Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Розробка фреймворку автоматизації тестування веб-сервісів електронної комерції на базі C# та NSelene

Виконав(ла): студент(ка) 6 курсу, групи СПм-61
спеціальності 121 Інженерія програмного забезечення
(шифр і назва спеціальності)

	(підпис)	Дранівський М.І. (прізвище та ініціали)
Керівник	(підпис)	Бойко І.В. (прізвище та ініціали)
Нормоконтроль	(підпис)	Бойко І.В. (прізвище та ініціали)
Завідувач кафедри	(підпис)	Петрик М.Р. (прізвище та ініціали)
Рецензент	(підпис)	Луцик Н.С. (прізвище та ініціали)

Тернопіль 2020

АННОТАЦІЯ/ABSTRACT

Кваліфікаційна робота магістра містить: 47 с., 15 рис., 1 табл., 22 джер.

Метою даної дипломної роботи розробка фреймворку автоматизації тестування веб сайтів електронної торгівлі на базі CMS PrestaShop.

Методи розробки базуються на уже існуючих рішеннях автоматизації тестування мовою програмування C#. Програмні інструменти та патерни які були використані при виконанні розробки програмного забезпечення: та її бібліотеки .NET фреймворку, середовище розробки Visual Studio, інструмент Selenium WebDriver, фреймворк NUnit, патерн проектування PageObject, фреймворк Extent Reports, надбудова NSelene.

Результатом виконаної роботи є програмне забезпечення, яке дозволить оптимізувати та спростити процес автоматизації тестування веб сайтів електронної комерції на основі платформи PrestaShop.

ЯКІСТЬ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ, C#, SELENIUM WEBDRIVER, NUNIT, PAGE OBJECT, NSELENE,PRESTASHOP , ЕЛЕКТРОННА КОМЕРЦІЯ .

The purpose of this thesis is to develop a framework for automating the testing of e-commerce websites based on CMS PrestaShop.

Development methods are based on existing solutions for automating testing in the C # programming language. Software applications and patterns used in the development of the program: C # programming language, .NET framework, Visual Studio development tool, Selenium WebDriver tool, NUnit framework, PageObject design pattern, Extent Reports framework, NSelene wrapper.

The result of the work done is software that will allow optimize and simplify the process of automating testing of e-commerce websites based on the PrestaShop platform.

QUALITY OF SOFTWARE, C#, SELENIUM WEBDRIVER, NUNIT, PAGE OBJECT, NSELENE, PRESTASHOP, E-COMMERCE.

ЗМІСТ

ВСТУП

Актуальність даної роботи полягає у необхідності автоматизації тестування веб сайтів електронної комерції на базі платформи PrestaShop, оскільки їх ручне тестування вимагає значних витрат часу та багаторазового здійснення однотипних дій.

Необхідність автоматизації обґрунтовується значним зростом популярності інтернет магазинів, як одних з найефективніших засобів ведення бізнесу у період пандемії та, відповідно, збільшенням вимог до їх якості.

Метою даної роботи є підвищення ефективності процесу тестування web-сервісів електронної комерції на основі CMS PrestaShop.

Для досягнення поставленої мети необхідно вирішити такі задачі:

- аналіз засобів для автоматизації тестування та вибір оптимальних;
- розробка програмного забезпечення для автоматизації тестування web-сервісів електронної комерції на основі CMS PrestaShop;

Об’єктом дослідження є процес автоматизації тестування web-сервісів електронної комерції на основі CMS PrestaShop.

Предметом дослідження є методи та засоби автоматизації тестування web-сервісів електронної комерції на основі CMS PrestaShop.

Наукова новизна отриманих результатів: запропоновано новий метод використання проекту Selenium WebDriver в обгортці NSelene, котра містить певні покращення у взаємодії із драйвером, що дозволяє писати більш стабільні та лаконічні тести. В даній роботі розглянуто спільне використання NSelene та Nunit, що дозволяє підвищити загальну швидкодію процесу тестування web-сервісів електронної комерції на основі CMS PrestaShop та розширити функціональні властивості цього процесу.

Практичною цінністю даної роботи є розроблене програмне та, що дозволить пришвидшити процес тестування web-сервісів електронної комерції на основі CMS PrestaShop та забезпечить проведення тестів у максимальній наближеності до тестових сценаріїв.

1 ОГЛЯД І АНАЛІЗ ТЕОРЕТИЧНИХ ВІДОМОСТЕЙ

1.1 Аналітичний огляд в області дослідження

Тестуванням програмного забезпечення називається процес технічного дослідження, який має на меті виявити інформацію про якість продукту. Техніки тестування передбачають як процеси пошуку багів та дефектів, так і перевірки програмних складових з метою їх оцінки. В процесі тестування оцінюються:

- те на скільки система відповідає вимогам проектування та розробки
- валідація полів та правильність поведінки системи при введенні різного набору тестових даних
- час виконання операцій системи
- практичність використання програмного продукту
- сумісність розроблюваної системи з іншим програмним забезпеченням та різноманітними операційними системами
- відповідність розроблюваної системи задачам замовника та бізнесу в цілому [1].

Беручи до уваги те, що кількість та різноманітність тестових сценаріїв навіть для простих програмних рішень може бути практично нескінченною, стратегія тестування повинна забезпечити глибоке тестування системи із врахуванням відведеного на нього часу та ресурсів.

Результатом такого стандартного підходу є те що програмне забезпечення тестується відповідно до раніше складеного тест плану з метою виявлення багів.

У підсумку тестування, користувачі та замовник зокрема повинні отримати повну вичерпну інформацію про стан програмної системи і її готовність до експлуатації. Як правило, тестування програмної системи виконується з моменту розробки коду. Сам процес розробки також включає планування подальшого рефакторінгу коду, тестування та відведення часу на виправлення знайдених багів [2].

Тестування програмного забезпечення включає різноманітні техніки, метою яких є перевірка відповідності між реальною і очікуваною поведінкою програмної системи. Така перевірка відбувається саме завдяки наперед визначеному комплексу тестів, які складаються відповідно до загальноприйнятих технік тест дизайну. Якість програмного забезпечення є суб'єктивним поняттям і не може бути абсолютною. Тестування програмного забезпечення не в змозі забезпечити ідеальну якість та гарантувати повну коректність роботи розроблюваного продукту. Варто брати до уваги ще й те що тестування програмного забезпечення є лише складовою повноцінного процесу забезпечення якості і є лише одним із заходів процесу контролю якості [3].

Зачасту, опис якості програмного забезпечення складається із понять надійності, практичності, безпечності і коректності, проте цей список значно більший і є добре описаним у стандарті ISO 9126. Вимоги до усієї супутньої документації описуються стандартом IEEE 829—1998 Standard for Software Test Documentation, в якому, крім переліку необхідних документів, описано також їх зміст [4].

Тестування програмної системи може виконуватися різними підходами, техніками та з використанням найрізноманітнішої кількості інструментів, але сам процес тестування у своїй суті є творчим процесом дослідження, який вимагає, окрім уважності, терпіння і аналітичних вмінь, навиків творчого нестандартного мислення.

Розробка перших програмних систем відбулася на замовлення міністерства оборони, адже як відомо саме в армія рушієм науково-технічного прогресу. Звісно, в процесі цих тестувань відбувався запис абсолютно усіх кроків і тестових процедур, результатів - цей процес був максимально формалізованим і документованим. На цьому етапі весь процес тестування був відгалуженим, але виконувався тими ж працівниками, котрі займалися розробкою цих програмних систем.

У 1960-х більшого значення почали надавати «вичерпному» тестуванню, яке мало на меті перевірити усі можливі шляхи виконання програми із усіма

можливими даними, проте було визнано що ця вимога не має сенсу і практично неможлива у виконанні. А все тому що кількість вхідних даних може мати найрізноманітніші варіації, а шляхів виконання при таких умовах також може бути просто безліч. Складність полягає також у тому що надзвичайно складно знайти помилки у архітектурі чи специфікації до такого програмного забезпечення. Згодом, з урахуванням усіх перелічених причин це тестування визнали неможливим та недоцільним.

На початку 1970-х підхід до тестування програмного забезпечення почав характеризувати тестування як процес перевірки коректності роботи продукту. На цей період лише припадає зародження програмної інженерії як такої, а верифікація розумілася лише як доказ правильності розроблюваного продукту.

Така концепція була логічною і теоретично могла розвиватися, проте на практиці виявилося що часові затрати є занадто великими і необгрунтованими, тому що цей підхід всеодно не охоплював усі аспекти тестування. Хоч цей підхід уже давно застарілий, але і в наші дні він має місце.

У 1980-х в тестування почали з'являтися такі поняття як запобігання дефектів, а проектування тестів почало розглядатися як найефективніший засіб запобігання помилок. Тестування наповнилося таким поняттям як запобіганням дефектам. Приблизно в цей же момент почали висловлюватись ідеї необхідності методології тестування та введення заходів забезпечення якості на усіх етапах розробки програмного забезпечення, адже, окрім готового коду, перевірки потребували і вимоги, і архітектура і самі тести безпосередньо.

Звичне до цього моменту тестування не могло забезпечити всеосяжної перевірки через те що інженери з якості просто не залучались до всіх етапів. Згодом новий підхід показав беззаперечні переваги, оскільки дозволив виявляти проблеми у вимогах та архітектурі системи на ранніх етапах, що в свою чергу позитивно відобразилося на термінах та бюджетах проектів.

Уже з середини 1980-х починають з’являтися перші інструменти автоматизованого тестування. З використанням комп'ютерів очікувалося що виконання тестів пришвидшиться і надійність цих тестів теж значною мірою

зросте. Проте ці інструменти були примітивними та не передбачали можливості використання скриптових мов для написання тестових сценаріїв.

Такі поняття як планування, проектування, розробка, супровід та виконання тестів з'явилися лише на початку 1990-х років. Цей етап є дуже важливим оскільки ознаменував перехід від звичайного тестування до повноцінного поняття забезпечення якості.

З середини 1990-х років з початком стрімкого розвитку інтернету почало розвиватися також гнучке тестування - як наслідок розвитку гнучких методологій програмування.

Уже 2000-х роках поняття тестування набуло більш широкого та точного визначення. Саме оптимізація бізнес-процесів відіграла ключову роль у формуванні розуміння забезпечення якості програмних систем [5].

Головною ідеєю сформованого підходу стала максимізація значення кожного з етапів життєвого циклу розробки для досягнення потрібного рівня якості, доступності та продуктивності.

1.1.1 Автоматизація процесу тестування програмного забезпечення

Автоматизація тестування являється частиною загального процесу тестування на стадії контролю якості у процесі розробки програмної системи. Це тестування відбувається з використанням різноманітних програмних засобів для виконання тестових сценаріїв та перевірки результатів проходження тестів, що значним чином позитивно впливає на час тестування і дозволяє спростити цей процес [6].

В автоматизації тестування існує два головні підходи: тестування на рівні коду та GUI-тестування. До автоматизації на рівні коду належить, зокрема, модульне тестування. Під автоматизацією GUI-тестування передбачається

імітація роботи користувача з веб сторінкою за допомогою тестових скриптів.

Найбільшого поширення набула сама автоматизація програмних систем саме через графічний інтерфейс користувача. Популярність цього підходу аргументується тим що такий спосіб тестування дозволяє імітувати роботу користувачів максимально наближено до реальних варіантів використання системи. Ще однією перевагою цього типу тестування є відсутність необхідності мати доступ до вихідного коду програми [6].

1.1.2 Переваги автоматизованого тестування

Повторюваність - одноманітність тестових сценаріїв, можливість повторного запуску тих же тестів та мінімізація “людського фактору” дозволяють часто використовувати один і той же набір регресійних тестів перед кожним релізом нової версії програмного забезпечення.

Програмований - завдячуючи сучасним мовам програмування, при розробці тестового сценарію можна створити потрібні умови для виконання тесту, які максиально наблизять тест до реальних умов відтворення - це можуть бути, наприклад, навантаження на ресурси мережі, які інженер з тестування не може створити засобами ручного тестування.

Швидке виконання - один раз правильно створений створений скрипт дозволяє заощадити велику кількість часу на виконання тестів в майбутньому.

Низькі затрати на підтримку - найбільше часу витрачається на написання тестових скриптів, проте коли вони є готові, то на їх постійну підтримку та оновлення витрачається мінімальна кількість часу

Звіти – в залежності від потреб бізнесу та команди, можна реалізувати найрізноманітніші способи звітування про проходження тестів.

Виконання без втручання - протягом виконання тестів інженер із

забезпечення якості може виконувати інші завдання, або ж можна налаштувати виконання тестів навіть у неробочий час для ще більшої мінімізації ресурсозатрат.

Беззаперечно автоматизація тестування має ряд переваг, які значно спрощують та оптимізовують процес забезпечення якості в цілому. Перспективність цього підходу важко переоцінити, адже саме автоматизація дозволяє забезпечити постійну стабільну перевірку програмних систем.

1.1.3 Недоліки автоматизованого тестування

Повторюваність - цей пункт є одночасно перевагою та недоліком, оскільки передбачає що протягом виконання заданого скрипту існує ризик існування багу у наближеному функціоналі. Цей баг буде знайдено протягом ручного тестування після проведення кількох додаткових операцій, але оминеться протягом виконання скрипту, оскільки не зачіпає саме того елементу, з яким існують проблеми.

Витрати на підтримку - автоматизація тестування може бути затратою у випадку систем які постійно розвиваються та змінюються, адже велика кількість змін у програмній системі побудує зміни у тестових скриптах.

Затрати на розробку - витрати на розробку автоматизованих тестів можуть розцінюватись як розробка додаткового програмного забезпечення, адже також включають роботу із фреймворками, бібліотеками, утилітами - відповідно все це потребує додаткових налаштувань, вмінь та знань. Це доволі складний процес, тому що по-факту відбувається розробка програми, що перевіряє іншу програму.

Вартість інструменту автоматизації. Так як більшість безкоштовних інструментів не містять зручного функціоналу, у комерційних проектах потрібно використовувати платні інструменти, вартість яких подекуди має досить високу ціну.

Пропуск дрібних помилок - потрібно завжди враховувати що скрипт завжди буде пропускати дрібні помилки, не пов'язані із сценарієм безпосередньо - це можуть бути як орфографічні помилки у написах, так і помилки некоректного розміщення елементів [6].

Зазвичай, автоматизація тестів відбувається для забезпечення регресійного тестування, а не для знаходження свіжих помилок, саме тому автоматизація є хорошим інструментом, проте не може вважатися повноцінною заміною ручному тестуванню. Автоматизовані тести не можуть вказати на незручність використання системи, чи дослідити першопричину виникнення нової помилки.

Тому, незважаючи на ряд переваг, автоматизація тестування залишається лише доповненням до ручного тестування, а не альтернативою йому.

1.2 Аналіз предметної області

Предметом дослідження даної магістерської роботи є автоматизація тестування веб сервісів для електронної торгівлі. Саме тому важливу роль відіграє розуміння особливостей даної предметної області, його різновиди та сучасний стан ринку програмного забезпечення та в цілому.

1.2.1 Роль електронної комерції у сучасному світі

Електронна комерція в загальному понятті полягає у використанні мереж та електронних платежів для ведення бізнесу.

Електронна комерція передбачає не лише можливість замовлення товарів з онлайн магазину. Вона включає усі можливі аспекти взаємодії організації з її зацікавленими сторонами у глобальній мережі. Електронна комерція існує вже

біля 30 років - від моменту коли комп'ютерна система Рейтерс почала використовуватися для проведення операцій на фондових біржах.

В електронній комерції повністю змінила підхід до торгівлі - тепер кожен може замовити будь-який товар із будь- якої точки світу, відпадає необхідність оренди великих приміщень для демонстрації товарів та все більше значення має просування реклами бренду в глобальній мережі інтернет.

Спочатку термін «електронна комерція» передбачав тільки можливіст електронного обміну даними для надсилання бізнес-документів, зокрема рахунок-фактуру на покупку чи голосування в електронному вигляді. З розвитком цієї галузі термін набув нового значення і почав використовуватися для ведення торгівлі товарам та послугами в мережі Інтернет [7].

Основною класифікацією електронної комерції є «бізнес-бізнес» (B2B), «бізнес-споживач» (B2C), «споживач-споживач» (C2C), «споживчий-бізнес» (C2B) і «мобільна комерція» (M-Commerce).

B2B. «Бізнес-бізнес» - це тип комерційної транзакції, або операція між однією компанією та іншою компанією щодо передачі послуг та продуктів. Можливим поясненням цього може бути те, що компанія включає інтернет-оптову торгівлю, в якій компанія продає матеріали, продукцію та послуги іншим компаніям на веб-сайті..

B2C. Електронна комерція на основі B2B означає доставку продукції "від компанії до компанії". Така модель є ідеальним рішенням для компанії, яка співпрацює з багатьма виробниками та постачає різні види продукції іншим компаніям. Дуже часто модель B2B використовується комплексними постачальниками офісної продукції. Співпрацюючи з діловими партнерами та збагачуючи свою діяльність системою B2B, таке підприємство легко продає власну продукцію та товари партнерів по всій країні на основі логістичної діяльності.

C2B. Це одна з найпопулярніших бізнес-моделей електронної комерції. Продажі, засновані на B2C - це інтернет-магазин з моделлю роздрібних продажів,

коли компанія продає різні види товарів приватним особам. Цей бізнес ведеться в Інтернеті, а не в стаціонарному магазині.

C2C- це бізнес-модель електронної комерції, яка значно відрізняється від попередніх двох. У цій моделі є фізична особа з обох сторін процесу купівлі-продажу. Прикладом організації, яка веде свій бізнес на основі цієї моделі, є eBay. У моделі C2C люди торгують між собою різними видами товарів в Інтернет-середовищі.

M-Commerce. Мобільна комерція, є однією з галузей електронної комерції. Ми можемо називати m-commerce транзакції тими, які ми здійснюємо за допомогою мобільних пристроїв, наприклад телефону чи планшета.

Обсяг онлайн продажів у світі поступово зростав на 25-30% протягом останніх років, проте пандемія у 2020 році спровокувала значний скачок у сфері електронної комерції і змінила ставлення до цієї сфери в цілому. Значно зріс попит на програмне забезпечення цього типу, а головне посилилися вимоги до якості такого програмного забезпечення, оскільки торгівля онлайн стала основним способом ведення бізнесу значної кількості підприємств.

1.2.2 Опис «PrestaShop» системи

«PrestaShop» - система керування вмістом для інтернет-магазинів з відкритим кодом. Система написана на PHP, для написання шаблонів використовується Smarty, для створення баз даних використовується MySQL.

Система призначена для малого та середнього бізнесу і має більше 310 стандартних функцій для швидкого створення функціонального магазину. Незважаючи на безкоштовність, «PrestaShop» готовий запропонувати цілком собі унікальні функції, на кшталт обліку залишків товарів на декількох складах. Подібні речі досить рідкісні і для комерційних рішень, а тут ця функція є безкоштовною.

Як сильну сторону, можна виділити юзабіліті. Панель управління досить добре продумана і навчання тих же менеджерів з продажу не потребує багато часу.

PrestaShop заснований на 3-рівневій архітектурі Model – View – Controller (MVC). Команда розробників вирішила не використовувати фреймворк PHP, такий як Zend Framework, Symfony або CakePHP, щоб забезпечити кращу читабельність і, таким чином, швидше редагування.

Це також сприяє кращій продуктивності, оскільки програмне забезпечення складається лише з рядків коду, який йому потрібен, і не містить купу додаткових загальних бібліотек.

Трирівнева архітектура має багато переваг:

- Читати код програмного забезпечення легше.
- Розробники можуть швидше додавати та редагувати код.
- Графічний дизайнер та інтегратори HTML можуть працювати з обмеженнями папки / themes без необхідності розуміти або навіть читати один рядок PHP-коду.
- Розробники можуть працювати над додатковими даними та модулями, якими можуть користуватися інтегратори HTML [8].

1.3 Постановка задачі

Метою дипломної роботи є створення програмного забезпечення для автоматизації тестування інтернет магазинів на базі CMS PrestaShop, яке буде відповідати наступним вимогам:

1. Містити інструменти інтеграції з системою PrestaShop за допомогою PrestaShop WebService API
2. Бути адаптованою до бізнес-процесів діяльності інтернет магазину.

3. Зберігати логи виконання тестів різного рівня деталізації у текстовому форматі.

4. Автоматично генерувати HTML звіти проходження тестів із детальною статистикою та логами по кожному тесту окремо.

5. Зберігати скріншоти тестів які не пройшли та відображати їх у звіті.

6. Містити інструменти автоматичного генерування даних, необхідних для проходження тестів.

Таким чином, при розробці даного програмного забезпечення буде досягнуто наступні цілі:

1. Скорочення часу на інтеграцію із системою PrestaShop.
2. Скасування необхідності підготовки набору тестових даних.
3. Покращення інформування та звітності щодо проходження тестів.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ПРОГРАМНОЇ СИСТЕМИ

2.1 Обґрунтування доцільності розробки системи

У сучасних умовах інструменти автоматизованого тестування відіграють важливу роль в процесі забезпечення якості програмних продуктів. Правильний вибір інструменту дозволяє оптимізувати та скоротити час і витрати на цей довготривалий процес.

Однією з найпопулярніших CMS на ринку програмного забезпечення для електронної торгівлі є PrestaShop. Така популярність платформи пояснюється безкоштовністю, зручністю у використанні, великим ком'юніті, та надзвичайно великою кількість найрізноманітніших тем та плагінів, що дозволяють максимально кастомізувати інтернет магазин під потреби та особливості бізнесу.

Проблема тестування веб сайтів, розроблених на базі PrestaShop полягає у тому що на ринку фреймворків автоматизованого тестування відсутній комплексний інструмент, який дозволив би провести тестування графічного інтерфейсу та API таких продуктів. Існуючі на ринку рішення вимагають складної і довгої адаптації фреймворку до потреб проекту і не містять жодних інтрументів, які могли б спростити автоматизацію тестування для інженера із забезпечення якості.

Можливість використовувати функції електронної торгівлі у тестах дозволяє будувати зручні для читання, логічні тести, які відповідаю налаштуванням платформи. Фреймворк містить також інструменти API тестування, які дозволять провести тестування в процесі розробки для перевірки коректності роботи базового функціоналу пов'язаного із основними елементами продаж, такими як продукти, замовлення, клієнти, способи доставки та іншими елементами управління інтернет-магазинами.

2.2 Аналіз конкурентів

Об’єктом дослідження цього розділу є фреймворки автоматизованого тестування які дозволяють написання тестів на платформі .NET з використанням мови програмування C#.

Більшість фреймворків автоматизованого тестування розробляється для мови програмування Java, тому ринок фреймворків з використанням мови програмування #C є молодим та незаповненим. Найпопулярнішими аналогами розробленого продукту є Atata Framework та фреймворк Gauge.

Atata – це тестовий фреймворк написаний на мові програмування C # для автоматизації Web тестування на основі Selenium WebDriver. Atata фреймворк складається із: компонентів; атрибутів для пошуку веб-компонентів; атрибутів налаштувань; стратегій; тригерів; методів та атрибутів для верифікації.

Перевагами фреймворку є те що всі об’єкти сторінок наслідується від класу Page/PageObject це також дозволяє використати Fluent підхід. Це підхід який представляє собою ланцюжок викликів, до самого закінчення виконання програми. Atata виконує пошук елементів веб сторінка за назвою змінної та контенту веб елемента. Фреймворк має велике ком'юніті і добре описану документацію.

Недоліками Atata є необхідність у збереженості послідовності викликів без можливості їх переривання, а також складність в освоєнні фреймворку [9].

Gauge - це один з новітніх інструментів автоматизації тестування з відкритим вихідним кодом для Mac, Windows, і Linux, розроблений компанією ThoughtWorks - тією ж компанією, що створила колись Selenium.

Можливості фреймворку:

- Написання специфікації з використанням markdown
- Написання скриптових тестів на мовах за вибором Javascript, C#, Python, Java, Ruby.

На відміну від інших інструментів, Gauge надає можливість використовувати код різних мов програмування та передавати змінні середовища без написання відповідного коду. Для цього потрібно помістити файли середовища в структуру проекту, визначену Gauge.

- Створення тестів в різних IDE за своїм вибором (Visual Studio, VS Code, IntelliJ і т. д.)

Gauge підтримує різні методи виконання тестів, такі як виконання на рівні специфікації, рівні сценарію, рівні каталогу і тд. За допомогою Gauge передача тестових даних в фреймворк не сильно змінюється в порівнянні з Cucumber, але Gauge має додаткову можливість передавати великі файли даних в тести без будь-яких зусиль з написання логіки синтаксичного аналізу. Gauge підтримує передачу даних у вигляді файлів і таблиць на етап тестування. Це дозволяє легко використовувати найбільш поширені формати файлів, такі як Excel, CSV і word.

- Запускання тестів паралельно

Gauge дозволяє паралельне виконання тестів, створюючи потоки виконання в залежності від кількості ядер процесора, доступних на машині.

- Створення користувацьких звітів

Gauge дозволяє генерувати звіти про тестування засобами фреймворку без написання додаткового коду, а також змінювати звіти, відповідно до потреб.

- Розробка спеціальних плагінів для індивідуальних вимог
- Хороша підтримка на ком'юніті

Gauge має детальну і добре пояснену документацію з прикладами і фрагментами коду. Документація містить демо-проекти з Gauge для початківців, які наглядно демонструють можливості фреймворку.

Gauge досить молодий фреймворк, який все ще знаходиться на стадії beta-тестування, саме тому його популярність на ринку невелика [10].

2.3 Шаблон проектування Page Object

Page Object - це шаблон проектування, найважливішою ідеєю якого є упаковка сторінки та її елементів у відповідні класи та їх поля. Важливо, що, на відміну від загальновизнаної номенклатури, класом не обов’язково повинна бути ціла сторінка - на практиці частіше трапляється, що це конкретна функціональна частина сторінки, наприклад панель або список результатів пошуку. Елементи сторінки починають відповідати полям класу, а дії, які можна виконувати з ними, починають відповідати методам цього класу. Таким чином створюється своєрідний API (інтерфейс програми), що дозволяє проводити тести для виконання дій на сторінці без посилання на шляхи до елементів. Це спрощує роботу з автоматизації та покращує читабельність самих методів тестування.

Основне правило для об’єкта сторінки полягає в тому, що він повинен дозволяти клієнту програмного забезпечення робити що-небудь і бачити все, що може людина. Він також повинен забезпечувати інтерфейс, який легко програмувати, і приховує основний віджет у вікні. Об'єкт сторінки повинен інкапсулювати механіку, необхідну для пошуку та обробки даних у самому елементі керування графічним інтерфейсом.

У типовому випадку код ділиться на класи, які обертають перевірені сторінки (наприклад, у папку з іменем Pages) і на тестові сценарії (наприклад, у папку Tests). Тести використовують об'єкти класу в папці Pages для виконання кожного кроку на сторінці. Що стосується тверджень, то є прихильники їх тестування, а також класів сторінок. Прихильники обох рішень мають свої аргументи, але найголовніше, здається, постійно дотримуватися обраного варіанту і не поєднувати два підходи.

Page Object в Selenium. Паттерн Page Object в Selenium реалізований за допомогою бібліотеки Page Factory і класу сторінки. Page Factory ініціалізує кожну змінну.

Класи веб-сторінок або об'єкти сторінок, що містять веб-елементи, потрібно ініціалізувати за допомогою Page Factory, перш ніж можна використовувати змінні веб-елементів. Це може бути зроблено просто шляхом використання initElements функціонувати на PageFactory:

```
PageFactory.InitElements(driver, page);
```

WebElement із посиланням на відповідний елемент на фактичній веб-сторінці на основі налаштованих «локаторів». Це робиться за допомогою анотацій @FindBy. Кожного разу, коли метод викликається за цією змінною WebElement , драйвер спочатку знаходить його на поточній сторінці, а потім моделює взаємодію [11].

Page Object дозволяє розробляти зручні, структуровані рішення для автоматизації тестування для проектів різного рівня складності.

2.4 Проектування об'єктної моделі фреймворка

Для створення об'єктної моделі фреймворка була вибрана мова UML (Unified Modeling Language).

Дана мова дозволяє у вигляді стандартизованих креслень і схем візуалізувати, специфікувати, конструювати і документувати програмні системи.

Мова придатна для складання моделей систем будь-якого масштабу та напрямку, його застосовують при моделюванні інформаційних системи великих і малих підприємств, Web-додатків, вбудованих систем реального часу [12].

Суб'єкт - це класифікатор (включаючи підсистему, компонент або навіть клас), що представляє бізнес, програмну систему, фізичну систему чи пристрій, що аналізується, розробляється або розглядається, має певну поведінку та застосовується до набору випадків використання.

Прецеденти (use case) - це детальний опис послідовностей дій (в тому числі і їх варіанти), що виконуються системою для досягнення актором певного конкрентого результату. Діаграма прецедентів використання призначена для опису функціонального призначення системи. Діаграма прецедентів фреймворка автоматизації тестування наведена на рисунку 2.1.

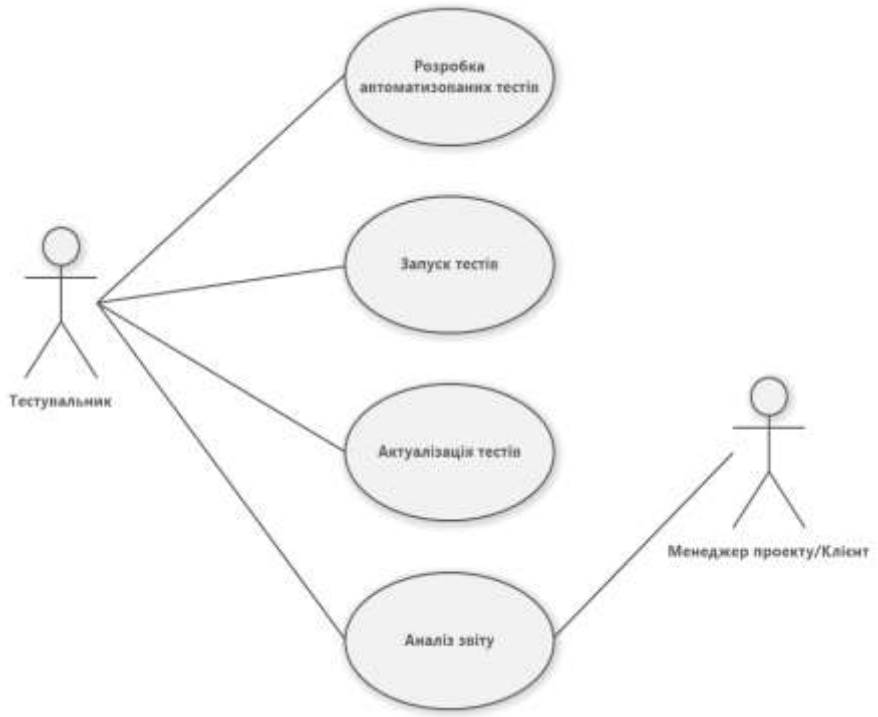


Рисунок 2.1. Діаграма прецедентів використання фреймворку автоматизації

Далі, перейдемо до розробки діаграми класів для фреймворка (Class Diagram). Клас - це основний складовий елемент інформаційної системи. Дане поняття є одним з основоположних в об'єктно мовах програмування. Відповідно, між класами в діаграмах UML і класами в програмному коді є відповідність, тому більшість засобів побудови діаграм UML дозволяють генерувати програмний код на обраною мовою програмування на основі діаграми класів. Кожен клас обов'язково має унікальну назву, а також атрибути і методи. Атрибутами називаються властивості класу, а методами - дії, які може виконувати об'єкт обраного класу, найчастіше методи класу призначені для зміни його властивостей. Абстрактна діаграма класів для розроблюваного фреймворку представлена на рисунку 2.2.

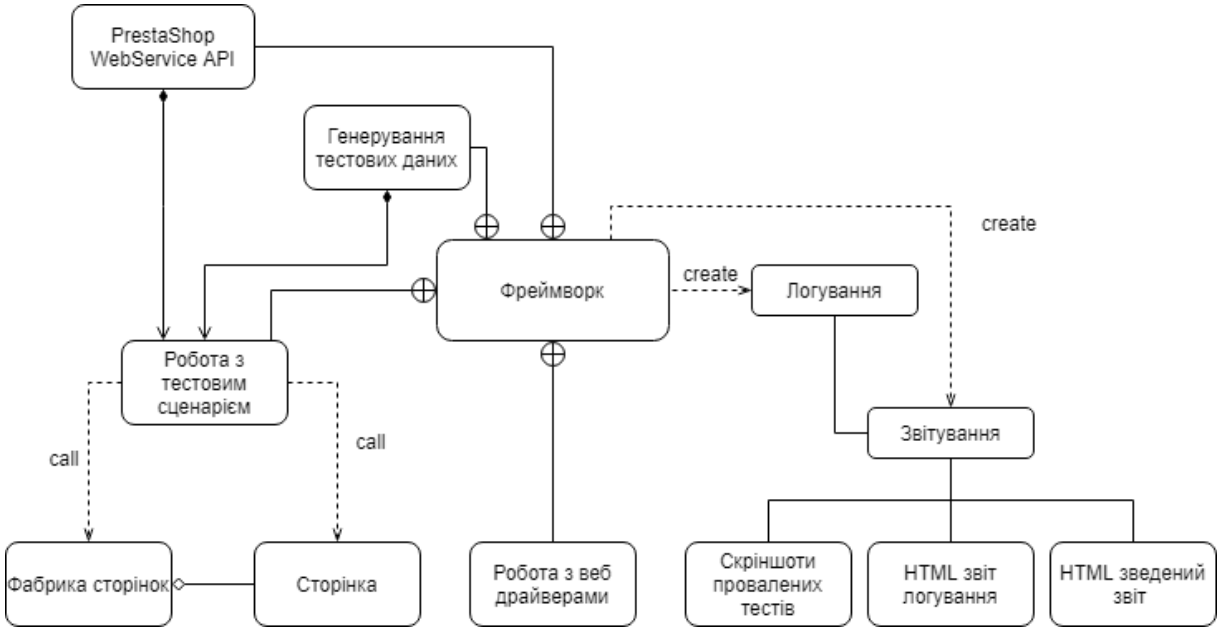


Рисунок 2.2. - Абстрактна діаграма класів фреймворку для автотестування

З даної діаграми видно, які сутності будуть розробляється у фреймворку і їх зв'язки. Більш точну UML діаграму буде представлено в процесі конструювання програмної системи.

2.5 Висновки до розділу

У цьому розділі було обґрунтовано доцільність розробки нового фреймворку тестування із врахування особливостей однієї з найпопулярніших CMS - PrestaShop. Також було проведено аналіз уже існуючих на ринку програмного забезпечення рішень та визначено їх основні переваги та недоліки. На основі проаналізованих даних було визначено основні функціональні елементи системи та запроектовано абстрактну діаграму класів програмного продукти, яки буде відповідати усім, раніше визначеним, вимогам.

3 КОНСТРУЮВАННЯ ФРЕЙМВОРКУ

3.1. Діаграма класів фреймворку

Розглянемо більш докладно кожен з класів, а також його поля і методи. Для цього класи розділені на основний функціонал і службовий функціонал. Основний функціонал є ядром діяльності фреймворка, а службовий функціонал відповідає за обслуговування основного, або за надання додаткових функцій для зручності розробки тестів [13].

Основний функціонал:

1. Фабрика сторінок (PageFactory) реалізована у фреймворку класом BasePagePR (див. рис. 3.1). Клас містить усі необхідні для роботи із веб сторінками поля та методи.

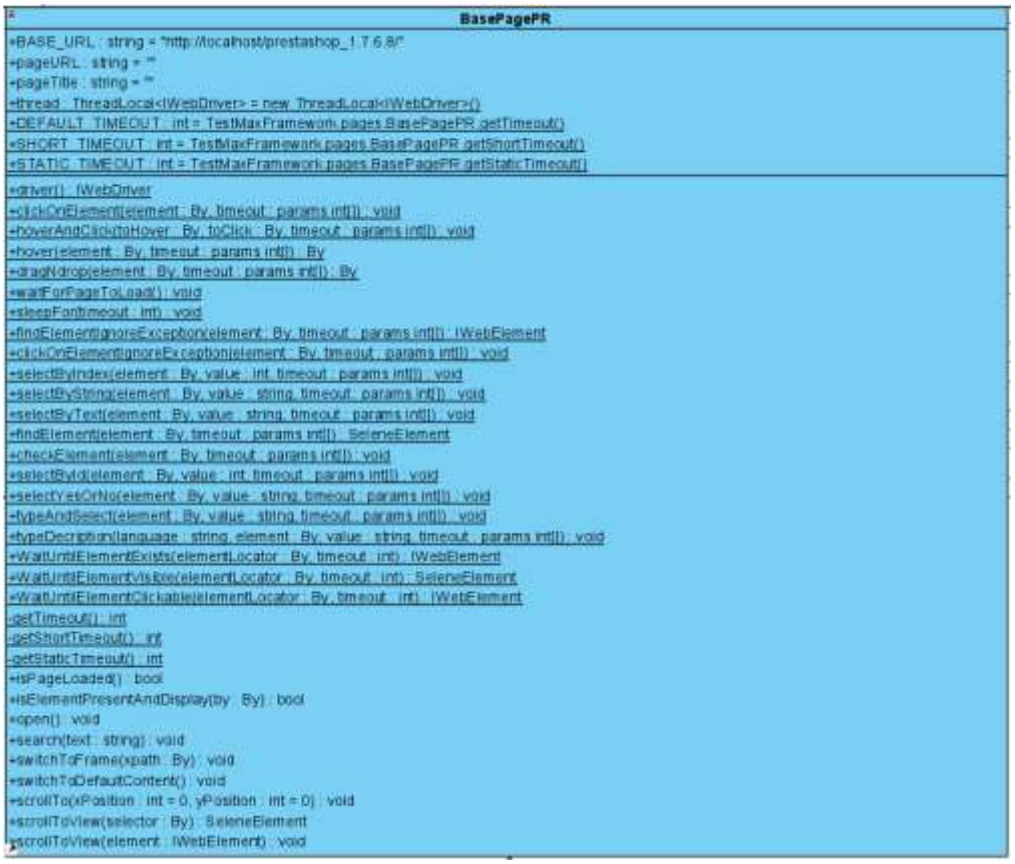


Рисунок 3.1 – UML діаграма класу BasePagePR

Фабрика сторінок є основним класом, так як на ньому заснована вся робота фреймворку. Він реалізований як проходження концепції PageObject, коли для представлення елементів і дій створюється клас конкретної сторінки (Page). BasePagePR є керуючим класом для пакетів сторінок(див. рис. 3.2).

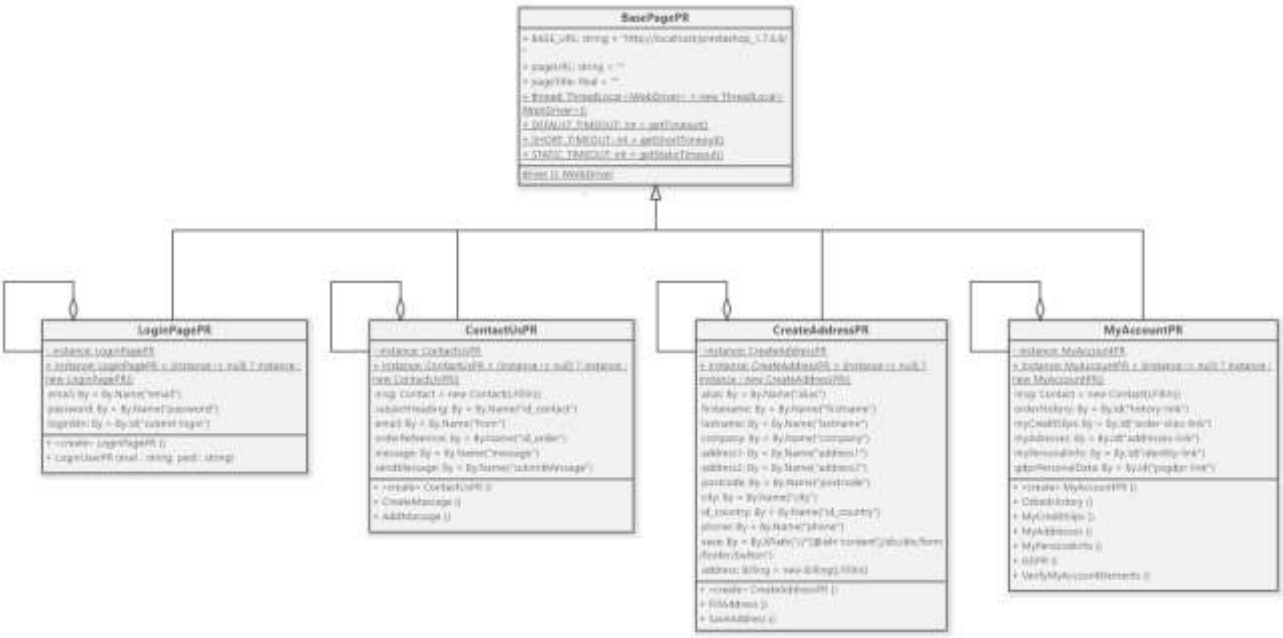


Рисунок 3.2 - UML діаграма фабрики сторінок

2. Ініціалізація драйверів у фреймворку відбуває класом DriverProvider (див. рис. 3.1), який дозволяє опридiлити версію операційної системи (Windows, Linux, MacOS) та ініціалізувати необхідний веб драйвер (IE, Google Chrome, Forefox).

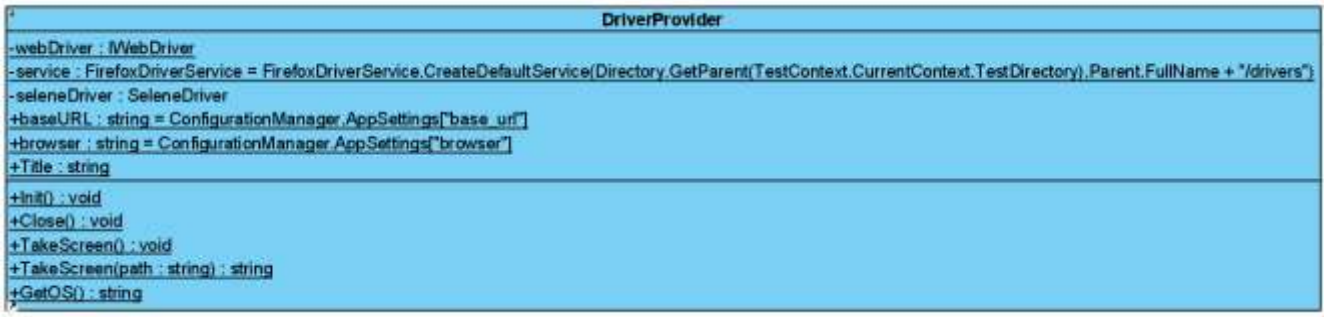


Рисунок 3.3 - UML діаграма класу DiverProvider

3. Генерування необхідних випадкових тестових даних відбувається з використанням пакету Bogus. На діаграмі представлено класи базових наборів даних, необхідних для тестування (див. рис. 3.4).

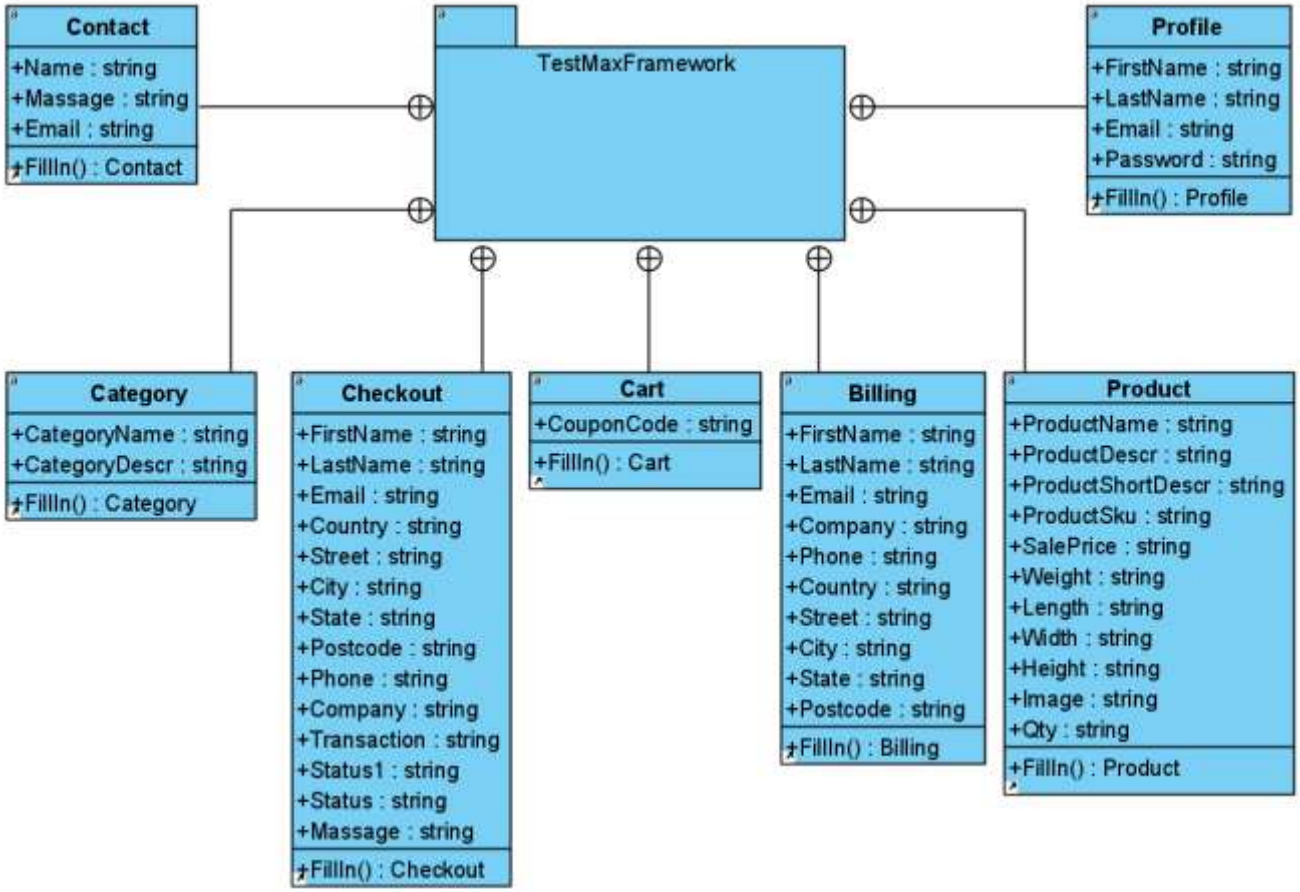


Рисунок 3.4 - UML діаграма класів генерування тестових даних

Вибір в сторону автоматичного генератора випадкових тестових даних, на відміну від завантаження попередньо підготовлених для тестування файлів, дозволяє уникнути “ефекту пестициду” у тестуванні, при якому тести “звикають” до однакового набору даних.

4. Генерування звітів та логування даних про проходження тестів у фреймворку реалізовано класом Report (див. рис. 3.5)., який дозволяє формувати журнал звітів та записувати дані про проходження тестів із створенням скріншотів невдало пройдених тестів.

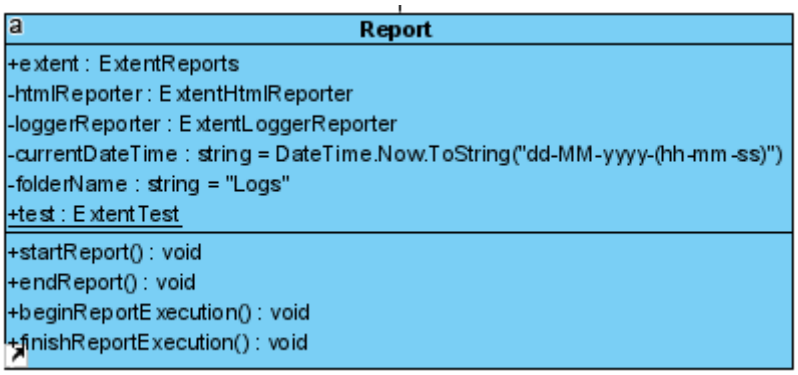


Рисунок 3.5 - UML діаграма класу Report

3.1.1 Характеристика результатної інформації

Звітна інформація, яка надходить за результатами роботи розробляється фреймворка, являє собою відомості про прогонах тестів, представлені в різних формах для різних категорій користувачів. Результати будуть представлятися в наступних формах:

1. Список. Логування є низькорівневі записи про порядок виконання інструкцій коду в процесі виконання певних методів. Логування здійснюється в консоль, або у вигляді запису в текстові файли.

2. Візуальний звіт про результати конкретних прогонів тестів у двох видах із різним рівнем деталізації логів. Візуальний звіт служить для виведення інформації для більш широкого кола користувачів, тому що не передбачає наявності у користувача глибоких знань про внутрішню структуру фреймворка. Даний звіт відображається, звичайно, в форматі інтерактивної веб-сторінки, результати прогонів конкретних тестів, тестових наборів, а також загальну статистику.

Загальна статистика включає: час виконання набору і кожного конкретного тесту, кількість успішно і неуспішно пройдених тестів, і опис успішно пройдених і неуспішно пройдених кроків для кожного конкретного тесту. Користувачами даної категорії результуючої інформації будуть усі учасники процесу розробки:

розробники тестів, тестувальники, особи які приймають рішення / менеджери, розробники тестованого ПО. На рисунку 3.6 представлено звіт про проведене тестування.

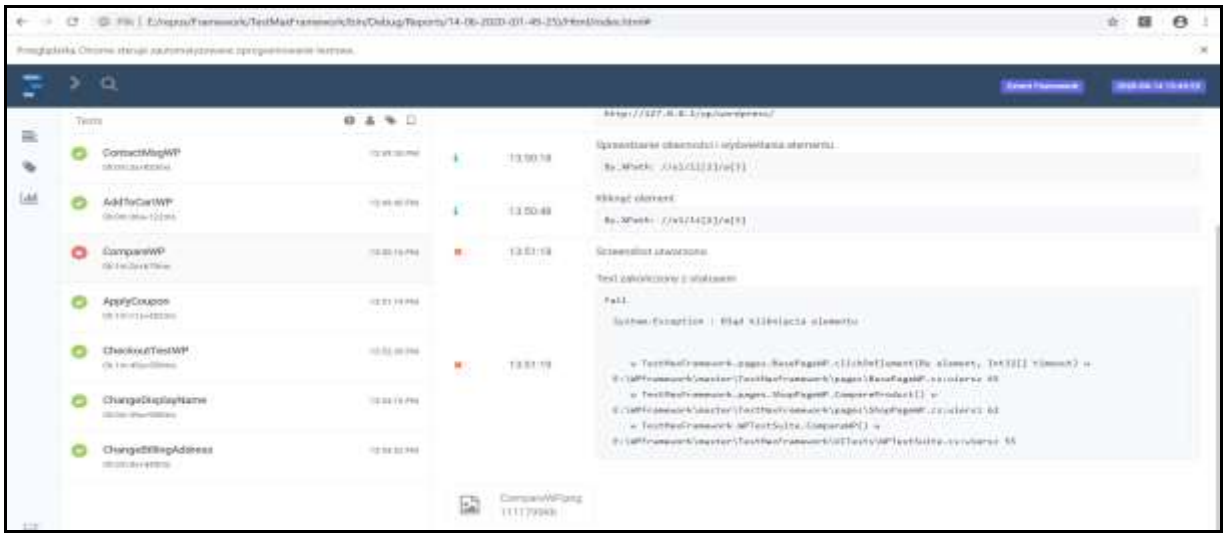


Рисунок 3.6 - Звіт про проведене тестування

3.1.2 Функціональна структура фреймворку

Функціональна структура фреймворку включає основні функції, такі як: створення об'єктів сторінок (Page), вчинення дій над ними в підключених браузерах, перевірки (Assertions) і ін. Також функціональна структура проекту включає службові внутрішні функції, такі як: управління веб-драйвером, формування звітних даних, зняття скріншотів. Більш детально, склад основного функціоналу:

1. Дії для роботи з веб-драйвером Selenium в рамках підходу PageFactory:

- ініціалізація веб-драйвера і зв'язок з конкретним браузером;

- отримання пакету класів сторінок, які будуть використані в поточних тестах

- отримання класу конкретної сторінки і ініціалізація елементів на ній;

- отримання інформації про оточення і поточних налаштуваннях;

- організація зберігання тестових даних;

- отримання інформації про стан браузера;

- налаштування параметрів браузера;

- відключення веб-драйвера і закриття вікна або вкладки браузера;

- очікування для різноманітних дій.

2. Функції для роботи з винятками:

- виключення ініціалізації браузера;

- виключення невідповідності версії браузера для конкретних тестів;

- виключення, пов'язані з пошуком і ініціалізацією конкретних елементів;

3. Робота з аспектами, тобто функції, які будуть виконуватися спільно з певними функціями, позначеними спеціальним шаблоном:

- аспекти для взаємодії на елементи;

- аспекти для роботи з винятками.

4. Функції для роботи з анотаціями, що представляють собою спеціальні позначки для методів і конкретних змінних об'єктів:

- анотації пошуку та ідентифікації веб-елементів;

- анотації для маркування окремих функцій для роботи зі сторінкою;

- анотації, що містять опис правил валідації для конкретних елементів.

Таким чином, функціонально проект абсолютно відповідає основним потребам користувачів, яким необхідно автоматизувати функціонал тестування веб-сайту інтернет магазину.

3.2 Використані бібліотеки та фреймворки

3.2.1 Selenium WebDriver

Selenium – це набір API з відкритим кодом, який використовується для автоматизації тестування веб-програми. Інструмент Selenium WebDriver застосовується для автоматизації тестування веб-додатків, щоб перевірити, чи

працює він, як очікувалося. Він має підтримку безлічі браузерів, таких як Firefox, Chrome, IE та Safari. Однак, використовуючи Selenium WebDriver, ми можемо автоматизувати тестування лише для веб-додатків. Він не підходить для десктоп додатків. Він також підтримує різні мови програмування, такі як C#, Java, Perl, PHP та Ruby для написання тестових сценаріїв. Selenium Webdriver (див. рис. 3.7) є незалежним від платформи, оскільки один і той же код може використовуватися в різних операційних системах, таких як Microsoft Windows, MacOS та Linux [14].

Рисунок 3.7 - Логотип Selenium WebDriver

Для знаходження потрібних елементів на веб-сторінці можна використати метод FindElement (або FindElements, якщо потрібно знайти декілька елементів), а також By, в якому задається спосіб пошуку елементу: через ідентифікатор елемента, по його імені, через XPath та CSS локатори.

3.2.2 Надбудова NSelene

NSelene це орієнтоване на користувача API для Selenium Webdriver для автоматизації мовою програмування C#. NSelene було створено за аналогією до Selene в Python та Selenide в Java.

Коли функціоналу драйвера не вистачає або він незручний і складний, над ним з’являється ще один рівень, який називається надбудовою. Надбудова – програма, яка взаємодіє з додатком через один або кілька драйверів, підвищуючи

зручність їх використання або розширюючи їх можливості. У надбудови можуть бути наступні функції:

- Модифікація поведінки (без зміни API). Наприклад, додаткове протоколювання, валідація даних, очікування виконання дії протягом певного часу.
- Підвищення зручності і / або рівня абстракції API через використання синтаксичного цукру – зручних назв функцій, коротших звернень до них, уніфікованого стилю написання тестів;
- Неявне управління драйвером, коли, наприклад, він ініціалізується автоматично, без необхідності прописувати кожну таку дію вручну;
- Спрощення складних команд на кшталт вибору події з календаря або роботи зі списками, які прокручуються;
- Реалізацію альтернативних стилів програмування, таких як процедурний стиль або fluent.

У розробці даного фреймворку NSelene значно спростив взаємодію із драйвером, проте його можливостей не завжди охоплювали всі необхідні операції, що можна віднести до недоліків даного інструменту.

3.2.3 Фреймворк NUnit

NUnit (див. рис. 3.8) - це фреймворк з відкритим кодом, призначений для написання та запуску тестів на мовах програмування Microsoft .NET . NUnit, як і JUnit , є аспектом тестової розробки (TDD), яка є частиною більшої парадигми дизайну програмного забезпечення, відомої як Extreme Programming (XP).

Рисунок 3.8 - Логотип NUnit

NUnit має графічний інтерфейс користувача (GUI), подібний до того, що використовується в JUnit. Тести можна проводити безперервно. Результати надаються негайно. Одночасно можна проводити кілька тестів. Не потрібно суб’єктивних людських суджень чи інтерпретацій результатів тестів. Простота фреймворку дозволяє легко виправити помилки у міру їх виявлення. Поточна версія NUnit написана на C # , мові об'єктно-орієнтованого програмування (OOP), що поєднує в собі потужність C ++ і простоту Visual Basic . NUnit є однією з сімейства пов'язаних платформ тестування, відомих як xUnit [15].

3.2.4 Serilog

Бібліотека підтримує всі основні функції логування, які є у log4net, Nlog, і інших відомих бібліотек:

Кілька загальноприйнятих типів запису:

- Verbose - найбільш низькорівневе і детальне логування (наприклад, прийшли аргументи в метод)
- Debug — дані для налагодження коду, на один рівень вище Verbose (наприклад, який метод запускали, і результат виконання)
- Warning — попередження для бізнес-процесу, не повинно містити debug дані (наприклад, запустили розрахунок зарплат)
- Error — помилка в додатку, яке не очікували
- Fatal — виняткова помилка зупиняє бізнес процеси додатки (наприклад, перенаправили користувача в PayPal і оплата покупця не дорівнює очікуваній сумі).

Різні типи зберігання, званих в Serilog стоком: текстовий файл, реляційні БД, NoSQL БД, Windows Events, HTTP (Hypertext Transfer Protocol) запити, і т. д. Зручна конфігурація через код, так і через .config файли.

Особливості:

- Ведення API (Application Programming Interface) журналу на основі формату з відомими рівнями, такими як налагодження, інформація, попередження, помилка і т. д.
- Синтаксис конфігурації C # і додаткова підтримка конфігурації XML (eXtensible Markup Language) або JSON (JavaScript Object Notation)
- Ефективність при включенні, надзвичайно низькі накладні витрати при відключенні рівня ведення журналу.
- Краща в своєму класі підтримка. Net Core, включаючи постачальника Microsoft.Extensions.Logging.
- Підтримка широкого спектру приймачів, включаючи файли, консоль, локальні і хмарні сервери журналів, бази даних і черги повідомлень.
- Складне збагачення журнальних подій контекстною інформацією, включаючи хмарні (LogContext) властивості, ідентифікатори потоків і процесів, а також ідентифікатори кореляції домену, такі як HttpRequestId.
- Об'єкти Logger з нульовим загальним станом з необов'язковим глобальним статичним класом Log.
- Формат-агностичний конвеєр протоколювання, який може генерувати події у відкритому тексті, JSON (JavaScript Object Notation), в об'єктах LogEvent в пам'яті (включаючи rx-Конвеєри) та інших форматах.

Логотип бібліотеки зображено на рисунку 3.9.

Рисунок 3.9 - Логотип Serilog

3.2.5 ExtentReports

Бібліотека ExtentReports (див. рис. 3.10) дозволяє легко вибирати ваші репортери, будь то HTML, електронна пошта або передача даних в MongoDB для Klov.

Рисунок 3.10 - Логотип ExtentReports

ExtentReports дозволяє індивідуально налаштувати кожен шаблон звіту, вказавши користувальницький CSS і / або JavaScript через конфігураційний XML або безпосередньо з коду.

Можливості бібліотеки:

- За допомогою бібліотеки ExtentReports можна створювати красиві, інтерактивні та докладні звіти для своїх тестів. Бібліотека дозволяє додавати події, скріншоти, теги, пристрої, авторів або будь-яку іншу релевантну інформацію, яка, на вашу думку, важлива для створення описового звіту, який можна повністю контролювати.
- Пошук по всіх збірках. Пошук тестів за їх назвою, статусом, датою виконання, атрибутами (тегами, авторам, пристроям), за виникаючими винятками, на різних вкладених рівнях або за ключовими словами в журналах або подіях.
- Атрибути тесту. Тег, пристрої, автори, винятки - кожна категорія дозволяє присвоювати тестам відповідні атрибути. Можна візуалізувати кожну категорію на панелі моніторингу і навіть відстежувати всі їх запуски і тести в одному поданні.
- Керувати декількома проектами. Нові проекти можуть бути створені з бібліотеки ExtentReports при виконанні тестів, в той час як збірки продовжують агрегувати існуючі проекти.

3.2.6 PrestaShop Web Service

PrestaShop надає доступ стороннім інструментам до бази даних свого магазину через CRUD API, інакше названою веб-службою. PrestaShop Web Service використовує архітектуру REST, щоб бути доступною на якомога більшій кількості платформ, оскільки протокол HTTP та файли XML зрозумілі більшості платформ, якщо не всім.

CRUD - це скорочення, що розшифровується як "створення, читання, оновлення та видалення". Це чотири основні операції з управління даними в програмі.

REST приблизно визначає стиль архітектури програмного забезпечення, який сприяє використанню методів HTTP під час створення веб-додатків, замість користувацьких методів або протоколів, таких як SOAP або WSDL. У ньому визначено кілька правил, у тому числі подібне до CRUD, яке описано нижче.

HTTP має кілька методів, які можуть виконувати обробку даних, як визначено в архітектурі REST, серед яких 4 основні методи, представлені у таблиці 3.1

Таблиця 3.1 - Методи обробки даних

HTTP/REST	CRUD	SQL
POST	Create	INSERT
GET	Read	SELECT
PUT	Update	UPDATE
DELETE	Delete	DELETE

Розглянемо більш детально схеми процесів, які виконуються за допомогою PrestaShop Web Service.

Створення сутності. Процес створення трохи складніший, ніж звичайне зчитування деяких даних з API, головним чином тому, що ми рідко управляємо даними через формат XML. У більшості випадків використання, користувачеві надається зрозумілий для людини компонент, такий як форма, а введені дані потім обробляються. Крім того, потрібно переконатися, що XML, надісланий веб-службі, є зрозумілим та повним.

Для створення використовується порожня схема, яка є порожнім поданням ресурсу. Цей порожній XML буде заповнений нашими даними, а потім відправлений до веб-служби за допомогою методу add ().На рисунку 3.11 зображено UML діаграму повного процесу створення сутності.

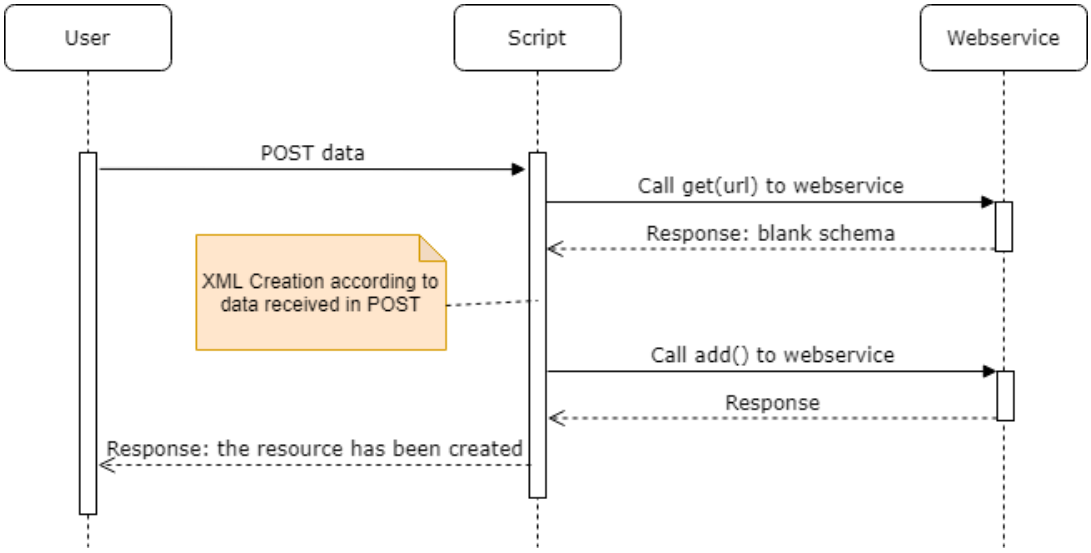


Рисунок 3.11 - UML діаграма діяльності процесу створення сутності

Процес оновлення схожий на процес створення, проте головна відмінність полягає в тому, що початковий вхід - це не порожній XML, а вже існуючий, тому ми використовуємо метод get () для отримання попередньо заповненого XML, а потім буде можливість оновити його поля . На рисунку 3.12 зображено UML діаграму повного процесу редагування сутності.

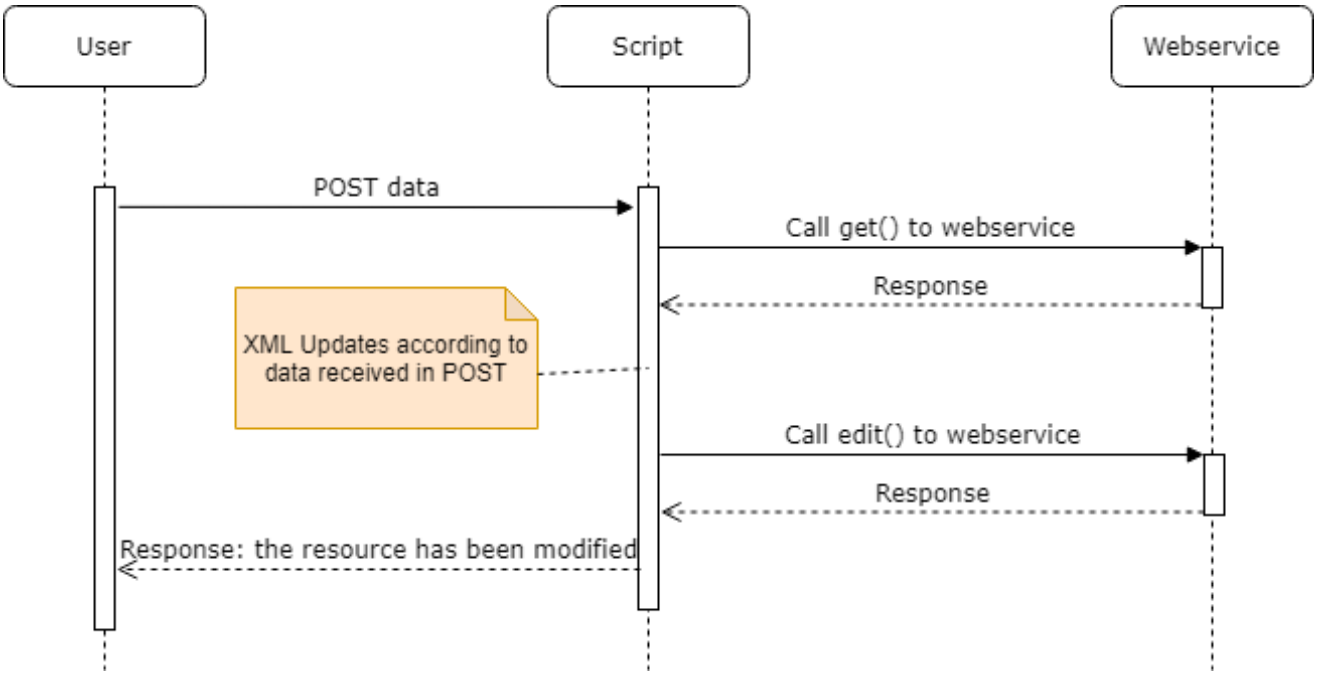


Рисунок 3.12 - UML діаграма діяльності процесу редагування сутності

3.3 Опис типових схем використання системи

Розроблюваний фреймворк призначений для тестування сайтів електронної комерції, зокрема інтернет-магазинів, саме тому особливістю цього проекту є оптимізація тестування звичних сценаріїв процесу продажу продуктів онлайн. Фреймворк розроблений з урахуванням особливостей платформи PrestaShop, тому включає в себе методи тестування функцій, які надає ця платформа.

Блок схема роботи інтернет-магазину, зображена на додатку А, відображає повний процес взаємодії клієнта із сайтом та представляє задачі, як набір елементів, операцій, потоків та даних, з яким працює фреймворк.

При створенні тестових сценаріїв необхідно виділити послідовності діяльностей в інтернет магазині. Як приклад, на рисунку 3.13 зображено діаграму послідовності покупки товару. На основі цієї послідовності визначаються передумови, кроки та дані необхідні для створення тесту. З використанням

розробленого фреймворку цей процес стає значно швидшим та зручнішим, а головне простішим у подальшій актуалізації та підтримці розроблених тестових сценаріїв.

Рисунок 3.13 - Діаграма діяльності покупки товару

3.4 Розробка тестових сценаріїв

На прикладі одного із тестових сценаріїв розглянемо процес творення та тестового сценарію та методи які використовуються для виконання обраного тесту. Для прикладу було обрано тест перевірки видалення категорії з адмін панелі. Тестови скрипт даного сценарію виглядає наступним чином:

```
[Test]
            public void CheckDeleteCategory()
        {
            // Передумови
            prestaApi.CreateCategory();

            // Кроки
            AdminCategoryPage          category          =
AdminCategoryPage.Instance;
            AdminDashboard dash = AdminDashboard.Instance;
            dash.open();
            dash.GoToCategories();
            category.DeleteCategory(1);
        }
```

Для виконання даного тесту необхідно виконати передумови, а саме, створити категорію, яку буде видалено у процесі виконання тесту. Для цього використовується метод створення категорії засобами API PrestaShop WebService:

```
prestaApi.CreateCategory();
```

У блоці кроків виконання тестового сценарію спочатку відбувається ініціалізація сторінок Dashboard та сторінки категорій в адмін панелі.

```
AdminCategoryPage category =
AdminCategoryPage.Instance;
AdminDashboard dash = AdminDashboard.Instance;
```

Після ініціалізації, відкривається сторінка Dashboard та відбувається перехід до сторінки категорій не через URL сторінки, а через UI - так як буде робити звичайний користувач.

```
dash.open();
dash.GoToCategories();
```

На сторінці категорій виконується видалення категорії із списку за вказаним параметром. У випадку даного сценарію, було передано параметр “1”, тому буде видалено першу категорію у списку - ту котра була створена в передумовах тесту.

```
category.DeleteCategory(1);
```

Результатом виконання тесту буде успішно видалена категорія.

3.5 Висновки до розділу

На базі спроектованої раніше абстрактної діаграми класів, було запроектовано і сконструйовано класи системи та їх взаємодія. Для конструювання фреймворку автоматизації тестування було використаного найпопулярніші та найбільш підходящі бібліотеки та інструменти, які дозволили зробити фреймворк багатофункціональним повноцінним інструментом тестування, який забезпечує зручну взаємодію із користувачем та дозволяє оптимізувати процес тестування в цілому. Також було розроблено та, на прикладі одного із тестових сценаріїв, продемонстровано роботу тестів, пов'язаних з основними бізнес процесами інтернет магазинів, для тестування яких і розробляється даний фреймворк.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1 Охорона праці

Оскільки, розробка програмного забезпечення автоматизації тестування передбачає використання комп'ютерної техніки, то виконавець такого процесу зобов'язаний забезпечити оптимальні умови праці з охорони праці і техніки безпеки. Зазвичай, в якості виконавця виступає колектив працівників підприємства, тому саме на керівництво цього підприємства покладається відповідальність за забезпечення безпечних умов праці.

Існує широкий перелік нормативно-правових актів, що займаються регулювання даного питання. Обов'язки роботодавця, що стосуються забезпечення працівникам безпечних та комфортних умов праці, а також права працівників на такі умови передбачено частиною 2 ст. 2 та ч. 1 ст. 21 КЗпП, а також ст. 13 Закону України «Про охорону праці».

У Кодексі законів про працю України записано, що обов'язком роботодавця або уповноваженого ним органу є проінструктувати працівника і визначити йому робоче місце. Перед початком роботи за укладеним договором, власник або уповноважений ним орган повинен проінструктувати працівника з техніки безпеки, гігієни праці, виробничої санітарії, і протипожежної охорони відповідно до Статті 153. "Створення безпечних і нешкідливих умов праці" [19].

На всіх підприємствах, в організаціях та установах повинні створюються безпечні і нешкідливі умови праці. Умови праці на робочому місці, санітарно-побутові умови, стан засобів колективного та індивідуального захисту, що використовуються працівником повинні відповідати вимогам нормативних актів про охорону праці.

На підприємстві повинні впроваджуватися сучасні засоби техніки безпеки, для запобігання виробничому травматизмові, і забезпечувати

санітарно-гігієнічні умови, що дозволяють запобігати виникненню професійних захворювань працівників. Власник не має права вимагати від працівника виконання роботи, яка може загрожувати життю людини, а також виконувати свою роботу в умовах, що не відповідають законодавству про охорону праці. Працівник може відмовитися від дорученої роботи, якщо виникла виробнича ситуація, небезпечна для його життя чи здоров'я або життя і здоров'я оточуючих людей, навколишнього середовища, тощо.

Також на підприємстві мають систематично проводитися інструктажі працівників з питань охорони праці та протипожежної охорони.

Власник або уповноважений ним орган зобов'язаний вживати заходів щодо полегшення і оздоровлення умов праці працівників шляхом впровадження прогресивних технологій, вимог ергономіки, позитивного досвіду з охорони праці, зниження та усунення запиленості та загазованості повітря у приміщеннях, зниження інтенсивності шуму, вібрації та випромінювань.

Однак, на працівника також покладається ряд зобов'язань щодо дотримання вимог до охорони праці на підприємстві, за порушення яких працівник може понести дисциплінарну, адміністративну, матеріальну чи кримінальну відповідальність [19].

Головними з причин виникнення професійних захворювань користувачів комп'ютерної техніки є вплив інфразвуку та вібрації.

Вібрація - це механічні коливання, що призводять до розладу життєвих функцій людини, шкідливо впливають на роботу обладнання та руйнують будівельні конструкції. Коливання тіл з частотою до 16 Гц організм сприймає як вібрацію, а коливання з частотою 16–20 Гц і більше – одночасно як вібрацію і звук.

Залежно від способу передачі вібрації тілу людини розрізняють:

- локальну (місцеву), що передається людині через руки;
- загальну, що передається на тіло людини через опорні поверхні тіла.

Довготривалий вплив на людину загальної вібрації призводить до розладу вестибулярного апарату, центральної та вегетативної нервових систем, захворювання органів травлення, а також серцево-судинної системи.

Місцева вібрація викликає порушення периферійного кровообігу і нервової системи та м'язово-суглобного апарату. Тривала дія локальних вібрацій часто призводить до вібраційної хвороби з незворотними змінами в цих системах. Профілактика впливу вібрацій на організм людини включає ряд заходів технічного, санітарно-гігієнічного та лікувального характеру. Найкращим захистом є дотримання нормативних параметрів інтенсивності вібрації.

Допустимі рівні вібрації передбачають допустимі значення коливальної швидкості, що передається на руки безпосереднім контактом із вібруючою поверхнею [20].

Інфразвук - це механічні коливання пружного середовища, що мають однакову із шумом фізичну природу, але різняться частотою коливань, яка не перевищує 20 Гц. У повітрі інфразвук поглинається незначно. У зв'язку з цим він здатний поширюватися на великі відстані.

У виробничих умовах інфразвук утворюється при роботі тихохідних електричних та механічних приводів комп’ютерів, що здійснюють обертальні або зворотно-поступальні рухи з повторним циклом до 20 разів за секунду.

Низькочастотні коливання з рівнем інфразвукового тиску, що перевищує 150 дБ, людина не в змозі перенести. Особливо несприятливі наслідки викликають інфразвукові коливання з частотою 2...15 Гц у зв'язку з виникненням резонансних явищ в організмі людини. Особливо небезпечною є частота 7 Гц, тому що вона може збігатися з α-ритмом біотоків мозку.

У відповідності до санітарних норм, рівні звукового тиску інфразвуку в октавних смугах із середньогеометричними частотами 2; 4; 8 та 16 Гц не повинні перевищувати 105 дБ, а в діапазоні частот 32 Гц - 102 дБ.

Джерелами шуму під час роботи з ПК є жорсткий диск, вентилятор блока живлення мережі, вентилятор, розташований на процесорі, швидкісні CD-ROM, механічні сканери, пересувні механічні частини принтера. Під час роботи

матричних голчастих принтерів шум виникає внаслідок переміщення головки принтера і в процесі удару голок головки по паперу. У разі роботи вентиляційної системи ПК, яка забезпечує оптимальний температурний режим електронних блоків, виникає аеродинамічний шум. Крім того, діють й інші зовнішні джерела шуму, не пов'язані з роботою ПК.

Зниження рівня шуму в приміщенні можна досягти так:

– використанням блоків живлення ПК з вентиляторами на гумових підвісках;

– використанням ПК, у яких термодавачі вмонтовані в блоці живлення та в критичних точках материнської плати (процесор, мікросхеми плати), які дають змогу програмним шляхом регулювати як моменти ввімкнення вентиляторів, так і швидкість їх обертання;

– переведенням жорсткого диска в неробочий режим, якщо комп'ютер не працює протягом визначеного часу;

– використанням ПК, у яких вентилятор на процесорі встановлено виробником (BOX-процесор);

– застосуванням материнських плат формату ATX та ATXкорпусів, що дає змогу регулювати автономну швидкість та моменти часу відмикання вентилятора блока живлення від електромережі;

– використанням менш швидкісних CD-ROM;

– заміною матричних голчастих принтерів струменевими і лазерними принтерами, які забезпечують під час роботи значно менший рівень звукового тиску;

– застосуванням принтерів колективного користування, розташованих на значній відстані від більшості робочих місць користувачів ПК;

– зменшенням шуму на шляху його поширення через розміщення звукоізолювального відгородження у вигляді стін, перетинок, кабін;

– акустичною обробкою приміщень – зменшення енергії відбитих звукових хвиль шляхом збільшення площі звукопоглинання (розміщення на поверхнях

приміщення облицювань, що поглинають звук, розташування в приміщеннях штучних поглиначів звуку) [21].

У цьому підрозділі було розглянуто основні вимоги до охорони праці, яких повинні дотримуватися роботодавець та працівник. Також було визначено який вплив на людський організм мають інфразвук та вібрації у різних своїх варіація та поради щодо зниження рівня шуму та вібрації в приміщення. Додаково було розглянуто різні джерела шуму для користувачів ПК, їх вплив та способи нівелювання їхньої негативної дії на організм.

4.2 Безпека в надзвичайних ситуаціях

Використання електроенергії на підприємстві потребує правильного поводження з нею, оскільки порушення правил електробезпеки може призвести до важкої і навіть смертельної травми.

Електробезпека – це система організаційних і технічних заходів і засобів, які забезпечують захист людей від шкідливої і небезпечної дії струму, електричної дуги, електромагнітного поля і статичної електрики.

Електричний струм, який проходить через організм людини, спричинює термічну, електролітичну, біологічну і механічну дії.

Термічна дія електричного струму призводить до опіків шкіри, нагрівання до високої температури кров’яних судин, нервів, серця, мозку та інших органів, які є на шляху струму, і зумовлює в них серйозні функційні розлади. Вона може спричинити руйнування тканин аж до їхнього обвуглення.

Електролітична дія електричного струму виявляється в електролізі (розкладі) рідин, у тому числі крові, що спричинює зміну їхнього фізи- ко-хімічного складу й органів у цілому, а також суттєво змінює функційний склад клітин.

Механічна дія електричного струму виявляється в розшаруванні тканин і навіть у відриві частин тіла. Дія електричного струму призводить до місцевих електротравм і загальних уражень – електроударів.

Електричні травми – це чітко виражені місцеві ушкодження тканин і органів людини, які виникають унаслідок дії електричного струму і від електричної дуги. Їх виліковують, і працездатність людини відновлюється повністю або частково.

З метою запобігання ушкодженням, що можуть виникнути з причин ураження електричним струмом, загоряння, коротке замикання тощо, розроблено загальний стандарт безпеки IEC 950. Стандартом електробезпечності для країн Європейської співдружності є Cemark. Протягом проектування систем електропостачання, монтажу силового електрообладнання та електричного освітлення будівель та приміщень для ПЕОМ потрібно дотримуватись вимог вищеназваних нормативно-правових актів, а також СН 357-77 "Инструкция по проектированию силового осветительного оборудования промышленных предприятий", затверджених Держбудом СРСР, ГОСТу 12.1.006, ГОСТу 12.1.030 "ССБТ. Электробезопасность. Защитное заземление, зануление", ГОСТу 12.1.019 "ССБТ. Электробезопасность. Общие требования и номенклатура видов защиты", ГОСТу 12.1.045, ВСН 59-88 Держкомархітектури СРСР "Электрооборудование жилых и общественных зданий. Нормы проектирования", Правил пожежної безпеки в Україні, ДСанПіН 3.3.2.007-98, розділів СНиП, що стосуються штучного освітлення і електротехнічних пристроїв, та вимог нормативно-технічної і експлуатаційної документації заводу-виробника ПЕОМ.

ЕОМ, периферійні пристрої ЕОМ та устаткування для обслуговування, ремонту та налагодження ЕОМ, інше устаткування (апарати управління, контрольно-вимірювальні прилади, світильники тощо), електропроводи та кабелі за виконанням та ступенем захисту мають відповідати класу зони за ПУЕ, мати апаратуру захисту від струму короткого замикання та інших аварійних режимів. Під час монтажу та експлуатації ліній електромережі потрібно унеможливити виникнення електричного джерела загоряння з причин короткого замикання та

перевантаження проводів, зменшити застосування проводів з легкозаймистою ізоляцією і, по мірі можливості, перейти на негорючу ізоляцію.

Лінія електромережі для живлення ЕОМ, периферійних пристроїв ЕОМ і устаткування для обслуговування, ремонту і налагодження ЕОМ виконується як окрема групова трипровідна мережа, шляхом прокладання фазового, нульового робочого і нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів.

Використання нульового робочого провідника в якості нульового захисного провідника забороняється. Нульовий захисний провід прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток живлення. Не допустимим є підключення на щиті до одного контактного затискача нульового робочого і нульового захисного провідників. Площа перерізу нульового робочого і нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника.

Всі провідники мають відповідати номінальним параметрам мережі і навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму і типам апаратури захисту, вимогам ПУЕ. У приміщенні, де одночасно експлуатується або обслуговується більше п'яти персональних ЕОМ, на помітному і доступному місці встановлюється аварійний резервний вимикач, що може повністю вимкнути електричне живлення приміщення, крім освітлення.

ПЕОМ, периферійні пристрої ПЕОМ і устаткування для обслуговування, ремонту і налагодження ЕОМ мають підключатися до електромережі тільки з допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення. Штепсельні з'єднання і електророзетки крім контактів фазового і нульового робочого провідників мають мати спеціальні контакти для підключення нульового захисного провідника. Конструкція їх має бути такою, щоб приєднання нульового захисного провідника відбувалося раніше ніж приєднання фазового і нульового робочого провідників. Порядок роз'єднання при

відключенні має бути зворотним. Необхідно унеможливити з'єднання контактів фазових провідників з контактами нульового захисного провідника.

Неприпустимим є підключення ПЕОМ і периферійних пристроїв ПЕОМ до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв.

Електромережі штепсельних з'єднань та електророзеток для живлення ПЕОМ, периферійних пристроїв слід виконувати за магістральною схемою, по 3...6 з’єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 36 В за своєю конструкцією повинні відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В і мають бути пофарбовані в колір, який візуально значно відрізняється від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Індивідуальні та групові штепсельні з'єднання та електророзетки необхідно монтувати на негорючих або важкогорючих пластинах з урахуванням вимог ПУЕ та Правил пожежної безпеки в Україні.

Електромережу штепсельних розеток для живлення ПЕОМ, периферійних пристроїв ПЕОМ при розташуванні їх уздовж стін приміщення прокладають по підлозі поряд зі стінами приміщення, як правило, в металевих трубах і гнучких металевих рукавах з відводами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання.

При розташуванні в приміщенні за його периметром до 5 ПЕОМ, використанні трипровідникового захищеного проводу або кабелю в оболонці з негорючого або важкогорючого матеріалу дозволяється прокладання їх без металевих труб та гнучких металевих рукавів.

Електромережу штепсельних розеток для живлення ПЕОМ при розташуванні їх у центрі приміщення, прокладають у каналах або під знімною підлогою в металевих трубах або гнучких металевих рукавах. При цьому не дозволяється застосовувати провід і кабель в ізоляції з вулканізованої гуми та інші матеріали, що містять сірку. Відкрита прокладка кабелів під підлогою

забороняється. Металеві труби та гнучкі металеві рукави повинні бути заземлені. Заземлення повинно відповідати вимогам НПАОП 40.1-1.21-98.

Для підключення переносної електроапаратури застосовують гнучкі проводи в надійній ізоляції.

Тимчасова електропроводка від переносних приладів до джерел живлення виконується найкоротшим шляхом без заплутування проводів у конструкціях машин, приладів та меблях. Доточувати проводи можна тільки шляхом паяння з наступним старанним ізолюванням місць з'єднання.

Є неприпустимими:

• експлуатація кабелів та проводів з пошкодженою або такою, що втратила захисні властивості за час експлуатації, ізоляцією; залишення під напругою кабелів та проводів з неізольованими провідниками;

• застосування саморобних подовжувачів, які не відповідають вимогам ПВЕ до переносних електропроводок;

• застосування для опалення приміщення нестандартного (саморобного) електронагрівального обладнання або ламп розжарювання;

• користування пошкодженими розетками, розгалужувальними та з'єднувальними коробками, вимикачами та іншими електровиробами, а також лампами, скло яких має сліди затемнення або випинання;

• підвішування світильників безпосередньо на струмопровідних проводах, обгортання електроламп і світильників папером, тканиною та іншими горючими матеріалами, експлуатація їх зі знятими ковпаками (розсіювачами);

• використання електроапаратури та приладів в умовах, що не відповідають вказівкам (рекомендаціям) підприємств- виготовлювачів [22].

У цьому розділі було оприділено поняття електричного струму, його термічну, електролітичну, біологічну і механічну дії на організм людини. Також було розглянуто актуальні стандарти, які регулюють вимоги щодо електробезпеки. Додатково було висвітлено нормативно-правові акти щодо монтажу, обслуговування та експлуатації периферійних пристроїв ЕОМ, а також вимоги що стосуються прокладання та ремонту ектромереж.

ВИСНОВКИ

В даній роботі було досліджено актуальний ринок програмного забезпечення фреймворків автоматизації тестування, визначено слабкі та сильні сторони наявних продуктів. Для розробки нового програмного проекту було прийнято рішення використовувати платформу .NET з мовою програмування C#, в якості драйвера для взаємодії із браузерами - Selenium WebDriver в обгортці NSelene, а для генерування звітів та логування - Extent Reporting Framework та Serilog відповідно.

Було досліджено основні підходи до автоматизації тестування програмного забезпечення, визначено, що автоматизація тестування – актуальне та важливе питання, що стрімко розвивається в сучасному світі та має багато переваг, такі як виключення людського фактору при тестуванні програмного забезпечення; скорочення часових витрат на тестування; спрощення процесів формування звітності тощо.

В ході виконання дипломного проекту було запроектовано та сконструйовано фреймворку автоматизації тестування веб сайтів електронної комерції на базі платформи PrestaShop. Фреймворк розроблений на основі шаблону Page Object із використанням Data Driven та Model Driven підходів тестування.

Фреймворк дозволяє використовувати методи, побудовані на основі базових варіантів використання інтернет магазинів з врахуванням ролей користувачів. Фреймворк автоматично генерує html звіт про проведене тестування із детальним різнорівневим логуванням виконання тестів та скріншотами провалених тестів. Система включає інструменти тестування PrestaShop WebService API, що дозволяє проводити тестування на різних стадіях розробки інтернет магазинів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Why is software testing necessary? [Electronic resource] – Mode of access: http://tryqa.com/why-is-testing-necessary/ – Last access: 2020. – Title from the screen.
2. Software Testing Fundamentals – Test Plan [Electronic resource] – Mode of access: http://softwaretestingfundamentals.com/test-plan/ – Last access: 2020. – Title from the screen.
3. What is software testing - Definition [Electronic resource] – Mode of access: https://www.softwaretestingmaterial.com/software-testing/ – Last access: 2020. – Title from the screen.
4. ISO/IEC 9126-2001 Software engineering – Product quality (Part 1 – 4).
5. Dooley, J. Software Development and Professional Practice [Text] – 2011 – 193 p.
6. Definition of System Testing [Electronic resource] – Mode of access: https://economictimes.indiatimes.com/definition/system-testing – Last access: 2019. – Title from the screen.
7. What is software testing - Definition [Electronic resource] – Mode of access: https://www.softwaretestingmaterial.com/software-testing/ – Last access: 2020. – Title from the screen.
8. PrestaShop Developer Documentation [Electronic resource] – Mode of access: https://devdocs.prestashop.com/ – Last access: 2020. – Title from the screen.
9. Atata - C#/.NET Test Automation Framework [Electronic resource] – Mode of access: https://atata.io/ – Last access: 2020. – Title from the screen.
10. Gauge: Open Source Test Automation Framework [Electronic resource] – Mode of access: https://docs.gauge.org/– Last access: 2020. – Title from the screen.
11. PageObject pattern [Electronic resource] – Mode of access: https://www.swtestacademy.com/page-object-model-c/ – Last access: 2020. – Title from the screen.
12. Буч, Г. Язык UML. Руководство пользователя: Пер. с англ. Мухин Н./Г. Буч, Д. Рамбо, А. Якобсон – М. : ДМК Пресс, 2006. – 496 с.

13. Clapp, J.A.; Stanten, S.F.; Peng, W.W. Software Quality Control, Error Analysis, and Testing [Text] – 1995 – 192 p.

14. Selenium WebDriver [Electronic resource] – Режим доступу: https://economictimes.indiatimes.com/definition/selenium-web-driver– Last access: 2020. – Title from the screen.

15. Definition NUnit [Electronic resource] – Режим доступу: https://searchsoftwarequality.techtarget.com/definition/NUnit – Last access: 2020. – Title from the screen.

16. Serilog — simple .NET logging with fully-structured events [Electronic resource] – Mode of access: https://github.com/serilog/serilog/wiki/ – Last access: 2020. – Title from the screen.

17. Extentreports-core | Extent Framework [Electronic resource] – http://www.extentreports.com/docs/versions/5/net/index.html – Last access: 2020. – Title from the screen.

18. The PrestaShop Webservice API:: PrestaShop Developer Documentation [Electronic resource] – https://devdocs.prestashop.com/1.7/webservice/ – Last access: 2020. – Title from the screen.

19. Кодекс законів про працю України : закон України від 10.12.1971 № 322-VIII [Електронний ресурс]. – Режим доступу: http://zakon.rada.gov.ua/ laws/show/322-08

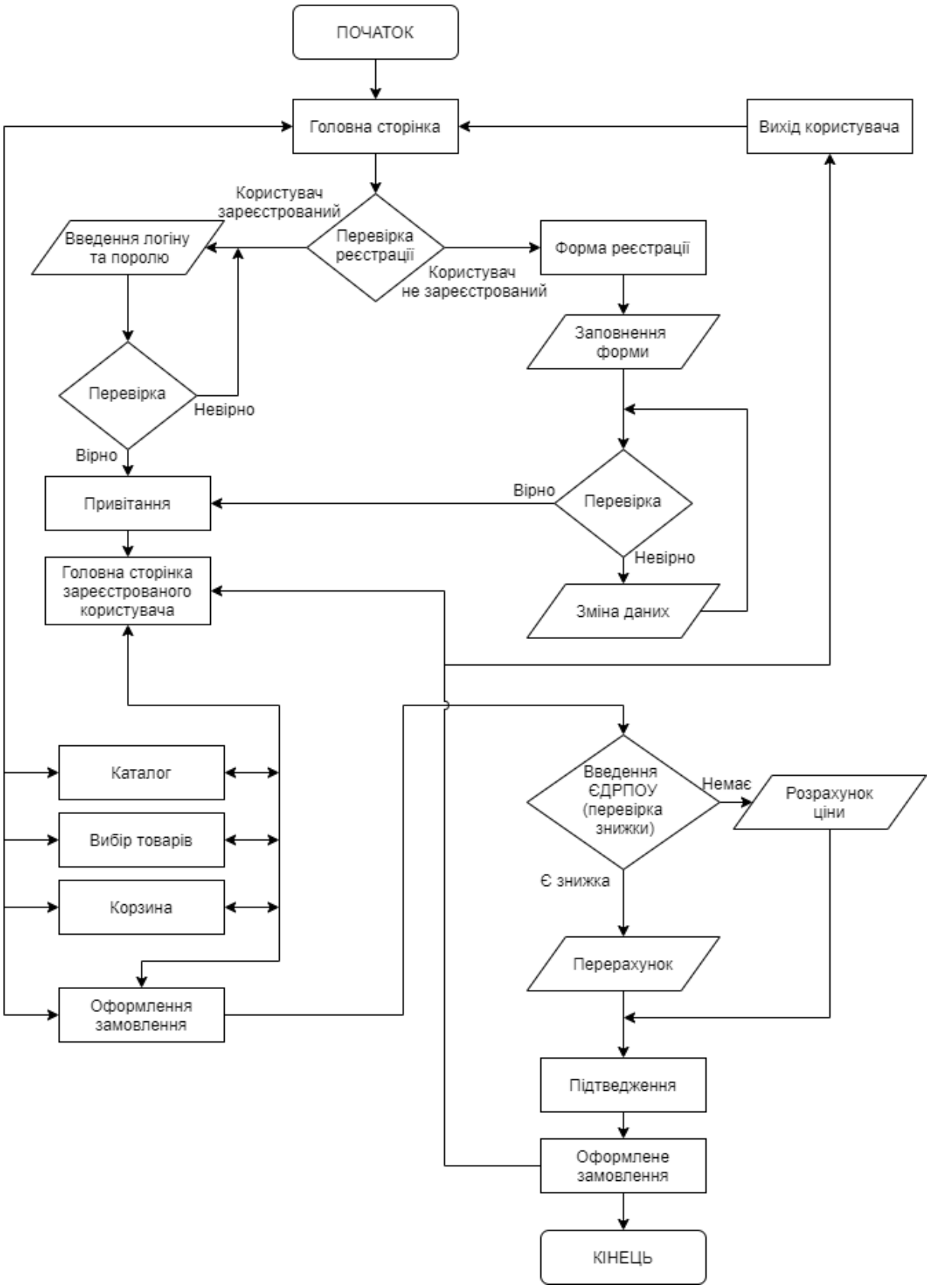
20. В.І. Голінько, М.Ю. Іконніков, Я.Я. Лебедєв Охорона праці в галузі інформаційних технологій [Текст] – Дніпропетровськ : НГУ – 2015. – 247 с.

21. Жидецький В.Ц. Охорона праці користувачів комп’ютерів. Львів: Афіша, 2000. 176 с.

22. Вимоги до електробезпеки у офісних приміщеннях [Електронний ресурс] – Режим доступу: https://cpo.stu.cn.ua/Oksana/posibnik/1140.html – Title from the screen.

ДОДАТОК А

БЛОК-СХЕМА РОБОТИ ІНТЕРНЕТ-МАГАЗИНУ


ПОЧАТОК
Головна сторінка
Вихід користувача
Користувач зареєстрований
Перевірка реєстрації
Введення логіну та поролю
Форма реєстрації
Користувач не зареєстрований
Заповнення форми
Перевірка
Невірно
Вірно
Привітання
Вірно
Перевірка
Невірно
Зміна даних
Головна сторінка зареєстрованого користувача
Каталог
Вибір товарів
Корзина
Оформлення замовлення
Введення ЄДРПОУ (перевірка знижки)
Немає
Розрахунок ціни
Є знижка
Перерахунок
Підтведження
Оформлене замовлення
КІНЕЦЬ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ
КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку проекту

«Розробка фреймворку автоматизації тестування веб-сервісів електронної комерції на базі C# та NSelene»

Розробники:
виконавець ст. гр. СПм-61
Дранівський М.І.

_____
(підпис)

керівник проекту
к.ф.-м.н., доцент Бойко І.В.

_____
(підпис)

Тернопіль 2020

ЗМІСТ

1. ПІДСТАВИ ДО РОЗРОБКИ

Розробка проводиться у відповідності до графіку навчального плану підготовки бакалаврів за спеціальністю 121 «Інженерія програмного забезпечення».

Тема проекту: «Розробка фреймворку автоматизації тестування веб-сервісів електронної комерції на базі C# та NSelene».

Термін виконання: до 30.11.2020 р.

2 ПРИЗНАЧЕННЯ ПРОГРАМНОГО ПРОЕКТУ

Фреймворк призначений для покращення процесу забезпечення якості а зокрема автоматизованого тестування веб сайтів електронної комерції.

Інформаційна система буде корисною в сфері якості програмного забезпечення.

Інформаційна система дозволить пришвидшити процес розробки автоматизованих тестів, скоротити та оптимізувати процес тестування API та базових UI функцій інтернет магазину.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОЕКТУ

3.1 Функціональні вимоги

Система повинна передбачати декілька рівнів логування даних про виконання тестів:

- інформаційний рівень
- рівень помилок
- рівень дебагінгу

Фреймворк повинен містити інструменти автоматичного генерування html звітів після завершення тестування.

Базова функціональність PrestaShop інтернет магазинів повинна буди імплементована у функціях фреймворку для легкого і швидкого написання тестових сценаріїв типу:

- додавання товарів до корзини
- перевірка чекаут форми
- перевірки форми зворотнього зв'язку
- перевірка створення замовлення
- перевірка відміни замовлення

3.2 Технічні вимоги

Вимоги до клієнтської частини: Windows 7 і вище, 2 Гб ОЗУ.

3.3 Програмні вимоги

Використання Selenium WebDriver, PrestaShop Web Service API
Розробка клієнтської частини: C#

4. ЕТАПИ РОЗРОБКИ

Розробка інформаційної системи проводиться в наступному порядку:

- аналіз предметної області, виявлення акторів та варіантів використання системи;
- вибір засобів розробки та архітектури системи;
- розробка програмного забезпечення системи;
- тестування інформаційної системи на реальних даних;
- оформлення супровідної документації;
- здача проекту.

Результати виконання кожного етапу проекту погоджуються з керівником проекту.

5. СУПРОВІДНА ДОКУМЕНТАЦІЯ

Для інформаційної системи повинні бути розроблені наступні документи:

- пояснювальна записка до проекту;
- презентація проекту;
- рецензія на проект;
- диск з проектом.

Пояснювальна записка до проекту оформляється згідно діючих вимог до нормоконтролю проектів.

6. ПОРЯДОК ЗДАЧІ ПРОЕКТУ

Розроблена інформаційна системи повинна відповідати вимогами, що складаються з перерахованих у п.3.1 цього документу характеристик.

Для здачі проекту необхідно підготувати весь перелік документів зазначений у п.5 цього документу.

Приймання проекту проводиться спеціально створеною комісією в термін зазначені в п.1 цього документу.

7. ВІДМІТКИ ПРО ВИКОНАННЯ ЕТАПІВ ТА ЗМІНИ В ПРОЕКТІ

Назва етапу	Відмітка*
Аналіз предметної області	
Проектування системи	
Розробка системи	
Використання системи	
Супровідна документація	

* відмітки про виконання етапу ставляться керівником проекту