УДК 004.054
DOI: 10.32626/2308-5916.2023-24.79-94

О. Ю. Тарновецька*, канд. фіз-мат. наук,
Н. І. Бойко,** канд. екон. наук,
Н. С. Пислар*,** Senior test automation,
Л. І. Д'яченко*, канд. тех. наук

*Чернівецький національний університет
імені Юрія Федьковича, м. Чернівці,
**Національний університет «Львівська політехніка», м. Львів,
***Global-E, Israel

МОДЕЛЮВАННЯ ТЕСТОВИХ СЦЕНАРІЇВ ДЛЯ ДОСЛІДЖЕННЯ ВІДМІННОСТЕЙ МІЖ БРАУЗЕРАМИ CHROME ТА HEADLESS CHROME

Введення автоматизації тестування має численні переваги в сучасному світі інформаційних технологій. До яких можна віднести скорочення часу тестування, спрощення процесу формування звітності та постійне покращення ефективності.

Застосування автоматизації тестування відкрило можливість ретельного аналізу відмінностей у роботі інтерфейсів браузерів Chrome та Headless Chrome. Виявлено, що продуктивність Headless браузера перевершує продуктивність його аналога з графічним інтерфейсом на 10,3%. Важливо враховувати, що хоча Headless Chrome є досить ефективним, він не завжди є універсальним для різних видів тестів.

У рамках дослідження обгрунтовано, що використання Selenium WebDriver для автоматизованого тестування надає потужні можливості для виконання як рутинних, так і складних тестових завдань, які важко виконати вручну. Вибір цього інструментарію зумовлений його багатофункціональністю, високою практичністю та сумісністю з різними мовами програмування.

Проведено докладний аналіз характеристик та атрибутів тестового веб-додатка, згідно якого розроблено модель функціонування програмного забезпечення для автоматизації тестових сценаріїв для проекту «trello.com» охоплює веб-додатки, які піддаються тестуванню через інтерфейс користувача. Отримані результати дозволили дослідити швидкість роботи Google Chrome і Headless Chrome та встановити їх переваги та недоліки. Зокрема, недоліки Headless браузера полягають у відсутності можливості використання певних функцій, які реалізовані в браузерах з графічним інтерфейсом, наприклад, випадаючі меню, що може призвести до збоїв у тестах. До недоліків Real Browser можна віднести високе споживання ресурсів, залежність від середовища, нестабільність та складність налаштувань.

© О. Ю. Тарновецька, Н. І. Бойко, Н. С. Пислар, Л. І. Д'яченко, 2023

Отже, вибір конкретної техніки тестування повинен залежати від конкретних вимог проекту, і для отримання оптимальних результатів може бути використана комбінація тестування Headless та Real Browser.

Ключові слова: *автоматизація тестування, швидкість браузерів, тестові сценарії, selenium webdriver, page object model, google chrome, headless chrome.*

Вступ. Автоматизація тестування – це процес виконання тестування комп'ютером за наперед описаним алгоритмом [1]. Вона покликана, в першу чергу, для того, щоб зекономити найважливіший ресурс в житті людей – час.

Автоматизація тестування може і, як правило, виконується на існуючих браузерах, що відрізняються своєю швидкодією [2], продуктивністю та іншими характеристиками, проте виконання тестів навіть на браузерах з найвищою швидкодією займає багато часу. Headless браузер – це веб-браузер без графічного інтерфейсу користувача, за допомогою якого можна забезпечити автоматичний контроль веб-сторінки в середовищі, подібному до популярних веб-переглядачів, але виконання відбувається через інтерфейс командного рядка.

Браузери такого типу особливо корисні для тестування вебзастосунків, оскільки вони можуть виконувати і розуміти HTML так само, як і браузер, включно з CSS, JavaScript і AJAX, які, зазвичай, недоступні при використанні інших методів тестування.

У них немає реального промальовування вмісту, тобто вони все малює в пам'яті, тому споживає менше пам'яті, у зв'язку з цим Headless-браузер працює швидше. Ще однією важливою властивістю є можливість його встановлення на «голий» Linux-сервер, тобто на операційну систему Ubuntu або Red Hat можна просто вставити бінарний файл або поставити пакет і браузер буде працювати з коробки.

При тестуванні цікавість представляє верхній компонент Browser UI. Це той самий інтерфейс користувача – вікна, меню, спливаючі повідомлення та все інше.

Такий вигляд має headless браузер. Ми повністю прибираємо інтерфейс користувача. Залишається лише компонент ядра.

Headless браузер – це тип програмного забезпечення, яке може отримати доступ до веб-сторінок, але не показує їх користувачеві, і може передавати вміст веб-сторінок до іншої програми. На відміну від звичайного браузера, при запуску headless браузера на екрані нічого не з'являтиметься, оскільки програми запускаються на сервері.

Як і звичайний браузер, headless браузер може аналізувати та інтерпретувати веб-сторінки (хоча поведінка може відрізнятися в різних реалізаціях браузера), тому він може надавати реальний контекст

браузера без будь-яких витрат на пам'ять та швидкість, на відміну від роботи повноцінного браузера з графічним інтерфейсом користувача. Headless браузери можуть бути не дуже корисними для серфінгу в Інтернеті, але вони є чудовим інструментом для тестування.

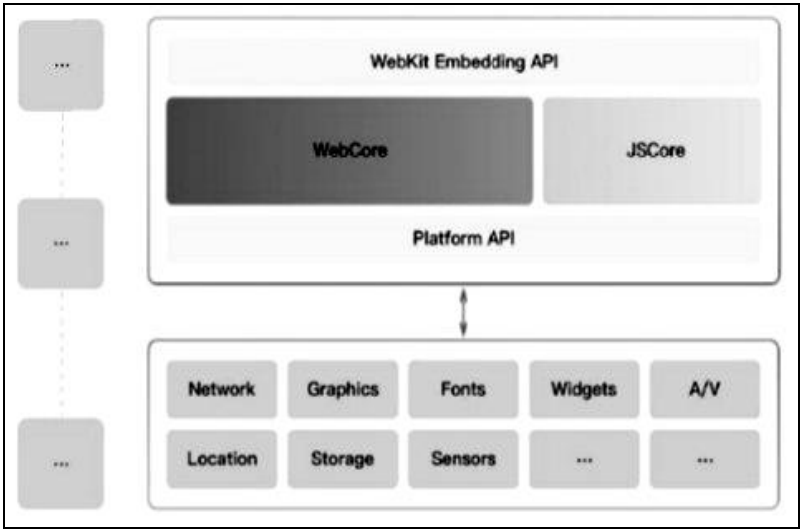

Рис. 1. *Схематичний вигляд headless браузера*

Тестування з використанням headless браузера має ряд своїх переваг з боку автоматизації, тестування макета та продуктивності. Також за допомогою headless тестування браузера можна запускати тести на машинах без графічного інтерфейсу користувача, створювати скріншоти і PDF-файли веб-сторінок, моніторити продуктивність мережевих додатків, знімати часову шкалу веб-сайту для діагностики продуктивності, симулювати кілька браузерів на одній машині без використання ресурсів.

Не зважаючи на всі переваги, які можна отримати за допомогою headless реалізації, насправді неможливо використовувати її для всіх потреб тестування і все ж більшу частину часу доведеться проводити тестування в реальних браузерах. Зрештою, кінцеві користувачі не використовують headless браузер, і тому не має потреби зосереджуватися на помилках, які трапляються лише в headless режимі. Щоб перевірити досвід користувачів і провести функціональний тест, потрібно імітувати реального користувача та отримати точний відгук про те, що їм насправді відображається на веб-сайті.

Об’єктом дослідження роботи є різниця швидкості робіт браузерів Real Chrome та Headless Chrome в автоматизації тестування.

Предметом дослідження є результат автоматизованих тестів сайту trello.com.

Метою роботи є дослідження відмінності робіт браузерів Real Chrome та Headless Chrome і, як приклад використання браузерів, застосування Selenium WebDriver для розробки автоматизованої системи, яка проводить тестування.

Огляд програмних засобів та методологій для автоматизованого тестування. Розглянемо та проаналізуємо програмні засоби, обрані для автоматизованого тестування, а саме: Selenium Webdriver та NUnit.

Selenium WebDriver – це веб-фреймворк, який дозволяє виконувати міжбраузерні тести[1]. Цей інструмент використовується для автоматизації тестування веб-додатків, щоб переконатися, що вони працюють належним чином. Selenium WebDriver дозволяє вибрати мову програмування для створення тестових скриптів, проте не здатний обробляти компоненти вікна, але цей недолік можна подолати за допомогою таких інструментів, як Sikuli, Auto IT тощо.

Архітектура Selenium WebDriver (рис. 2) складається з чотирьох основних компонентів: Клієнтська бібліотека Selenium, Протокол JSON через HTTP, Драйвери браузера, Браузери.

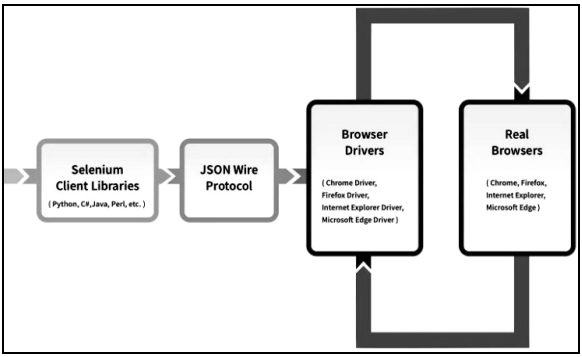

Рис. 2. *Архітектура Selenium WebDriver*

Selenium забезпечує підтримку кількох бібліотек, таких як Ruby, Python, Java тощо, прив'язки яких потрібно завантажити з офіційного сайту Selenium [3].

JSON – це абревіатура від JavaScript Object Notation. Це відкритий стандарт, який забезпечує транспортний механізм для передачі даних між клієнтом і сервером в мережі Інтернет та забезпечує підтримку різних структур даних, таких як масиви та об'єкти, що значно полегшує читання та запис даних із JSON. JSON служить REST (Representational State Transfer) API, який обмінюється інформацією між серверами HTTP.

Selenium надає драйвери, специфічні для кожного браузера, і, не розкриваючи внутрішньої логіки функціональності браузера, драйвер браузера взаємодіє з відповідним браузером, встановлюючи безпечне з'єднання. Ці драйвери браузера також специфічні для мови, яка використовується для автоматизації тестових випадків, як-от C#, Python, Java тощо.

Коли тестовий сценарій виконується за допомогою WebDriver, у фоновому режимі виконуються такі завдання:

1. Генерується HTTP-запит, який доставляється драйверу браузера для кожної команди Selenium
2. Запит HTTP отримує драйвер через HTTP-сервер
3. Усі кроки/інструкції, які мають виконуватися в браузері, вирішує сервер HTTP
4. Потім сервер HTTP отримує статус виконання і, у свою чергу, надсилає його назад сценаріям автоматизації.

Переваги та недоліки Selenium WebDriver. Selenium WebDriver – це один з найпопулярніших інструментів з відкритим вихідним кодом, з яким легко почати тестування веб-додатків. Це також дозволяє виконувати перевірку на сумісність між браузерами.

До переваг можна віднести те, що Selenium WebDriver:

1. Підтримує кілька операційних систем, таких як Windows, Mac, Linux, Unix тощо.
2. Забезпечує сумісність з низкою мов, включаючи Python, Java, Perl, Ruby тощо.
3. Забезпечує підтримку сучасних браузерів, таких як Chrome, Firefox, Opera, Safari та Internet Explorer.
4. Selenium WebDriver завершує виконання тестових скриптів швидше в порівнянні з іншими інструментами.
5. Більш стислий API (інтерфейс прикладного програмування), ніж Selenium RC.
6. Забезпечує сумісність з iPhoneDriver, HtmlUnitDriver і AndroidDriver.

Недоліки WebDriver:

1. Підтримка нових браузерів недоступна в порівнянні з Selenium RC
2. Для автоматичного створення результатів тесту він не має вбудованого функціоналу.

В загальному Selenium WebDriver працює в три кроки:

1. Тестові команди перетворюються на HTTP-запит за допомогою дротового протоколу JSON.
2. Перед виконанням будь-яких тест кейсів кожен браузер має власний драйвер, який ініціалізує сервер.

3. Потім браузер починає отримувати запит через свій драйвер.

Розглянемо приклад із фрагментом коду нижче:

```
WebDriver driver = new ChromeDriver ();
driver.Navigate().GoToUrl("google.com");
```

Наведений вище код призведе до запуску браузера Chrome, який перейде на веб-сайт google.com.

Після виконання програми кожен рядок коду/скрипту перетворюється на URL-адресу. Протокол JSON Wire через HTTP робить це можливим. Потім ця URL-адреса передається до драйверів браузера. На цьому етапі наша клієнтська бібліотека перекладає код у формат JSON і взаємодіє з ChromeDriver.

Для отримання HTTP-запитів кожен драйвер браузера використовує HTTP-сервер. Як тільки драйвер браузера отримує URL-адресу, він обробляє запит, передаючи його справжньому браузеру через HTTP. І тоді всі команди в скриптах Selenium будуть виконані.

Є два типи запитів – GET і POST. Запит GET призводить до відповіді, яка буде згенерована в кінці браузера, і вона буде надіслана через HTTP до драйвера браузера, і, врешті-решт, драйвер браузера за допомогою дротового протоколу JSON надсилає його в інтерфейс користувача. Запит POST створює певну сутність або надсилає інформацію з клієнта на сервер.

Перехресне браузерне тестування – це процес тестування веб-сайту або веб-програми в різних комбінаціях браузерів і версій браузера. Частка ринку браузерів для настільних комп'ютерів є чітким показником фрагментації, яка існує на ринку настільних браузерів.

Тестування на сумісність між браузерами – це підхід, за допомогою якого ви можете перевірити свій веб-сайт (або веб-додаток) у різних браузерах, версіях браузера та операційних системах. Це важливо для забезпечення однакового використання продукту для всіх браузерів і платформ (або операційних систем).

Варто зазначити, що Selenium – це популярний фреймворк з відкритим кодом, який використовується для міжбраузерного тестування або автоматизованого тестування веб-сайтів (або веб-додатків). Фреймворк Selenium забезпечує мовні прив'язки для популярних мов програмування, таких як Python, Java, JavaScript, C#, Ruby і PHP.

Загалом, Selenium – це потужна платформа автоматизації тестування, яка не тільки важлива в тестуванні веб-автоматизації, але й допомагає прискорити загальний випуск програмного забезпечення за допомогою інтеграції з популярними інструментами CI/CD, такими як Jenkins, TeamCity, GitLab CI тощо. Єдиним недоліком Selenium є те, що його не можна використовувати для автоматизації десктопних

програм. Платформа Selenium розроблена таким чином, щоб оновлення (або покращення) у браузерах мали мінімальний вплив на сценарії тестування. Взаємодія з WebElements на сторінці, яка відображається у браузері, полегшується за допомогою драйверів браузера.

Кожен веб-продукт складається з інтерфейсу та бекенду. HTML, CSS і JavaScript є одними з найбільш широко використовуваних веб-мов для розробки інтерфейсу. З іншого боку, серверну частину можна розробити за допомогою популярних програм PHP, Python, Java та інших [5, 6].

Крім того, кожен веб-браузер використовує різний механізм візуалізації та механізм JavaScript, що може призвести до різного досвіду в різних веб-переглядачах. Наприклад, Google Chrome використовує механізм Blink, Mozilla Firefox використовує механізм Gecko, Apple Safari використовує механізм WebKit тощо.

Роль механізмів візуалізації у браузерах зіграла важливу роль у важливості кроссбраузерного тестування веб-сайтів. Перехресне тестування потрібне, щоб переконатися, що відвідувачам вашого вебсайту надається однаковий досвід, незалежно від браузера, версії браузера чи ОС, що використовується для доступу до них.

Покриття браузера є невід'ємним аспектом тестового покриття. Основним завданням кроссбраузерного тестування є проведення тестування в різних браузерах і комбінаціях версій[7].

Використання Selenium для кроссбраузерного тестування допомагає в ретельному тестуванні коду [8] в кількох браузерах; тим самим допомагаючи досягти кращого покриття браузера. Це, у свою чергу, допомагає покращити роботу користувачів у цільових веб-переглядачах. Матриця сумісності браузера може сприяти визначенню пріоритетів веббраузерів, які використовуються цільовою аудиторією.

NUnit – це платформа модульного тестування [9] для всіх мов .Net, перевагами якої є:

1. Можливість запуску тестів з консолі, у Visual Studio через тестовий адаптер або за допомогою сторонніх програм.
2. Можливість виконувати тести паралельно.
3. Потужна підтримка тестів на основі тестових даних.
4. Підтримка кількох платформ, включаючи .NET Core, Xamarin Mobile, Compact Framework і Silverlight.
5. Можливість створення категорій тестів, щоб забезпечити вибірковий запуск.
6. Можливість консольного запуску (nunit3-console.exe), який використовується для пакетного виконання тестів. Console runner працює через NUnit Test Engine, який надає йому можливість завантажувати, досліджувати та виконувати тести. Коли тести мають

виконуватися в окремому процесі, двигун використовує для їх виконання програму nunit-agent.

NUnit використовує атрибути (рис. 3) для роботи в тестових класах і методах, основними є:

- TestFixture – Позначає клас, який містить тестові методи;
- Test – Позначає метод у тестовому класі. Цей метод перетворюється у тест і починає відображатись у тест провіднику;
- SetUp – Зазначає, що метод помічений атрибутом виконається безпосередньо перед кожним тестом;
- TearDown – Зазначає, що метод помічений цим атрибут буде виконуватись після кожного тестового методу;
- Repeat – Зазначає, що метод має бути виконаний декілька разів;
- Order – Зазначає, що метод має бути виконаний у відповідному порядку.

```
namespace Tests
{
    [TestFixture]
    Ссылок: 0
    public class SmokeTests : BaseTest
    {
        [Test]
        Ссылок: 0
        public void NegativeLogin()
        {
            InitPage init = new InitPage(driver);
            init.ClickLogin();
            LoginPage login = new LoginPage(driver);
            Wait(5);
            login.EnterUsername("hello");
            login.EnterPassword("world");
            login.ClickLogin();
            Wait(5);
            var errorMessage = login.GetErrorMessage();
            Assert.AreEqual("Не існує акаунта, пов'язаного з цим ім'ям користувача", errorM
        }
```

Рис. 3. *Приклад тесту написаного за допомогою технології NUnit*

Технологію NUnit було використано для створення якісних автотестів та конфігурації автоматизованого фреймворку. NUnit має різні конфігураційні можливості запуску тест кейсів в різних режимах (паралельно, послідовно).

Для UI тестування було використано Selenium та Selenium Web Driver – популярний фреймворк для тестування веб-додатків. Його основною перевагою є кроссбраузерне тестування (тестування одного й того самого функціоналу на різних браузерах).

З рис. 3 видно приклад використання патерну «Page Object Model» – патерну проектування для створення класів для кожного елементу.

Відповідно до цього патерну, кожна сторінка додатку або сайту повинна мати власний клас, що їй відповідає [2].

Кожен клас-сторінка складається з наступних регіонів:

- Конструктор без параметрів.
- Конструктор, що приймає драйвер.
- Строкові змінні, що описують XPath елементу (опціонально).
- Веб-елементи, що описують елементи з веб сторінки, яку автоматизуємо.
- Методи, що виконують певні дії над веб-елементами (за потреби).

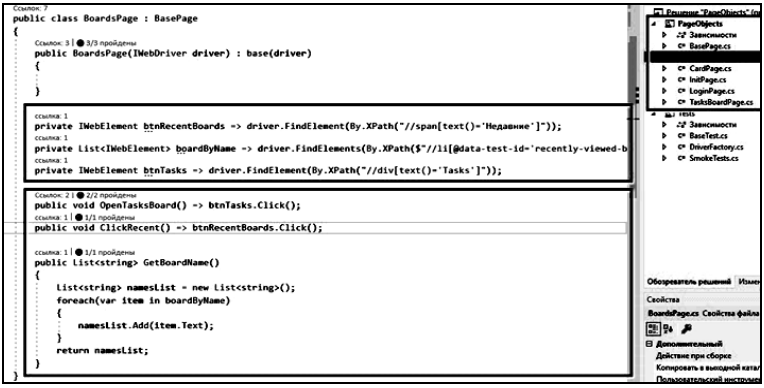

Рис. 4. *Приклад патерну «Page Object Model»*

У кожного з таких методів, назва повинна показувати, яку дію цей метод виконує.

За принципом Page Object Model, всі сторінки є нащадками однієї – базової. Зазвичай, вона містить загальні для всіх методи або обгортки для методів. Прикладами таких методів в моєму проекті є наступні:

- NavigateByUrl – метод, що відкриває посилання, передане в параметри.
- Wait – метод, що чекає певну кількість секунд, передану в параметри.
- RefreshPage – метод, що оновлює сторінку браузера.

Дослідження швидкості Headless браузера на прикладі проекту для автоматизації. Для дослідження швидкості роботи headless браузера було написано 5 end-to-end тест кейсів, тобто тестових сценаріїв, що повністю імітують дії користувача. Як приклад сайту, що тестується, використано trello.com. В якості порівняння другим браузером було обрано Google Chrome – найпопулярніший, станом на 2023 рік, веб-браузер в світі, що використовує користувацький інтерфейс.

Для точності експерименту, тестування було проведене тричі на кожному браузері. Результат виконання тестів подано у таблиці 1. Всі числові значення в таблиці зазначені в секундах.

Таблиця 1

Результат запуску тестів тричі на кожному браузері

	Chrome			Headless		
	1 спроба	**2 спроба**	**3 спроба**	**1 спроба**	**2 спроба**	**3 спроба**
Negative Login	15,1	16,1	16,6	13,9	13,8	13,6
Positive Login	12,1	13,4	13,9	11,4	11,7	12,6
ViewAllBoards	23,8	25	25,1	12,4	17	17,6
FillCardWithData	31,3	32,9	31,5	34,4	32,7	30,6
AddCardToBoard	30,1	30,5	29,7	33	30,1	29,9
Total	112,4	117,9	116,8	105,1	105,3	104,3

Колір в таблиці 1 означає успішність проведення тестування. Клітинки, виділені зеленим кольором позначають успішні тести. Червоні клітинки позначають невдалі тести. Даний приклад демонструє, що є тести, які не працюють на headless браузері і вимагають користувацький інтерфейс для роботи. Суми результатів всіх тестів у спробі подані у вигляді діаграми (рис. 5)

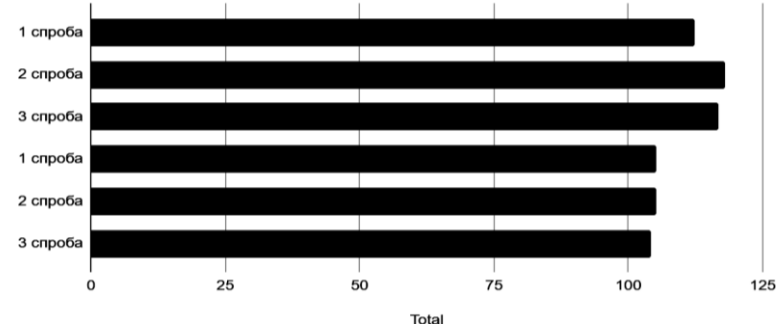

Рис. 5. *Суми результатів всіх тестів в спробах*

Перші три спроби демонструють швидкість роботи браузера з користувацьким інтерфейсом, другі три спроби демонструють швидкість роботи headless браузера. З діаграми видно, що headless браузер займає менше часу на виконання аналогічних задач.

Середні значення для кожної спроби (115.7 секунд для браузера з користувацьким інтерфейсом та 104.9 секунд для headless браузера) показують, що headless браузер швидше звичайного на 10,3%.

Реалізація, призначення та область застосування. Автоматизація – це процес виконання тестування роботом або комп'ютером. Вона допомагає, в першу чергу, зекономити найважливіший ресурс в житті людей – час. Автоматизація тестування може виконуватись на великій кількості браузерів, проте виконання тестів на звичайних браузерах займає багато часу.

Для розробки програмного продукту, що забезпечує адекватність дослідження, використано модульну архітектуру. Структура розробленого фреймворку подано на рис.6.

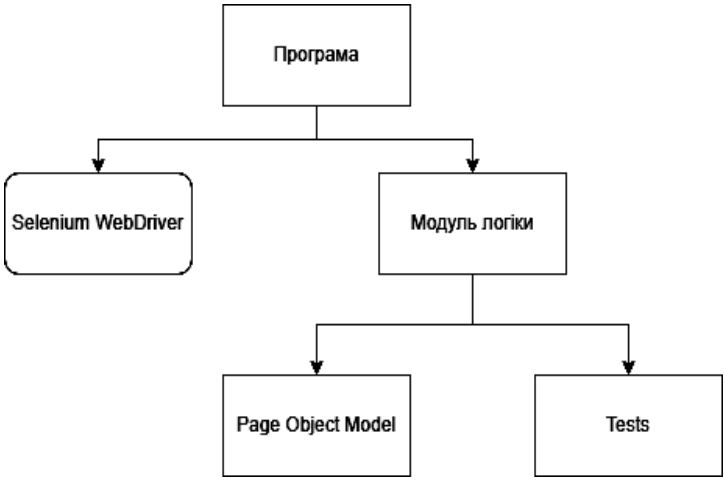

Рис. 6. *Загальна структура фреймворку*

Як видно з рис. 6, програмне забезпечення спроектовано на основі компонентів логіки, які використовують модулі NUnit та Selenium WebDriver.

В кожному класі з закінченням Page міститься обов'язкові 3 регіони:

- Конструктори.
- Елементи.
- Методи.

Також в класах можуть бути додаткові регіони:

- XPath.
- Допоміжні методи.

XPath – це один із шляхів, за яким можна знайти елемент на сторінці.

За правилами, XPath називають точно так само, як і елемент, якому він відповідає, і додається закінчення «XPath».

Елементи за правилами повинні мати префікс – залежно від того, який елемент описується.

Нижче наведені основні префікси і їх означення:

- btn – button;
- chk – checkbox;
- rdo – radiobutton;

- opt – option;
- ddl – dropdownlist;
- txt – text field.

Елементи можуть бути знайдені не тільки за допомогою XPath, а ще за наступними типами:

- ID;
- Class;
- Name;
- Css selector;
- XPath;
- Value;
- Text.

У Selenium WebDriver є власні реалізовані методи, що дуже полегшують роботу автоматизатора. Методи, що використовувались в програмі:

- Click() – натискає на елемент;
- SetText() – вписує текст, що передається в параметри;
- Scroll() – рухає колесо миші;
- ExecuteJS() – виконує переданий в параметри код мовою JavaScript;
- Refresh() – оновлює сторінку;
- Dispose() – закриває браузер.

Патерн POM і взаємодія класів-сторінок відповідно до модулів даної програми зображена на рис. 7.

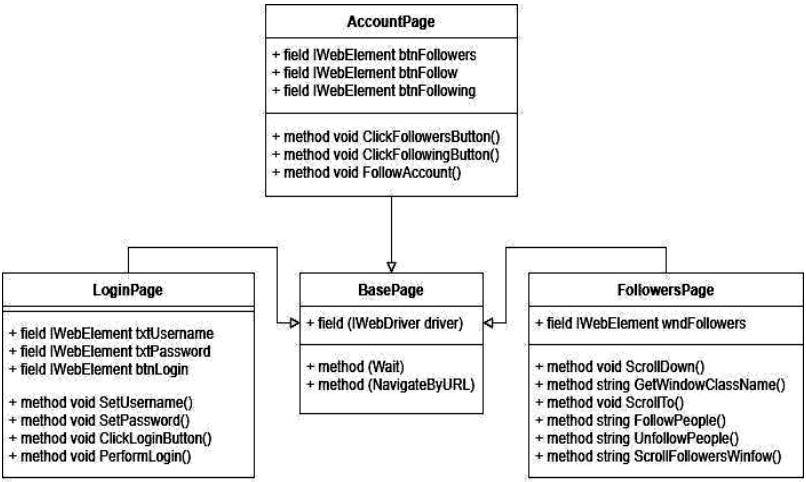

Рис. 7. *Діаграма POM класів*

Вхідними даними продукту є автоматизовані тести веб-сайту для менеджменту роботи – trello.com. (рис. 8).

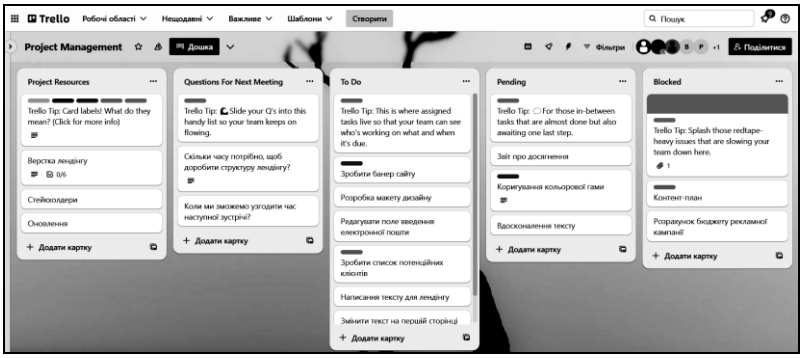

Рис. 8. *Інтерфейс веб-сайту для менеджменту роботи trello.com*

Вихідні дані – це результат запуску тестів в середовищі Visual Studio 2019, що містить у собі всю інформацію про виконані дії в програмі (рис. 9).

Тестирование	Длительн
Tests (5)	2 мин
Tests (5)	2 мин
SmokeTests (5)	2 мин
AddCartToBoard	30,4 с
FillCardWithData	33,3 с
NegativeLogin	16,5 с
PositiveLogin	13,5 с
ViewAllBoards	25,2 с

Рис. 9. *Результат запуску тестів в середовищі Visual Studio 2019*

Наприкінці тесту середовище розробки Visual Studio 2019 відобразить звіт про тестування [10]. Користувач може клацнути будь-який тест лівою кнопкою миші, а потім переглянути детальну інформацію на панелі нижче. Якщо цей текст замалий для користувача, ви можете клацнути правою кнопкою миші тест і вибрати «Журнал тестування». Потім результат тесту буде розгорнуто на весь екран.

Висновки. За допомогою автоматизації тестування було досліджено відмінності роботи UI Chrome та Headless Chrome та показано, що швидкість headless браузера більша за швидкість браузера з користувацьким інтерфейсом на 10,3%, але він підходить не для всіх видів тестів. У процесі дослідження визначено, що при автоматизованому тестуванні, тобто тестуванні програмного забезпечення для перевірки та порівняння фактичних результатів з очікуваними, Selenium WebDriver використову-

ється для виконання повторюваних завдань та інших тестових завдань, які важко виконати вручну, або які будуть потребувати великих часових затрат. Автоматизація є важливим підходом до постійного вдосконалення в сучасному світі, вона має багато переваг, таких як скорочення часу тестування, спрощення процесу звітності тощо.

В роботі проаналізовано характеристики та атрибути тестового веб-додатка. Досліджено різні методи автоматизації тестування вебдодатків, проведено детальний аналіз технологій, в результаті якого було обрано Selenium WebDriver, оскільки цей продукт має багато переваг у порівнянні з подібними продуктами, має високу практичність і може використовуватися в поєднанні з великою кількістю мов програмування. Розроблене програмне забезпечення для автоматизації тестових сценаріїв проекту «trello.com», охоплює веб-додатки, які тестуються за допомогою тестування інтерфейсу користувача. У роботі використано фреймворк NUnit, який дозволяє розробникам створювати та запускати тести для перевірки правильності роботи їх програмного коду, вирізняється своєю простотою, широкими можливостями властивостей тестів, підтримкою паралельного виконання тестів, підтримкою Data-Driven тестування, інтеграцією з іншими інструментами розробки тощо.

До недоліків headless браузера можна віднести неможливість досягти деяких специфічних функцій, які справді виконуються браузерами з User Interface. Наприклад, функціонал випадаючих меню в headless браузері не працює і призводить до падіння тестів. Відповідно до цілей тестування, можна вибрати, яка техніка є кращою та корисною для тестувальника. Наприклад, у разі участі користувачів може бути обрано тестування Real Browser, а якщо вимог до відображення інтерфейсу немає, для швидкого проведення тестування краще перейти до тестування Headless Browser. Більш ефективним буде тестування з поєднанням як Headless, так і Real Browser, що дозволить подолати окремі обмеження та недоліки кожного з них.

Список використаних джерел:

1. Durga Shree N., Sree Dharinya S, Dasari Vijayasree, Nadendla Sai Roopa, Anugu Arun. A Review on the Process of Automated Software Testing. 2022. URL: https://doi.org/10.48550/arXiv.2209.03069
2. Azza Mohamed, Ibrahim Ismail. A Performance Comparative on Most Popular Internet Web Browsers. *Procedia Computer Science.* 2022. Vol. 215. P. 589-597. URL: https://doi.org/10.1016/j.procs.2022.12.061
3. Офіційний сайт Selenium Browser Automation. URL: http://www.selenium.org.
4. Документація Selenium WebDriver. URL: https://www.selenium.dev/-documentation/webdriver/

5. Joseph Albahari C# 10 in a Nutshell: The Definitive Reference, Paperback. 2022, 1058 p.
6. Teguh Rijanandi, Faisal Dharma Adhinata Choosing the Right Programming Language in Making a Website Backend Using the Waterfall Method. *International Journal of Recent Contributions from Engineering, Science & IT (iJES).* 2022. Vol. 10 (02). P. 62-69. URL: https://doi.org/10.3991/ijes.v10i02.30845.
7. Leandro N. Sabaren, Maximiliano A. Mascheroni, Cristina L. Greiner, Emanuel Irrazábal A Systematic Literature Review in Cross-browser. *Journal of Computer Science & Technology.* 2018. Vol. 18. № 1. URL: https://doi.org/10.24215/16666038.18.e03
8. Rohit Khankhoje, Web Page Element Identification Using Selenium and CNN: A Novel Approach. *Journal of Software Quality Assurance (JSQA).* 2023. № 1 (1). P. 1-17. URL: https://doi.org/10.13140/RG.2.2.17110.42569.
9. Sohail Sarwar, Yasir Mahmood, Zia Ul Qayyum Artificial Intelligence. *Methodology, Systems, and Applications.* 2014. Vol. 8722. URL: https://doi.org/10.1007/978-3-319-10554-3_25
10. Durga Shree Nagabushanam, Sree Dharinya, Dasari Vijayasree, Nadendla Sai Roopa A Review on the Process of Automated Software Testing. 2022. URL: https://doi.org/10.48550/arXiv.2209.03069

SIMULATING TEST SCENARIOS TO EXPLORE THE DIFFERENCES BETWEEN CHROME AND HEADLESS CHROME

The introduction of test automation has numerous advantages in today's world of information technology. These include reducing testing time, simplifying the reporting process, and continuously improving efficiency.

The use of test automation opened up the opportunity to thoroughly analyze the differences in the operation of Chrome and Headless Chrome browser interfaces. It was found that the performance of the Headless browser exceeds the performance of its counterpart with a graphical interface by 10.3%. It is important to note that although Headless Chrome is quite effective, it is not always universal for different types of tests.

As part of the study, it is justified that the use of Selenium WebDriver for automated testing provides powerful opportunities for performing both routine and complex test tasks that are difficult to perform manually. The choice of this toolkit is due to its multifunctionality, high practicality and compatibility with various programming languages.

A detailed analysis of the characteristics and attributes of the test web application was carried out, according to which the software for automating test scenarios was developed for the "trello.com" project, covering web applications that are subject to testing through the user interface. The obtained results made it possible to investigate the speed of Google Chrome and Headless Chrome and establish their advantages and disadvantages. In particular, the disadvantages of the Headless browser

are the lack of the ability to use certain functions that are implemented in browsers with a graphical interface, for example, drop-down menus, which can lead to failures in tests. Disadvantages of Real Browser include high resource consumption, environment dependency, instability and complexity of settings.

Therefore, the choice of a specific testing technique should depend on the specific requirements of the project, and a combination of Headless and Real Browser testing may be used to obtain optimal results.

Key words: *test automation, browser speed, test scenarios, selenium webdriver, page object model, google chrome, headless chrome.*

Отримано: 10.11.2023