УДК 004.051
DOI https://doi.org/10.32689/maup.it.2022.1.4

Галина КИРИЧЕК
кандидат технічних наук, доцент кафедри комп'ютерних систем та мереж, Національний університет «Запорізька політехніка», вул. Жуковського, 64, Запоріжжя, Україна, індекс 69063 (kirgal08@gmail.com)

ORCID: 0000-0001-5725-5942

Марія ТЯГУНОВА
кандидат технічних наук, доцент кафедри комп'ютерних систем та мереж, Національний університет «Запорізька політехніка», вул. Жуковського, 64, Запоріжжя, Україна, індекс 69063 (mary.tyagunova@gmail.com)

ORCID: 0000-0002-9166-5897

Анна КУРАЧ
студент, Національний університет «Запорізька політехніка», вул. Жуковського, 64, Запоріжжя, Україна, індекс 69063 (annchubich@gmail.com)

ORCID: 0000-0002-0829-1864

Halyna KYRYCHEK
PhD in Technical Sciences, Associate Professor at Computer Systems and Networks Department, National University "Zaporizhzhya Polytechnic", 64 Zhukovsky str., Zaporizhzhia, Ukraine, postal code 69063 (kirgal08@gmail.com)

Mariia TIAHUNOVA
PhD in Technical Sciences, Associate Professor at Computer Systems and Networks Department, National University "Zaporizhzhya Polytechnic", 64 Zhukovsky str., Zaporizhzhia, Ukraine, postal code 69063 (mary.tyagunova@gmail.com)

Anna KURACH
Student, National University "Zaporizhzhya Polytechnic", 64 Zhukovsky str., Zaporizhzhia, Ukraine, postal code 69063 (annchubich@gmail.com)

Бібліографічний опис статті: Киричек Г., Тягунова М., Курач А. Автоматизоване тестування веб-платформ з використанням Java та Selenium. *Інформаційні технології та суспільство.* 2022. Вип. 1 (3). С. 31–37. DOI: https://doi.org/10.32689/maup.it.2022.1.4

Bibliographic description of the article: Kyrychek, H., Tiahunova, M., Kurach, A. (2022). Avtomatyzovane testuvannia veb-platform z vykorystanniam Java ta Selenium [Automated testing of web platforms using Java and Selenium]. *Informatsiini tekhnolohii ta suspilstvo – Information technology and society*, 1 (3), 31–37. DOI: https://doi.org/10.32689/maup.it.2022.1.4

АВТОМАТИЗОВАНЕ ТЕСТУВАННЯ ВЕБ-ПЛАТФОРМ З ВИКОРИСТАННЯМ JAVA ТА SELENIUM

Анотація. *На даний час тестування програмного забезпечення є одним із основних етапів забезпечення контролю його якості та ефективності використання. Перехід на автоматизацію дозволяє скоротити час тестування і значно прискорити цій процес. Система, яка пропонується для реалізації, дозволить швидше та якісніше виконувати автоматизовані тести.* ***Метою роботи*** *є реалізація системи автоматизованого тестування веб-платформ із використанням мови програмування Java та інструменту Selenium, з підтримкою усіх популярних браузерів і операційних систем. Для досягнення основної мети пропонується вирішити наступні* ***завдання****: дослідити аналоги систем автоматизованого тестування; реалізувати метод автоматизованого тестування веб-платформ із використанням мови програмування Java та інструменту Selenium; розробити алгоритм запуску тестів в рамках системи для різних браузерів та операційних систем; навести метод автоматизованого тестування із використанням Maven та Selenium Web Driver та результати тестування системи в порівнянні з аналогами.* ***Наукова новизна.*** *Авторами пропонується застосовувати загальний фреймворк автоматизованого тестування, як систему, набір умов, концепцій та практик, спрямованих на перевикористання, зменшення витрат на підтримку, підвищення надійності, швидкості та якості виконання тестів, включаючи його використання широким колом фахівців, включаючи розробників та спеціалістів з ручного тестування.* ***Висновком****, у роботі є те, що структура тестів реалізована за допомогою анотацій, що є зрозумілою для користувача та інформує систему про призначення поміченого коду. Окрім цього, при запуску та виконанні автоматизованих тестів за допомогою TestNG, використовується багатопотоковість, яка дозволяє одночасно виконувати декілька тестів. Перевагою системи є: підтримка різних браузерів та операційних систем; кращі швидкісні характеристики; детальна система звітів за результатами тестувань; умовна безкоштовність та реалізація проекту з відкритим вихідним кодом.*

Ключові слова: *тестування, автоматизація, веб-додатки, артефакт, фреймворк, Page Object pattern.*

AUTOMATED TESTING OF WEB PLATFORMS USING JAVA AND SELENIUM

Abstract. *Currently, software testing is one of the main stages of ensuring control over its quality and efficiency. Switching to automation reduces test time and significantly speeds up the process. The system proposed for implementation will allow you to perform automated tests faster and better. The* ***aim*** *of the work is to implement an automated testing system of web platforms using the Java programming language and the Selenium tool, with support for all popular browsers and operating systems. To achieve the main aim, it is proposed to solve the following* ***tasks****: to research the analogs of automated testing systems; to implement a method of automated testing of web platforms using the Java programming language and the Selenium tool; to develop an algorithm for running tests within the system for different browsers and operating systems; to give the method of automated testing using Maven and Selenium Web Driver and the results of testing the system in comparison with analogs.* ***Scientific novelty.*** *The authors proposed to use the general framework of automated testing as a system, set of conditions, concepts, and practices aimed at reuse, reduce maintenance costs, improve reliability, speed, and quality of test performance, including its use by a wide range of professionals, including developers and specialists in manual testing. The* ***conclusion*** *of the paper is that the structure of the tests is implemented using annotations, which are understandable to the user and inform the system about the purpose of the observed code. In addition, when running and running automated tests with TestNG, multithreading is used, which allows you to run multiple tests at the same time. The advantages of the system are: support for different browsers and operating systems; best speed characteristics; detailed system of reports on test results; conditional free and open source project implementation.*

Key words: *testing, automation, web applications, artifact, framework, Page Object pattern.*

Актуальність проблеми. У сучасному світі розробка програмного забезпечення (ПЗ) постійно еволюціонує, тому тестування є одним з основних етапів контролю якості програмного забезпечення при його проектуванні, під час реалізації та при подальшому супроводженні [1]. При цьому автоматизоване тестування використовує програмні засоби для запуску тестів в процесі перевірки ПЗ, із попередньо вказаними очікуваними результатами, що допомагає скоротити час при проведенні тестувань. Актуальність вирішення цих проблем в зростанні кількості використань гнучких методів реалізації програмного забезпечення, основними напрямами яких є оптимізація виробничого процесу та мінімізація ризиків, шляхом зведення етапів розробки до серії коротких циклів. Тестування є процесом, спрямованим на виявлення характеристик інформаційних систем та демонстрації відмінностей між їх потрібним та фактичним станами [1; 2]. Окрім того воно є частиною процесу валідації при перевірці достовірності та верифікації, а точніше доказу того, що вірогідний факт або твердження є істинним [3]. При цьому автоматизація тестування ПЗ є методом, який виконується з використанням спеціальних програмних засобів, які необхідні для виконання набору тестових прикладів [4]. Тому, після автоматизації набору тестів, втручання тестувальника в їх виконання більше не потрібне [5]. Метою автоматизації є зменшення кількісті тестових сценаріїв, які запускаються вручну, не виключи ручне тестування. При цьому, поєднання двох підходів часто є найбільш виграшним варіантом, у якому частка тестів автоматизованих та виконаних вручну залежить від вимог проекту, його бюджету, експертизи команди та термінів проведення тестувань [3; 5].

Аналіз останніх досліджень та публікацій. При виконанні досліджень розглянуто п’ять фреймворків, які використовуються для автоматизованого тестування, це: Katalon, JBehave, Test Complete, Ranorex та Robot Framework. Авторами проаналізовані їх переваги та недоліки.

Katalon є дуже популярним інструментом при автоматизації процесу тестування у початківців Quality Assurance (QA) інженерів і має функції спрощеного запису скриптів та тестування із використанням ключових слів. Але використання мови Groovy, яка побудована на Java та завантаження безлічі бібліотек для аналізу тестових даних і об'єктів та записів журналу робить процес більш повільнішим, ніж у Java. Також в Katalon відсутні деталізовані звіти та немає можливості одночасного запуску декількох тестів [6]. JBehave є одним із найкращих фреймворків тестування Java, він підтримує розробку на основі поведінки, приділяючи особливу увагу взаємодії з користувачем та автоматизації дій. Але у JBehave високий поріг входження для новачків та він має складний етап установки тестового фреймворку [7]. TestComplete є ефективним інструментом для тестування десктопних, мобільних і веб-додатків та допомагає реалізувати свої тест-кейси у різних скриптових мовах: JavaScript, Python, VBScript, Delphi Script та JavaScript. Але він доступний тільки для операційної системи (ОС) Windows; має високу вартість та в ньому відсутній спрощений запис скриптів [8]. Ranorex Studio є приватним корпоративним інструментом для автоматизації тестування графічного інтерфейсу Windows, веб-додатків і мобільних додатків. При редагуванні записів або створенні тестів він використовує платформу Microsoft. NET і мови програмування C# і VB.NET, є платним та недостатньо поширеним [9]. Robot Framework (RF) є відкритим фреймворком автоматизації тестування для приймального тестування Acceptance Test Driven Development (ATDD) і Robotic Process Automation (RPA) та його ядро може запускатися на Jython (Java-реалізація Python) та IronPython (Python для .NET framework) [10; 11].

Виходячи з характеристик досліджених фреймворків, авторами пропонується застосовувати загальний фреймворк автоматизованого тестування, як систему, набір умов, концепцій та практик, спрямованих на перевикористання, зменшення витрат на підтримку, підвищення надійності, швидкості та якості виконання тестів, включаючи його використання широким колом фахівців, включаючи розробників та спеціалістів з ручного тестування.

Постановка завдання. Метою роботи є реалізація системи автоматизованого тестування веб-платформ із використанням мови програмування Java [2; 11], інструменту Selenium [1] та з підтримкою всіх популярних браузерів та операційних систем. Об'єктом дослідження є процес автоматизації тестування web-сервісів з використанням мови програмування Java та інструменту Selenium. Предметом є моделі, методи та програмні засоби автоматизації тестування web-сервісів. Процедуру автоматизованого тестування виконано із використанням Maven та Selenium Web Driver [1]. Структура проекту застосовує паттерн Page Object та Maven у середовищі Intellij IDEA.

Для досягнення основної мети авторами пропонується реалізувати: метод автоматизованого тестування веб-платформ із використанням мови програмування Java та інструменту Selenium; алгоритм запуску тестів в рамках системи, що дозволить проводити тестування із підтримкою різних браузерів та операційних систем; метод автоматизованого тестування із використанням Maven та Selenium Web Driver та модель системи з застосуванням паттерну Page Object та Maven у середовищі Intellij IDEA [12; 13]. Система, яка пропонується для реалізації, дозволить швидше та якісніше виконувати автоматизовані тести.

Виклад основного матеріалу. Система реалізована у вигляді фреймворка автоматизованого тестування веб-платформ. Середовище IntelliJ IDEA будує синтаксичне дерево, коли тільки вводимо код. В процесі вводу та виконання воно аналізує код, виявляє помилки та пропонує рішення, видаючи лише один варіант для автодоповнення, один, але вірний. Це робить розробку набагато більш оптимізованою за існуючі аналоги. Авторами для реалізації фреймворку обрано паттерн проектування PageFactory. Кожну веб-сторінку проекту описуємо як об'єкт класу. Взаємодію користувача наводимо у методах класу, а в тесті залишаємо лише бізнес-логіку. Паттерн Page Object у Selenium реалізовано за допомогою бібліотеки PageFactory та класу сторінки. Page Object є окремим класом, що містить локатори елементів, методи роботи з ними і конструктор, що приймає в якості параметру об'єкт WebDriver.

Методи класу Page Object можуть повертати об'єкти інших Page Object класів. За допомогою цього можна відтворити копію переходів та поведінки веб-програми. Одним із наслідків такого підходу є те, що необхідно моделювати як успішні так і неуспішні методи. Або, наприклад, якщо натискання на елемент може відкривати різні сторінки в залежності від умов, то також необхідно створювати різні методи для кожного необхідного випадку [12].

В роботі для тестування веб-сторінок браузерів застосовано Selenium WebDriver, який надає автотестам доступ до браузера [1; 3]. Він підтримує основні мови програмування: C #, Ruby, Java, Python, Perl, тощо; зв'язується із браузером; йому не потрібен Server, а необхідні три основні програмні компоненти: браузер, роботу якого треба автоматизувати; драйвер браузера для керування ним та скрипт/тест із набором команд для драйвера браузера. Головними особливостями реалізованого фреймворку є декларативний опис проекту (усі необхідні параметри налаштовуються за замовчанням) та гнучке управління залежностями. Maven вміє складати проекти підвантажувати до свого локального репозитарію сторонні бібліотеки, вибирати необхідну версію пакету та обробляти транзитивні залежності. Файл pom.xml у кореневому каталозі є файлом опису проекту, на основі якого здійснюються всі операції Maven.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-
4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>seleniumTestNgFramework</groupId>
  <artifactId>seleniumTestNgFramework</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>
  <name>seleniumTestNgFramework</name>
  <url>http://maven.apache.org</url>
  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <build>
    <plugins>
      ...
    </plugins>
    <resources>
       ...
    </resources>
  </build>
  <dependencies>
       ...
  </dependencies>
</project>
```

Тег project є базовим та містить всю інформацію про проект. У заголовку вказано інформацію, необхідну Maven для розуміння файлу pom.xml. Тег modelVersion вказує на поточну версію POM. Ці два теги зазвичай генеруються автоматично, міняти їх не потрібно. Далі формується унікальний ідентифікатор проекту: теги groupId та artifactId. Тег version теж входить до цієї групи. Він зазвичай генерується та оновлюється автоматично. Після номера версії йде суфікс -SNAPSHOT. Він означає, що проект перебуває у стадії розробки. Коли програмне забезпечення випускається фреймворк прибирає цей суфікс, а якщо розробка продовжиться він автоматично збільшить номер версії. Разом ці три теги дозволяють однозначно ідентифікувати артефакт. Тег name містить ім'я артефакту, що відображається, а url - посилання на сайт. Оскільки сайт не заданий під час створення, pom.xml містить нагадування про це у вигляді коментаря. Крім того, можна додати короткий опис до description. Ці теги використовуються для формування документації. Далі йде дуже важливий блок dependencies. У ньому описуються всі використовувані у проекті залежності. Кожну виділяємо тегом dependency та вказуємо унікальні ідентифікаційні дані. Maven сам завантажує транзитивні залежності. Крім того, за допомогою тега scope можна вказати етапи, на яких використано артефакт. При цьому життєвий цикл Maven складається з 9 етапів. Автотестування відбувається у фоновому режимі і може проходити паралельно з роботою будь-яких програм. Метод прискорення швидкості виконання автоматизованих тестів базується на декількох основних принципах (рис. 1).

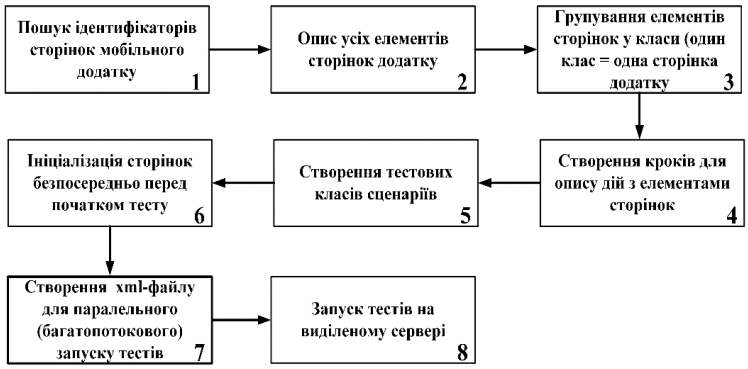

Рис. 1. Метод прискорення швидкості виконання тестів

Основні принципи роботи реалізованої системи автоматизації тестування (рис. 2): створення тестового сценарію; пошук та опис елементів веб-сторінки (використовуються різні види локаторів); групування усіх елементів в окремі класи; використання анотацій при створенні тестів; проведення попередньої ініціалізації об'єктів сторінок та реалізація багатопотоковості при запуску та виконанні автоматизованих тестів за допомогою TestNG.

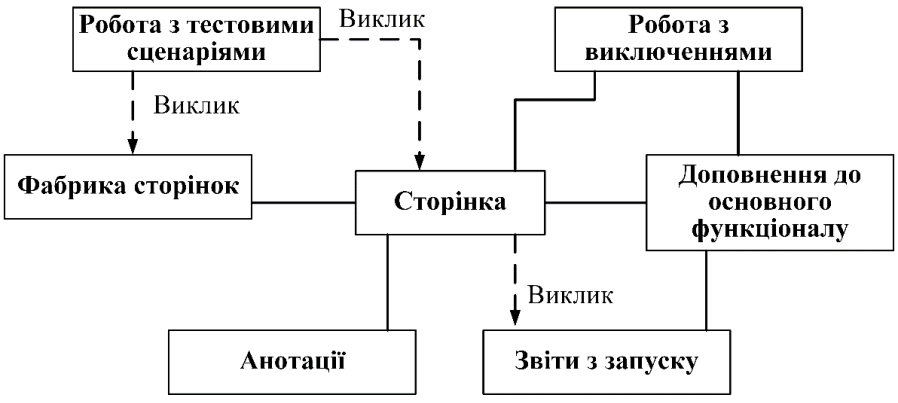

Рис. 2. Класи фреймворку для авто тестування

Алгоритм роботи системи для запуску тесту HomePageTest.java включає: запуск BasePage.java в якому реалізовані загальні методи обробки помилок; клас HomePage.java з ініціалізацією елементів веб-сторінки за допомогою локаторів та методи роботи з ними; BaseTest.java реалізує загальні пара-

метри та методи; HomePageTest.java реалізує виклик методів порівняння контенту за тест-кейсом. Усі інші тести, а саме LoginTest, LogoutTest, NegativeCartTest та PositiveCartTest реалізовано аналогічно. Основний функціонал є ядром роботи фреймворку, а службовий відповідає за надання додаткових функцій при реалізації тестів. Для створення звітності розроблено декілька видів звітів у форматі HTML.

Тестування системи. На основі проведених тестувань авторами узагальнені швидкісні показники системи, яка реалізована, та фреймворків Katalon та Ranorex. В процесі тестування проведено по десять запусків тест-ранів для тест-кейсів: HomePageTest, LoginTest та LogoutTest. Кожен тест-ран реалізовано із використанням фреймворків Katalon та Ranorex. У таблицях наведено час у секундах при виконанні автотесту для кожної із систем, що досліджуються. Результати спроб для HomePageTest тест-рану наведено у таблиці 1. У цьому класі запрограмовані переходи між сторінками: пошук по сайту; аналіз результатів пошуку та UI тестування переходу до корзини.

Таблиця 1

Результати тестів для HomePageTest тест-рану

тест-ран / t, сек	1	2	3	4	5	6	7	8	9	10
Реалізована система, T_1	51,545	57,327	50,036	47,457	51,672	55,431	53,214	48,016	52,567	58,318
Katalon, T_2	54,543	52,755	48,645	55,532	57,145	58,116	49,056	51,355	55,014	53,558
Ranorex, T_3	60,023	57,345	61,754	55,644	58,462	60,016	58,612	57,016	63,045	57,764

Розрахунок середньо арифметичного для часу виконання тестів із використанням кожної з досліджуваних систем при n=10, де n є кількістю спроб:

$$T_1 = \sum_{i=1}^{n} t_i = 52{,}558 \text{ сек.}; \; T_2 = \sum_{i=1}^{n} t_i = 53{,}571 \text{ сек.}; \; T_3 = \sum_{i=1}^{n} t_i = 58{,}968 \text{ сек.}$$

Графік порівняння отриманих результатів тестування зображено на рисунку 3.

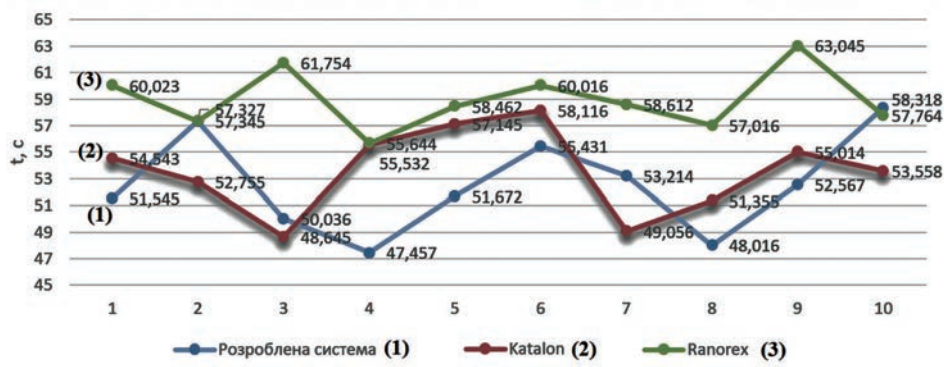

Рис. 3. Результат виконання тест-рану HomePageTest

Аналогічні результати (табл. 2) отримали для LoginTest тест-рану.

Таблиця 2

Результати тестів для LoginTest тест-рану

тест-ран / t, сек	1	2	3	4	5	6	7	8	9	10
Реалізована система, T_1	23,329	20,776	19,867	24,660	21,459	21,056	22,048	19,888	20,116	21,558
Katalon, T_2	24,551	26,412	24,002	23,761	27,558	26,005	27,389	26,779	25,338	26,339
Ranorex, T_3	27,337	28,004	27,118	26,003	29,408	26,117	28,450	26,449	27,448	26,709

Розрахунок середньо арифметичного для часу виконання тестів із використанням кожної з досліджуваних систем при

$$\text{n=10: } T_1 = \sum_{i=1}^{n} t_i = 21{,}475 \text{ сек.}; \; T_2 = \sum_{i=1}^{n} t_i = 25{,}813 \text{ сек.}; \; T_3 = \sum_{i=1}^{n} t_i = 27{,}304 \text{ сек}$$

Графічно результати дослідження для LoginTest тест-рану наведені на рисунку 4.

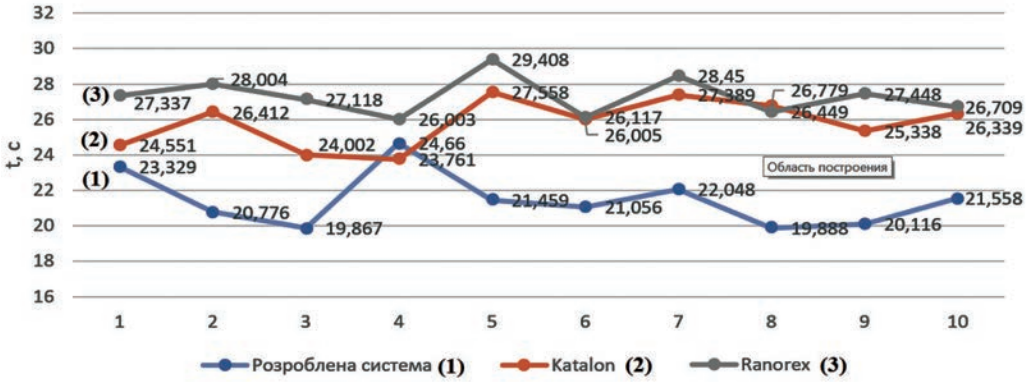

Рис. 4. Результат виконання тест-рану LoginTest

Далі наведемо результати дослідження при виконанні тест-рану LogoutTest, який перевіряє процес входу користувача до системи (табл. 3).

Таблиця 3

Результати тестів для LogoutTest тест-рану

тест-ран / t, сек	1	2	3	4	5	6	7	8	9	10
Реалізована система, T_1	34,772	32,004	33,568	32,038	35,001	34,117	32,882	31,669	34,002	33,006
Katalon, T_2	35,885	34,641	32,038	32,278	33,110	34,725	31,890	33,955	34,805	35,679
Ranorex, T_3	36,489	37,038	35,814	35,890	34,771	33,993	34,278	35,743	37,778	38,055

Розрахунок середньо арифметичного для часу виконання тестів із використанням кожної з досліджуваних систем при

$$n=10: T_1 = \sum_{i=1}^{n} t_i = 33{,}305 \text{ сек.}; T_2 = \sum_{i=1}^{n} t_i = 33{,}900 \text{ сек.}; T_3 = \sum_{i=1}^{n} t_i = 35{,}984 \text{ сек.}$$

Графічно результати дослідження наведені на рисунку 5.Аналізуючи результати тестувань можна зробити висновок, що за швидкісними показниками реалізована система має перевагу над двома іншими, порівнювальними системами. Окрім цього треба враховувати, що дослідження проводилось на короткотривалих тестах, тому для реальних систем, коли тести виконуються декілька годин, розрив у показниках швидкості буде більший.

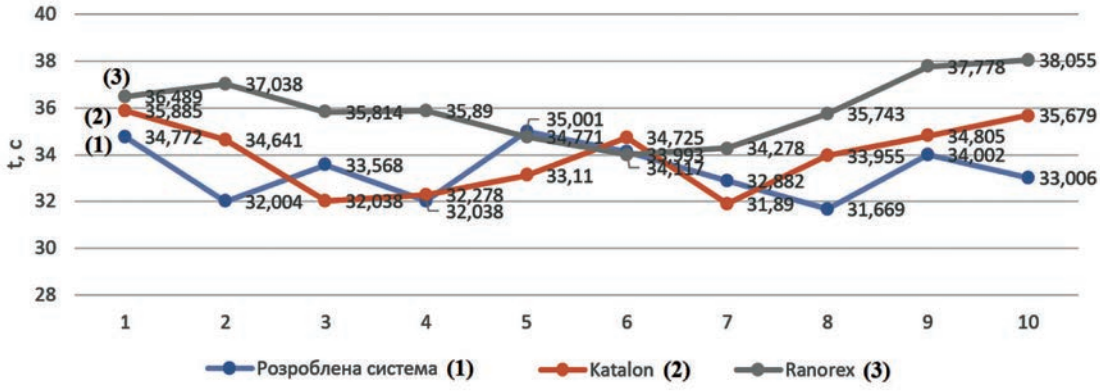

Рис. 5. Результат виконання тест-рану LogoutTest

Висновки. Всі елементи веб-сторінок згруповані в окремі класи – об'єкти сторінок. Це забезпечує швидке знаходження та заміну інформації на веб-сторінці. Структура тестів реалізована за допомогою анотацій, що забезпечує зрозумілу структуру для користувача та дає зрозуміти системі призначення коду, який був помічений. Окрім цього реалізована багатопотоковість при запуску та виконанні авто-

матизованих тестів за допомогою TestNG, що дозволяє виконувати декілька тестів одночасно. До переваг реалізованої системи відносимо: підтримку різних браузерів: Google Chrome, Mozilla Firefox, Edge; підтримку основних ОС: Windows, MacOS, Linux; кращі швидкісні характеристики у порівнянні з фреймворками Katalon та Ranorex; наявність детальної системи звітів по результатам тестування; умовна безкоштовність; реалізація проекту з відкритим вихідним кодом.

Список використаних джерел:

1. Krishna V. V., Gopinath G. Test Automation of Web Application Login Page by Using Selenium Ide in a Web Browser. Management. 2021. P. 713-732.
2. Rudkovskyi O. R., Kirichek G. G. Interaction support system of network aplications. CEUR Workshop Proceedings 2832. 2020. P. 11-23.
3. Groeneveld F., Mesbah A., Van Deursen A. Automatic invariant detection in dynamic web application's. TUD-SERG-2010-037. 2010. P. 1-10.
4. Tiahunova M., Kyrychek H., Bohatyrova T., Moshynets D. System and method of automatic collection of objects in the room. CEUR Workshop Proceedings 3077. 2021. P. 174-186.
5. Altiero F. et all. Inspecting Code Churns to Prioritize Test Cases. IFIP. Springer, Cham. 2020. P. 272-285.
6. Srivastava N., Kumar U., Singh P. Software and Performance Testing Tools. Journal of Informatics Electrical and Electronics Engineering. 2021. 2(01). P. 1-12.
7. Okolnychyi A., Fögen K. A study of tools for behavior-driven development. Full-scale Software Engineering/Current Trends in Release Engineering. 2016. P. 7-12.
8. Lenka R. K., Mamgain S., Kumar S., Barik R. K. Performance Analysis of Automated Testing Tools: JMeter and TestComplete. IEEE. 2018. P. 399-407.
9. Bhargava S., Jain P. B. Software Quality Assurance Methodology with GUI Testing Tool: Ranorex. Journal of Software Engineering Tools & Technology Trends. 2018. 5(2). P. 11-17.
10. Bisht S. Robot framework test automation. Packt Publishing Ltd, 2013.
11. Tiahunova M., Tronkina O., Kirichek G., Skrupsky S. The neural network for emotions recognition under special conditions. CEUR Workshop Proceedings 2864. 2021. P. 121-134.
12. Khan R., Qahmash A., Hussain M. R. Automatic Testing for Web Application Using HP-ALM Tool. International Journal of Engineering Research and Technology. 13(12). 2020. P. 4662-4665.
13. Mann M., Tomar P., Sangwan O. P. Automated software test optimization using test language processing. Int. Arab J. Inf. Technol. 2019. 16(3). P. 348-356.

References:

1. Krishna, V. V., Gopinath, G. (2021). Test Automation of Web Application Login Page by Using Selenium Ide in a Web Browser. Management (pp. 713-732).
2. Rudkovskyi, O. R., Kirichek, G. G. (2020). Interaction support system of network aplications. In CEUR Workshop Proceedings 2832 (pp. 11-23).
3. Groeneveld, F., Mesbah, A., Van Deursen, A. (2010). Automatic invariant detection in dynamic web applications. Technical Report Series TUD-SERG-2010-037
4. Tiahunova, M., Kyrychek, H., Bohatyrova, T., Moshynets, D. (2021). System and method of automatic collection of objects in the room. In CEUR Workshop Proceedings 3077 (pp. 174-186).
5. Altiero, F., Corazza, A., Di Martino, S., Peron, A., Starace, L. L. L. (2020). Inspecting Code Churns to Prioritize Test Cases. In IFIP International Conference on Testing Software and Systems (pp. 272-285). Springer, Cham.
6. Srivastava, N., Kumar, U., Singh, P. (2021). Software and Performance Testing Tools. Journal of Informatics Electrical and Electronics Engineering, 2(01), 1-12.
7. Okolnychyi, A., Fögen, K. (2016). A study of tools for behavior-driven development. Full-scale Software Engineering/ Current Trends in Release Engineering, 7-12.
8. Lenka, R. K., Mamgain, S., Kumar, S., Barik, R. K. (2018). Performance Analysis of Automated Testing Tools: JMeter and TestComplete. In International Conference on Advances in Computing, Communication Control and Networking (ICACCCN) (pp. 399-407). IEEE.
9. Bhargava, S., Jain, P. B. (2018). Software Quality Assurance Methodology with GUI Testing Tool: Ranorex. Journal of Software Engineering Tools & Technology Trends, 5(2), 11-17.
10. Bisht, S. (2013). Robot framework test automation. Packt Publishing Ltd.
11. Tiahunova, M., Tronkina, O., Kirichek, G., Skrupsky, S. (2021). The neural network for emotions recognition under special conditions. In CEUR Workshop Proceedings 2864 (pp. 121-134).
12. Khan, R., Qahmash, A., Hussain, M. R. (2020). Automatic Testing for Web Application Using HP-ALM Tool. International Journal of Engineering Research and Technology, 13(12), 4662-4665.
13. Mann, M., Tomar, P., Sangwan, O. P. (2019). Automated software test optimization using test language processing. Int. Arab J. Inf. Technol., 16(3), 348-356.