УДК 004.942; 536.2
DOI https://doi.org/10.32838/2663-5941/2021.2-1/20

Д'яченко Л.І.
Чернівецький національний університет імені Юрія Федьковича

Ілін А.В.
Чернівецький національний університет імені Юрія Федьковича

Шумиляк Л.М.
Чернівецький національний університет імені Юрія Федьковича

Газдюк К.П.
Чернівецький національний університет імені Юрія Федьковича

Тарновецька О.Ю.
Чернівецький національний університет імені Юрія Федьковича

ОГЛЯД ПРОГРАМНИХ ЗАСОБІВ ТА МЕТОДОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМ АВТОМАТИЗОВАНОГО ТЕСТУВАННЯ

У цій роботі розглянуто методи автоматизованого тестування програмного забезпечення, досліджено прийоми та інструменти для створення автоматизованих тестів та надано рекомендації стосовно доцільності застосування тих чи інших інструментів для тестування різних типів програмного забезпечення. Розглянуто платформи з відкритим кодом NUnit, що розроблена для написання та запуску тестів мовами програмування Microsoft.NET, xUnit – це загальна назва для кількох модулів тестування, які виводять свою структуру та функціональність із SUnit Smalltalk та тестове рішення для автоматизації.NET SpecFlow, яке побудоване на парадигмі BDD (Behavior Driven Development). Описано переваги використання Selenium WebDriver для автоматизації тестування вебдодатків та розглянуто системи для відображення результатів тестування, а саме: ExtentReports – це бібліотека звітів для автоматизованих тестів та Allure Framework – це гнучкий полегшений багатомовний інструмент звіту про тестування.

З метою оцінки правильності поданих рекомендацій було розроблено програмне забезпечення для автоматизації тестових сценаріїв спеціалізованого програмного забезпечення, створений макет для написання дефект-репортів та тестових сценаріїв. Основним завданням системи є аналіз ефективності використання автоматизованого тестування для перевірки якості програмного забезпечення. Система показує, як можна заощадити час тестування, використовуючи автоматизовані тести. Технології тестових рушіїв NUnit та XUnit було використано для запуску тестів та механізму перевірки даних. Розроблене ПЗ покриває тестами вебдодаток за допомогою інтеграційного, UI- та API-тестування. Система показала значне заощадження часу тестування (ми отримали реальне скорочення часу, витраченого на виконання тестування у шість разів), використовуючи автоматизовані тести та виконуючи їх паралельно.

Ключові слова: *автоматизоване тестування, Selenium WebDriver, тестові сценарії, дефект-репорт, інтеграційне тестування.*

Постановка проблеми. Автоматизація тестування – це техніка тестування програмного забезпечення, яка використовує спеціальні програмні засоби для виконання набору тестів. Ручне тестування проводиться людиною, яка ретельно виконує кожен етап тестування, використовуючи створені раніше тестові сценарії.

Програмне забезпечення для автоматизації цих процесів може вводити дані в систему, що підлягає тестуванню, порівнювати очікувані з фактичними результатами та генерувати докладні звіти. Автоматизація випробувань вимагає значних вкладень як коштів, так й інших ресурсів.

Послідовні цикли розробки часто вимагають виконання одного і того ж набору тестів неодноразово. За допомогою інструментів автоматизації тестування можна записати цей набір тестів і повторно відтворити його за необхідності. Після автоматизації набору тестів втручання людини не потрібне. Метою автоматизації є зменшення кіль-

кості тестових сценаріїв для запуску вручну, а не взагалі усунення ручного тестування.

Автоматизація тестування – найкращий спосіб підвищити його ефективність, охоплення тестом і швидкість виконання окремих тестових сценаріїв у процесі тестування програмного забезпечення.

Нині є різні підходи до автоматизації тестування [1, с. 35, 54]. Комплексна стратегія тестування включає в себе три рівні автоматизації:

1) рівень модульного тестування (Unit Tests Layer) – компонентні тести, що зазвичай пишуться розробниками;

2) рівень функціонального тестування (Functional Test Layer, or Service/API Tests Layer) – тестування через доступ до функціонального шару, не беручи до уваги призначений для користувача інтерфейс;

3) рівень тестування інтерфейсу користувача (GUI Test Layer) дає змогу тестувати не тільки сам інтерфейс користувача, а й реалізовувати функціональне тестування, виконуючи операції, що використовують бізнес-логіку програми.

Для забезпечення найкращої якості додатку рекомендується автоматизувати всі три рівні. Тести верхнього рівня (тестування інтерфейсу) вважаються більш складними та дорогими в розробці. Співвідношення може змінюватися залежно від специфіки тестованого продукту.

Автоматизація тестування програмних продуктів через графічний інтерфейс користувача набирає нині дедалі більшої популярності. І це не дивно, оскільки тільки в цьому разі додаток буде протестовано тим же способом, яким він буде використовуватися кінцевими користувачами. Плюсами такого виду тестування є те, що тестування відбувається на реальних пристроях із різною конфігурацією, різними операційними системами, в різних браузерах.

До мінусів цього виду автоматизованого тестування можна зарахувати те, що тести проходять довго, багато зусиль витрачається на їх підтримку (при зміні інтерфейсу потрібне внесення змін у всі тести, які використовують оновлені елементи), тести можуть бути нестабільними (можуть «падати» через спроби звернутися до елементів інтерфейсу, які ще не завантажилися, нестабільно працювати з конфігураціями пристроїв для запуску тестів, відмінними від тієї, на якій тести створювалися).

Реалізується такий вид автоматизації через використання спеціалізованих інструментів, які дають змогу спілкуватися з інтерфейсом програми, що тестується [2].

Data-driven testing – тестування на основі даних – це метод тестування програмного забезпечення, при якому дані тестування зберігаються у форматі таблиці або таблиць. Кероване даними тестування дає змогу запускати один скрипт, що виконує сценарії для усіх даних із таблиць, та очікувати виведення результатів у тій самій таблиці. Цей метод також називають тестуванням на основі таблиць, або параметризованим тестуванням.

Data Driven Framework – це система автоматизації тестування, в якій вхідні значення зчитуються з файлів даних і зберігаються у змінних у тестових скриптах. Це дозволяє тестувальникам об'єднувати як позитивні, так і негативні тести. Вхідні дані в структурі, керованій даними, можуть зберігатися в .xls, .xml, .csv файлах та базах даних.

Keyword-based testing – це техніка сценаріїв, яка використовує файли даних, що містять ключові слова, пов'язані з тестованою програмою. Ці ключові слова описують набір дій, необхідних для виконання певного кроку.

Тест, керований ключовими словами, складається з ключових слів високого та низького рівня, включаючи аргументи ключових слів. Він складається для опису дії тестового випадку. Цей метод також називається тестуванням на основі таблиці або тестуванням на основі дії [2].

Задля продуктивної роботи треба вибрати зручний для автоматизації інструмент. Правильний вибір інструменту є, як правило, складним завданням. По-перше, слід визначити вимоги, вивчити різні інструменти та їх можливості, встановити очікування від цього інструмента та перейти до підтвердження концепції.

Існує велика кількість додатків для автоматизації тестування, такі як: HP LoadRunner, HP QuickTest Professional, HP Quality Center; Segue SilkPerformer; IBMRational FunctionalTester, IBM Rational PerformanceTester, IBM Rational TestStudio;

За допомогою цих додатків та інструментів тестувальник може:

– встановити продукт;
– створити тестові дані;
– взаємодіяти з GUI;
– визначати проблеми.

Проте не варто забувати, що автоматизоване тестування не може повністю замінити ручне, адже це затратний процес, який вимагає багато ресурсів.

Нині багато проєктів визнають переваги автоматизованого тестування, можливість збільшення

покриття тестами, яке зменшить час на регресивне тестування.

Наразі є великий вибір утиліт і фреймворків, які можуть значно полегшити та автоматизувати процес тестування [3, с. 125, 215].

Постановка завдання. У статті буде більш детально розглянуто та проаналізовано програмні засоби, вибрані для реалізації системи автоматизованого тестування.

Виклад основного матеріалу дослідження. NUnit – це платформа з відкритим кодом, розроблена для написання та запуску тестів мовами програмування Microsoft.NET.

У NUnit тести можна запускати постійно, також є можливість одночасного запуску кількох тестів. Результати видаються негайно. Ніяких суб'єктивних суджень людини або тлумачень результатів тесту не потрібно. Простота фреймворку дає змогу легко виправити помилки в міру їх виявлення.

NUnit використовує різноманітні атрибути задля впровадження своєї функціональності в тестові класи й методи. До основних атрибутів можна зарахувати такі:

– TestFixture – клас, який містить тестові методи;

– Test – метод у тестовому класі. Цей метод перетворюється на тест і починає відображатись у тест-провіднику;

– SetUp – метод, позначений атрибутом, виконується безпосередньо перед кожним тестом;

– TearDown – метод, позначений цим атрибутом, буде виконуватись після кожного тестового методу;

– Repeat – метод має бути виконаний кілька разів;

– Order – метод має бути виконаний у відповідному порядку.

xUnit – це загальна назва для кількох модулів тестування, які виводять свою структуру та функціональність із SUnit Smalltalk. Назви багатьох фреймворків є варіацією "SUnit", зазвичай замінюючи "S" першою літерою (або літерами) в назві передбачуваної ними мови ("JUnit" для Java, "RUnit" для R тощо). Ці фреймворки та їх загальна архітектура в сукупності відомі як "xUnit" [4, с. 340].

SpecFlow – це тестове рішення для автоматизації .NET, побудоване на парадигмі BDD (Behavior Driven Development). SpecFlow використовується для визначення, управління та автоматичного виконання зручних для читання тестів прийняття в проєктах .NET (Full Framework та .NET Core).

Тести SpecFlow пишуться з використанням технології Gherkin, що дає змогу писати тестові сценарії на зрозумілій мові. SpecFlow використовує офіційний парсер Gherkin, який підтримує понад 70 мов. Потім ці тести прив'язуються до коду вашої програми за допомогою так званих прив'язок, що дозволяє виконувати тести, використовуючи вибрану систему тестування. Також можна виконувати тести за допомогою власного спеціалізованого тестового рушія SpecFlow, SpecFlow + Runner.

Selenium WebDriver – це зручний інструмент, за допомогою якого зручно автоматизувати програмне забезпечення з відкритим кодом. Selenium WebDriver – це колекція API з відкритим кодом, які використовуються для автоматизації тестування вебдодатків [4, с. 451].

Структура взаємодії Selenium WebDriver з бібліотеками мов програмування показана на рис. 1.

Засіб Selenium WebDriver використовується для автоматизації тестування вебдодатків. Він підтримує багато браузерів, таких як Firefox, Chrome, IE та Safari. Однак, використовуючи Selenium WebDriver, можна автоматизувати тестування лише для вебдодатків. Він не відповідає вимогам для віконних програм.

Selenium WebDriver також підтримує різні мови програмування, такі як C#, Java, Perl, PHP та Ruby для написання тестових скриптів. Selenium WebDriver не залежить від платформи, оскільки один і той же код може використовуватися в різних операційних системах, таких як Microsoft Windows, macOS та Linux. Це один із компонентів сімейства Selenium, який також включає Selenium IDE, Selenium API та Selenium Grid.

Selenium WebDriver не обробляє компонент вікна, але це обмеження можна подолати, використовуючи зовнішні інструменти, такі як AUTO IT tool, Sikuli тощо. Він має різні стратегії розташування, такі як ID, Ім'я, Текст посилання, Частковий текст посилання, Назва класу, Селектор CSS та Xpath. WebDriver також має кращу підтримку динамічних вебсторінок, таких як Ajax, де елементи вебсторінки можуть змінюватися без перезавантаження самої сторінки. Використовуючи різні jar-файли, ми також можемо тестувати API, проводити тест бази даних тощо, використовуючи Selenium WebDriver.

Там, де це можливо, WebDriver використовує власну функціональність рівня операційної системи, а не команди JavaScript на основі браузера для керування браузером. Це обходить проблеми з

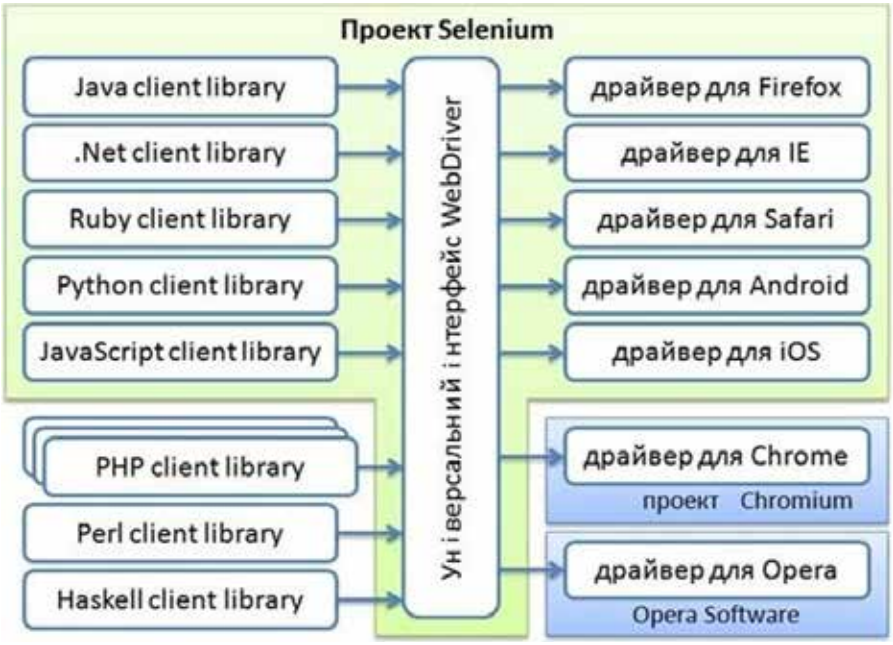

Рис. 1. Структура взаємодії Selenium WebDriver із бібліотеками мов програмування

незначними відмінностями між власними командами та командами JavaScript, включаючи обмеження безпеки.

Selenium 2.0 має на меті надати базовий набір будівельних блоків, з яких розробники можуть створити власну мову для конкретного домену (DSL). Один із таких DSL вже існує: проєкт Watir на мові Ruby має багату історію гарного дизайну. Watir-webdriver реалізує API Watir як обгортку для Selenium WebDriver в Ruby. Watir-webdriver створюється повністю автоматично на основі специфікації WebDriver та специфікації HTML [5, с. 181].

За своїми ознаками Selenium WebDriver є специфікацією програмного інтерфейсу для управління браузером, референтними реалізаціями цього інтерфейсу для кількох браузерів, набором клієнтських бібліотек для цього інтерфейсу кількома мовами програмування.

WebDriver дає змогу використовувати мову програмування на ваш вибір під час розробки тестів.

Єдиними недоліками WebDriver є те, що він не має вбудованих команд для автоматичного формування результатів тесту.

Cross Browser Testing – це тип функціонального тестування для перевірки того, чи працює вебпрограма належним чином у різних браузерах.

Якщо використовується Selenium WebDriver, то можна автоматизувати тестові кейси за допомогою браузерів Internet Explorer, FireFox, Chrome, Safari. Для одночасного виконання тестових сценаріїв із різними браузерами на одній машині можна інтегрувати фреймворк TestNG із Selenium WebDriver.

Крос-браузерне тестування – це техніка тестування вебдодатків за допомогою різних веббраузерів. Selenium може підтримувати різні типи браузерів для автоматизації. Він може бути інтегрований із TestNG для проведення мультибраузерного тестування. З параметрів у testing.xml передається ім'я браузера, а в тестовому сценарії можна створити посилання на WebDriver відповідно.

Різні веббраузери дотримуються стандартів Open Web, але вони мають власні інтерпретації. Оскільки кожен із них відображає HTML, CSS та JavaScript унікальними способами, ретельного налагодження вихідного коду вебсайту недостатньо, щоб гарантувати, що сайт буде виглядати та поводитись так, як передбачалося в різних браузерах (або різних версіях одного браузера).

Веброзробникам доводиться абстрагувати відмінності браузера. Перехресне тестування сумісності браузерів допомагає в цьому, визначаючи помилки сумісності для браузера, щоб їх можна було швидко налагодити. Це допомагає гарантувати, що вебсайт не відчужує значну частину своєї цільової аудиторії просто тому, що вебсайт не працює в операційній системі браузера.

Нижче наведено найкращі практики, яких слід дотримуватися під час проведення тестування в кількох браузерах за допомогою Selenium.

Варто вибирати бібліотеки та фреймворки обережно. Перш ніж приступати до розробки, потрібно знати, що найновіші фреймворки CSS можуть допомогти створити яскравіший та динамічний інтерфейс користувача, але вони можуть бути несумісними з кожним браузером, оскільки кожен із них має певний механізм візуалізації. Перш ніж використовувати найновіші бібліотеки або фреймворки CSS або JS, варто прочитати інструкції щодо браузера, щоб визначити, чи підтримують вони ці бібліотеки та фреймворки.

Використання належних бібліотек та програм для запуску завдань JavaScript відіграє важливу роль у розробці вебдодатків, тому дуже важливо використовувати правильні ресурси JavaScript, які відповідають вимогам сайту та пропонують сумісну підтримку браузера.

Є безліч інструментів JavaScript на вибір. Grunt та Gulp виділяються тим, що їх можна автоматизувати відповідно до вимог. Вони також покращують загальну продуктивність програми, забезпечуючи компіляцію, підключення та інші функції, які покращують якість коду після компіляції [6, с. 145, 184].

Служби контролю якості мають вирішити, в яких браузерах та операційних системах сайт має бути протестований. Кожен браузер має кілька версій, а деякі браузери навіть оновлюються постійно – принаймні, раз на місяць. Таким чином, необхідно перевірити, який браузер, версії браузера та ОС необхідні для крос-браузерного тестування за допомогою Selenium. Пріоритетними є комбінації, які, ймовірно, використовуватимуть найбільші сегменти цільової аудиторії.

Internet Explorer не підтримує розширені стилі та фреймворк CSS. Є ймовірність, що багато елементів дизайну сайту будуть спотворені при доступі через IE. Щоб виправити це, варто створити окрему таблицю стилів для IE. InternetExplorerDriver – реалізація WebDriver для браузера Internet Explorer.

Проведення крос-браузерного тестування за допомогою Selenium є надзвичайно важливим, оскільки воно забезпечує сумісність вебдодатків та забезпечує надійну взаємодію з користувачем. Користувач може вибрати будь-який браузер. Це може бути Google Chrome, Safari або Opera Mini. Наприклад ви, як розробник, користуєтесь Google Chrome. Оскільки ви розробляєте вебсайт, який стосується Chrome, він буде правильно відображатись у браузері. Але коли ви публікуєте сайт, то припускаєте, що кожен із ваших користувачів використовує Google Chrome. Потрібно пам'ятати, що Google Chrome має 62% частку в статистиці браузерів. Отже, ви просто втратили 38% своїх користувачів, навіть не давши їм змогу побачити продукт [7, с. 189, 210].

ExtentReports – це бібліотека звітів для автоматизованих тестів. Logger – це просто об'єкт для реєстрації повідомлень або подій для певної системи чи програми. ExtentReports використовує стиль ведення журналу, щоб додати інформацію про тестові сесії, такі як створення тестів, додавання знімків екрана, призначення тегів та додавання подій або серії кроків.

Усі методи в ExtentReports є багатопотоковими. Рекомендованим використанням є підтримка одного екземпляра об'єкта ExtentReports для тестового сеансу.

Allure Framework – це гнучкий полегшений багатомовний інструмент звіту про тестування, який не лише демонструє стислий опис про те, що було перевірено у формі вебзвіту, але дозволяє кожному, хто бере участь у процесі розробки, отримати максимум корисної інформації із щоденного виконання тестів.

З позиції програмістів та тестувальників звіти Allure скорочують загальний життєвий цикл дефектів: невдачі тесту можна розділити на помилки та непрацюючі тести, а також журнали, кроки, пристосування, вкладення, історію та інтеграцію з TMS та системами відстеження помилок, тому відповідальні розробники та тестувальники матимуть всю інформацію під рукою [8, с. 214].

З позиції менеджерів Allure дає чітке уявлення про те, які особливості були висвітлені, де згруповані дефекти, як виглядає графік виконання та багато інших зручних речей. Модульність та розширюваність Allure гарантують, що завжди можна щось налаштувати, щоб зробити використання Allure зручним для користувача.

Практичне застосування інструментів для автоматизованого тестування ПЗ

Для перевірки доцільності автоматизації тестування програмних продуктів було створено власний фреймворк, що автоматизує тестування спеціалізованого програмного забезпечення.

Основним завданням системи є аналіз ефективності використання автоматизованого тестування для перевірки якості програмного забезпечення. Система показує, як можна заощадити час тестування, використовуючи автоматизовані тести та виконуючи їх паралельно.

Технології тестових рушіїв NUnit та XUnit було використано для запуску тестів та механізму перевірки даних. Використання цих двох технологій

також обґрунтовується порівнянням їхньої імплементації. XUnit запускає всі тестові скрипти одразу в паралельному режимі, що є не дуже зручним для автоматизованого тестування, але ідеально підходить для тестування модулів. XUnit не містить тестового прогресу та контексту (додаткова інформація впродовж виконання тесту), через що в подальшому важко відслідкувати причину падіння тестового сценарію. NUnit своєю чергою запускає тести в послідовному режимі, але містить багато різних конфігураційних можливостей для запуску тестових сценаріїв у різних режимах послідовності виконання та поділу їх на тестові набори.

SpecFlow використовується для наочного опису методології BDD (Behavior Driven Development), яка своєю чергою заохочує співпрацю між усіма, хто бере участь у розробці ПЗ: розробниками, тестувальниками та представниками бізнесу, такими як власники продуктів або бізнес-аналітики. Процес тестування був побудований саме на основі методології BDD оскільки за її допомогою було створено тестові сценарії та скрипти, написані не тільки за допомогою коду, але й за допомогою ключових фраз мовою Gherkin, які є зрозумілими для всіх зацікавлених сторін.

UI-тестування полягає в імітації роботи користувача з тестованим продуктом. Із цією метою було використано Selenium та Selenium Web Driver, який є портативним фреймворком для тестування вебдодатків. Його перевагами є підтримка багатьох мов програмування (наприклад C#, Groovy, Java, Perl, PHP, Python, Ruby та Scala) та можливість взаємодії з вебелементами вебсторінки для крос-браузерного тестування.

POM (Page Object Model) – це шаблон дизайну, що створює сховище об'єктів для елементів вебінтерфейсу. Згідно з цією моделлю, для кожної вебсторінки у програмі має бути відповідний клас сторінки. У цій роботі цей клас сторінки визначає елементи цієї вебсторінки, а також містить методи, які виконують операції з цими елементами. Це дало змогу зменшити дублювання коду та полегшило технічне обслуговування. POM є основою автоматизованого UI-тестування.

Extended Web Driver – це своєрідна обгортка Selenium Web Driver для кастомізації будь-яких дій та елементів, яка приховує роботу з Selenium, що допоможе швидкій зміні інструменту для автоматизації вебсторінок.

Для відображення звіту тестових сценаріїв було використано Allure Framework та Extent Report Framework. Ці інструменти не тільки демонструють стисле уявлення про те, що було протестовано в формі вебзвіту, але дозволяють кожному, хто бере участь у процесі розробки, отримати максимум корисної інформації з повсякденного виконання тестів.

Іншим типом тестування, яке було розглянуто, є API-тестування. Це тип тестування програмного забезпечення, який перевіряє Application Programming Interface (API). Метою тестування API є перевірка функціональності, надійності, продуктивності та безпеки інтерфейсів програмування. Під час тестування API замість використання стандартних засобів введення/виведення використовується програмне забезпечення для надсилання викликів до API, отримання вихідних даних та записування реакції системи. В основному API-тестування концентрується на рівні бізнес-логіки архітектури програмного забезпечення. У цій роботі для тестування API використовується інструмент RestSharp, який своєю чергою є найпопулярнішою клієнтською бібліотекою HTTP для .NET. Зручними функціями, які відділяють RestSharp від інших технологій і є причиною застосування цього інструменту, є автоматична серіалізація та десеріалізація, виявлення типів запитів і відповідей та різноманітність автентифікацій.

Важливою частиною тестування є підготовка тестових даних, яка має бути швидкою та зручною. Для підготовки тестових даних використовується робота з базою даних, а саме технологія Entity Framework. Це набір технологій в ADO.NET, які підтримують розробку орієнтованих на дані програмних додатків. Іншою ціллю застосування Entity Framework є перевірка записів у БД, оскільки ця технологія спрощує роботу з БД і виключає потребу в будь-яких інших адаптерах для підключення до БД.

Структура розробленого фреймворку подана на рис. 2.

Як видно з рис. 2, програмне забезпечення спроєктовано на основі компонентів логіки, які своєю чергою використовують БД, модулі NUnit та Selenium Web Driver.

Під час запуску звіту у html-форматі ми бачимо всю інформацію про виконання тестів:

– відображення кількості тестів, які пройшли чи провалилися;

– відображення кількості кроків, які пройшли чи провалилися;

– відображення відповідної інформації у вигляді кругової діаграми;

– інформація про початок, кінець та час виконання тестів;

У разі натискання на певний розділ відображаються тести з цього розділу.

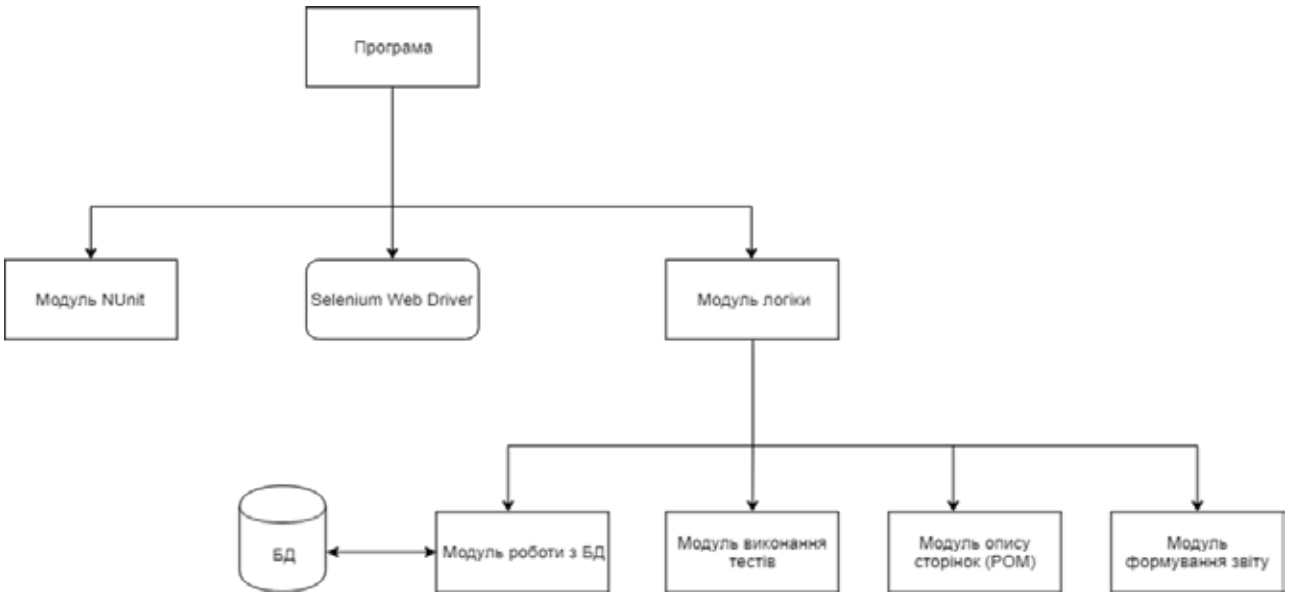

Рис. 2. Загальна структура фреймворку

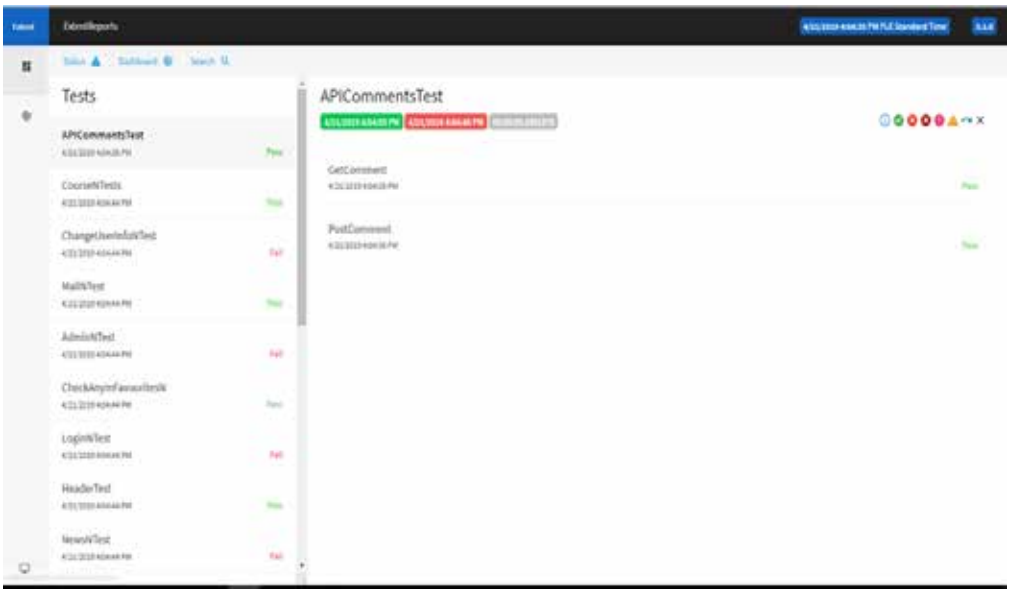

Рис. 3. Графічне відображення звіту

Результати дослідження. У цьому розділі представлені результати порівняння швидкості тестування ПЗ із використанням розробленого фреймворку та за допомогою ручного тестування.

Таблиця 1

Результати часових витрат на тестування розробленого ПЗ

Вид тестування	Затрачений час (год.)
Ручне тестування	18
Автоматизоване тестування (з використанням розробленого фреймворку)	3

Як бачимо, ми отримали реальне скорочення часу, затраченого на виконання тестування у шість разів. Хоча варто заначити, що інструменти автоматизованого тестування, а також навчання їх використанню коштують недешево, тому треба ретельно оцінювати бюджет. Для виконання ручного тестування не треба створювати додаткових програмних інструментів та витрачати на це ресурси. Але сам час виконання тестових сценаріїв вручну досить довгий. Певні види тестування взагалі неможливо здійснити за допомогою ручного методу, наприклад, тестування навантаження (не можна змоделювати велику кількість користувачів вручну).

Також до недоліків автоматизованого тестування часто зараховують відсутність «людського погляду». Можливе існування помилок, які помітить тільки людина.

Автоматизоване тестування не рекомендується використовувати для програмних продуктів, які часто змінюються. Тестувальникам доведеться витрачати багато часу на написання нових інструментів та модернізацію існуючих. В той час, як із ручним тестуванням, незначні зміни можуть бути досліджені відразу, без написання коду і його виконання.

Взагалі для тестування більшості реальних проєктів використовують комбінацію обох видів тестування, що дає змогу отримати від тестування максимальний результат.

Висновки. У процесі цього дослідження було визначено, що автоматизація тестування використовується для автоматизації повторюваних завдань та інших завдань тестування, які важко або ресурсозатратно виконувати вручну. Автоматизація тестування – вагоме питання, яке постійно вдосконалюється в сьогоднішньому світі та має багато переваг: зменшення часових затрат на тестування; спрощення процесів формування звітності та багато іншого.

Також було проаналізовано особливості та властивості тестування web-додатків. Внаслідок аналізу для автоматизації тестування більшості вебпроектів рекомендується використовувати Selenium WebDriver, адже цей продукт має багато переваг порівняно з аналогами, практичний у застосовуванні та може використовуватися разом із великою кількістю мов програмування.

Для оцінки правильності поданих рекомендацій було розроблено програмне забезпечення для автоматизації тестових сценаріїв спеціалізованого програмного забезпечення, створений макет для написання дефект репортів та тестових сценаріїв. Розроблене ПЗ покриває тестами вебдодаток за допомогою інтеграційного, UI- та API-тестування. Система показала значне заощадження часу тестування, використовуючи автоматизовані тести та виконуючи їх паралельно.

Список літератури:

1. Майерс Г., Баджет Т., Сандлер К. Мистецтво тестування програм. Москва, 2012. 270 с.
2. Автоматизоване тестування web-додатків. Київ, 2010. URL: http://www.sworld.com.ua/index.php/ru/technical-sciences-b217/informatics-computer-science-and-automation-b217/29858-b217-014. (дата звернення: 20.01.2021).
3. Кріспін Л., Грегорі Д. Гибкое тестирование: практическое руководство для тестировщиков ПО и гибких команд. Москва, 2011. 463 с.
4. Липаев В.В. Тестирование компонентов и комплексов програм : підручник. Москва-Берлін, 2015. 528 с.
5. Азарский К.И. Тестирование. Легкий старт. Москва, 2014. 226 с.
6. Бейзер Б.В. Тестирование черного ящика. Технологии функционального тестирования программного обеспечения и систем. Санкт-Петербург, 2012. 318 с.
7. Copeland L. A Practitioner's Guide to Software Test Design. London, 2013. 238 с.
8. Whittaker J.A. Exploratory Software Testing. Boston, 2011. 253 с.

Diachenko L.I., Ilin A.V., Shumyliak L.M., Hazdiuk K.P., Tarnovetska O.Yu. **SOFTWARE AND METHODOLOGIES FOR THE IMPLEMENTATION OF AUTOMATED TESTING SYSTEMS**

This paper discusses the methods of automated software testing, explores techniques and tools for creating automated tests and provides recommendations on the feasibility of using certain tools for different types of software testing. There are considered some automated platforms for software testing: NUnit open-source platforms that was designed to write and run tests in Microsoft.NET programming languages, xUnit is a common name for several testing modules that derive their structure and functionality from SUnit Smalltalk and a test solution for.NET – SpecFlow automation, which built on the paradigm of BDD (Behavior Driven Development). The benefits of using Selenium WebDriver to automate web application testing are described, and systems for displaying test results are discussed. We described ExtentReports – a report library for automated tests and Allure Framework – a flexible, easy-to-use multilingual test report tool.

To assess the correctness of the submitted recommendations, software for test automation of specialized software was developed, a layout for writing defect reports and test scenarios was created. The main task of the system is to analyze the effectiveness of the automated testing usage for verifying the software quality. The system shows how you can save testing time using automated tests. NUnit and XUnit test engine technologies were used to run the tests and the data validation mechanism. The developed software covers the web application with tests using integration, UI and API testing. The system showed significant savings in testing time (we obtained a real reduction in the time spent on testing in six times.), using automated tests and running them in parallel.

Key words: *automated testing, Selenium WebDriver, test scenarios, defect report, integration testing.*