УДК 004.415.53

Костів Н.А.

Національний університет «Львівська політехніка»

Кафедра електронних обчислювальних машин

Розробка фреймворку для автоматизованого тестування WEB-сервісів

Досліджено існуючі утиліти для автоматизованого тестування, визначено їх характеристики, переваги та недоліки. Запропоновано структурну схему фреймворку та алгоритм її роботи.

Ключові слова: тестування Web-сервісів, автоматизоване тестування, фреймворк

Existing utilities for automated testing are studied, their characteristics, advantages and disadvantages are determined. The structural scheme of the framework and the algorithm of its work are suggested.

Key words: testing Web-services, automated testing, framework

Вступ. Тестування ПЗ є важливою частиною життєвого циклу ПЗ та процесу контролю якості, оскільки дозволяє перевірити і оцінити відповідність роз-робленого ПЗ до вимог його специфікацій та функціональності. Тестування програмного забезпечення є затратною та трудомісткою діяльністю. Згідно досліджень Pierre Audoin Consultants [1], у 2014році світові витрати на тестування (включаючи тестування ПЗ та обладнання) перевищували 100 міль-ярдів євро. Щороку ця цифра тільки зростає. Деякі задачі тестування, такі як регресивне тестування, можуть бути трудомісткими та вимагати багато часу якщо виконувати їх вручну. Крім того, мануальне тестування може недостатньо ефективно знаходити деякі класи помилок. При належному плануванні та впровадженні, автоматизоване тестування може принести різні переваги в порівнянні з ручним тестуванням. Можна примітити, що перепровіряти один і той же функціонал, виключаючи людський фактор, коли тестувальник перестає помічати помилки, в тому числі некоректності в відображенні елементів графічного інтерфейсу, дозволяє не пропустити такі помилки до кінцевого користувача. В той же час як автоматизація не має такого недоліку.

Стан проблеми. Сучасне програмне забезпечення є складним багатофункціональним об'єктом. Його ручна перевірка вимагає значних трудових і тимчасових витрат. На допомогу приходять засоби автоматизації тестування, які підвищують якість, забезпечують повторне використання тестів при коригуванні ПЗ. Якість не є абсолютною, це суб'єктивне поняття. Тому тестування, як процес своєчасного виявлення помилок та дефектів, не може повністю забезпечити коректність програмного забезпечення. Воно тільки порівнює стан і поведінку продукту зі специфікацією. При цьому треба розрізняти тестування програмного забезпечення й забезпечення якості програмного забезпечення, до якого належать всі складові ділового процесу, а не тільки тестування.

Хоча UI і може бути протестований тільки вручну, люди часто схильні до неефективності. Деякі помилки можуть залишитися непоміченими. Існує необхідність у автоматизації тестування веб-ресурсів, якщо їх ручне тестування вимагає значних витрат часу та багаторазового здійснення однотипних дій. Автоматизація цього процес може збільшити скорочення часових витрат і збільшити спектр можливих тестів, наприклад протестувати поведінку системи при надсиланні багато запитів одночасно.

Постановка задачі. Розробити фреймворк для автоматизованого тестування WEB-сервісів. Розробити структурну схему та описати алгоритм роботи фреймворку.

Розв'язання задачі. Для розв'язку поставленої задачі було вирішено взяти за основу Selenium WebDriver – для тестування інтерфейсу користувача, RestSharp – для тестування API запитів, SqlClient – для перевірки результатів тестових сценаріїв у БД.

Selenium – це проект з декількох інструментів, кожен з яких передбачає свій власний підхід до автоматизації тестування. Більшість інженерів-тестувальників, які працюють з Selenium, фокусуються на одному-двох інструментах з цього набору, які найкраще відповідають їхнім потребам.. У сукупності набір інструментів Selenium надає багатий набір можливостей, спеціально зібраних разом для тестування всіх типів веб-додатків. Selenium надає кілька варіантів для ідентифікації елементів інтерфейсу, порівняння очікуваного і спостережуваного результату поведінки тестованої програми. Однією з ключових особливостей фреймворку на основі Selenium є можливість запуску одних і тих же тестів в різних браузерах.[3]

Selenium WebDriver реалізований для багатьох мов програмування.Серед них Java, Python, Ruby, C# та інші [4]. Кожна з них має певні переваги та недоліки. Після проведення аналізу та враховуючи додаткові інструменти для розробки автоматизованих тестів, мовою програмування було обрано C# та платформу .NET Core 3.0 а також обрано гібридний підхід data-driven тестування та data-driven фреймворку.

Структура фреймворку (подано на рис. 1)містить наступні елементи: збір даних – етап отримання даних, яка використовується в процесі виконання тесту, це можуть бути як логін та пароль користувача чи інші дані які користувач «вводитиме» в певні поля так і значення які будуть перевірятись в ході дій тесту. Джерелом цих даних може бути як звичайний Excel документ так і певний work item з TFS чи JIRA; вибір конфігурації – етап вибору конфігурації системи де буде виконано тест, може містити в собі як розширення екрану, назву браузера, посилання на ресурс який тестується, посилання на бази даних та ін.; керування тестовими даними – етап опрацювання тестових даних, де отримані дані розбиваються на відповідні параметри та ітерації, для можливості їх використання в тест кейсі; модуль тест кейсів – етап виконання тестових скріптів, містись в собі скріпти, основані на юніт-тест фреймворку (NUnit, MSTest v1/v2, XUnit), кожен крок яких відповідає кроку у тест кейсі; модуль керування фреймворком – представлений у вигляді набору класів та методів які написані для симуляції дій користувача; модуль керування API запитами – відповідає за відправлення та отримання API запитів, їх обробку у відповідні «пейлоади»(клас з полями які відповідають отриманому JSON від API запиту); модуль запитів у базу даних – керує запитами у базу даних; модуль WebDriver – симулює дії користувача у системи, такі як клік, drag and drop, натискання клавіш чи їх комбінацій та ін.; керування браузером – етап на якому створюється та закривається процес браузеру в якому відбувається тестування, ведеться записування дій користувача в вигляді скріншотів та відео запису; формування звіту – етап форсування звіту про виконання тесту, містить в собі результати виконання, логи, фото та відео записи виконання тесту.

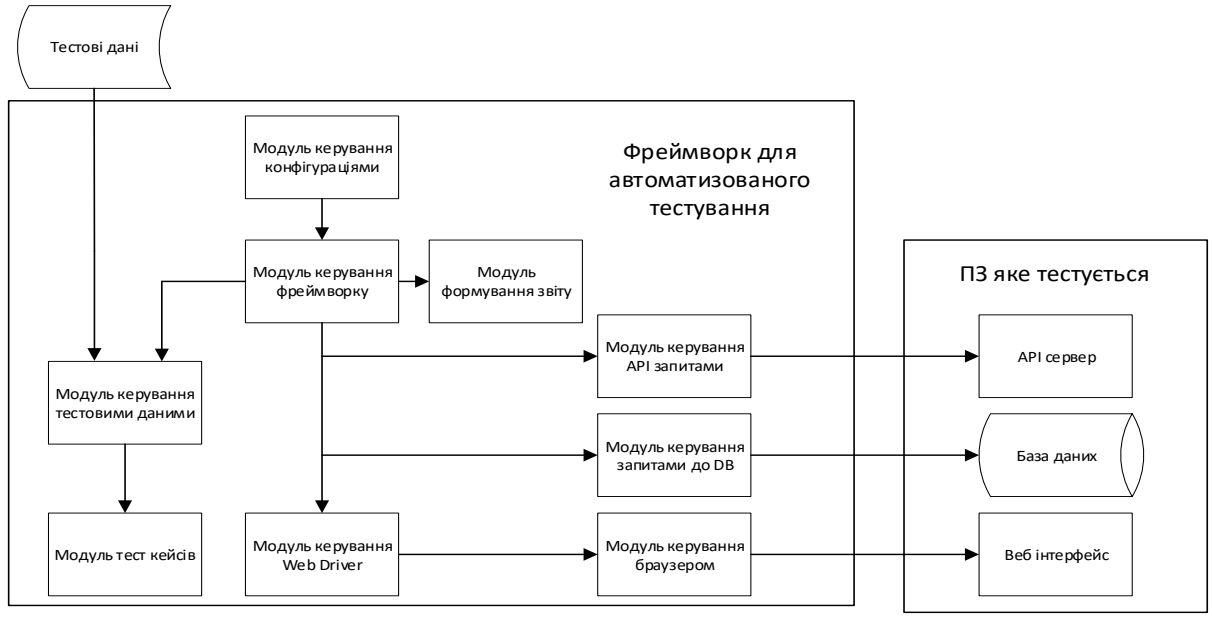

Рис. 1. Структурна схема для фреймворку автоматизованого тестування

Алгоритм роботи фреймворку складається з наступних кроків: завантаження тестових даних – крок підключення до джерела даних, отримання та опрацювання даних; вибір конфігурації – крок вибору браузера, розширення та інших параметрів; вибір чи використовувати браузер – для певних сценаріїв браузер не є необхідним, тому його створення є надлишковим; запуск браузера – крок створення екземпляру драйверу та його налаштування; запис відео та скріншотів – початкова точка коли починають запис відео та створення скріншотів під час проходження тесту; проходження тесту – крок виконання тесту згідно тестового випадку(test case); закриття процесів та підключень – зупинення процесів та підключень що більше не є необхідними після виконання тесту; опрацювання помилки – етап обробки помилки та додавання інформації що пішло не так до звіту; видалення логів – крок при успішному виконанні тесту, коли відео, скріншоти, є непотрібними і їх непотрібно опрацьовувати; публікація звіту – крок опрацювання та надсилання звіту про тест чи набір тестів на опрацювання інженеру, що містить в собі повну інформацію про виконання тесту. Алгоритм подано на рис 2.

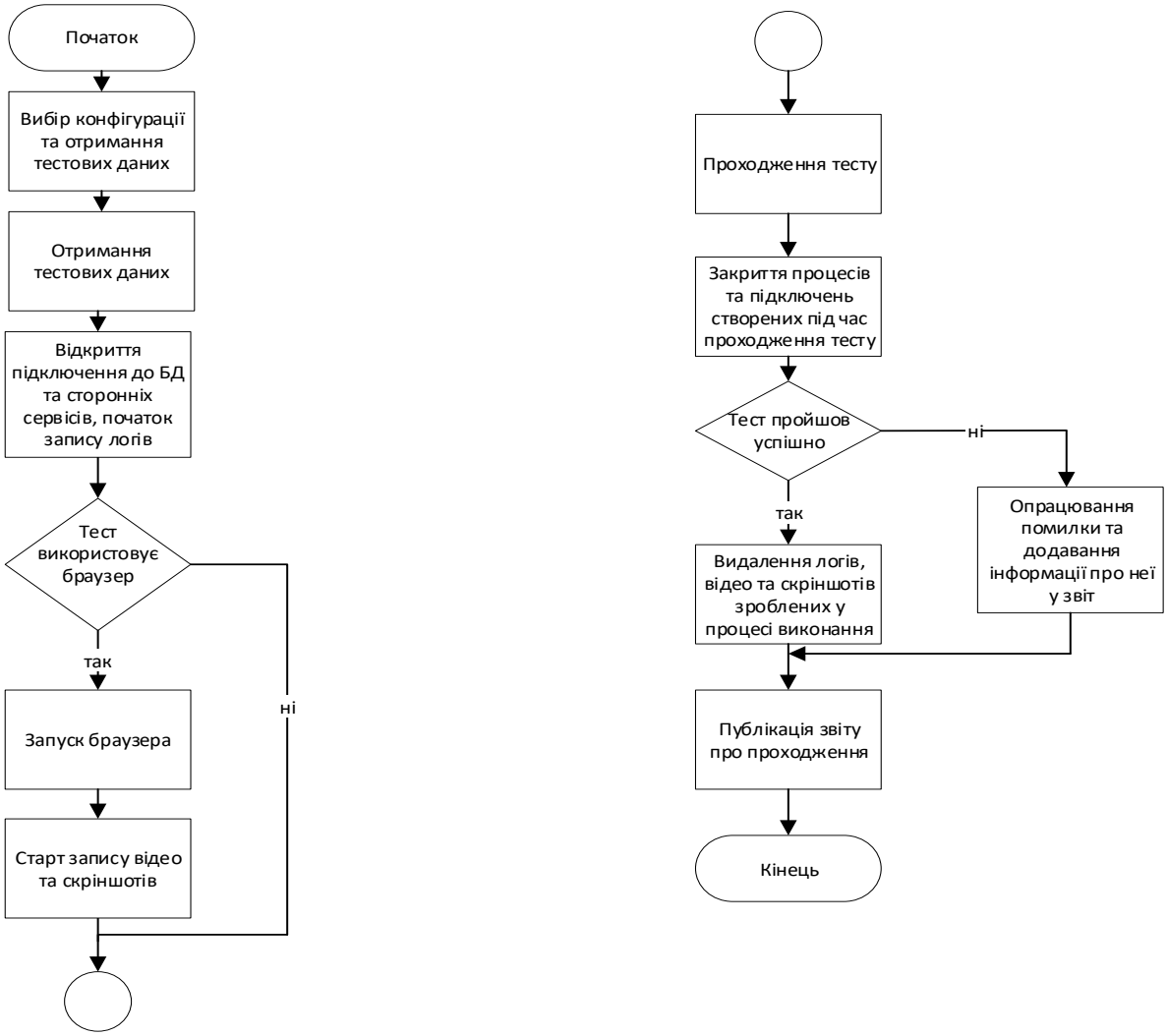

Рис. 2. Алгоритм роботи фреймворку автоматизованого тестування

Висновки. У даній роботі запропоновано підхід до розробки фреймворку для автоматизованого тестування Web-сервісів. Обрано стек технологій які найкраще оптимально поєднують з обраною мовою тестування, а саме NUnit, NLog, Selenium WebDriver, Selenium Grid, SqlClient, RESTSharp та Newtonsoft.Json. Розроблено структурну схему та алгоритм роботи фреймворку.

Література.

1. Pierre Audoin Consultants (PAC), "Software testing spends to hit Euro 100bn by 2014" 2014.
2. Alan Page, Ken Johnston, Bj Rollison (2009) How We Test Softwre at Mi-crosoft
3. Тестування програмного забезпечення [Електронний ресурс]: Режим доступу: https://uk.wikipedia.org/wiki/Тестування_програмного_забезпечення.
4. Selenium [Електронний ресурс]: Режим доступу: https://www.selenium.dev/

5. Hamilton B. NUnit Pocket Reference / Hamilton B. – Sebastopol:O'Relly Media, 2014. – 85c. – ISBN 0-596-00739-6.
6. Garousi, V., & Elberzhager, F. (2017). Test Automation: Not Just for Test Execution. IEEE Software, 34(2), 90- 96.
7. Implementing Automated Software Testing [Електронний ресурс] Режим доступу: http://methodsandtools.com/archive/archive.php?id=94.
8. Whittaker J.A. How to Break Web Software: Functional and Security Testing of Web Applications and Web Services/ M. Andrews, J. A. Whittaker // Addison-Wesley Professional, – 2006. –219p.
9. Daich G.T. et al. Software Test Technology Report. / STSC, August – 1994. – 70p.