**GEEKBRAINS**

**ФАКУЛЬТЕТ РАЗРАБОТЧИК-ТЕСТИРОВЩИК**

ВЫПУСКНАЯ ДИПЛОМНАЯ РАБОТА НА ТЕМУ:

**«Автоматизация тестирования с целью уменьшения трудозатрат при установке новой версии продукта на примере интернет-магазина одежды, обуви и аксессуров»**

Печерских Алёна Сергеевна

Самара, 2023 г.

# СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc140888885)

[1. ИНСТРУМЕНТЫ ДЛЯ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ 4](#_Toc140888886)

[1.1. Selenium 4](#_Toc140888887)

[1.2. Puppeteer 7](#_Toc140888888)

[1.3. Cypress 8](#_Toc140888889)

[1.4. Playwright 9](#_Toc140888890)

[1.5. Сравнение и выводы 11](#_Toc140888891)

[2. ВЫДЕЛЕНИЕ БАЗОВЫХ СЦЕНАРИЕВ ИСПОЛЬЗОВАНИЯ ВЕБ-ПРИЛОЖЕНИЯ LCWAIKIKI.KZ 15](#_Toc140888892)

[2.1. Описание функционала веб-приложения lcwaikiki.kz 15](#_Toc140888893)

[2.2. Описание базовых сценариев использования веб-приложения lcwaikiki.kz 15](#_Toc140888894)

[ЗАКЛЮЧЕНИЕ 16](#_Toc140888895)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И РЕСУРСОВ 17](#_Toc140888896)

[ПРИЛОЖЕНИЯ 18](#_Toc140888897)

# ВВЕДЕНИЕ

Цель данной дипломной работы по специальности «Разработчик-Тестировщик» – исследование инструментов для реализации автоматизации тестирования базовых сценариев использования продукта на примере интернет магазина lcwaikiki.kz, а также разработка приложения для автоматизированной валидации функционала, которое позволит уменьшить трудозатраты на проверку работоспособности веб-портала после обновления.

Автоматизация тестирования после установки новой версии продукта является критичной задачей в особенности для поддержки крупного продукта, т.к. позволяет избежать огромных затрат, которые имеют место быть в случае ручной проверки.

Исходя из цели, в дипломной работе поставлены и решены следующие задачи:

1. Рассмотреть основные способы взаимодействия с браузером. Выделить плюсы и минусы того или иного способа;
2. Выбрать инструмент взаимодействия с браузером для реализации автоматизированных тестов;
3. Ознакомиться с функционалом интернет-магазина lcwaikiki.kz, выделить базовые сценарии использования, которые должны тестироваться после установки новой версии продукта (веб-приложения);
4. Реализовать сценарии автоматизированного тестирования для проверки базового функционала lcwaikiki.kz с использованием Java;
5. Собрать финальный билд для запуска автотестов с помощью команды в Windows/Linux.

Задачи выше были решены с использованием следующих инструментов: Java как язык разработки приложения для автотестов, Intellij Idea как среда разработки, Playwright как инструмент для взаимодействия с браузером, Maven как инструмент для сборки финального приложения для запуска в Windows/Linux.

# ИНСТРУМЕНТЫ ДЛЯ АВТОМАТИЗИРОВАННОГО ТЕСТИРОВАНИЯ

## Selenium

Selenium – это проект, в рамках которого разрабатывается серия программных продуктов с исходным кодом (open source), открытым в декабре 2004 года [1, 2, 3]:

* Selenium WebDriver;
* Selenium RC;
* Selenium Server;
* Selenium Grid;
* Selenium IDE.

**Selenium WebDriver** – это программная библиотека для управления браузерами. Часто употребляется также более короткое название WebDriver. Это целое семейство драйверов для различных браузеров, а также набор клиентских библиотек на разных языках, позволяющих работать с этими драйверами.

Selenium WebDriver – это основной продукт, разрабатываемый в рамках проекта Selenium.

В рамках проекта Selenium разрабатываются драйверы для браузеров Firefox, Internet Explorer и Safari, а также драйверы для мобильных браузеров Android и iOS. Драйвер для браузера Google Chrome разрабатывается в рамках проекта Chromium, а драйвер для браузера Opera (включая мобильные версии) разрабатывается компанией Opera Software. Поэтому они формально не являтся частью проекта Selenium, распространяются и поддерживаются независимо. Но логически, конечно, можно считать их частью семейства продуктов Selenium.

Аналогичная ситуация и с клиентскими библиотеками – в рамках проекта Selenium разрабатываются библиотеки для языков Java, .Net (C#), Python, Ruby, JavaScript. Все остальные реализации не имеют отношения к проекту Selenium, хотя, возможно, в будущем, какие-то из них могут влиться в этот проект.

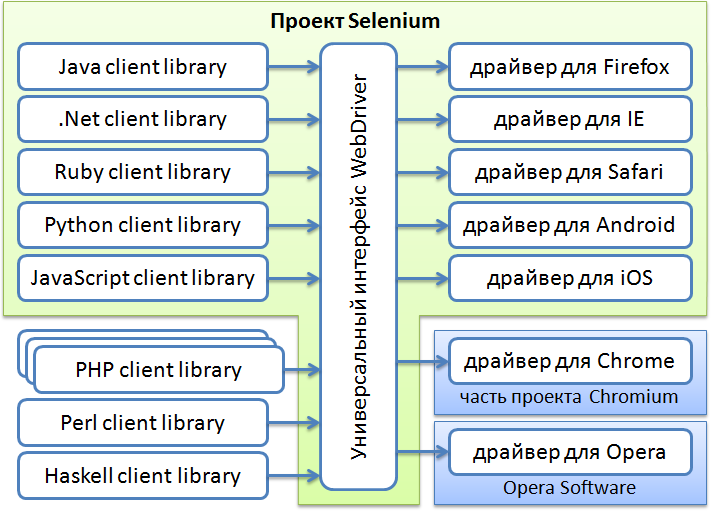


Рисунок 1 – Структура Selenium WebDriver

**Selenium RC** – это предыдущая версия библиотеки для управления браузерами. Аббревиатура RC в названии этого продукта расшифровывается как Remote Control, то есть это средство для «удалённого» управления браузером.  
Эта версия с функциональной точки зрения значительно уступает WebDriver. Сейчас она находится в законсервированном состоянии, не развивается и даже известные баги не исправляются. А всем, кто сталкивается с ограничениями Selenium RC, предлагается переходить на использование WebDriver.

Иногда Selenium RC называется также Selenium 1.0, тогда как WebDriver называется Selenium 2.0. Хотя на самом деле дистрибутив версии 2.0 включает в себя одновременно обе реализации – и Selenium RC, и WebDriver. А вот когда выйдет версия 3.0 – в ней останется только WebDriver.

С технической точки зрения WebDriver не является результатом эволюционного развития Selenium RC, они построены на совершенно разных принципах и у них практически нет общего кода. Объединяет их лишь тот факт, что обе реализации были сделаны в рамках проекта Selenium. Ну, или если быть совсем точным, WebDriver сначала был самостоятельным проектом, но в 2008 году произошло слияние и сейчас WebDriver представляет собой основной вектор развития проекта Selenium.

**Selenium Server –** это сервер, который позволяет управлять браузером с удалённой машины, по сети. Сначала на той машине, где должен работать браузер, устанавливается и запускается сервер. Затем на другой машине (технически можно и на той же самой, конечно) запускается программа, которая, используя специальный драйвер RemoteWebDriver, соединяется с сервером и отправляет ему команды. Он в свою очередь запускает браузер и выполняет в нём эти команды, используя драйвер, соответствующий этому браузеру.

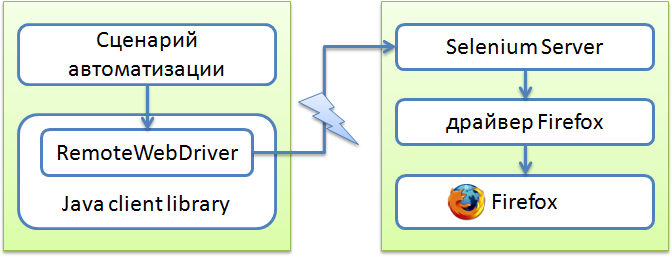


Рисунок 2 – Логика работы Selenium Server

Selenium Server поддерживает одновременно два набора команд – для новой версии (WebDriver) и для старой версии (Selenium RC).

**Selenium Grid –** это кластер, состоящий из нескольких Selenium-серверов. Он предназначен для организации распределённой сети, позволяющей параллельно запускать много браузеров на большом количестве машин. Selenium Grid имеет топологию «звезда», то есть в его составе имеется выделенный сервер, который носит название «хаб» или «коммутатор», а остальные сервера называются «ноды» или «узлы». Сеть может быть гетерогенной, то есть коммутатор и узлы могут работать под управлением разных операционных систем, на них могут быть установлены разные браузеры. Одна из задач Selenium Grid заключается в том, чтобы «подбирать» подходящий узел, когда во время старта браузера указываются требования к нему – тип браузера, версия, операционная система, архитектура процессора и ряд других атрибутов.

Ранее Selenium Grid был самостоятельным продуктом. Сейчас физически продукт один – Selenium Server, но у него есть несколько режимов запуска: он может работать как самостоятельный сервер, как коммутатор кластера, либо как узел кластера, это определяется параметрами запуска.

**Selenium IDE –** плагин к браузеру Firefox, который может записывать действия пользователя, воспроизводить их, а также генерировать код для WebDriver или Selenium RC, в котором выполняются те же самые действия. В общем, это «Selenium-рекордер».

Тестировщики, которые не умеют (или не хотят) программировать, используют Selenium IDE как самостоятельный продукт, без преобразования записанных сценариев в программный код. Это, конечно, не позволяет разрабатывать достаточно сложные тестовые наборы, но некоторым хватает и простых линейных сценариев. [1]

## Puppeteer

Библиотека Puppeteer (Node.js библиотека) предоставляет высокоуровневый API для управления браузерами на основе Chromium, включая Microsoft Edge, с помощью протокола DevTools (Chrome DevTools Protocol — это протокол для автоматизации Chromium). [4, 5] Разработчиком данной библиотеки является Google, а сам проект стартовал в 2012 году. [6]

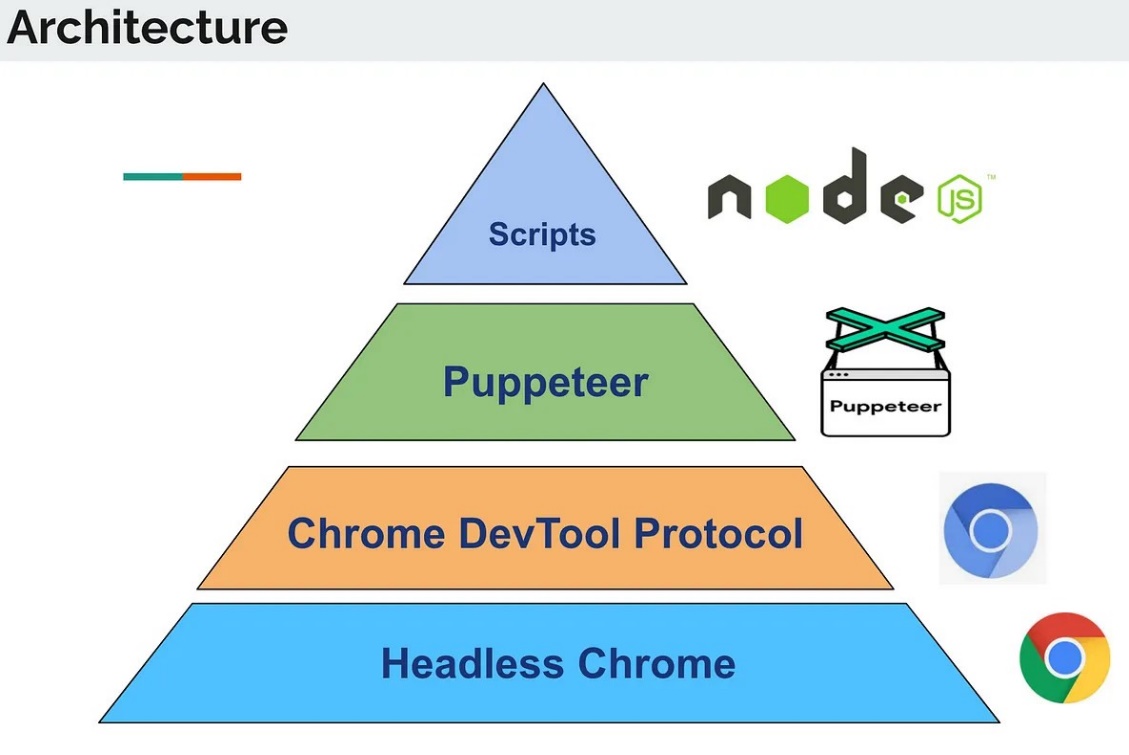


Рисунок 3 – Архитектура работы Puppeteer [7]

Puppeteer запускает браузеры в безголовом режиме (headless) по умолчанию. Такие браузеры не отображают пользовательский интерфейс. Также есть возможность настроить Puppeteer для запуска полного браузера, аналогичного тому, который мы запускаем каждый день.

## Cypress

Cypress – это еще один инструмент (JavaScript инструмент [8]) для тестирования фронтэнда веб-приложений. [9] Большинство инструментов (например Selenium) для автотестирования работают вне браузера и посылают удаленные команды браузеру. [10, 11] В свою очередь, Cypress имеет иной подход: при работе мы видим на экране браузер. Он не обязательно открыт, он может быть headless, но он есть, и он открывает особое приложение самого Cypress, которое состоит из нескольких фреймов: в одном фрейме открывается продукт, который мы тестируем, в другом фрейме запускаются тесты.

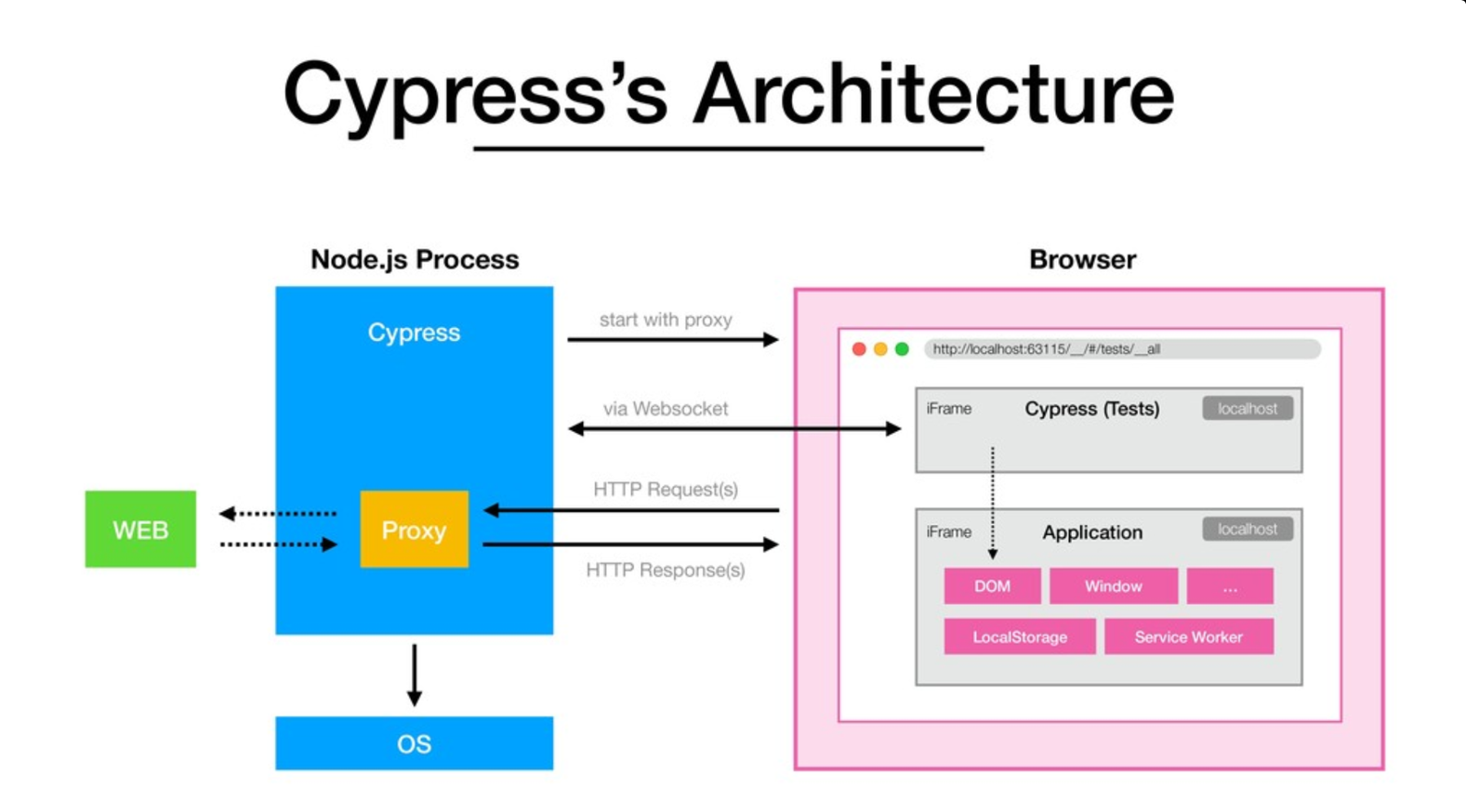


Рисунок 4 – Архитектура работы Cypress

Код тестов пишется на JavaScript, поэтому они могут выполняться непосредственно в браузере — ведь это нативный для него язык. Благодаря такой архитектуре, можно сказать, что Cypress имеет доступ не только к фронтэенду, но и к бэкенду веб-приложения. Таким образом, этот инструмент имеет возможность действовать на сетевом уровне, отслеживая и подменяя веб-траффик.

## Playwright

Playwright – написанная Microsoft (2020 год [6]) Node.js библиотека с одним API, автоматизирующим работу сразу с тремя самыми популярными движками Chromium (браузеры Chrome, Opera, Edge), Gecko (браузер Firefox) и Webkit (браузер Safari). [12, 13].

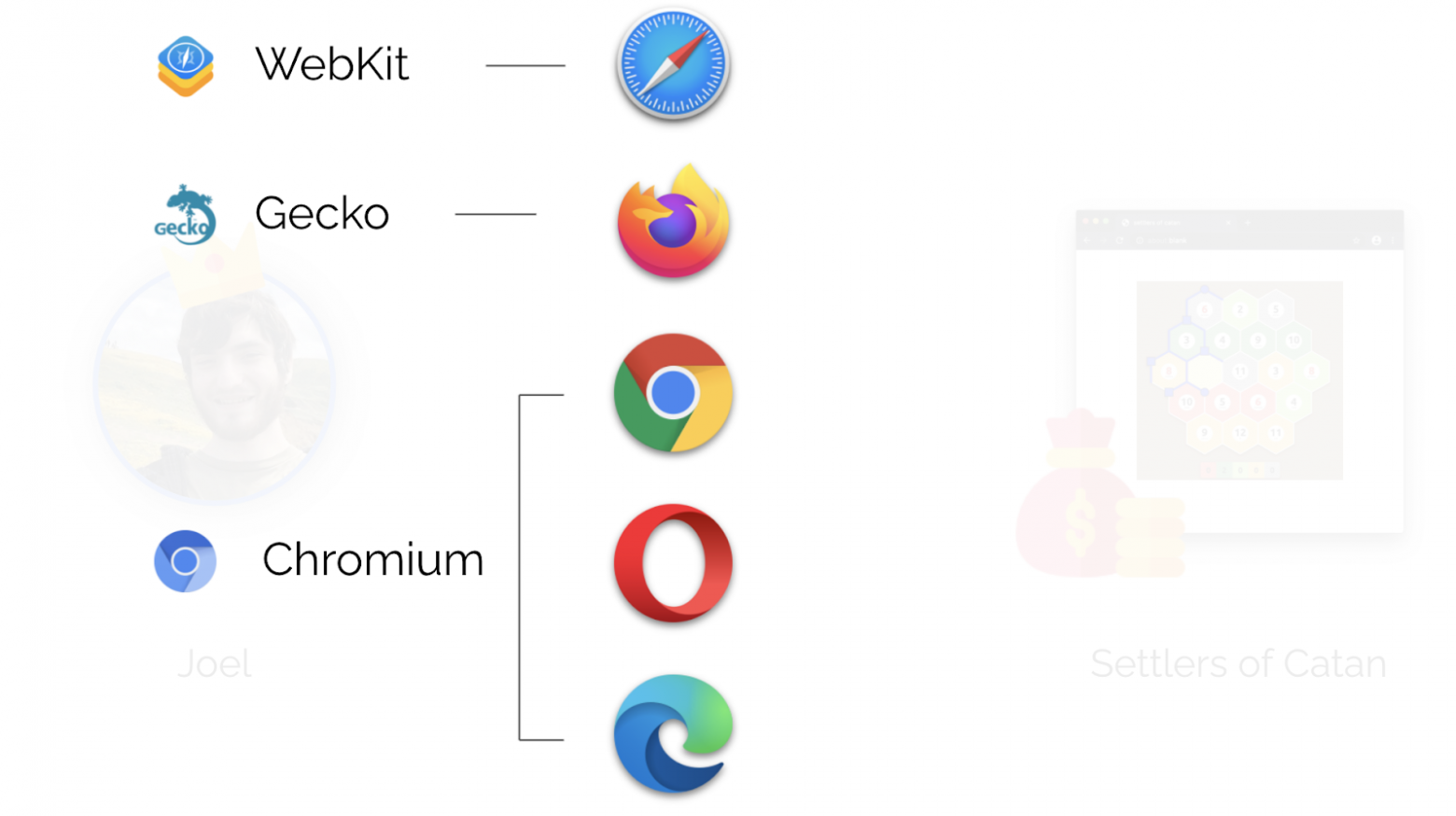


Рисунок 5 – Браузеры и движки на которых они работают

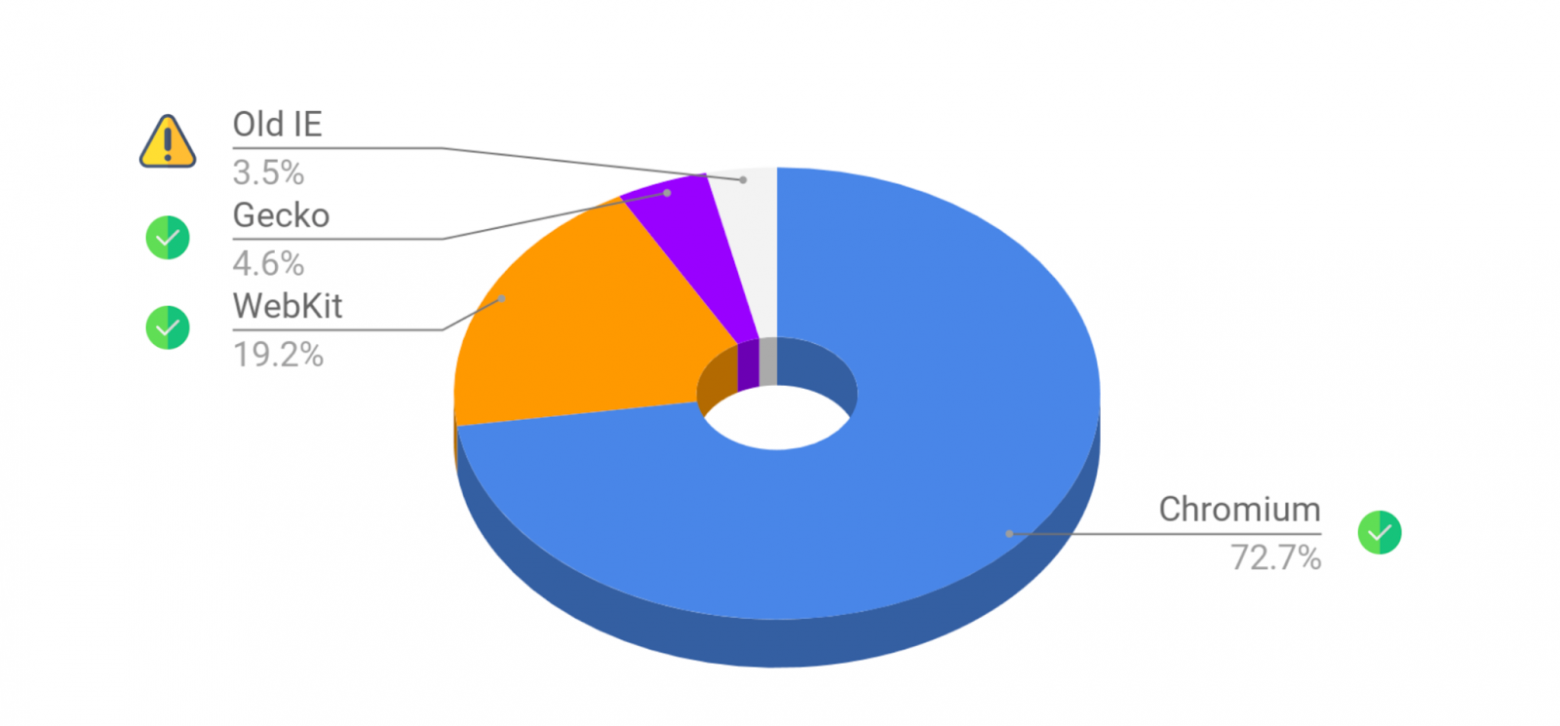


Рисунок 6 – Распространенность движков в интернете

Playwright – большая инициатива, которая объединяет несколько направлений.

Первое направление — сборка и дистрибуция WebKit на трех платформах (Windows, Linux, Mac). Возможно скачать этот браузер к себе на Windows и поиграть с ним, «посерфить» — всё это работает и в headless-режиме тоже. Поэтому окна могут не появляться, и можно спокойно запускать тесты на WebKit в фоне.

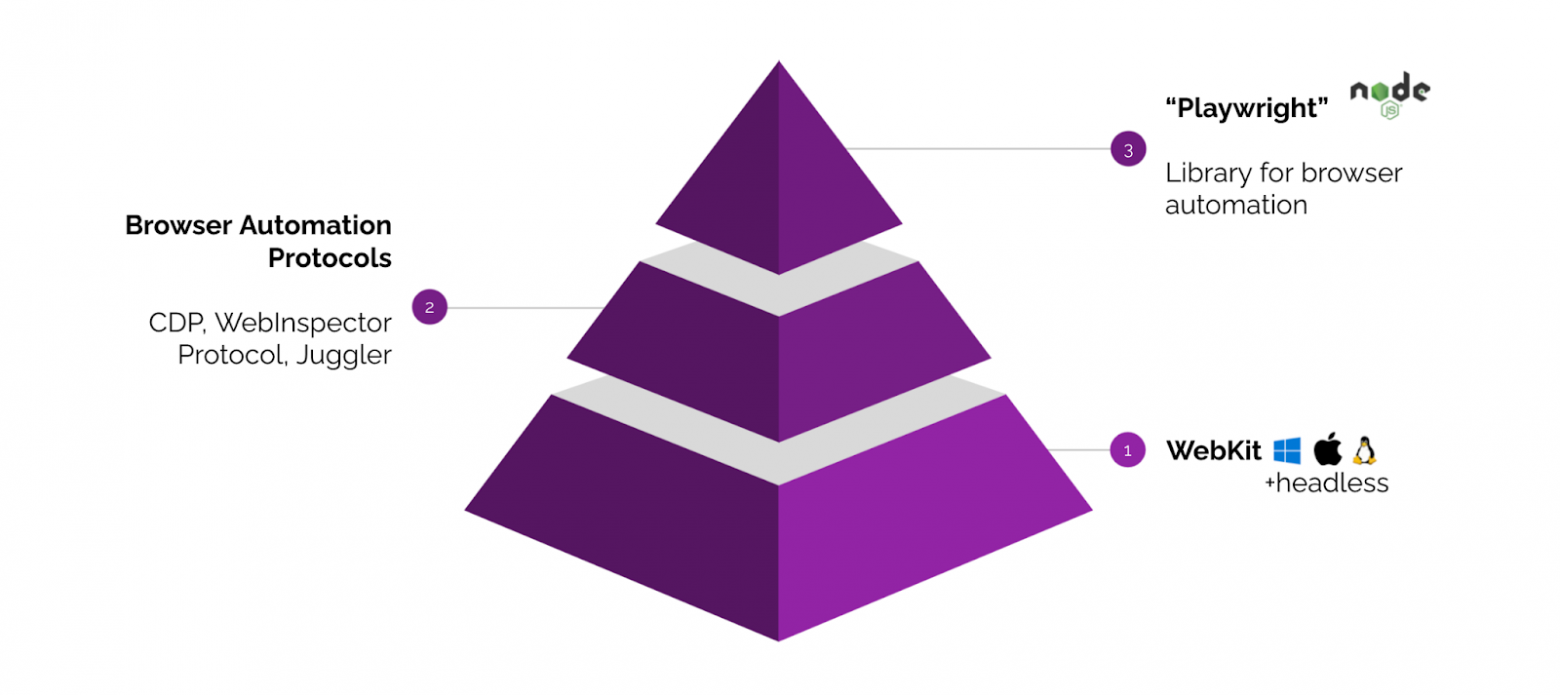


Рисунок 7 – Направления в рамках Playwright инструментария

Второе направление — это браузерные протоколы автоматизации. Выше уже упоминался Chrome DevTools Protocol (это протокол для автоматизации Chromium). В Safari есть похожий протокол по идеологии — WebInspector Protocol. В Firefox используется протокол Juggler — это аддон, который реализует похожую вещь на Chrome DevTools Protocol, однако в терминах движка Gecko, чтобы он был естественным для браузерных разработчиков.

Третье направление – библиотека Playwright. Она умеет разговаривать с каждым из трех протоколов, собирает их и предоставляет один удобный API, чтобы можно было управлять тремя разными браузерами.

Также в 2022 году появился собственный тестраннер, ставший важным направлением развития.

Стоит отметить, что тесты, написанные с помощью Playwright, выполняются в изолированной среде с чистого листа, называемой контекстами браузера.

## Сравнение и выводы

Ниже приведены сравнительные таблицы инструментов, описанных выше. [6]

Таблица 1 – Сравнение Selenium/Puppeteer/Cypress/Playwright по основным параметрам

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Selenium** | **Puppeteer** | **Cypress** | **Playwright** |
| Разработчик |  | Google |  | Microsoft |
| Старт проекта | 2004 год | 2012 год | 2015 год | 2020 год |
| Поддерживаемые браузеры | Chrome, Firefox, Edge, Internet Explorer, Opera, Safari | Браузеры семейста Chromium (Chrome, Opera, Edge) | Браузеры семейста Chromium (Chrome, Opera, Edge), Firefox | Браузеры семейста Chromium (Chrome, Opera, Edge), Firefox, WebKit |
| Код | Весь код открытый | Весь код открытый | Весь код открытый | Весь код открытый |
| Язык программирования | Java, Python, C#, Ruby, JavaScript и другие | Java, Python, C#, JavaScript, TypeScript и другие | JavaScript, TypeScript | Java, Python, C#, JavaScript, TypeScript |
|
|
|
|
|
| Явное сетевое управление | нет | нет | Представляет общеконтекстный перехват сети для заглушки и имитации сетевых запросов | Представляет общеконтекстный перехват сети для заглушки и имитации сетевых запросов |
| Безголовый режим (headless) для браузеров | Не все драйверы поддерживают безголовый режим | Поддерживает безголовый режим | Поддерживает безголовый режим | Поддерживает безголовый режим |
| OS | Нужны Mac для Safari и Windows для IE (Edge) | Работает на 3-х платформах (Windows, Mac, Linux) | Работает на 3-х платформах (Windows, Mac, Linux) | Работает на 3-х платформах (Windows, Mac, Linux) |

Таблица 2 – Сравнение Selenium/Puppeteer/Cypress/Playwright по принципу работы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Selenium** | **Puppeteer** | **Cypress** | **Playwright** |
| Принцип работы | В Selenium WebDriver есть три процесса: 1.Selenium WebDriver; 2.Драйвер браузера, такой как ChromeDriver, GeckoDriver (для Firefox), и т.п.; 3.Cам браузер. Все связи между этими процессами означают, что тесты Selenium занимают много времени. | Puppeteer запускает headless-браузер и использует DevTools протокол, поэтому тесты проходят быстрее и стабильнее по сравнению с Selenium. | Код, который используется в тестовых сценариях Cypress, не запускается вне браузера, как в WebDriver. Его запускает браузер.  Фактически, он выполняет тестовый код и код тестируемого приложения. | Тесты, написанные с помощью Playwright, выполняются в изолированной среде с чистого листа, называемой контекстами браузера. |
|
|
|
|

Таблица 3 – Сравнение Selenium/Puppeteer/Cypress/Playwright по скорости работы [14]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Selenium** | **Puppeteer** | **Cypress** | **Playwright** |
| Среднее время выполнения (в секундах) | 3.66 | 2.22 | 10.35 | 3.19 |
| Стандартное отклонение (в секундах) | 0.65 | 0.41 | 2.67 | 1.23 |

Если необходимо тестировать веб-приложение на Internet Explorer или на подлинном Safari, подойдет только **Selenium** - он является проверенным временем продуктом.

Если важна кроссбраузерность (поддержка нескольких браузеров), и поддержка Internet Explorer необязательна (он перестает поддерживаться в 2021-2022 году), то подходит [**Playwright**](https://playwright.dev/docs/). Это достаточно молодой продукт, но быстро развивающийся.

Если кроссбраузерность не важна и можно запускать тесты на одном браузере, тогда хорошо подойдет **Puppeteer**. Он самый быстрый, а также Сhromium, на данный момент времени, самый стабильный движок.

Если важно делать Network Mocking (работа на сетевом уровне, имитация сетевых запросов) в тестах, то можно использовать [**Cypress**](https://www.cypress.io/), там больше готовых сценариев чем на [Playwright](https://playwright.dev/docs/).

Для написания автотестов в ходе дипломной работы будет использоваться инструмент Playwright, т.к. кроссбраузерность крайне важна, а использование одного API для взаимодействия с разными движками является крайне удобным. Также Playwright является самым молодым и бурно развивающимся продуктом и его использование позволит «быть на острие» технологий.

# ВЫДЕЛЕНИЕ БАЗОВЫХ СЦЕНАРИЕВ ИСПОЛЬЗОВАНИЯ ВЕБ-ПРИЛОЖЕНИЯ LCWAIKIKI.KZ

## Описание функционала веб-приложения lcwaikiki.kz

Lcwaikiki.kz является типичным примером стандартного интернет-магазина одежды, обуви и аксессуаров. Интернет-портал предоставляет возможность создания личного кабинета (или же войти в уже существующий профиль), ведения списка понравившихся товаров, а также корзины. Предоставляется удобный поиск по каталогу, навигация по которому осуществелена с помощью строки поиска или перехода на соответвующую категорию той или иной продукции. Также большую часть начальной страницы веб-приложения занимает реклама и описание промоакций/распродаж. Также стоит отметить, что поддерживается разнообразная локализация, т.е. отличное отображение для различных стран. Отдельным пунктом вынесена возможность просмотра статуса заказа. Также присутсвуют ссылки на отдельные мобильные приложения на Google Play и App Store.

## Описание базовых сценариев использования веб-приложения lcwaikiki.kz

Под базовым сценарием использования подразумевается такой сценарий, который не подвержен частому изменению. Задачей данной дипломной работы является написание автотестов, которые могли бы запускаться после каждого обновления веб-приложения, проверяя, что основной функционал сайта доступен и работает согласно бизнес-требованиям. Редактировать логику тестов с каждой новой версией – накладная активность, именно по этой причине было решено покрыть тестами лишь базовый функционал.

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ И РЕСУРСОВ

1. <https://habr.com/ru/articles/152653/>

1. <https://ru.wikipedia.org/wiki/Selenium>

1. <https://www.selenium.dev/>
2. <https://learn.microsoft.com/ru-ru/microsoft-edge/puppeteer/>
3. <https://pptr.dev/>
4. <https://habr.com/ru/articles/566348/>
5. <https://javascript.plainenglish.io/puppeteer-basics-3be7f9f82a08>
6. <https://habr.com/ru/companies/plesk/articles/557830/>
7. <https://en.wikipedia.org/wiki/Cypress_(software)>
8. <https://www.cypress.io/>
9. <https://www.tutorialspoint.com/cypress-architecture-test-automation>
10. <https://playwright.dev/docs/>
11. <https://habr.com/ru/companies/jugru/articles/652919/>
12. <https://www.checklyhq.com/blog/cypress-vs-selenium-vs-playwright-vs-puppeteer-speed-comparison/>

# ПРИЛОЖЕНИЯ