Тестирование web-приложений

Ключевые особенности и этапы тестирования WEB-приложений, обзор используемых технологий и применяемых решений.

Введение

Тестирование WEB-приложений – это наблюдение за функционированием программы в искусственно созданных обстоятельствах, которое позволяет определить ее соответствие поставленным требованиям, например: производительность WEB-приложения, его степень защиты и комфорт использования для рядового пользователя.

Задачи тестирования WEB-приложений:

-Нахождение ошибок в работе программы и эффективное их устранение;
-Проверка приложения на соответствие конкретным требованиям;
-Проверка качества работы создателей приложения;
-Обработка информации, которая станет основой для принятия последующих действий.
Информация, полученная в процессе тестирования, позволяет выстроить последующую стратегию развития приложения. Важно понимать, что тестирование WEB-приложений является важным этапом, так как даже самые успешные программисты таких интернет-гигантов как Google, Sony и Facebook могут допустить ошибки, которые в последствии могут стать причиной уязвимости WEB-приложения и утечки информации его пользователей.

Особенности архитектуры WEB-приложения

Согласно статистике на 2016 год, которую проводил Международный союз электросвязи, в интернет-пространстве уже имеется свыше одного миллиарда приложений, и порядка 3,5 миллиарда людей со всей планеты пользуются разного рода интернет-приложениями.

WEB-приложение является клиент-серверным приложением, где в качестве клиента выступает браузер. При этом бизнес-логика приложения может выполняться как на стороне клиента, так и на стороне веб-сервера, обрабатывающего запросы пользователей IT-продукта.

Типичная архитектура веб-приложения выглядит следующим образом:

Веб-страницы

Веб-страницы состоят из HTML, CSS и Javascript-кода, а также могут включать в себя дополнительные элементы, такие как изображения или видео. Веб-страницы отображаются непосредственно на стороне пользователя.

Сервер

Сервер принимает HTTP-запросы от пользователей и выдает им соответствующие HTTP-ответы, которые могут содержать веб-страницы для отображения, либо данные (обычно в формате JSON), используемые Javascript-логикой для динамического обновления веб-страницы. Веб-сервер может быть реализован с использованием различных языков программирования, таких как Java, Perl, Ruby, Python, PHP и других.

База данных

Классическая теория воплощает в себе две «стороны» приложения, но в таком алгоритме активной «стороной» выступает третий элемент, такой как база данных. Базу данных не относят к веб-серверу, но большая часть веб-приложений не могут без нее полноценно функционировать, так как база данных содержит в себе динамическую информацию, такую как учетные и пользовательские данные.

Под термином «база данных» понимается информационная модель, которая имеет различный набор качеств, которые можно систематизировать. База данных функционирует за счет систем управления базами данных (сокращенно СУБД). Одними из наиболее известных СУБД являются MySQL, PostgreSQL и др.

Особенности тестирования WEB-приложений

Клиент является первым элементом архитектуры, где под клиентом понимается браузер, и здесь сразу возникает вопрос тестирования кроссбраузерности. Кроссбраузерность — это тестирование правильности работы WEB-приложения в различных браузерах и на различных операционных системах.
Кроссбраузерное тестирование проводит проверку по таким пунктам:

-Функционал, который реализуется на «стороне» клиента;
-Проверка правильности отображения графических элементов web-приложения;
-Интерактивность веб-приложения и др.

В диагностике нуждаются в первую очередь популярные браузеры, такие как Google Chrome, однако в случае наличия требований поддержки очень старых браузеров, может потребоваться тестирование, например, в браузере Internet Explorer.

При тестировании веб-приложения отдельное внимание нужно уделить валидаторам верстки. Для этого не требуется особых знаний, так как оценка соответствия качества верстки современным стандартам может выполняться за счет автоматических средств.

Следующая особенность веб-приложения – это формы заполнения, и здесь нужно выделить такие элементы, как:

Проверка обязательности заполнения полей с учетом их маркировки;
Проверка отправки форм на соответствие ожидаемому результату;
Проверка использования чит-листов для тестирования форм;
Проверка выдачи информационных сообщений;
Проверка реализации экранирования символов;
Проверка отправки подтверждающих писем, которые приходят на почту клиента, после того как он заполнит форму;

Проверка возможности использования специальных инструментов для диагностики форм, таких как Web Developer Toolbar.

Виды и этапы тестирования web-приложений

Виды и этапы тестирования WEB-приложений определяются исходя из стратегии тестирования. Это делается для того, чтобы определить какие ресурсы нужно привлечь в настоящий момент, чтобы только созданная программа была эффективно запущена в работу и функционировала без сбоев.

Выделяются такие виды тестирования программ, как:

Тестирование функционала;
Оценка удобства использования веб-приложения;
Оценка интерфейса;
Оценка производительности;
Оценка безопасности.

При функциональном тестировании определяют работает ли каждая функция WEB-приложения согласно спецификации требования. Сюда входит проверка работы ссылок, форм пользователя, проверка кода HTML и CSS, тестирование workflow и др.

При тестировании удобства пользования определяют характеристики взаимодействия пользователя с web-приложением, чтобы обнаружить недостатки и эффективно их устранить. При такой диагностике обращают внимание на простоту обучения, навигацию, общий вид и др.

При тестировании интерфейса пользователя оценку проходит графический интерфейс WEB-приложения. В ходе диагностики оценивают, внешний вид интерфейса, его удобство и простоту использования. Важно отметить, что на этом этапе оценивают совместимость веб-приложения с различными браузерами и версиями браузеров.

При тестировании производительности проводится проверка веб-приложения на способность выдерживать большие нагрузки. Здесь нужно отметить, что так называемое стресс-тестирование будет определять пределы производительности веб-приложения вплоть до полного отказа работы.

Нагрузочное тестирование проводится для того, чтобы определить, сколько пользователей могут одновременно обращаться к одной странице без потери качества их обслуживания.

Еще один важный этап тестирования веб-приложений – это диагностика безопасности, которая позволяет оценить степень защиты приложения от разного рода уязвимостей, атак и других рисков, как правило, приводящих к большим потерям.

Задача диагностики безопасности – это обнаружение угроз, которые разработчики должны устранить еще на этапе запуска веб-приложения в работу.

Виды и этапы тестирования веб-приложения должны быть отражены в документе, в котором будут перечислены все ошибки. На основе данного документа разработчики должны провести полное устранение ошибок и выполнить повторное тестирование, чтобы убедиться в том, что все риски и ошибки действительно устранены.

Если говорить о функциональном тестировании WEB-приложений, то возможна дополнительная инструментальная поддержка, которая будет включать следующие шаги:
-Разработка модели WEB-приложения;
-Разработка тестового сценария;
-Анализ результатов.

Данный подход позволяет обнаружить и перечислить классические уязвимости, такие как генерация на странице скрипта (XSS), также уязвимости XSRF, PHP, SQL и др. **Среди уязвимостей особенное внимание уделяют такой как несанкционированный доступ**

к учетной записи и переполнение буфера. Рекомендованные к использованию чит-листы – это XSS Filter, Evasion Cheat Sheet, MySQL SQL Injection Cheat Sheet.

Аналитики и разработчики должны еще до начала диагностики приложения составить план, в котором будут следующие пункты тестирования:
- Цель диагностики;
- Вид диагностики;
- Определение специфики и ЦА приложения;
- Перечень операционных систем и браузеров, которые будут задействованы в ходе диагностики.

Обзор технологий для тестирования WEB-приложений

Программное обеспечение становится всем более сложным, при этом оно должно корректно работать, и без его тестирования обойтись невозможно. Важно понимать, что изолировать код, чтобы исправлять ошибки сейчас стало все сложнее.

Сегодня как эффективный вариант диагностики используют модульное тестирование, которое снижает риск возникновения ошибок, уменьшает их, и благодаря его использованию разработчики могут получить документацию для тестируемых кодов. Наравне с модульным тестированием используется регрессия и автоматизация UI. Примеры популярных библиотек для тестирования: QUnit, JSUnit, YUI Test, xUnit.

Модульный тест может проверить лишь одну функцию за один тест, при этом используя xUnit вы вначале задаете условия, затем руководствуетесь в процессе тестирования этими условиями, и в финале проверяете вывод диагностики.
Еще одна технология – это QUnit (инфраструктура JavaScript-кода). Ее можно считать облегченной версией тестирования, которая проста и удобна, с ее помощью можно тестировать любой код.

Как было указано выше, тестирование UI с помощью CUIT может выполняться в автоматизированном режиме, в нем возможно записывать новые тесты и редактировать текущие.

Приведённые выше технологии юнит-тестирования можно использовать вместе с Selenium или аналогичными технологиями, симулирующими реальное поведение пользователя в браузере, что позволяет покрыть большую часть встречающихся на практике случаев и выявить большинство потенциальных проблем.

Прогнозы на будущее (тенденции, новые методы и технологии применения)

Рассматривая современные тенденции, первое, на что следует обратить внимание это развитие веб-браузеров. **В настоящее время на рынке преобладают браузеры, основанные на Chromium, и единственным конкурентом с альтернативным движком является браузер Firefox.** Учитывая сложность современных веб-стандартов, маловероятно, что в ближайшее время будут появляться новые браузерные движки, поэтому можно предположить, что тестирование клиентской части будет сведено к двум упомянутым выше браузерам.

Можно предположить, что в будущем особую популярность обретут облачные SaaS решения для проведения тестирования, которые будут позволять удобным образом проверять наиболее популярные случаи – причём, возможно, даже без необходимости привлечения специалиста. Тем не менее, с большой долей уверенности можно сказать, что существующие инструменты будут актуальны ещё продолжительное время, и, вероятнее всего, продолжат совершенствоваться разработчиками.

Заключение

Подводя итог, для тестирования современных веб-сайтов и веб-приложений имеется существенное количество технологий и подходов, что, при должном их использовании позволяет выявить большую часть ошибок и потенциальных проблем, позволяя существенно повысить качество конечного продукта или услуги, что, в свою очередь, может существенно отразиться на состоянии бизнеса – увеличить лояльность клиентов, повысить метрики удержания и, в конечном счёте, благополучно сказаться как на прибыли компании, так и на удобстве пользователей. При этом следует принимать во внимание, что тестирование имеет собственные издержки – в частности, время специалистов, затрачиваемое на покрытие тестами необходимого функционала.

Список литературы

Функциональное тестирование Web-приложений на основе технологии UniTesk / А.А.Сортов, А.В.Хорошилов – стр. 92.
Применение UniTesk к тестированию встроенных систем2004/Пакулин Н.В. – стр. 117.
Автоматизация тестирования Web-приложений, основанных на скриптовых языках 2008/Силаков Д.В. – стр. 160.