ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

"НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО"

(УНИВЕРСИТЕТ ИТМО)

Факультет программной инженерии и компьютерной техники

Направление (специальность) 09.04.04 Программная инженерия

Образовательная программа Веб-технологии

Дисциплина — Проектирование и анализ языков веб-решений

Курсовой проект (работа)

ТЕМА: Сравнительный анализ JavaScript фреймворков для автоматизации функционального тестирования

ВЫПОЛНИЛ

Студент группы P41081 _____ Д. И. Рева

№ группы подпись, дата ФИО

ПРОВЕРИЛ преподаватель _____ И. Б. Государев

ученая степень, должность подпись, дата ФИО

САНКТ-ПЕТЕРБУРГ

2021 г.

Оглавление

1 Введение

Веб-приложения интегрированы в жизнь практически каждого человека. За ответом на определенный вопрос чаще всего люди обращаются к Интернету, большинство задач (повседневных или рабочих) выполняются с помощью веб-технологий.

Отсюда следует, что и разработка подобного рода продуктов пользуется популярностью. Один из этапов создания веб-приложения – тестирование или оценка качества. Этот шаг позволяет:

- проверить готовое решение на отсутствие дефектов и на соответствие требованиям;
- предоставить конечным пользователям продукт, который гарантированно даст им возможность эффективно удовлетворить свои потребности и достичь своих целей.

Эти факты обуславливают актуальность темы данного исследования.

Цель курсовой работы – сравнительный анализ JavaScript фреймворков автоматизированного тестирования для формирования представления об организации процесса автоматизации тестирования в современных реалиях, а также о возможностях используемых для этого инструментов.

Объект исследования – JavaScript фреймворки тестирования.

Задачи исследования:

1. Провести обзор зарубежных и отечественных научных источников, описывающих процесс тестирования веб-приложений и используемых инструментов.
2. Подобрать необходимые для анализа JavaScript фреймворки.
3. Провести эксперимент и собрать необходимые метрики для каждого инструмента автоматизации тестирования.
4. Провести сравнительный анализ, используя собранные в ходе эксперимента метрики.

2 Основные понятия

В настоящее время интернет играет важную роль в жизни людей. Взаимодействие с информацией (ее поиск, хранение и передача) в большинстве своем построено на работе с веб-приложениями. Их необходимость и популярность растет, а, соответственно, растут потребности пользователей. Требования все больше усложняются, что меняет все этапы цикла создания приложений:

- разработку требований;
- программную реализацию веб-приложения;
- тестирование;
- дальнейшую поддержку.

Этап тестирования необходим для проверки соответствия приложения требованиям клиентов. Более того, этот процесс применяется с целью предупреждения дефектов, которые могут впоследствии помешать пользователям эффективно работать с программным продуктом.

Сфера тестирования веб-приложений достаточно хорошо развита. Тестирование можно классифицировать по нескольким признакам:

- по исполнителю тестовых сценариев:
  - мануальное тестирование;
  - автоматическое тестирование;
- по объекту тестирования [1]:
  - функциональное тестирование;
  - usability тестирование;
  - тестирование совместимости;
  - performance тестирование.

Для осуществления каждого типа тестирования существует множество инструментов и фреймворков. Чтобы организовать эффективный процесс оценки качества веб-приложения, необходимо выбрать стек технологий, подходящий под специфику проверяемого приложения.

3 Сравнительный анализ фреймворков для автоматизации функционального тестирования

3.1 Исследование существующих инструментов

На данный момент существует множество инструментов для автоматизации тестирования веб-приложений, начиная от юнит- и интеграционного тестирования и заканчивая тестированием интерфейса и функциональным тестированием. В данной курсовой работе рассматриваются фреймворки для функционального тестирования, следовательно далее приведен список подходящих под данную задачу инструментов:

- **Selenium**

Selenium – экосистема для автоматизации тестирования веб-приложений на всех браузерах и операционных системах. Более того, Selenium – бесплатный инструмент с открытым исходным кодом [2]. В комплекс средств семейства Selenium входят следующие модули [3]:

1. Selenium IDE – инструмент, который позволяет создавать автоматизированные тесты. Реализован как надстройка над браузером Mozilla Firefox. Недостаток модуля в его простоте, он не позволяет использовать условия и циклы, а также исключать дублирования кода.
2. Selenium Remote Control (RC) – одна из версий Selenium API, сервер, который использовался до объединения его с Selenium WebDriver. Данный продукт обеспечивает взаимодействия с DOM-элементами веб-страницы.
3. Selenium 2 (WebDriver) – последняя версия Selenium, является основной ветвью развития проекта. Данный модуль использует только WebDriver API, что дает возможность не запускать отдельное клиентское приложение (как было в Selenium RC). WebDriver не требует никакой установки,

необходимо только его скачивание. Также эта версия драйвера взаимодействует с DOM-элементами напрямую, имитируя действия пользователя в браузере.

4. Selenium Grid позволяет выполнять несколько тестов параллельно в разных браузерах и платформах, его использование построено на работе с Docker контейнерами, в которых и разворачивается взаимодействие автоматических тестов с браузерами.

- **TestCafe**

TestCafe[4] является альтернативой инструментам, основанным на Selenium. Это библиотека с открытым исходным кодом, написанная на JavaScript и ориентированная на тестирование. TestCafe внедряется в страницы в виде JS-скрипта, она не контролирует браузер, как это делает Selenium. Это позволяет TestCafe выполняться в любых браузерах, включая мобильные, и иметь полный контроль над тем, что происходит в JavaScript.

- **Cypress**

Cypress[5] – конкурент и аналог TestCafe. Работает по такому же принципу, что и TestCafe, то есть внедряет тесты в код веб-страниц. Cypress позволяет разрабатывать не только end-to-end тесты, но и интеграционные, и юнит-тесты.

- **Puppeteer**

Puppeteer[6] - это библиотека для Node.js, которую разработала компания Google. Она предоставляет удобное Node.js API для управления браузером Chrome без пользовательского интерфейса. Подобный браузер — это обычный Chrome (версия 59 и выше), который запускается с флагом --headless. Когда браузер выполняется в таком режиме, он предоставляет API для управления им, а Puppeteer — это, как уже было сказано, JS-инструмент, созданный Google для управления браузером.

- **Playwright**

Playwright[7] – кросс-браузерный аналог Puppeteer от Microsoft. Достаточно новый и развивающийся инструмент. Playwright наследует преимущества Puppeteer, добавляя к этому кросс-браузерность и более оптимизированный процесс запуска тестовых сценариев.

3.2 Выработка критериев сравнения фреймворков

Тестирование веб-приложения имеет свои особенности. Следовательно, необходимо подбирать фреймворк для тестирования, исходя из этих особенностей. В работе [8] приводится набор метрик, для сравнения инструментов автоматизации функционального тестирования:

- поддерживаемые браузеры;
- поддерживаемые языки программирования;
- инструмент с лицензией или с открытым кодом.

В контексте данной курсовой работы второй критерий не несет какой-либо информации, так как рассматриваются только JavaScript фреймворки. Третий критерий также не требует анализа, так как инструменты из пункта 3.1 все с открытым кодом.

Исходя из сказанного выше, необходимо выделить следующие критерии для оценки инструментов тестирования:

- кросс-браузерность;
- скорость и простота установки фреймворка;
- скорость выполнения автоматического тестового сценария.

3.3 Планирование эксперимента

Эксперимент заключается в сравнении различных JavaScript инструментов автоматизации функционального тестирования. В качестве объекта тестирования в эксперименте планируется использовать сайт университета ИТМО. Анализируемые инструменты:

- Selenium;
- TestCafe;
- Cypress;

- Puppeteer;
- Playwright.

Запуск экспериментальных сценариев будет организован с использованием Docker. Это позволит получить независимые от мощности локальной машины результаты производительности фреймворков тестирования.

С помощью каждого фреймворка необходимо разработать три тестовых сценария:

1. Открытие сайта университета и создание скриншота. Данный кейс необходим для проверки существования данной функциональности во фреймворке. Создание скриншотов в ходе теста достаточно важно для организации процесса логирования и создания отчетов с результатами тестирования.
2. Открытие страницы поступления в магистратуру и валидация шагов в алгоритме поступления. Данный кейс включает в себя открытие новой вкладки в браузере по клику на ссылку. Это позволит исследовать возможности работы с несколькими вкладками.
3. Заведомо провальный сценарий, чтобы оценить обработку ошибок фреймворком для тестирования.

В ходе эксперимента планируется собрать следующие метрики для сравнения инструментов автоматизации тестирования:

- кросс-браузерность;
- скорость и простота установки фреймворка;
- скорость выполнения автоматического тестового сценария.

3.4 Проведение эксперимента

3.4.1 Selenium

Кросс-браузерность. Selenium поддерживает все типы браузеров. Данное свойство обеспечивается за счет использования различных драйверов под разные браузеры.

Скорость и простота установки. Selenium возможно установить с помощью Node.js. Кроме прямого функционала Selenium устанавливаются драйверы для разных браузеров.

3.4.2 TestCafe

Кросс-браузерность. TestCafe поддерживает различные браузеры, они перечислены на их официальном сайте[4].

Скорость и простота установки. Инструмент устанавливается с помощью команды node i testcafe.

3.4.3 Cypress

Кросс-браузерность. Cypress поддерживает различные браузеры.

Скорость и простота установки. Возможна установка с помощью node i cypress команды, а также с помощью загрузки и запуска исполняемого файла. Кроме фреймворка для создания тестов устанавливается среда для конфигурации Cypress и среда для запуска тестов.

3.4.4 Puppeteer

Кросс-браузерность. Анализируя документацию Puppeteer[6], можно сделать вывод о том, что данный фреймворк возможно использовать только при тестировании Chromium и Chrome. Более того, при установке данного инструмента устанавливается необходимая версия Chromium, которая и будет использоваться по умолчанию. Также стоит отметить, что недавно появилась экспериментальная версия Puppeteer, поддерживающая Firefox Nightly.

Скорость и простота установки. Puppeteer является npm модулем, что дает возможность установить его единственной командой npm i puppeteer. Вместе с данным модулем установится Chromium браузер, если его нет на локальной машине.

3.4.5 Playwright

Кросс-браузерность. Инструмент поддерживает различные браузеры.

Скорость и простота установки. Playwright устанавливается с помощью npm i playwright команды. Кроме прямого функционала устанавливаются драйверы разных браузеров (firefox, chromium, webkit).

Результаты по третьему критерию – скорость выполнения автоматического тестового сценария – приведены в пункте ниже.

3.5 Анализ результатов эксперимента

Скорость выполнения тестового сценария измерялась следующим образом. С помощью каждого фреймворка было разработано три тестовых сценария. Каждый из сценариев был запущен 10 раз для расчета среднего значения скорости выполнения сценариев. Тесты запускались внутри Docker контейнера.

Временные показатели собирались по-разному для каждого фреймворка. Cypress и TestCafe самостоятельно подсчитывают время прохождения сценария. Selenium, Playwright и Puppeteer такого функционала не имеют, поэтому подсчет производился с помощью performance.now() функции JavaScript.

Первым сценарием было открытие веб-страницы и получение ее скриншота. Все тестовые фреймворки обладают данным функционалом. Сравнение по времени представлено на рисунке 1.

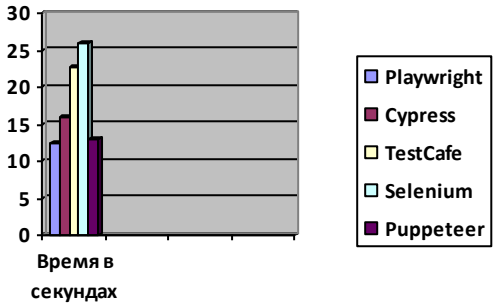

Рисунок 1. Время выполнения сценария 1 в различных фреймворках

Второй сценарий включает в себя следующие шаги:

1. Открытие сайта;
2. Переход на страницу поступления в магистратуру;
3. Валидация шагов в алгоритме поступления.

Важно отметить, что в Cypress реализация данного сценария невозможна по двум причинам:

- Cypress не поддерживает тестирование нескольких вкладок в браузере (на втором шаге при клике по ссылке открывается новая вкладка).

  Эту проблему возможно решить, если получить у ссылки href атрибут и перейти по этой ссылке на той же вкладке с помощью Cypress команды cy.visit(). Однако при таком подходе была найдена следующая проблема.

- Cypress запрещает тестировать различные домены в одном тестовом сценарии (URL главной страницы ИТМО отличается от URL страницы поступления в магистратуру).

Сравнение фреймворков по времени прохождения сценария 2 представлено на рисунке 2.

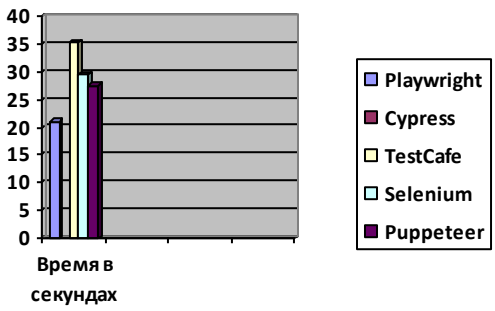

Рисунок 2. Время выполнения сценария 2 в различных фреймворках

Третий сценарий задуман быть провальным, чтобы оценить обработку ошибок различными инструментами автоматизации тестирования. Cypress и TestCafe предоставляют читабельные описания ошибок, статусы прохождения тестов. Более того, в Cypress предоставляется ссылка на документацию, где можно более подробно изучить проблему. Selenium, Playwright и Puppeteer не обрабатывают успешный статус прохождения тестов, он никак не логируется. Провальные тесты описываются вместе со стектрейсом, однако данный

лог не читабелен. Сравнение фреймворков по времени прохождения сценария 3 представлено на рисунке 3.

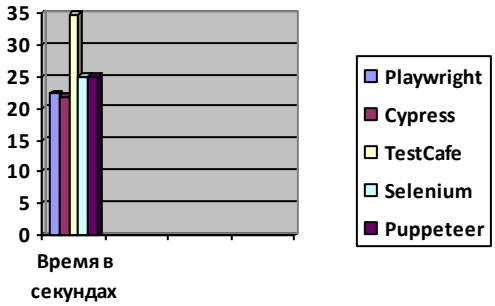

Рисунок 3. Время выполнения сценария 3 в различных фреймворках

Выводы:

Исходя из полученных результатов, среди JavaScript фреймворков для автоматизации тестирования лидируют Cypress и Playwright. Какой из них выбрать, зависит от конкретной задачи.

Cypress – инструмент, ориентированный на тестирование. Он подходит для освоения автоматизации тестирования, благодаря качественно организованной среде для работы: полная и понятная документация, среда для разработки, запуска тестов и анализа результатов тестирования. Однако данный инструмент не подходит для тестирования сложных приложений, в которых существует возможность работы в нескольких вкладках или же в которых происходит редирект пользователя на URL с другими доменами.

Playwright – новый инструмент для автоматизации действий в веб-приложении. Он снискал славу благодаря качественной документации и высокой скорости исполнения сценариев. Данный инструмент подходит для кроссбраузерного тестирования, тестирования приложений, использующих несколько вкладок. Однако Playwright недостаточно удобен с точки зрения анализа результатов тестирования, данный процесс необходимо дополнительно настраивать и дорабатывать, что предполагает высокий уровень программирования у специалиста, разрабатывающего архитектуру тестирования веб-приложения.

4 Заключение

В ходе реализации курсового проекта были решены все поставленные задачи:

- проведен обзор зарубежных и отечественных научных источников, описывающих процесс тестирования веб-приложений и используемых инструментов;
- подобраны необходимые для анализа JavaScript фреймворки;
- проведен эксперимент и собраны необходимые метрики для каждого инструмента автоматизации тестирования;
- проведен сравнительный анализ с использованием собранных в ходе эксперимента метрик.

Были сделаны следующие выводы:

- Cypress и Playwright лидируют среди JavaScript фреймворков автоматизированного тестирования, так как достаточно удобны в установке и использовании, поддерживают тестирование различных типов браузеров, а также предоставляют возможность наиболее быстро организовывать запуск тестовых сценариев.
- Cypress – полноценная платформа для тестирования, которая внутри себя содержит все элементы для автоматизации (разработка скриптов, их запуск в различных браузерах, логирование и создание отчетов о прохождении сценариев). Но данный инструмент не поддерживает некоторые функции сложных веб-приложений (например, использование нескольких вкладок в браузере или переходы по ссылкам с различными доменами).
- Playwright – инструмент для автоматизации действий в веб-приложении. В нем есть возможность запускать тестовые кейсы в различных браузерах, в нескольких вкладках и с разными URL. Единственный отмеченный недостаток – недоработанная система логирования результатов тестирования.

5 Список литературы

1. Umair M. Quality Assurance Process for Web Applications.
2. Selenium [Электронный ресурс] URL: https://www.selenium.dev/ (дата обращения: 06.12.2020).
3. Янгунаев В. М., Янгунаева Е. А. Исследование средств семейства Selenium для автоматизированного тестирования веб-приложений //Вестник научных конференций. – ООО Консалтинговая компания Юком, 2016. – №. 1-5. – С. 218-219.
4. TestCafe [Электронный ресурс] URL: https://testcafe.io/ (дата обращения 07.05.2021).
5. Cypress [Электронный ресурс] URL: https://www.cypress.io/ (дата обращения 07.05.2021).
6. Puppeteer [Электронный ресурс] URL: https://github.com/puppeteer/puppeteer (дата обращения: 07.05.2021).
7. Playwright [Электронный ресурс] URL: https://github.com/microsoft/playwright (дата обращения: 07.05.2021).
8. Lakshmi D. R., Mallika S. S. A review on web application testing and its current research directions //International Journal of Electrical and Computer Engineering. – 2017. – Т. 7. – №. 4. – С. 2132.

6 Приложения

6.1 Код тестовых сценариев

Код тестовых сценариев можно найти в репозитории https://github.com/direva/QAAutomationFrameworks .