновления. Внедрение серверных и браузерных технологий повысило риски наличия дефектов при разработке веб-приложений, отмечается, что около 70% используемых веб-приложений содержат дефекты из-за плохой реализации или отсутствия тестирования [1]. С началом эры Web 1.0, созданной Тимом Бернерсом-Ли, интернет был ограничен статичными страницами, требующими постоянного обновления для добавления новой информации [2]. Тестирование было сосредоточено на функциональности и совместимости браузеров, при этом инструменты для тестирования были ограничены.

Переход к Web 2.0 ознаменовался появлением социальных сетей и платформ для совместной работы, что увеличило сложность веб-приложений и потребность в разнообразных методах тестирования, включая безопас-

Статья получена 10 марта 2024. Данная работа представляет собой результат выполнения выпускной квалификационной работы.

П. А. Редкин, РТУ МИРЭА (e-mail: redkin_pavel02@mail.ru).

А. С. Алёшкин, РТУ МИРЭА (e-mail: antony@testor.ru).

ность и нагрузочное тестирование [2]. Инструменты, такие как Selenium и JMeter, начали использоваться для более эффективного выявления проблем.

Web 3.0, известный как семантический веб, привнес новые вызовы для тестирования, требуя от специалистов глубокого понимания технологий блокчейн и адаптацию методов тестирования под децентрализованные приложения [3]. Распределённое тестирование веб-приложений выступает здесь как эффективный ответ на вызовы реального мира, где приложениям приходится работать в условиях широкого спектра сетевых сред и под высокой нагрузкой от большого числа пользователей. Этот подход предполагает выполнение тестов на множестве компьютеров или серверов, что позволяет создать условия, максимально приближенные к реальным сценариям использования приложений, и тем самым обеспечивает более точную оценку производительности, масштабируемости и надежности веб-приложений.

В современном мире тестирование веб-приложений сталкивается с проблемами совместимости, управления зависимостями, безопасности и оптимизации ресурсов, что подчеркивает важность автоматизации и комплексного подхода в обеспечении качества всего процесса разработки решений.

III. Комплексный подход к тестированию веб-приложений

Тестирование веб-приложений — это не просто проверка функциональности, это комплексная задача, включающая в себя анализ безопасности, оценку производительности веб-приложения, проверку совместимости с различными устройствами и браузерами, а также удостоверение в том, что пользовательский интерфейс интуитивно понятен конечному пользователю.

Современный ландшафт тестирования веб-приложений характеризуется широким спектром инструментов и технологий, способствующих обеспечению качества и надежности разработки. Эти инструменты поддерживают различные аспекты тестирования, от автоматизированных фреймворков и облачных платформ до специализированных инструментов для нагрузочного тестирования и анализа безопасности — выбор имеющихся сегодня средств впечатляет своим разнообразием и возможностями. Они помогают командам разработчиков встроить тестирование в процессы непрерывной интеграции и доставки (CI/CD).

Существует множество видов и техник тестирования, каждый из которых имеет свои цели, методы и инструменты, способствующие обеспечению качества и надежности разработки. Тестирование веб-приложений, распределенное тестирование требуют комплексного подхода, который должен включать в себя:

- Функциональное тестирование, для проверки соответствия приложения спецификациям и требованиям.
- Тестирование безопасности, для выявления уязвимостей и предотвращения потенциальных атак.
- Нагрузочное тестирование, для оценки производительности приложения под высокой нагрузкой
- Юзабилити-тестирование, для обеспечения удобства и интуитивности пользовательского интерфейса.

IV. Автоматизация тестирования

Автоматизация тестирования значительно увеличивает эффективность процесса тестирования и повышает его охват, что позволяет быстрее выявлять дефекты и сокращать общее время на тестирование. Это особенно важно для обеспечения качества в условиях Agile и DevOps. [4]. Основная цель автоматизации тестирования заключается в повышении эффективности и охвата тестами в рамках разработки программного обеспечения. Создав качественный набор тестов один раз, мы сможем использовать его каждый раз после внесения некоторых изменений в веб-приложение. Кроме того, современные веб-приложения построены на основе многоуровневой архитектуры. Следовательно, чтобы протестировать общее поведение веб-приложения, а значит провести распределенное тестирование, нам необходимо выполнить определенный набор методов тестирования, который подбирается в зависимости от решаемой задачи. Инструменты автоматизации тестирования — наиболее важный компонент в цепочке инструментов разработки, для данного этапа [5].

Автоматизация — это универсальный подход в программировании и тестировании. Автоматизация – процесс перевода повторяющего ручного труда в автоматический режим. Автоматизированное тестирование программного обеспечения — часть процесса тестирования, на этапе контроля качества, в процессе разработки программного обеспечения. Оно использует программные средства для выполнения тестов и проверки результатов выполнения, что помогает сократить общее время тестирования и упростить его процесс [6].

Касательно тестирования веб-приложений с помощью автоматизации понимают создание тестовых сценариев, эмулирующих действия тестировщика для повторяющегося ручного труда или создания необходимых условий для тестирования. Исходя из исследования [6] можно сделать вывод, что автоматизация облегчает работу ручных тестировщиков, покрывая некоторые области, но, нельзя сказать, что автоматизация полностью заменяет их. Она позволяет покрыть достаточную область набора тестов.

Как показывают рассмотренные практики (см. таблицу 1.1), главным недостатком внедрения автоматизации тестирования веб-приложений становится необходимость значительных финансовых вложений в разработку и поддержку собственного фреймворка тестирования. На первом этапе автоматизации необходимо создать фреймворк, используя один из широко используемых языков программирования в сочетании с тестовыми библиотеками и инструментами для логирования. Для обеспечения взаимодействия между кодом и веб-браузером применяется библиотека Selenium Webdriver, дополненная драйверами для определенных браузеров. [6] Разрабатываемый фреймворк должен соответство-

вать ряду критериев, включая:

— создание отчетов в понятном формате;
— возможность проведения запросов к базам данных;
— настройку тестового окружения с учетом различных параметров;
— интеграцию с системами непрерывной интеграции (CI);
— управление различными наборами данных для выполнения тестов.

Таблица 1.1. Преимущества и недостатки автоматизации тестирования

Преимущества	Недостатки
Скорость выполнения тестов быстрее ручной проверки	Большие денежные и временные затраты на разработку фреймворка
Возможность тестирования областей, которые невозможно протестировать вручную (нагрузочное тестирование, объемное тестирование)	Пропуск мелких ошибок, так как тесты выполняют только запрограммированные проверки
Автоматические тесты могут выполняться без наблюдения тестировщика	Необходимость настроек и поддержки окружений
В результате выполнения автотестов формируются отчеты, позволяющие наблюдать за динамикой изменения процента дефектов	Необходимость постоянной поддержки фреймворка и используемых в нем библиотек в актуальном состоянии
Меньшие затраты на поддержку – написанные скрипты периодически требуют вмешательства, но затраты на их поддержку ниже, чем проведение того же объема работ вручную	Стоимость некоторых инструментов для автоматизации слишком высока
Все тесты выполняются однообразно, что гарантирует независимость результата от внешних воздействий	Необходимость формирований дополнительных наборов данных, специальных для сред тестирования

V. Инструменты для автоматизации тестирования

В автоматизации тестирования используются следующие инструменты:

Selenium — это один из самых популярных и широко используемых инструментов для автоматизации тестирования веб-приложений. Он поддерживает множество языков программирования, включая Java, C#, Python и Ruby, а также обеспечивает совместимость с большинством современных веб-браузеров. Selenium WebDriver позволяет создавать сложные сценарии взаимодействия с веб-страницами для тестирования их поведения в различных условиях.

Appium представляет собой инструмент для автоматизации тестирования мобильных приложений. Он поддерживает тестирование как нативных, так и гибридных мобильных приложений и работает на платформах iOS, Android и Windows. Appium основан на протоколе WebDriver и позволяет использовать те же API для тестирования мобильных приложений, что и Selenium для веб-приложений.

Postman широко используется для тестирования API, позволяя разработчикам легко создавать, делиться, тестировать и документировать API. Это инструмент, который предоставляет удобный пользовательский интерфейс для отправки запросов к API, получения ответов и анализа их содержимого, что делает его неоценимым помощником в процессе разработки и тестирования API.

Каждый из этих инструментов предлагает уникальные возможности и преимущества, позволяя командам выбирать наиболее подходящие решения в зависимости от специфики проекта, технических требований и предпочтений в методологии тестирования. Использование автоматизации тестирования улучшает эффективность процессов разработки и тестирования, сокращая время выхода продукта на рынок и повышая его качество.

VI. Нагрузочное тестирование

Нагрузочное тестирование — важнейший элемент тестирования производительности, целью которого является создание зеркала вероятной нагрузки на любое программное обеспечение, веб-приложение или веб-сайт. Его основная задача — выявить и устранить любые проблемы производительности до того, как программный продукт станет общедоступным или до того, как система станет полностью работоспособной [7].

Исходя из исследования [8], в котором авторы определяют важность нагрузочного тестирования (НТ) за его способность выявить как именно система работает в экстремальных условиях, и убедиться, способна ли она справиться с пиковым взаимодействием с пользователем. Это становится важным для устранения потенциальных проблем, которые могут снизить производительность системы и отрицательно повлиять на удобство работы пользователя, например, медленное время отклика, повышенное количество ошибок или даже сбои в системе. Выявление этих проблем до того, как система будет полностью работоспособна, позволяет их устранить, гарантируя пользователям беспрепятственный и эффективный пользовательский опыт.

Существует ряд основных правил НТ для обеспечения эффективной работы приложения или системы под высокой нагрузкой:

Настройка тестовой среды задачей которого заключается в создании тестовой среды, максимально приближенной к реальной производственной среде, с точным воспроизведением всех её компонентов, включая аппаратное и программное обеспечение, сетевые настройки и базы данных. Это необходимо для обеспечения достоверности результатов тестирования и их применимости к реальным условиям эксплуатации системы.

Разработка сценариев нагрузочного тестирования, которые должны отражать наиболее типичные или критические задачи, выполняемые системой. Кроме того, необходимо определить критерии производительности, такие как время отклика, пропускная способность и допустимый уровень ошибок.

Выполнение теста включает в себя запуск и мониторинг тестовых сценариев при заданных условиях нагрузки с использованием автоматизированных инструментов.

Анализ результатов тестирования необходим для оценки производительности системы с точки зрения достигнутых критериев. В случае выявления проблем с производительностью следует провести анализ, чтобы

определить причины недостаточной производительности и узкие места в архитектуре системы.

Авторы статьи [7] отмечают, что ***архитектура нагрузочного тестирования*** является важной основой для анализа и повышения эффективности веб-приложений. В её центре лежит процесс имитации множества пользовательских запросов, направленных на приложение через интернет.

При проведении нагрузочного тестирования крайне важно иметь специальную среду тестирования, которая максимально точно отражает производственную среду. Такие инструменты, как Puppet и Docker, могут помочь в настройке среды тестирования и управлении ею.

VII. Инструменты для нагрузочного тестирования

Инструменты нагрузочного тестирования — это бесценные программные утилиты, которые помогают оценить производительности системы в различных условиях нагрузки. Они инструменты имитируют действия множества одновременных пользователей, позволяя наблюдать за временем отклика системы, выявлять узкие места и обеспечивать удобство использования в реальных условиях.

Первым и одним из самых универсальных инструментов является ***Apache JMeter***: этот инструмент представляет собой приложение с открытым исходным кодом, созданное на Java и разработанное специально для тестирования функциональности и производительности нагрузки. JMeter, разработанный Apache Software Foundation, является универсальным и способен моделировать нагрузки по широкому спектру сервисов и протоколов, таких как HTTP, HTTPS, JDBC, LDAP и SOAP. Благодаря расширяемому ядру, которое можно адаптировать с помощью плагинов, оно обеспечивает гибкость, необходимую для различных сценариев тестирования.

LoadRunner, предоставленный Micro Focus, является более тяжелым инструментом нагрузочного тестирования, но в то же время широко распространенным. Его основная функция — моделировать реальные условия нагрузки в вашей системе, обнаруживать узкие места в производительности и прогнозировать поведение системы. Благодаря поддержке широкого спектра прикладных сред, платформ и баз данных LoadRunner является универсальным выбором для удовлетворения разнообразных требований нагрузочного тестирования. Он предлагает подробные показатели производительности, полезные для настройки и оптимизации производительности системы.

Gatling: инструмент нагрузочного тестирования и тестирования производительности с открытым исходным кодом, в первую очередь предназначенный для веб-приложений. Gatling использует простой предметно-ориентированный язык (DSL) для создания и поддержки тестовых сценариев. Он поддерживает протокол HTTP/2 и позволяет записывать и создавать сценарии прямо из браузера. Инструмент также предоставляет подробные отчеты о производительности, которые легко анализировать.

VIII. Тестирование на безопасность

Тестирование на безопасность веб-приложений — это процесс идентификации, анализа и устранения угроз и уязвимостей в веб-приложениях. Этот вид тестирования направлен на обеспечение защиты данных, функциональности и инфраструктуры веб-приложения от несанкционированного доступа, атак злоумышленников и других потенциальных угроз безопасности.

В исследовании [9] подчёркивается, что каждая единица информации в веб-приложении должна быть всесторонне защищена, чтобы она не могла привести к нарушению целостности или краже данных. Узнать бреши в уязвимостях безопасности можно с помощью методов тестирования на проникновение. Оценка уязвимостей, которая представляет собой метод сканирования лазеек на веб-сайтах, существующих в веб-приложении. Путем выполнения сканирования уязвимостей можно, к примеру, обнаруживать SQL-инъекции и другие уязвимости [10].

В статье [11] говорится, что инъекция – лидер из самых опасных методов взлома, целью которых преимущественно является получение данных пользователей, в частности логинов и паролей, которые позволят авторизоваться в системе с правами какого-либо пользователя. Наиболее распространенным видом инъекций является SQL-инъекция (SQLi), которая использует различные уязвимости сайта для отправки вредоносного кода в запросах к серверу базы данных (БД). Для SQL-инъекции могут быть использованы любые входные данные сайта: логин, пароль, ключевые слова для поиска, вводимые тэги, строки запроса, файлы cookie и т.д. Другими видами инъекций являются: NoSQL-инъекции, межсайтовые сценарии (Cross-Site Scripting, XSS), выполнение команд операционной системы (ОС), LDAP-инъекции (Lightweight Directory Access Protocol), инъекции шаблонов (EL, Expression Language), инъекции выражений OGNL (Object Graph Navigation Library), инъекции XML, XPATH и т.д.

IX. Инструменты для тестирования на безопасность

OWASP ZAP (Zed Attack Proxy): Разработанный OWASP (Open Web Application Security Project), ZAP или Zed Attack Proxy представляет собой мультиплатформенный инструмент тестирования безопасности веб-приложений с открытым исходным кодом. ZAP используется для поиска ряда уязвимостей в веб-приложении на этапе разработки, а также на этапе тестирования. Инструмент тестирования безопасности идеально подойдет как новичку из-за интуитивно понятного интерфейса, так и опытному пользователю. ZAP написан на Java. Помимо использования в качестве сканера, ZAP также можно использовать для перехвата прокси-сервера для ручного тестирования веб-страницы.

Burp Suite: Интегрированный платформенный решение для тестирования безопасности веб-приложений. Burp Suite предлагает широкий спектр инструментов для проведения как автоматизированных, так и ручных тестов безопасности.

X. Юзабилити-тестирование и его инструменты

Юзабилити-тестирование веб-сайта — это процесс проверки возможностей веб-сайта с целью убедиться, что веб-сайт прост в использовании, эффективен и понятен для пользователей. При проведении технико-экономического тестирования веб-сайта можно использовать несколько методов, один из которых — это метод «мышление вслух», разработанный Джейкобом Нильсеном, всемирно известным экспертом по юзабилити и пользовательскому опыту (UX) [12].

В исследовании [13] отмечают, что изменения в технологиях, включая доступ к пользователям в любом месте в любое время, наряду с изменениями в объеме тестирования (от очень крупных исследований к очень маленьким), означают, что определение юзабилити-тестирования должно расширяться, чтобы охватить методы и практики, которые поддерживают тестирование во многих различных окружениях и в самых разных условиях.

К преимуществам метода «мышление вслух» относится возможность у юзабилити-тестировщиков понять точку зрения пользователей на использование интерактивных продуктов и получить немедленную обратную связь.

Также отметим преимущество в выявлении проблем удобства использования. С помощью метода «мышление вслух» пользователи могут помочь выявить проблемы с удобством использования, которые можно преодолеть на ранних этапах разработки интерактивного продукта, что минимизирует предвзятость исследователя.

К слабым сторонам метода «мышление вслух» при тестировании юзабилити включают в себя: незаинтересованность пользователей в предоставлении обратной связи, когда их просят откровенно высказать свои мысли при использовании интерактивного продукта.

Метод «мышление вслух» менее эффективен при тестировании когнитивных аспектов, таких как понимание пользователем функций интерактивного продукта. В этом случае более подходят более конкретные методы тестирования, такие как когнитивное пошаговое тестирование или детальное тестирование. В то же время метод мышления вслух имеет тенденцию фокусироваться на пользовательском опыте при использовании интерактивных продуктов в изолированных ситуациях и не отражает пользовательский опыт в более сложных и реальных ситуациях.

К инструментам для Юзабилити-тестирования относятся:

UserTesting: позволяет получить обратную связь от пользователей в реальном времени.

Optimal Workshop: набор инструментов для различных аспектов юзабилити-тестирования, включая карточную сортировку.

Lookback.io: позволяет проводить удаленные юзабилити-исследования с записью экрана и видео реакций пользователей.

Данные инструменты предоставляют тестировщикам ценные данные о поведении пользователей, позволяя улучшить удобство использования веб-приложений и обеспечить положительный пользовательский опыт.

XI. Архитектура комплекса тестирования веб-приложений

В разработке современных веб-приложений автоматизация тестирования играет ключевую роль, обеспечивая высокое качество продукта и его безопасность. Комплекс для автоматизации тестирования представляет собой многоуровневую систему, включающую в себя следующие основные компоненты:

— Веб-интерфейс для пользователей

Фронтенд: создан для удобства пользователей, предоставляя интерактивный интерфейс для конфигурации и запуска тестов, а также для просмотра результатов. Этот слой обеспечивает прямое взаимодействие с пользователем, делая процесс настройки тестов интуитивно понятным и эффективным.

Бэкенд: отвечает за обработку запросов от фронтенда, включая взаимодействие с модулями планирования и исполнения тестов. Бэкенд также связывается с базой данных для сохранения конфигураций и результатов тестирований, обеспечивая централизованное хранение и доступ к данным.

— Модуль планирования тестов

Этот модуль принимает настройки тестирования от пользователя, проводит их валидацию и подготавливает к запуску. Он также генерирует задачи для исполнительных агентов, управляя процессом тестирования в соответствии с указанными настройками.

— Исполнительные агенты

Функциональное тестирование: используют Selenium для проведения тестов пользовательского интерфейса, имитируя действия конечных пользователей.

Тестирование производительности: применяют JMeter для создания искусственной нагрузки на веб-приложение, имитируя работу множества пользователей.

Тестирование безопасности: включает в себя использование инструментов, таких как OWASP ZAP, для выявления уязвимостей в безопасности веб-приложения.

API тестирование: осуществляют запросы к API веб-приложения (через Swagger), проверяя корректность работы интерфейсов приложения.

— Система управления

Осуществляет сбор и анализ логов исполнения тестов от агентов, предоставляя данные о ходе тестирования и выявленных проблемах. Эта система играет важную роль в контроле качества исполнения тестов.

— База данных

Служит для хранения конфигураций тестов и результатов их выполнения, обеспечивая быстрый доступ к истории тестирования и возможность анализа результатов.

— Модуль анализа и отчетности

Обрабатывает результаты тестирования, генерируя подробные отчеты и дашборды с результатами для пользователей. Отчеты представлены в формате OK/NOK с подробным описанием результатов тестов, что позволяет пользователям легко интерпретировать данные и

принимать решения о дальнейших действиях. На рис. 1 представлена общая схема работы программного комплекса.

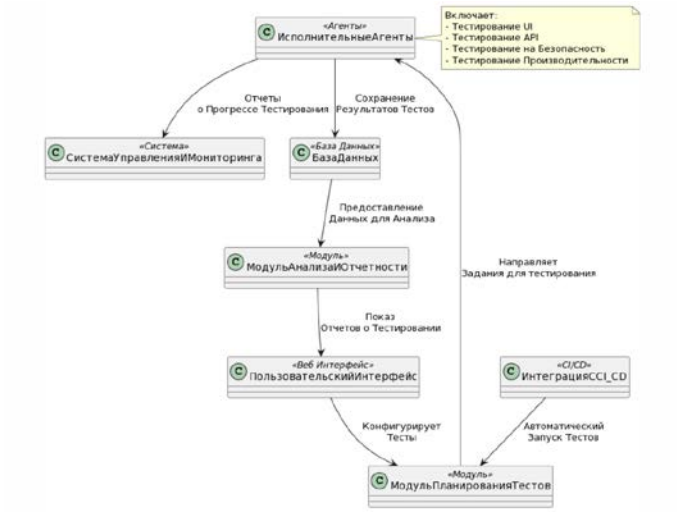

Рис. 1. Схема работы программного комплекса

XII. Сценарий работы пользователя/системы распределенного тестирования веб-приложений

Шаг 1: Настройка теста через веб-интерфейс.

Пользователи системы начинают с входа в систему через веб-интерфейс, представленный фронтенд-решением. Они переходят к форме настройки тестов, где им предоставляется возможность выбрать тип тестирования, такой как функциональное тестирование, тестирование производительности, тестирование на безопасность или API тестирование. Далее, пользователи указывают параметры для теста, включая URL веб-приложения, критерии успеха, требуемую нагрузку и другие важные детали. Завершающим действием этапа является сохранение конфигурации теста и отправка данных на бэкенд для дальнейшей обработки.

Шаг 2: Обработка и планирование теста.

После получения данных бэкенд передает информацию модулю планирования тестов. Этот модуль осуществляет валидацию настроек теста и подготавливает его к исполнению, генерируя задачи для исполнительных агентов согласно указанным параметрам.

Шаг 3: Исполнение теста.

Исполнительные агенты приступают к выполнению тестов согласно полученным задачам. Различные агенты отвечают за разные аспекты тестирования: Selenium используется для функционального тестирования интерфейса, JMeter создает нагрузку для тестирования производительности, OWASP ZAP сканирует приложение на наличие уязвимостей для тестирования безопасности, а Swagger применяется для тестирования API. Результаты и логи собираются агентами в процессе исполнения.

Шаг 4: Сбор результатов.

Система управления аккумулирует данные исполнения и результаты от всех агентов. Эта информация затем агрегируется и сохраняется в базе данных для последующего анализа.

Шаг 5: Анализ результатов и отчетность.

Модуль анализа и отчетности извлекает данные из базы и обрабатывает их, создавая детализированные отчеты и дашборды с результатами тестирования. Эти отчеты представляются пользователям через веб-интерфейс в понятном формате (OK/NOK), сопровождаясь подробными расшифровками результатов.

Шаг 6: Просмотр результатов

Пользователи возвращаются в веб-интерфейс для просмотра сгенерированных отчетов и дашбордов. Анализ результатов позволяет выявить потенциальные проблемы и узкие места в веб-приложении, направляя команду разработки на их устранение для улучшения качества и безопасности продукта.

Этот сценарий демонстрирует, как интегрированная система распределенного тестирования обеспечивает комплексную эффективную проверку веб-приложений, значительно повышая их качество и безопасность.

XIII. Заключение

Комплексное тестирование веб-приложений не только выступает в качестве залога создания продуктов высокого качества, надежности и безопасности, но и демонстрирует влияние на успешность и репутацию разработчика на рынке. Адаптация к постоянно развивающимся интернет-технологиям и внедрение современных методик тестирования требуют от специалистов глубоких знаний и постоянного самосовершенствования. Внедрение автоматизации тестирования, применение передовых инструментов для нагрузочного тестирования, обеспечение высокого уровня безопасности и оптимизация юзабилити веб-приложений открывают новые горизонты для повышения их качества и эффективности.

В этой связи, ключевую роль играет не только выбор подходящих инструментов и методик, но и стратегия их интеграции в процессы разработки решений. Интеграция тестирования в цикл непрерывной интеграции и доставки (CI/CD) позволяет не только сократить время разработки и ускорить выход продукта на рынок, но и гарантировать высокое качество конечного продукта на всех этапах его создания. Система распределенного тестирования, описанная в данной статье, представляет собой один из эффективных подходов к реализации этих задач, позволяя командам разработки быстро идентифицировать и устранять потенциальные проблемы, тем самым значительно улучшая качество и безопасность веб-приложений.

Таким образом, в мире, где цифровые технологии постоянно эволюционируют, а ожидания пользователей непрерывно растут, комплексное тестирование становится неотъемлемой частью успешной разработки веб-приложений. Оно требует глубокого понимания как технических аспектов, так и потребностей конечных пользователей, представляя собой искусство и науку одновременно. В итоге успешное тестирование приводит к созданию продуктов, которые не только функциональны, но и приносят радость и удовлетворение пользователям, способствуя их лояльности и доверию к разработчику подобных решений.

Библиография

[1] S. Doğan, A. Betin-Can, V. Garousi, “Web application testing: A systematic literature review,” Journal of Systems and Software, vol. 91, pp. 174-201, 2014. https://doi.org/10.1016/j.jss.2014.01.010.

[2] K. Jacksi, , S.M. Abass, “Development History Of The World Wide Web,” International Journal of Scientific & Technology Research, vol. 8, pp. 75-79, 2019.

[3] H. Li, R. Dang, Y. Yao, H. Wang, “A Review of Approaches for Detecting Vulnerabilities in Smart Contracts within Web 3.0 Applications,” Blockchains, vol. 1, pp. 3-18, 2023. https://doi.org/10.3390/blockchains1010002.

[4] S. Hanna, A.A.-S. Ahmad, “Web applications testing techniques: a systematic mapping study,” International Journal of Web Engineering and Technology, vol. 17, no.4, pp. 372-412, 2022. https://doi.org/10.1504/IJWET.2022.10054339.

[5] H. M. Ali, M.Y. Hamza, T.A. Rashid, “A Comprehensive Study on Automated Testing with The Software Lifecycle,” The Journal of Duhok University, vol. 26, no. 2, pp. 613-620, 2023. https://doi.org/10.26682/csjuod.2023.26.2.55.

[6] О.П. Берегейко, А.С. Дубовский, “Автоматизация тестирования веб-приложений,” Вестник магистратуры, № 12-4(63), с. 39-41, 2016. https://cyberleninka.ru/article/n/avtomatizatsiya-testirovaniya-veb-prilozheniy.

[7] S. Biswas, S. Hoda, “What is Load Testing? Complete Tutorial With Best Practices,” URL: https://www.lambdatest.com/learning-hub/load-testing (дата обращения: 07.03.2024).

[8] H. Malik, H. Hemmati, A.E. Hassan, “Automatic detection of performance deviations in the load testing of Large Scale Systems,” In Proceedings - International Conference on Software Engineering, pp. 1012-1021, 2013. https://doi.org/10.1109/ICSE.2013.6606651.

[9] A. Pandey, P. Chandra, R. Mahule, S. Das, S. Gupta, G. Prasad, Web testing and security audit of web application,” Chhattisgarh Journal of Science and Technology, vol. 18, no.2, pp. 125-127, 2021.

[10] Y. Armando, R. Rosalina, “Penetration Testing Tangerang City Web Application With Implementing OWASP Top 10 Web Security Risks Framework,” JISA (Jurnal Informatika dan Sains), vol. 6, pp. 105-109, 2023. https://doi.org/10.31326/jisa.v6i2.1656.

[11] Trofymenko, Olena & Dyka, Anastasiia & Loboda, Yuliia. (2023). Analysis of vulnerabilities and security problems of web applications. System technologies. 3. 25-37. https://doi.org/10.34185/1562-9945-3-146-2023-03.

[12] К.М. Барнум, *Основы юзабилити-тестирования*. М.: ДМК Пресс, 2021.

[13] S. Wicaksono, *Usability Testing*, 2023. https://doi.org/10.5281/zenodo.7705056.

Web Application Distributed Testing Software System

Pavel A. Redkin, Anton S. Aleshkin

***Abstract* — This article focuses on web application testing, which plays a key role in creating high-quality, dependable, and secure products. Modern adaptation to Internet technologies requires the introduction of modern testing methods, as well as in-depth knowledge and constant self-improvement of the processes of creating solutions. The article discusses the main stages of such improvement: the introduction of test automation, the use of advanced tools, and the provision of an overall high level of security. The key aspects of testing are load testing, security testing, and usability testing, as well as approaches to executing and analyzing test results.**

The article emphasizes the importance of an end-to-end approach to quality assurance for web applications and explores the role of automation in simplifying and streamlining testing processes. The article describes the various tools used for testing, including Apache JMeter, LoadRunner, Gatling for load testing, and OWASP ZAP and Burp Suite for security testing. The authors also discuss usability assessment methods and introduce modern usability testing tools such as UserTesting, Optimal Workshop, and Lookback.io.

The integration of testing into the "continuous integration and delivery" (CI/CD) cycle is allows you to reduce the development time, and, at the same time, guarantee the high quality of the product at all stages of its creation.

The result of the technology review is a distributed testing system, which is an effective approach to implementing test tasks, allowing development teams to quickly identify and fix even potential problems.

***Keywords*— test automation, load testing, web application testing, security testing.**

REFERENCES

[1] S. Doğan, A. Betin-Can, V. Garousi, “Web application testing: A systematic literature review,” Journal of Systems and Software, vol. 91, pp. 174-201, 2014. https://doi.org/10.1016/j.jss.2014.01.010.

[2] K. Jacksi, , S.M. Abass, “Development History Of The World Wide Web,” International Journal of Scientific & Technology Research, vol. 8, pp. 75-79, 2019.

[3] H. Li, R. Dang, Y. Yao, H. Wang, “A Review of Approaches for Detecting Vulnerabilities in Smart Contracts within Web 3.0 Applications,” Blockchains, vol. 1, pp. 3-18, 2023. https://doi.org/10.3390/blockchains1010002.

[4] S. Hanna, A.A.-S. Ahmad, “Web applications testing techniques: a systematic mapping study,” International Journal of Web Engineering and Technology, vol. 17, no.4, pp. 372-412, 2022. https://doi.org/10.1504/IJWET.2022.10054339.

[5] H. M. Ali, M.Y. Hamza, T.A. Rashid, “A Comprehensive Study on Automated Testing with The Software Lifecycle,” The Journal of Duhok University, vol. 26, no. 2, pp. 613-620, 2023. https://doi.org/10.26682/csjuod.2023.26.2.55.

[6] O.P. Beregeiko, A.S. Dubovskii Avtomatizatsiia testirovaniia veb-prilozhenii // Vestnik magistratury, № 12-4(63), pp. 39-41, 2016. https://cyberleninka.ru/article/n/avtomatizatsiya-testirovaniya-veb-prilozheniy. (In Russian)

[7] S. Biswas, S. Hoda, “What is Load Testing? Complete Tutorial With Best Practices,” URL: https://www.lambdatest.com/learning-hub/load-testing (дата обращения: 07.03.2024).

[8] H. Malik, H. Hemmati, A.E. Hassan, “Automatic detection of performance deviations in the load testing of Large Scale Systems,” In Proceedings - International Conference on Software Engineering, pp. 1012-1021, 2013. https://doi.org/10.1109/ICSE.2013.6606651.

[9] A. Pandey, P. Chandra, R. Mahule, S. Das, S. Gupta, G. Prasad, Web testing and security audit of web application,” Chhattisgarh Journal of Science and Technology, vol. 18, no.2, pp. 125-127, 2021.

[10] Y. Armando, R. Rosalina, “Penetration Testing Tangerang City Web Application With Implementing OWASP Top 10 Web Security Risks Framework,” JISA (Jurnal Informatika dan Sains), vol. 6, pp. 105-109, 2023. https://doi.org/10.31326/jisa.v6i2.1656.

[11] Trofymenko, Olena & Dyka, Anastasiia & Loboda, Yuliia. (2023). Analysis of vulnerabilities and security problems of web applications. System technologies. 3. 25-37. https://doi.org/10.34185/1562-9945-3-146-2023-03.

[12] C.M. Barnum. *Usability Testing Essentials*: Ready, Set ...Test! 2nd Edition. Imprint: Morgan Kaufmann. 2020

[13] S. Wicaksono, *Usability Testing*, 2023. https://doi.org/10.5281/zenodo.7705056.

About the authors

Pavel Redkin, bachelor of the Institute of Cybersecurity and Digital Technologies of MIREA - Russian Technological University (MIREA, MGUPI, MITHT, VNIITE, ROSNIIT and AP, IPK of the Ministry of Education and Science of the Russian Federation), Moscow, Russia. Currently, works on a final qualifying work named "Development of a software package for distributed testing of web applications".

Anton Aleshkin, Ph.D. of Technical Science, currently employed as the associate professor for the Institute for Cybersecurity and Digital Technologies of MIREA - Russian Technological University (MIREA, MGUPI, MITHT, VNIITE, RosNIIIT and AP, IPK of the Ministry of Education and Science of the Russian Federation), Moscow, Russia. The present research is focused on Computer Science, computer and transport networks and mathematical modelling with percolation theory. Anton takes research about Smart Cities and transport behavior in it. His papers appeared in Mathematics, Journal of Physics, Russian Technological Journal, and some conference proceedings.