Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

БАКАЛАВРСЬКА ДИПЛОМНА РОБОТА

на тему:

«Інформаційна мережева система тестування сайтів міжнародних готелів»

ПОЯСНЮВАЛЬНА ЗАПИСКА

Виконав: студент 4 курсу, групи
1КІ-18б
напряму підготовки (спеціальності)
123 – «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)
Щербань К.П.
(прізвище та ініціали)
Керівник Крупельницький Л.В.
(прізвище та ініціали)
Рецензент Лукічов В.В.
(прізвище та ініціали)

Допущено до захисту
д.т.н., проф. Азаров О.Д.

" 22 " червня 2022 р.

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

Освітньо-кваліфікаційний рівень бакалавр

Спеціальність 123 — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

обчислювальної техніки

проф. Азарову О.Д.

«08» 02 2022 р.

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Щербаню Костянтину Павловичу

1 Тема роботи: «Інформаційна мережева система тестування сайтів міжнародних готелів»

Керівник роботи к.т.н., доц. каф.ОТ Крупельницький Леонід Віталійович

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "24" 03 2022 року № 66

2 Строк подання студентом проекту (роботи) 22. 06. 2022р.

3 Вихідні дані до проекту (роботи) — системи тестування, технічний опис системи тестування.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз інформаційних мережевих систем,огляд інсуючих систем автоматизованого тестування та вибір засобів для розробки, реалізація інформаційної мережі системи тестування, висновки, перелік джерел посилання, додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): класифікація готелів за функціональним призначенням, приклад системи автоматизованого тестування, архітектура системи.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1 — Консультанти розділів таблиці

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1–3	к. т. н., доцент каф. ОТ Крупельницький Л.В.		

7 Дата видачі завдання 10.02.2022 року.

8 Календарний план приведені в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів проекту (роботи)	Примітка
1	Постановка задачі роботи	11.02.22	виконано
2	Огляд інформаційних джерел	12.02-28.02.22	виконано
3	Огляд і аналіз сучасного стану галузі	01.03-12.03.22	виконано
4	Проектування інформаційної мережевої системи	13.04.-17.04.22	виконано
5	Підготовка матеріалів та опис розробки	20.03-22.03.22	виконано
6	Оформлення пояснювальної записки та ілюстративного матеріалу	23.04-29.04.22	виконано
7	Аналіз виконання роботи, висновки, додатки	30.04-17.05.22	виконано
8	Перевірка якості виконання бакалаврського проекту та усунення недоліків	18.05.22	виконано

Студент (підпис) Щербань.К.П. (прізвище та ініціали)

Керівник роботи (підпис) Крупельницький Л.В. (прізвище та ініціали)

АНОТАЦІЯ

Щербань К.П. Інформаційна мережева система тестування сайтів міжнародних готелів. Бакалаврська робота зі спеціальності 123 — Комп'ютерна Інженерія, Вінниця: ВНТУ, 2022. Пояснювальна записка містить 67 сторінки, 16 рисунків та 19 посилань.

Мета виконання бакалаврської дипломної роботи – створення інформаційної мережевої системи тестування сайтів міжнародних готелів.
шляхом оцінки їх зручності та доступності.

Об'єкт - інтернет-технології, що застосовуються при створенні сайтів.

Предмет дослідження - засоби для розробки серверної частини розподіленої системи, методи та критерії їх варіантного аналізу.

Задачами дослідження є проведення варіантного аналізу засобів для розроби сайтів, на основі якої створено веб-додаток, що дозволяє бронювати номери у готелях. Практичне значення бакалаврської дипломної роботи у можливості використання її результатів для віддаленого бронювання номерів у потрібному готелі.

Ключові слова: веб-сервісу, замовник, виконавець .NET Core, React

ABSTRACT

Shcherban K.P. Information network system for testing the sites of international hotels. Bachelor's thesis in specialty 123 — Computer Engineering, Vinnytsia: VNTU, 2022. The explanatory note contains 67 pages, 16 figures and 19 references.

The purpose of the bachelor's thesis is to create an information network system for testing the sites of international hotels.

by assessing their convenience and accessibility.

Object - Internet technologies used in the creation of sites.

The subject of research - tools for the development of the server part of the distributed system, methods and criteria for their variant analysis.

The objectives of the study are to conduct a variant analysis of tools for site development, based on which a web application has been created that allows you to book rooms in hotels. The practical significance of the bachelor's thesis in the possibility of using its results for remote booking of rooms in the desired hotel.

Keywords: web service, customer, .NET Core artist, React

ЗМІСТ

ВСТУП

Інформаційні технології є однією з найбільш важливих складових соціально-культурного сервісу і туризму. Ефективність їх застосування багато в чому визначає продуктивність діяльності у сфері туристичного бізнесу, для якого надійність і оперативність збирання, оброблення і передавання інформації стають усе більш актуальними.

Успішна робота сучасної туристичної фірми або підприємства соціально-культурного сервісу неможлива без використання спеціалізованих програмних ресурсів з їх автоматизації, а також застосування глобальних і локальних комп'ютерних мереж. Тут центром формування, просування і реалізації туристичного продукту є туристичний офіс, міра автоматизації якого істотно впливає на якість продукту, його собівартість і визначає виживання туристичної фірми в ринковому висококонкурентному середовищі.

Мета виконання бакалаврської дипломної роботи — створення інформаційної мережевої системи тестування сайтів міжнародних готелів шляхом оцінки їх зручності та доступності.

Задачами дослідження є:

— реалізація інформаційної мережевої системи тестування;

— аналіз інформаційних систем;

— огляд існуючих систем автоматизованого тестування;

— проведення варіантного аналізу засобів для розроби сайтів, на основі якої створено веб-додаток, що дозволяє бронювати номери у готелях.

Практичне значення бакалаврської дипломної роботи у можливості використання її результатів для віддаленого бронювання номерів у потрібному готелі

Об’єктом постають інтернет-технології, які застосовуються при створенні сайтів, що тестуються.

Під предметом дослідження розуміємо засоби для розробки серверної частини розподіленої системи, методи та критерії їх варіантного аналізу.

1 АНАЛІЗ ІНФОРМАЦІЙНИХ МЕРЕЖЕВИХ СИСТЕМ.

1.1 Сутність індустрії гостинності та її ретроспективний аналіз

Гостинність є одним з фундаментальних понять людської цивілізації. Гостинність має три похідні з латині, французької та англійської: Hospitalis (лат.) — гостинний; Hospitality (англ.) — гостинність; Hospice (фр.) — притулок мандрівника. Загалом, гостинністю вважають сукупність засобів розміщення, харчування і розваг для туристів, а також традиції прийому гостей в різних культурах. Головні чинники, що вплинули на становлення та розвиток сфери гостинності можна розглянути на рисунку 1.

З розвитком надання послуг гостинності людям, котрі з певних причин покидали свої домівки, гостинність перетворилася на професію, а потім і на справжню індустрію.

Індустрія гостинності – це бізнес, який спрямований на забезпечення мандрівників житлом, харчуванням, а також на організацію їхнього дозвілля. Індустрія гостинності об’єднує всі суміжні галузі економіки, що спеціалізуються на обслуговуванні через спеціалізовані підприємства: готелі, ресторани, туристичні агентства, національні парки, парки культури та відпочинку та ін. Зокрема, в США ця індустрія є другою за значенням за кількістю робочих місць. Тому одна з головних задач індустрії гостинності полягає в розвитку послуг та культури сервісу. Індустрія гостинності є невід’ємною частиною туристичної індустрії та готельної індустрії. В межах готельної індустрії розвивається готельний бізнес.

Сфера гостинності як фундаментальна складова туристичної індустрії адекватна розвитку основних етапів еволюції людського суспільства. Ретроспективний аналіз розвитку сфери гостинності дозволяє виділити декілька етапів цього процесу:

1-й етап — стародавній світ II тис. до н. е.–V ст. — виникнення перших закладів гостинності;

2-й етап — VI–XV ст. (епоха середньовіччя) — початок формування спеціалізованих закладів розміщення;

3-й етап — XVI–XX ст. (епоха відродження → епоха промислової революції, індустріалізації суспільства→перша світова війна) — розвиток готельної справи;

4-й етап — 20-і роки XX ст.–початок XXI ст. — сучасний розвиток готельного господарства.

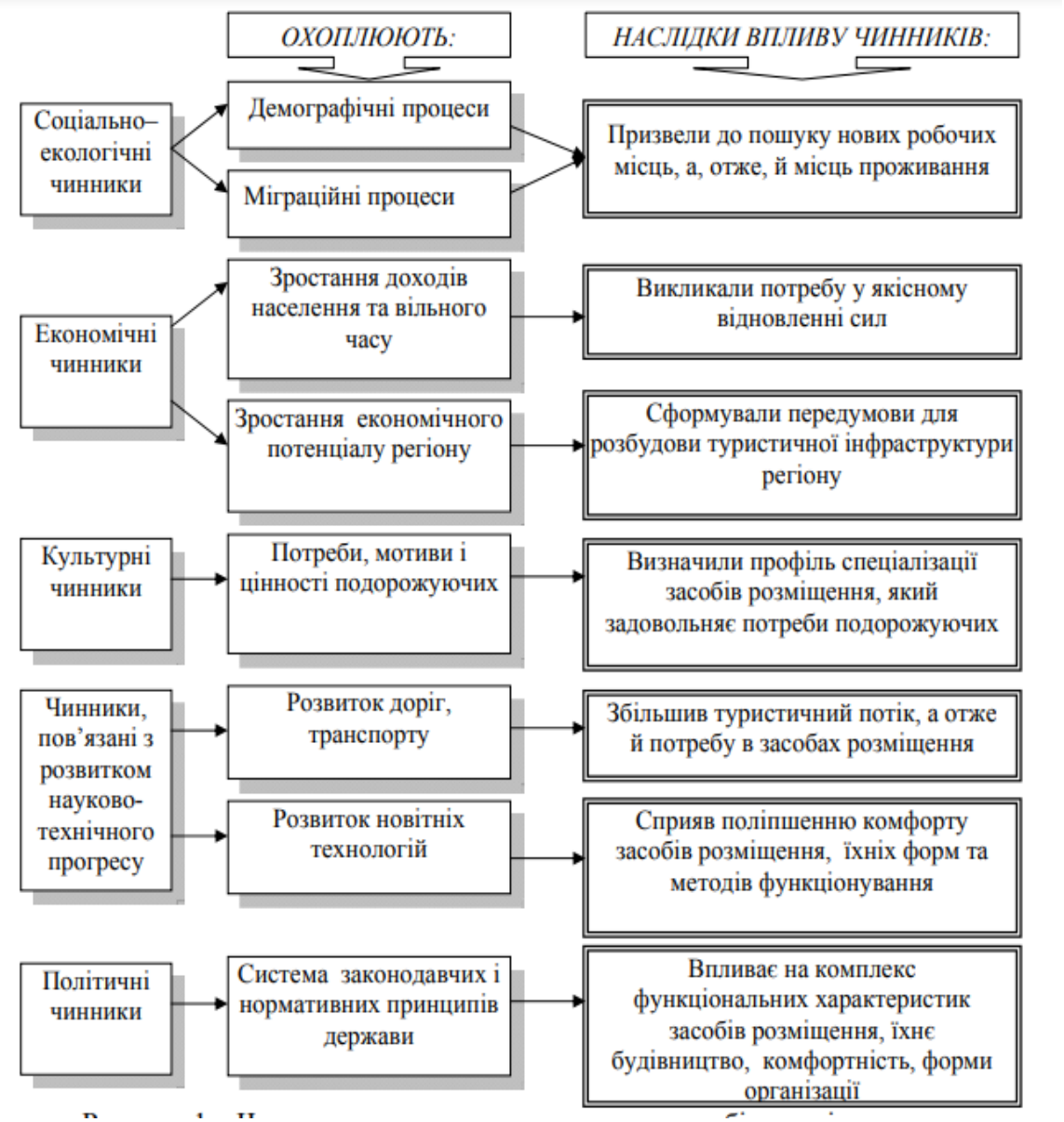

Рисунок 1.1 — Чинники становлення та розвитку засобів розміщення

1.2 Виникнення перших закладів гостинності

В епоху античності родину, що приймала гостей в своєму будинку, називали гостинною. За притулок в цей період грошей не брали. Мандрівники завжди могли знайти привітний прийом в приватних будинках. Перші ж згадки про гостьові

підприємства припадають на період античності. Ними були караван-сараї, таверни, постоялі двори. Караван-сараї представляли собою 2-х, 3-х поверхові будинки. Частину приміщень першого поверху займав господар зі своєю родиною, решта приміщень була облаштована для прийому гостей. На цьому ж поверсі передбачалося місце для стійла коней. Все це було обнесено оборонними мурами для захисту від розбійників та непогоди (рис. 2).

Рисунок 1.2 —Залишки караван-сараю Ribat-I Sharaf, 1124 р. до н. е.

Ці заклади не були схожими один на одного, в кожній місцевості дизайн приміщень караван-сараю залежав від клімату, рельєфу, вимог безпеки, доступності будівельних матеріалів. Якщо в період становлення в мусульманських країнах послуги караван-сараїв надавалися безкоштовно, що пояснювалося релігійними міркуваннями, то пізніше дах та харчування протягом трьох діб надавалися за рахунок держави, решту днів необхідно було оплатити мандрівникові.

На територіях східніше Константинополя, крім такого «пансіону», бідним мандрівникам додатково безкоштовно видавалася пара взуття. Караван-араї розташовувалися на території Туреччини, Ірану, Афганістану, північної Індії (палац Тадж-Махал). Виникнення перших спеціалізованих підприємств готельної індустрії пов’язано з Древнім Римом та Древньою Грецією (близько 50-й р. до н. е). На той час велика кількість чиновників і купців активно подорожували у справах. У зв’язку з потребою в їх розміщенні формувалася доволі розгалужена мережа державних та приватних постоялих дворів і таверн. Римляни будували засоби розміщення як засіб відшкодування

витрат на будівництво доріг. Внеском греків в розвиток гостинності стало проведення Олімпійських ігор, на час проведення яких зводили палаткові містечка, збільшували кількість таверн та інших закладів гостинності.

1.3 Початок формування спеціалізованих закладів розміщення (епоха середньовіччя)

У Європі після падіння Римської імперії діяльність закладів розміщення почала поступово завмирати і лише з часом, у зв'язку з масовими подорожами купців, підмайстрів, ремісників, учнів, артистів, паломників, знову починають розвиватися найрізноманітніші форми надання притулку, які на той час були безкоштовними. Заради любові до ближнього притулки надавалися церковними організаціями, княжими дворами. Подальший розвиток готельної справи ознаменувався появою едикту Карла Великого (768–814 рр.) (імператора Великої Римської імперії), відповідно до якого на монастирі і церкви покладався обов'язок утримання «госпіцій».

Наступний помітний період в розвитку готельного господарства пов'язаний зі встановленням в Європі регулярної поштової і транспортної мережі на кінній тязі (диліжанси в Західній Європі, станції ямщиків в Росії). Уздовж поштових трас з'явилися поштові станції для державного 19 транспорту, що служили також і місцем відпочинку; вони захищали від негоди і спрощували процедуру зміни коней. До XV ст. постоялі двори приєдналися до поштових станцій.

У XVII сторіччі стала очевидною відмінність між міськими (hotel) та сільськими (auberge, inn) готелями. Надалі ці відмінності лише посилювалися.

Пропозиція готельних послуг еволюціонувала в суворій відповідності з мінливим попитом. На рубежі XVI–XVII ст. основні вимоги, що визначають споживчий попит на послуги готелів, були в цілому сформовані.

Само слово «готель» з'явилося в XVIII ст. У Франції спочатку готелем іменували багатоквартирну будівлю, в якій квартири здавалися на місяць, на тиждень і навіть на один день, а пізніше – міський палац магната, місце перебування представництва іноземної держави або міських властей. Скоро цей термін широко розповсюдився і в Америці.

Інтенсивний розвиток готельної справи починається у XIX сторіччі. Зростання запитів заможної клієнтури щодо мандрівок і відпочинку стимулює виникнення сучасних готельних підприємств з розкішними апартаментами, високим рівнем комфорту і широким асортиментом послуг. Готельний бізнес перетворюється на важливу галузь економіки з високим рівнем доходності. Виникають великі сучасні готелі, розташовані у спеціально призначених для цієї мети спорудах на зразок приватних резиденцій або величних і гарних державних особняків.

Сьогодні у світі налічується понад 300 готельних ланцюгів. За кількістю об'єднаних у групи готелів в Європі лідирує Велика Британія, де розташовано 15% готельних груп. У Франції зосереджено 10% таких груп, в Іспанії — 5%, у Швейцарії — 5% усіх готельних мереж світу. Практика ведення бізнесу свідчить, що готелі, які входять до мережі, мають на 60% більше прибутку, ніж незалежні готелі. У таблиці 1 наведений рейтинг найбільших готельних ланцюгів світу.

1.4 Типи готелів за функціональним призначенням

Всі готелі за функціональним призначенням поділяють на дві великі групи: транзитні готелі та цільові готелі (рис.1.4)

Транзитні готелі призначені для обслуговування споживачів готельних послуг в умовах короткочасної зупинки.

До транзитних засобів розміщення відносять мотелі та кемпінги.

До цільових готелів належать готелі ділового призначення і готелі для відпочинку.

Готелі ділового призначення обслуговують осіб, що перебувають в ділових поїздках і відрядженнях.

Курортні готелі передбачають надання послуг розміщення, дієтичного харчування, лікувально-оздоровчих програм, розташовані на територіях, що дають можливості для відпочинку, оздоровлення і лікування в природно-кліматичних умовах.

Таблиця 1 — Рейтинг найпотужніших готельних мереж світу

Рейтинг за кількістю номерів		Назва мережі	Кількість номерів у:		Кількість готелів у:	
у 2013 р.	у 2012 р.		2013р	2012р	2013р	2012р
1	1	ING (Inter Continental Hotels Group), Великобританія	693072	658348	4700	4480
2	2	Hilton Hotels Corp., США	652378	631131	3992	3861
3	3	Marriott International, США	638793	622279	3672	3595
4	4	Wyndham Hotels Group, США	627437	613126	7342	7205
5	6	Choice Hotels International, США	497023	502460	6198	6203
6	5	Accor, Франція	450199	531714	3515	4426
7	7	Starwood Hotels & Resorts Worldwide, США	328055	315346	1121	1076
8	8	Best Western International, США	311611	295254	4024	4018
9	9	Home Inns (+ Motel 168), Китай	214070	176562	1772	1426
10	10	Carlson Hotels Worldwide, США	166245	165802	1077	1077

Основна проблема таких готелів — це сезонність попиту.

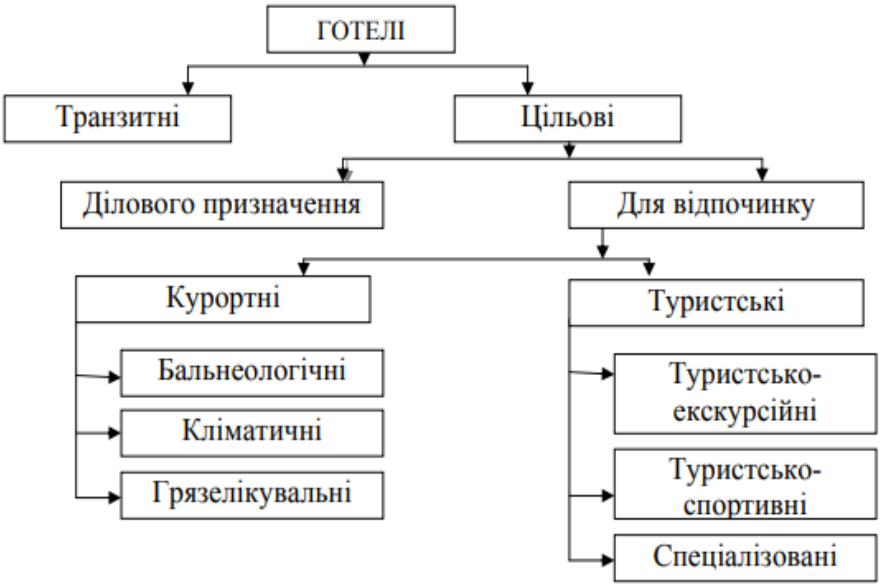

Рисунок 1.3 — Класифікація готелів за функціональним призначенням

.

2 ОГЛЯД ІСНУЮЧИХ СИСТЕМ АВТОМАТИЗАЦІЇ ТЕСТУ ТА ВИБІР ЗАСОБІВ ДЛЯ РОЗРОБКИ

2.1 Системи автоматизованого тестування

Існує безліч систем для автоматизованого тестування веб-сайтів, але їх основним мінусом є вартість. Навіть при мізерному функціоналі і низької швидкості роботи вони можуть мати високу вартість. Існує так само ряд безкоштовних і відкритих систем, але більшість їх створюють тести за методом record and play. Далі розглянемо низку систем автоматизованого тестування.

2.1.1 Selenium

Selenium — це Java-додаток, який може аналізувати файли певної структури для того, щоб знаходити в них команди для маніпуляції браузером і команди для виконання певних дій та перевірок [7].

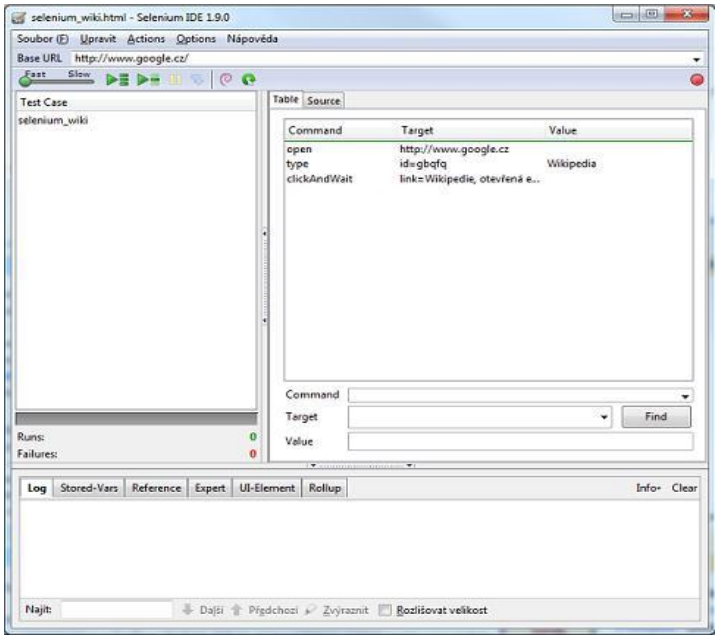

Рисунок 2.1 — Інтерфейс Selenium

Selenium як проект було розпочато у червні 2004 року, а вже у грудні 2004 року він став відкритим. Спочатку проект вела компанія ThoughtWorks, науковим керівником якої є Мартін Фаулер. Selenium підтримує Microsoft Internet Explorer, Google

Chrome, Mozilla Suite та Mozilla Firefox для Microsoft Windows, Linux та Apple Macintosh. У рамках проекту Selenium також випускається інструмент Selenium IDE, що є версією досить популярної бібліотеки Selenium в GUI-оболонці. Реалізовано це у вигляді розширення до браузера Firefox, розміром близько 240 кб, включаючи сам Selenium. Цей інструмент дозволяє записувати і відтворювати скрипти, що являють собою звичайні HTML-сторінки з однією таблицею, що містить команди. Створення автотестів у Selenium IDE відбувається за моделлю record and play [7].

Selenium підтримує більшість сучасних платформ та браузерів від Internet Explorer до Safari.

В інших браузерах часткова є підтримка, залежно від ОС, браузера та налаштувань безпеки браузера [7].

2.1.2 HP QuickTest Professional

HP QuickTest Professional (QTP) — один з провідних інструментів автоматизації функціонального тестування, є флагманським продуктом компаніїHPу своїй лінійці. Для розробки автоматизованих тестів QTP використовує мовуVBScript[14].

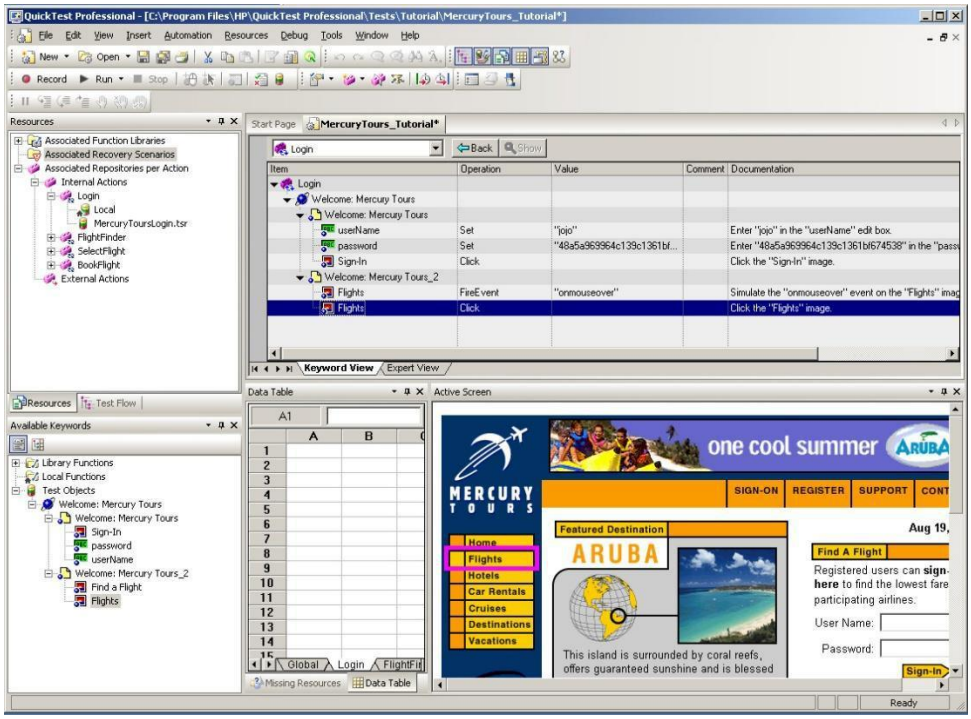

Рисунок 2.2 — Інтерфейс HP QuckTest Professional

QTP підтримує такі технології:

— Windows Presentation Foundation;
— Web services;
— Macromedia Flex;
— Ajax;
— Delphi;
— .NET;
— VisualBasic;
— ActiveX;
— Java;
— Oracle;
— SAPSolution;
— PowerBuilder;
— Siebel;
— PeopleSoft;
— VisualAge;
— Stingray.

Компанія HP рекомендує використання QTP в інтеграції зHP Quality Center для встановлення зв'язку тестів з вимогами, зберігання тестів, управління їх запуском, формування звітів [14].

На відміну від низки інших продуктів для автоматизації функціонального тестування, QTP дозволяє контролювати генерований текст скрипту в процесі запису дій користувача, за рахунок чого знижується час, необхідний для розробки тесту [14].

В QTP інформація про всі об'єкти екранного інтерфейсу зберігається в спеціальному репозиторії (Object Repository), що новому користувачеві може здатися непрозорим. Замовчання на вибір істотних властивостей кожного типу об'єктів екранного інтерфейсу можуть бути налаштовані окремо, наприклад, вікно може визначатися заголовком, а стовпець таблиці - шириною і порядковим номером в таблиці [14].

Існує вбудований механізм порівняння текстових даних з використанням регулярних виразів[14].

2.1.3 IBM Rational Functional Tester

Rational Functional Tester надає тестувальникам засоби автоматизованого тестування, що дозволяють виконувати функціональне тестування, регресивне тестування, тестування інтерфейсу користувача і тестування кероване даними. Rational Functional Tester інтегрований зIBM Rational Quality Manager- потужним набором засобів управління тестуванням, виявлення дефектів, управління версіями сценаріїв тестування та управління вимогами. Інструментальні засоби, що входять до складу цієї платформи, прискорюють розробку додатків, значно полегшуючи координацію та комунікацію всередині колективу розробників [15].

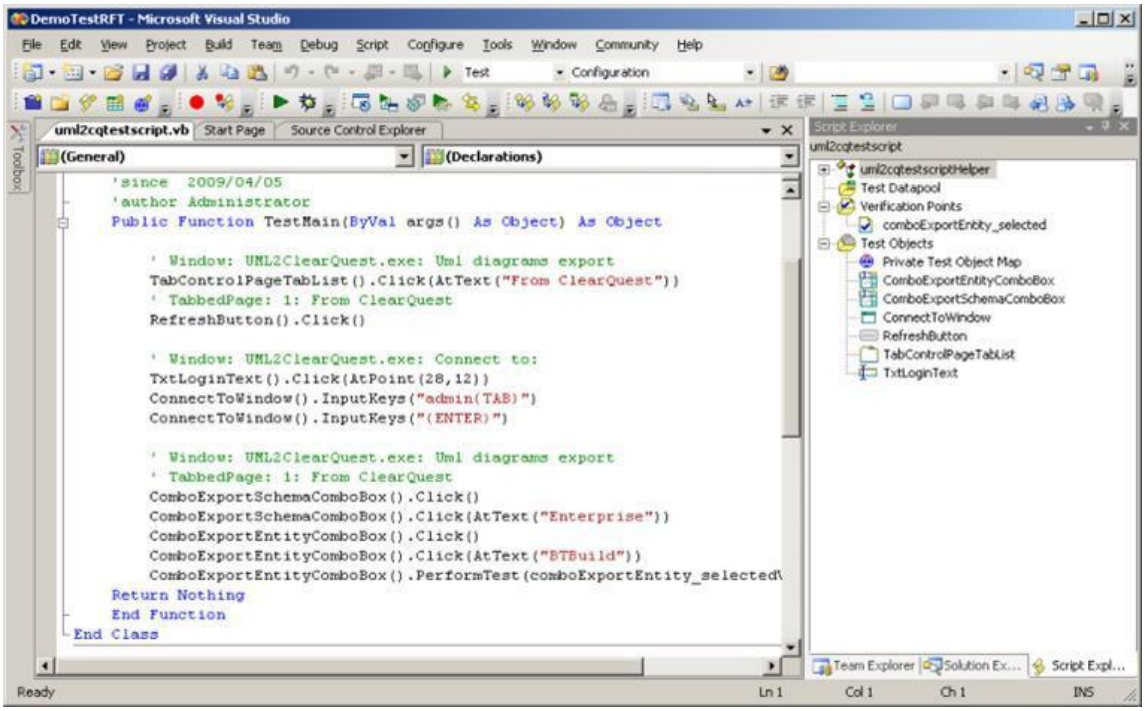

Рисунок 2.3 — Інтерфейс IBM Rational Functional Tester, інтегрованого в Microsoft Visual Studio

За допомогою інструментів, що є в IBM Rational Quality Manager, весь колектив розробників буде мати чітку інформацію про результати тестування. Зокрема, розро-

бники зможуть легко отримати доступ до інформації про виявлені дефектами тестувальників, що допоможе легко відтворити знайдені помилки. Крім того, управлінській ланці буде надана інформація, необхідна для оцінки ступеня ризиків за кожним етапом проекту [15].

IBM Rational Quality Manager звільняє тестувальників від обов'язку вручну відстежувати зміни у вимогах, що досягається шляхом автотичної позначки тестових сценаріїв, що посилаються на змінені вимоги.

Основні можливості:

— тестування додатків на Java (J2EE, J2SE, SWT, засоби управління AWT/JFC);

— тестування Web-додатків (HTML, DHTML, XML, JavaScript, Java-аплети);

— написання тестових сценаріїв Java;

— використання технології ScriptAssure та перевірка зміни прикладних даних;

— інтеграція із колективними комунікаційними засобами Rational;

— підтримує тестування додатків 3270 (zSeries) та 5250 (iSeries) з використанням розширення Functional Tester Extension для додатків на основі терміналів;

— безшовна інтеграція у середовище Eclipse, WebSphere Studio та Rational XDE Developer [15].

2.2 Порівняння існуючих систем

У ході порівняння існуючих систем було виявлено, що серед них немає тієї системи, яка б повністю задовольнила всі потреби в організації процесу автоматизованого тестування, при цьому була б безплатною і зручно розширюваною.

Система QuickTest Professional від компанії HP є одним з лідерів ринку для створення автотестів, але на жаль вартість її ліцензії від 8000 до 10000 доларів, що є вкрай високою ціною в умовах українського ринку IT компаній.

Система Rational Functional Tester має схожий функціонал з QuickTest Professional. Так само є платною, вартість її ліцензії досягає 6000 доларів.

Система Selenium IDE хоч і є безкоштовною, має гарний функціонал і відкритий код, проте потребує доопрацювання. За допомогою цієї системи можна генерувати автотести мовою Java, але підтримка даних тестів буде вкрай трудомістким завданням яка навряд чи окупиться в поточних темпах розвитку та розробки проектів.

2.3 Формування вимог до системи

Спочатку необхідно точно сформулювати основні ідеї, які визначають функціонал системи автоматизованого функціонального тестування. Грамотно складені вимоги до проекту є гарантією того, що він буде виконаний максимально якісно і матиме весь необхідний функціонал.

Таким чином, до системи автоматизованого функціонального тестування, що розробляється, пред'являються такі вимоги:

— простота створення створення і підтримки тестів навіть для людини, яка раніше не працювала в рамках автоматизації тестування, але володіє навичкою програмування;

— система повинна бути оптимізованою, швидкодіючою і здатною здійснювати тестування на різних рівнях від інтеграційного до системного;

— запускати тести на різних платформах та пристроях, для проведення тестування сумісності;

— наявність розвиненої системи звітів, існує необхідність в автоматичному формуванні різних видів звітів за станом проекту (кількість багів, швидкодія і т.д.);

— система повинна складатися з окремих модулів, що відповідають за окреме завдання розробки тестів;

— функціонал системи повинен бути легко розширюваним.

2.4 Вибір основної мови програмування

Для початку необхідно вибрати мову програмування, за допомогою якого можна було б реалізувати систему, що задовольняє всім вимогам і дозволяє взаємодіяти з найпопулярнішими інструментами тестування. Вибір був зроблений на користь Java мови.

Java — високорівнева об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (в подальшому придбаною компанією Oracle).

Програми Java зазвичай транслюються у спеціальний байт-код, тому вони можуть працювати на будь-який віртуальній Java-машині незалежно від комп'ютерної архітектури. Перевагою подібного способу виконання програм є повна незалежність байт-коду операційної системими і обладнання, що дозволяє виконувати Java-програми на будь-якому пристрої, для якого існує відповідна віртуальна машина. Іншою важливою особливістю технології Java є гнучка система безпеки, в рамках якої виконання програми повністю контролюється віртуальною машиною. Будь-які операції, які перевищують встановлені повноваження програми (наприклад, спроба несанкціонованого доступу до даних або з'єднання з іншим комп'ютером), викликають негайне переривання [8].

Часто до вад концепції віртуальної машини відносять зниження продуктивності. Ряд удосконалень дещо збільшив швидкість виконання програм на Java:

— застосування технології трансляції байт-коду в машинний код безпосередньо під час роботи програми (JIT-технологія) з можливістю збереження версій класу в машинному коді;

— широке використання платформно-орієнтованого коду (native-код) у стандартних бібліотеках;

— апаратні засоби, що забезпечують прискорену обробку байт-коду (наприклад, технологія Jazelle, підтримувана деякими процесорами фірми ARM)[8].

Ідеї, закладені в концепцію та різні реалізації середовища віртуальної машини Java, надихнули безліч ентузіастів на розширення переліку мов, які могли б бути використані для створення програм, що виконуються на віртуальній машині. Ці ідеї

знайшли також вираз у специфікації загальномовної інфраструктури CLI, закладеної в основу платфор-ми.NET компанією Microsoft[8].

Для розробки проекту було обрано середовище (IDE) IntelliJ IDEA Comunity Edition (рисунок 2.4) виробництва компанії JetBrains. Дана IDE є безкоштовною, призначена для розробки проектів на Java і має великі можливості розширення за рахунок плагінів, що підключаються.

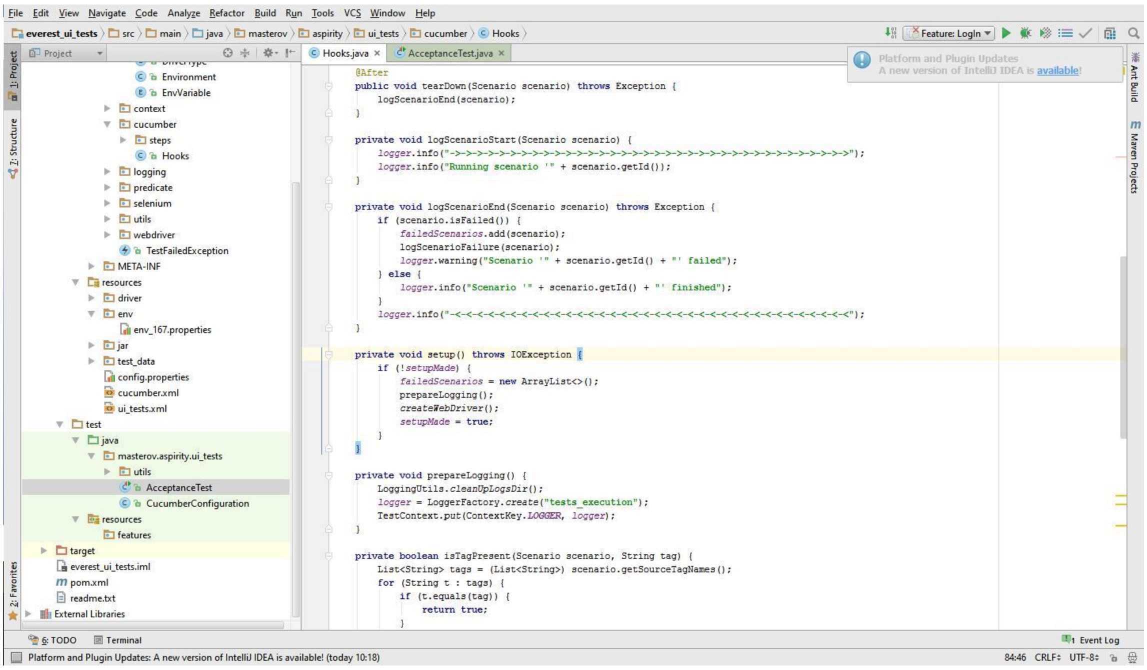

Рисунок 2.4 — Інтерфейс IntelliJ IDEA

2.5 Вибір технологій та специфікації

2.5.1 Cucumber

Однією з великих проблем автоматизації тестування є підтримка, формалізація та розуміння тест кейсів. Наприклад, якщо в компанію прийшла нова людина, то вона витратить велику кількість часу, щоб розібратися, що відбувається в момент роботи того чи іншого тесту. Мінімізувати цю проблему допоміг фреймворк Cucumber.

Рисунок 2.5 — Логотип фреймворку Cucumber

Cucumber використовується для написання програмного коду з методології Behavior-Driven-Development (BDD), що перекладається як "розробка рухома поведінкою". Суть методології в тому, щоб спочатку формалізувати вимоги, а потім уже відштовхуючись від цих вимог реалізовувати функціонал. У разі системи автоматизованого тестування цей підхід удобний для формалізації тест кейсів природною мовою, а потім реалізації кожного кроку у вигляді програмного коду. Це дозволяє внести ясність і чіткість у сам тест кейс, прискорює розуміння його роботи і того, що і як він тестує [9]. Файли з формалізованими тест кейсами мають розширення .feature і зазвичай мають назву самого тест кейсу, наприклад Login.feature (рисунок 2.6).

```
Feature: Trying To Login
  Scenario: I'm going to loginpage and trying to login
    Given i'm navigating to home page
    Then i'm trying to login as admin
```

Рисунок 2.6 — Приклад формалізованого тесту кейсу LogIn.Feature

Ключове слово Feature використовується для формалізації назви тест-кейс, а ключове слово Scenario служить для швидкого опису самого алгоритму тестування або очікуваного результату.

Для зв'язку формалізованого тест кейсу та його описом мовою програмування використовуються ключові слова:

— Given;
— When;
— Then;
— And.

Працюють ці ключові слова подібно до посилань в Інтернеті, говорячи про те, що існує якийсь спосіб, який відноситься саме до цього кроку тест кейсу. Cucumber підтримує безліч природних мов для формування тест кейсів, включаючи Російський, що робить його одним з найсильніших фреймворків для досягнення простоти і зрозумілості тестів і тест кейсів, які до них відносяться.

Так само Cucumber має вбудований генератор звітів що дозволяє наочно (рисунок 2.7) зобразити на якому моменті сталася помилка, що перешкодила отриманню очікуваного результату.

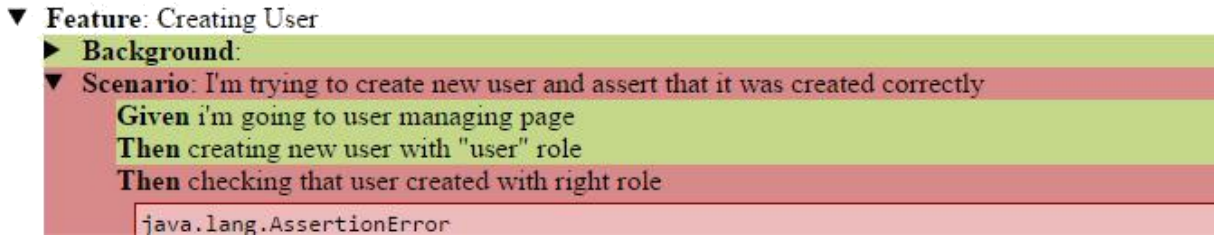

Рисунок 2.7 — Звіт згенерований фреймворком Cucumber

2.5.2 Selenium WebDriver

Для роботи з будь-яким сайтом або веб-програмою використовуються браузер. Браузер — прикладне програмне забезпечення для перегляду веб-сторінок. Перш ніж взаємодіяти з сайтом, потрібно вирішити завдання взаємодії з браузером як таким. На допомогу прийшов Selenium WebDriver. Цей драйвер є ключовим елементом системи і дозволяє імітувати роботу користувача з браузером: клікати на посилання, заповнювати поля, очищати поля, натискати на кнопки, переходити за посиланням і т. д. WebDriver здатний взаємодіяти з більшістю популярних браузерів: InternetExplorer, Mozilla FireFox, Google Chrome і т.д.

2.5.3 Бібліотека тестування jUnit

Далі були проведені дослідження популярних бібліотек для тестування, і вибір був зроблений на користь бібліотеки jUnit. Вона дозволить додати у будь-яке місце тесту якусь точку порівняння фактичного та очікуваного результату.

jUnit — бібліотека для модульного тестування програмного забезпечення мовою Java [12]. Створена Кентом Беком та Еріком Гаммою, JUnit належить сім'ї фреймворків xUnit для різних мов програмування, що бере початок у SUnit Кента Бека для Smalltalk. jUnit породив екосистему розширень — jMock, EasyMock, DbUnit, HttpUnit і т. д. jUnit був портований на інші мови, включаючи PHP (PHPUnit), C# (NUnit),

Python (PyUnit), Fortran (fUnit), Delphi (DUnit, Free Pascal (FPCUnit), Perl (Test::Unit), C++ (CPPUnit), Flex (FlexUnit)) , JavaScript (JSUnit), COS (COSUnit) [12].

jUnit дозволяє перевіряти деякі твердження описані в тесті, чи то вихідний параметр, чи рівність рядків. Приклад перевірки рівності рядків:

```
String nameOne = "Готель 1";
String nameTwo = "Готель 2";
assert(nameOne.equalsIgnoreCase(nameTwo));
```

Результатом роботи даного коду буде помилка перевірки умови (assertationError), тому що рядок nameOne не дорівнює рядку nameTwo.

2.5.4 Система автоматичного складання проекту Maven

В результаті з'явився набір деяких програмних засобів, кожне з яких відповідає за своє певне завдання. Для Java існує система складання Maven, призначена для автоматичного складання проектів, створена спільнотою Apache Software Foundation.

Для отримання структури проекту Maven використовує POM мову, що є підмножиною мови XML. У файлах опису проекту міститься його специфікація, конфігуруються залежності від інших проектів, індивідуальні фази процесу побудови проекту та список плагінів, що реалізують порядок складання.

3 РЕАЛІЗАЦІЯ ІНОРМАТИВНОЇ МЕРЕЖІ СИСТЕМИ ТЕСТУВАННЯ

3.1 Розробка архітектури системи

Розроблена архітектура системи має, при першому відкритті проекту на новій машині в IDE, імпортувати файл pom.xml для автоматичного збирача Maven. Встановлюються залежності між компонентами проекту і завантажуються бібліотеки, що відсутні.

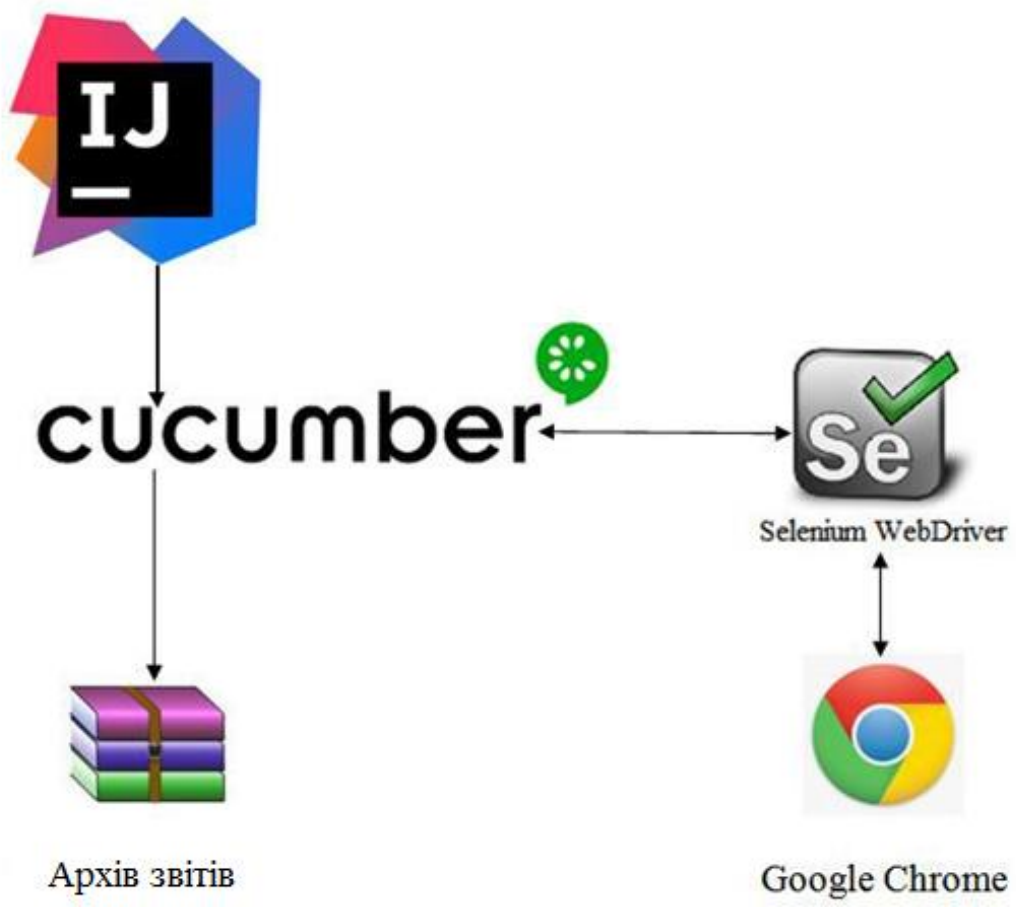

Рисунок 3.1 — Розроблена архітектура системи

Варто відзначити, що для роботи з браузером Google Chrome потрібно завантажити драйвер для Selenium WebDriver із зовнішнього ресурсу.

Після складання проект повністю готовий для створення нових та виконання вже існуючих тестів.

3.2 Основні класи проекту

Основними класами у проекті є класи AcceptanceTest та клас Hooks. Ці класи працюють у парі, дозволяючи реалізовувати гнучкі і стійкі тести. Код класу AcceptanceTest представлений у додатку А. Код класу Hooks представлений у додатку.

При запуску класу AcceptanceTest відбувається ініціалізація кадру Cucumber, який шукає ключові слова Before, Test і After, а потім викликає методи, що відповідають ключовим словам.

3.2.1 Before

На даній стадії необхідно очистити або створити директорію для звітів і запустити браузер, потім необхідно з файлу .feature отримати ім'я сценарію і вивести в консоль, що почав виконуватись наш сценарій. Якщо за раз виконується безліч сценаріїв, то відбувається перевірка на дублювання сценаріїв, якщо ми вже перевіряли даний сценарій, то повторна перевірка проведена не буде. За ці дії відповідає метод setUp() класу Hooks.

3.2.2 Test

На даній стадії відбувається ініціалізація об'єкта Cucumber, який виконуватиме тест та взаємодіятиме з Selenium WebDriver, та виконання тестового сценарію відповідно до формалізованого тесту кейсу.

3.2.3 After

Після отримання результату тесту, чи він позитивний чи негативний, відбувається завершення роботи WebDriver і створення звітних файлів у папці зі звітами. Також у консоль виводиться інформація про завершення тесту

3.3 Опис об'єктів веб-сторінки

Для того, щоб WebDriver знав, до якого елемента слід застосовувати ту чи іншу дію, потрібно описати елементи на сторінці за допомогою одного з двох способів:

— XPATH;

— CSS.

Отримавши конкретний опис елемента зі сторінки, необхідно створити клас, з назвою сторінки сайту, що тестується, і описати в ньому елементи (рисунок 3.2).

```
@Locate(how = XPATH, using = ".//*[@id='side-menu']/li[3]/ul/li[3]/a")
public PageObject requestNewUser;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[1]/div[2]/input")
public PageObject firstNameRequest;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[2]/div[2]/input")
public PageObject lastNameRequest;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[3]/div[2]/input")
public PageObject emailRequest;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[4]/div[2]/input")
public PageObject companyRequest;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[6]/div/button[3]")
public PageObject requestBtn;
@Locate(how = XPATH, using = ".//*[@id='page-wrapper']/div[2]/ui-view/div/div[2]/form/div[6]/div/button[1]")
public PageObject cancelBtn;
@Locate(how = CSS, using = ".alert")
public PageObject succsessAlert;
```

Рисунок 3.2 — Приклад опису елементів сторінки

3.3.1 XPATH

XPATH — мова запитів до елементів XML-документа. Розроблено для організації доступу до частин документа XML у файлах трансформації XSLT і є стандартом консорціуму W3C.XPath покликаний реалізувати навігацію по DOM в XML.XPath використовується компактний синтаксис, відмінний від прийнятого в XML. У 2007 році завершилася розробка версії 2.0, яка тепер є складовою мови XQuery1.0. У грудні 2009 року почалася розробка версії 2.1, яка використовуєXQuery1.1 [19].

Для знаходження XPATH елемента існує зручне розширення для Mozilla Firefox, що називається FirePath (рисунок 3.3). Це розширення максимально зручне у використанні, для знаходження XPATH необхідного елемента, достатньо натиснути

на кнопку інтерфейсу із зображенням стрілки миші та клікнути по необхідному елементу на сторінці. У діалоговому вікні виведеться підсвічений вихідний код елемента, а в рядку пошуку виведеться його XPATH.

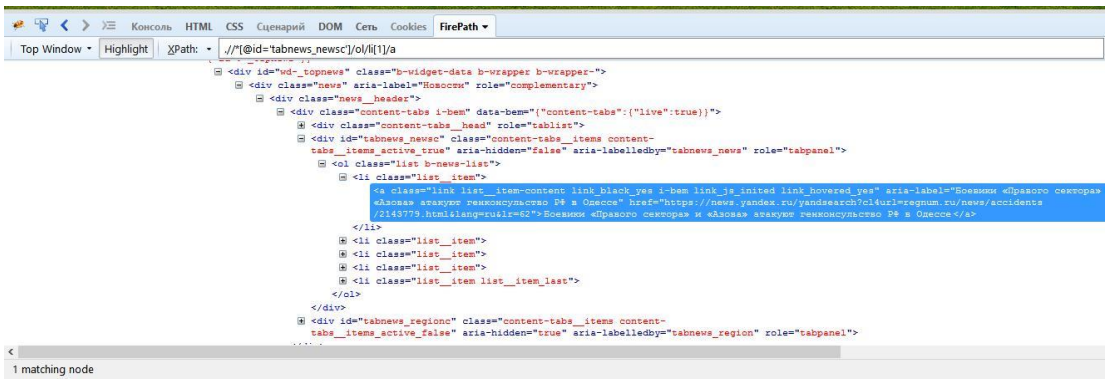

Рисунок 3.3 — Інтерфейс розширення FirePath

3.3.2 CSS

CSS — формальна мова описи зовнішнього вигляду документа, написаного з використанням мови розмітки [20].

Переважно використовується як опис опису оформлення зовнішнього вигляду веб-сторінок, написаних за допомогою мов розмітки HTML і XHTML, але може також застосовуватися до будь-яких XML-документам, наприклад, до SVG або XUL[20].

Для вибору елемента на сторінці за допомогою CSS необхідно скласти селектор CSS. Для отримання якісного селектора необхідно відкрити в браузері інструменти розробника, навести мишу на цікавий для нас елемент, у вікні інструментів розробника підсвітиться код, що відповідає за розташування даного елемента на сторінці. Після цього необхідно визначити ключові атрибути елемента (клас батьків, наявність дочірніх класів, наявність специфічних атрибутів і т.д.) і сформувати CSS селектор.

Також для складання селектора існує утиліта SuperSelector. Ця утиліта написана мовою JavaScript, максимально проста у використанні та дуже гнучка у налаштуванні. SuperSelector можна викликати на будь-якій веб-сторінці, для цього достатньо створити закладку в браузері і активувати її на потрібній сторінці простим натисканням миші. Утиліта запуститься, з'явиться інтерфейс, а потім достатньо натиснути Ctrl +

Click на необхідний елемент сторінки для отримання CSS селектора. На рисунку 3.4 зображено приклад роботи SuperSelector.

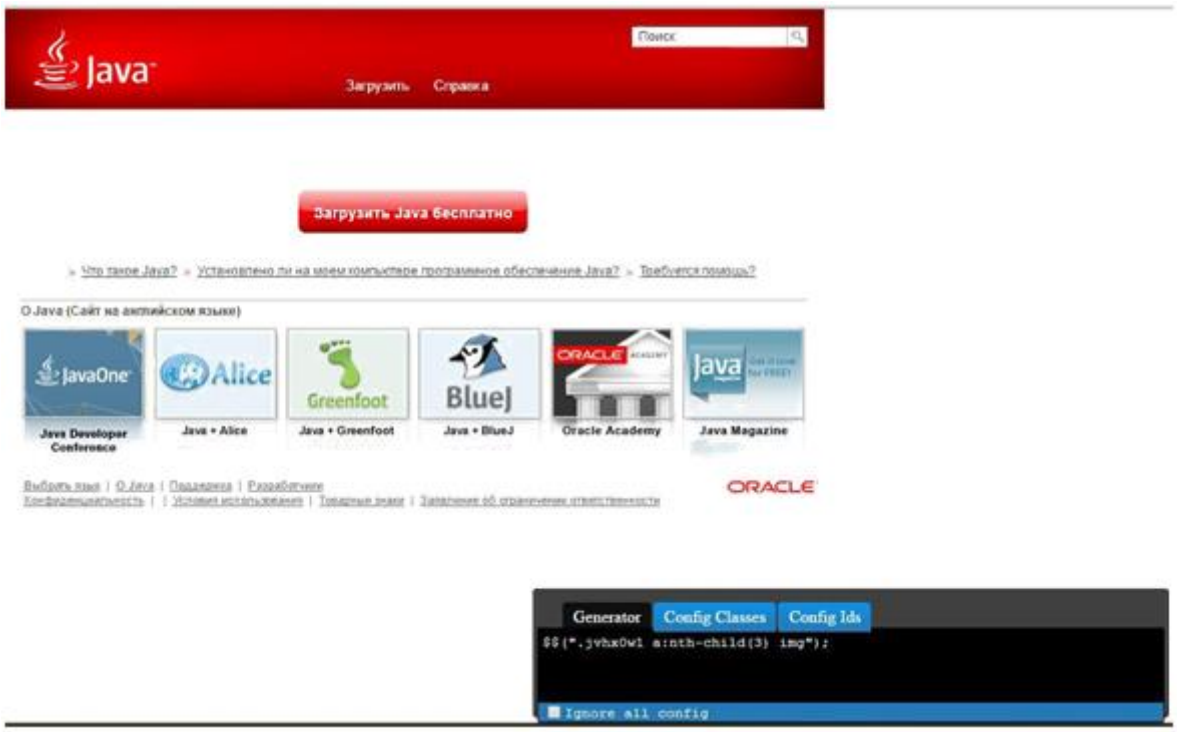

Рисунок 3.4 — Приклад роботи SuperSelector

3.4 Додаткові класи для WebDriver

Selenium WebDriver має потужний API за допомогою якого стає можливим кодування команд для роботи з браузером.

Основні команди API Selenuim WebDriver:

— get() — метод дозволяє перейти URL на веб сторінку;

— findElement() — метод, який дозволяє WebDriver виявити елемент на сторінці за допомогою відомих XPATH або CSS;

— click() — метод, який дозволяє натиснути на будь-який об'єкт на сторінці

— sendKeys() — метод, який дозволяє ввести будь-який текст у полі введення. Цей метод можна викликати для будь-якого елемента на сторінці, наприклад, для симулювання натискання гарячих клавіш;

— clear() — метод, який дозволяє очистити текстове поле.

Звернення до WebDriver з використанням даних команд у такому вигляді робить код тестів дуже громіздким і важко читаним, тому було прийнято рішення написати класи та методи, що спрощують взаємодію з API.

WebDriverFactory — клас містить методи ініціалізації драйвера, потрібний для конкретного браузера.

WebDriverHelper — клас містить методи, що спрощують виклик деяких часто повторюваних функцій WebDriver. Наприклад метод close-CurrentWebDriver() завершує роботу поточного драйвера і виводить те-куще повідомлення в лог і консоль. Метод navigateTo(String url) служить для переходу на веб-сторінку за посиланням, яке передається у вигляді рядкової змінної String url. Також існує метод disableAskLeavePagePo-pup() що відповідає за відключення повідомлення, що запитує підтвердження на залишення сторінки, що вимагає реакції користувача, що дозволяє безперешкодно завершити роботу драйвера.

PageObject() — клас на основі якого відбувається опис об'єктів на сторінці, а також який містить методи, які дозволяють чекати певного стану сторінки, наприклад метод waitUntilIs-Clickable() дозволяє дочекатися поки елемент на сторінці можна буде клікнути, а метод waitUntilAppears() дозволяє дочекатися поки конкретний елемент з'явиться сторінці.

BasePage — клас контейнер для базових об'єктів які зустрічаються майже на кожній веб-сторінці (header, menu, footer і т.д.), а також містить методи signIn() і signOut() відповідальні за вхід і вихід з акка-унта користувача веб-програми.

3.5 Система генерації звітів з проведених тестів

Була додана додаткова система запису подій для виведення інформації в режимі реального часу в консоль середовища розробки з можливістю генерації файлу, що містить звіти про проведені системою тести.

Клас LoggerFactory містить метод create() за допомогою якого відбувається ініціалізація системи запису подій, створюються директорії містять звіти, а також самі файли зі звітами у форматі .log (рисунок 3.5).

```
16:40:03 INFO: Instantiating Chrome driver
16:40:03 INFO: Looking up for driver driver/chromedriver.exe...
16:40:11 INFO: ->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->->
16:40:11 INFO: Running scenario 'creating-user;i'm-trying-to-create-new-user-and-assert-that-it-was-created-correctly
16:40:11 INFO: Navigating to home page
16:40:54 INFO: Attempting to login as 'admin' with password 'qwerty123456'...
16:40:54 INFO: Waiting until element 'usernameField' appears..
16:40:54 INFO: Clicking on 'usernameField'
16:40:54 INFO: Looking for 'usernameField' element
16:40:54 INFO: Sending keys [admin] to 'usernameField'
16:40:54 INFO: Looking for 'usernameField' element
16:40:55 INFO: Clicking on 'passwordField'
16:40:55 INFO: Looking for 'passwordField' element
16:40:55 INFO: Sending keys [qwerty123456] to 'passwordField'
16:40:55 INFO: Looking for 'passwordField' element
16:40:55 INFO: Clicking on 'signInBtnPopup'
16:40:55 INFO: Looking for 'signInBtnPopup' element
16:40:55 INFO: Waiting until element 'usersSideMenu' appears..
16:40:57 INFO: Clicking on 'usersSideMenu'
16:40:57 INFO: Looking for 'usersSideMenu' element
16:40:57 INFO: Waiting until element 'manageUsersSubMenu' appears..
16:40:57 INFO: Clicking on 'manageUsersSubMenu'
16:40:57 INFO: Looking for 'manageUsersSubMenu' element
16:41:03 INFO: Waiting until element 'addBtn' appears..
16:41:03 INFO: Waiting until element 'addBtn' is clickable..
16:41:03 INFO: Waiting until element 'addBtn' is enabled..
16:41:03 INFO: Waiting until attribute 'disabled' of element 'addBtn' doesn't exist..
16:41:03 INFO: Clicking on 'addBtn'
16:41:03 INFO: Looking for 'addBtn' element
16:41:03 INFO: Waiting until element 'fullName' appears..
16:41:04 INFO: Sending keys [Jeremy Oyster] to 'fullName'
16:41:04 INFO: Looking for 'fullName' element
16:41:04 INFO: Clearing input of 'email'
16:41:04 INFO: Looking for 'email' element
16:41:04 INFO: Sending keys [gioxproo@yomail.info] to 'email'
16:41:04 INFO: Looking for 'email' element
```

Рисунок 3.5 — Приклад звіту про проведення тесту

Клас LoggerUtils містить метод zipTestLogs(), за допомогою якого відбувається архівація звітів про проведений тест і розміщення архіву (рисунок 3.6) в директорії містить дані звіти. Адреса директорії встановлюється в конфігураційному файлі. Також цей клас містить метод cleanUpLogsDir(), який служить для відчистки директорії зі звітами і метод closeLoggerHandlers() який служить для завершення роботи системи запису подій.

Клас RecordFormatter містить формат дати, який буде записано в Звіти.

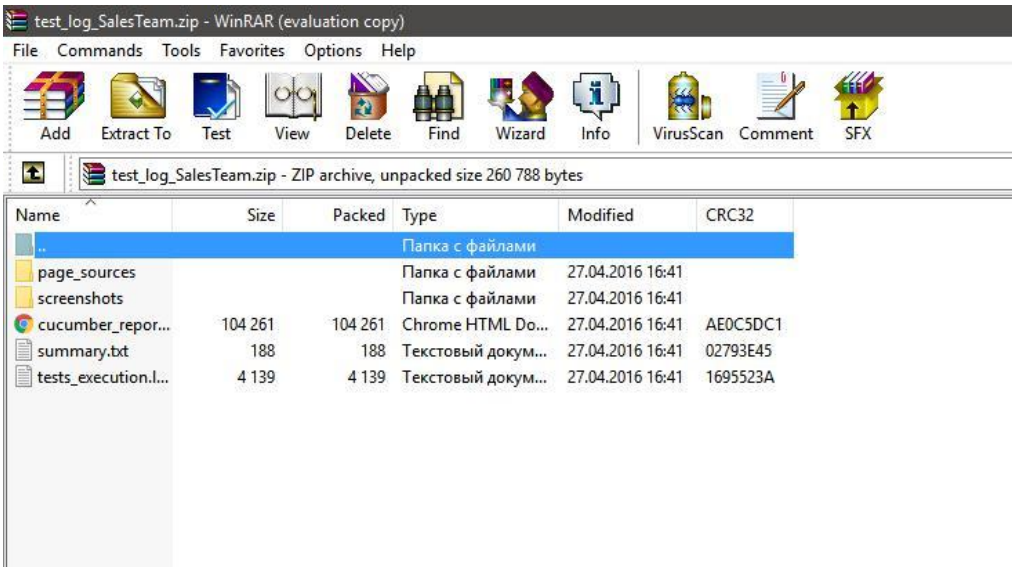

Рисунок 3.6 — Приклад архіву, отриманого за допомогою zipTestLogs()

3.6 Класи збору додаткової інформації

Для отримання додаткової інформації про веб-сторінку було реалізовано клас TestUtils, який містить такі методи.

SaveBrowserScreenshot(String outputDir, String fileName) - метод, який дозволяє зберегти знімок вікна браузера з ім'ям переданим рядковою змінною fileName в папку, адреса якої передається рядковою змінною outputDir. Якщо ім'я файлу не вказано, стандартне ім'я error.png.

SaveCurrentPageSource(String outputDir, String fileName) - метод, що зберігає вихідний код сторінки в папку з ім'ям переданим рядковою змінною fileName в папку, адреса якої передається рядковою змінною outputDir.

Sleep(int seconds) - метод дозволяє зупинити хід виконання тесту на необхідну кількість секунд, що передається цілою змінною seconds. Після закінчення необхідного часу тест продовжить своє виконання.

3.7 Файли конфігурації

Для зручності конфігурування системи було прийнято рішення винести основні дані, такі як тип драйвера, адресу папки для зберігання звітів, адресу папки для зберігання знімків вікна браузера та адресу папки для збереження вихідного коду сторінки

в окремий файл config.properties, що робить код тесту ще більше читається і легко модифікується. Наприклад, якщо нам потрібно змінити тип драйвера, достатньо просто змінити змінну driver.type, а також якщо необхідно зберегти скріншот у папку для зберігання скріншотів, достатньо отримати значення змінної screenshots.dir.

Також для конфігурації Cucumber був створений клас CucumberConfiguration, в якому необхідно вказати змінну features, що містить шлях до .feature файлу з необхідним формалізованим тест кейсом, змінну glue, що містить шлях до закодованих кроків і класу Hooks, а також змінну format, в яку вказується формат отриманого звіту.

```
package masterov.aspirity.ui_tests;

import cucumber.api.CucumberOptions;

@CucumberOptions(
        features = "src/test/resources/features/TryingToGoToPageByURL.feature",
        //features = "src/test/resources/features/LogIn.feature",
        //features = "src/test/resources/features/CreatingQuote.feature",
        //features = "src/test/resources/features/CreatingUser.feature",
        //features = "src/test/resources/features/RequestingNewUser.feature",
        //features = "src/test/resources/features/CreatingQuoteAfterAcceptingNewUser.feature",
        format = {"html:target/cucumber"},
        glue = {"masterov.aspirity.ui_tests.cucumber"}
)
public class CucumberConfiguration {
}
```

Рисунок 3.7 — Приклад змісту класу CucumberConfiguration

3.8 Структура проекту

На рисунку 3.8 зображено структуру отриманого проекту.

В папці src містяться всі фали класів, необхідні для коректної роботи системи.

В папці conf знаходяться класи, що містять методи для збору даних про оточення, на якому проводяться тести і дозволяють сконфіурувати систему тестування на основі отриманих даних.

В папці cucumber знаходяться класи, що містять інструкції складені на основі формалізованих кроків, що знаходяться в папці test/resources/features, а також клас Hooks.

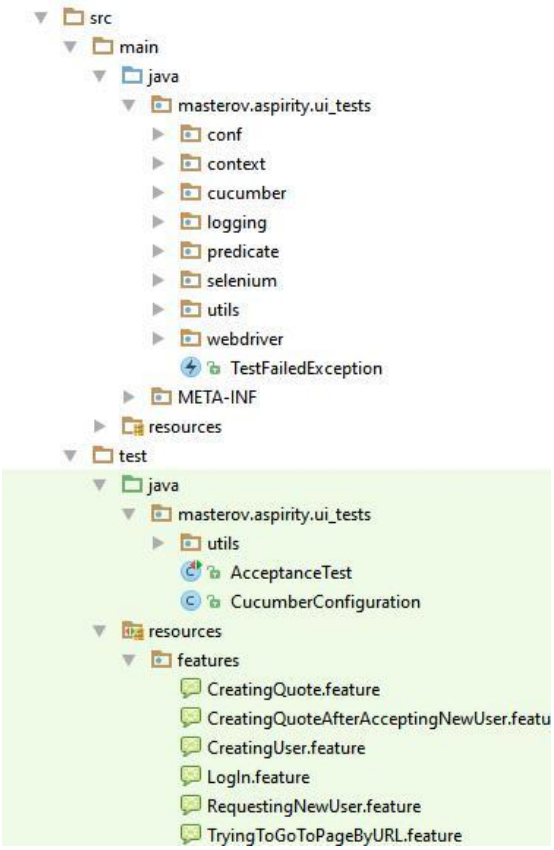

Рисунок 3.8 — Структура проекту

В папці predicate знаходяться класи, що дозволяють системі тестування розуміти описані об'єкти тестованих веб-сторінок, а також класи, що містять методи, що дозволяють перевірити існування деяких елементів тестованої веб-сторінки на основі WebDriver API.

В папці selenium знаходяться класи, що містять опис об'єктів з тестованих веб-сторінок, а також базові класи для опису об'єктів веб-сторінок PageObject та BasePage.

В папці logging знаходяться класи, що відповідають за збір та виведення інформації про проведення тесту.

В папці webdriver знаходяться класи ініціалізації Selenium WebDriver, а також класи, що спрощують роботу з WebDriver.

ВИСНОВКИ

В результаті вийшов потужний інструмент, а саме, інформаційна мережева система тестування сайтів, що задовольняє всім заявленим вимогам і володіє достатнім рівнем модульності для подальшого розвитку та розширення.

У хода роботи було реалізована інформаційна мережева система тестування; проаналізовано інформаційні системи; розглянуто різновиди існуючих систем автоматизованого тестування; проведено варіантний аналіз засобів для розроби сайтів, на основі якої створено веб-додаток, що дозволяє бронювати номери у готелях.

На даний момент розроблена система може бути застосована для тестування сайтів міжнародних готелів, пошуку інформації, адже у часи війни власникам готелів важлива безперебійна робота їх сайтів, а біженцям та переселенцям потрібний оперативний доступ до достовірної інформації. Система, в подальшому, може бути рекомендована, як потужний і корисний інструмент для створення та проведення автоматизованого функціонального тестування сайтів будь-якого вмісту.

У перспективі розвитку проекту планується доробити інтеграцію з білд сервером Jenkins для реалізації безперервної інтеграції та тестування на всіх стадіях розробки проекту сайтів.

Також планується реалізація взаємодії системи з базами даних за допомогою бібліотеки JDBC.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Selenium [Електронний ресурс] - Режим доступу: https://en.wikipedia.org/wiki/Selenium_(software)

2. Java [Електронний ресурс] - Режим доступу: https://ru.wikipedia.org/wiki/Java

3. BDD за допомогою Cucumber [Електронний ресурс] — Режим доступу: https://habrahabr.ru/post/62958/

4. Cucumber [Електронний ресурс] — Режим доступу: http://cucumber.io

5. Apache Maven [Електронний ресурс] - Режим доступу: https://ua.wikipedia.org/wiki/Apache_Maven

6. jUnit [Електронний ресурс] - Режим доступу: https://ru.wikipedia.org/wiki/JUnit

7. Selenium [Електронний ресурс] - Режим доступу: selenium2.ru

8. HPQuickTestProfessional [Електронний ресурс]- Режим доступу: https://ru.wikipedia.org/wiki/HP_QuickTest_Professional

9. IBM Rational Functional Tester [Електронний ресурс] — Режим доступу https://en.wikipedia.org/wiki/Rational_Functional_Tester

10. Функціональне тестування [Електронний ресурс] — Режим доступу: http://www.protesting.ru/testing/types/functional.html

11. Приймальний тестування або Прийомо-здавальне випробування (Acceptance Testing) [Електронний ресурс] - Режим доступу: http://www.protesting.ru/testing/levels/acceptance.html

12. XPATH [Електронний ресурс] - Режим доступу: https://ua.wikipedia.org/wiki/XPath

13. CSS [Електронний ресурс] - Режим доступу: https://ru.wikipedia.org/wiki/CSS

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

_____ проф., д.т.н. О.Д. Азаров

«___»_____2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання дипломної роботи

«Інформаційна мережева система тестування сайтів міжнародних готелів»

08-23.БДР.018.00.000 ПЗ

Науковий керівник к.т.н., доц. каф. ОТ

_____ Крупельницький Л.В.

Студента групи 1КІ-18б

_____ Щербань К.П

Вінниця 2022

1 Підстава для виконання бакалаврської дипломної роботи (БДР)

Підставою для розробки даної бакалаврської дипломної роботи є наказ ВНТУ № від «__» ___ 2022 року та рішення засідання кафедри обчислювальної техніки (протокол №_1_ від «_____»__ 2022 року).

2 Мета та призначення розробки

Теоретичне та практичне дослідження інформаційних мережевих систем тестування сайтів

3 Перелік задач

Перелік задач, що повинні бути виконані:

— реалізація інформаційної мережевої системи тестування;
— аналіз інформаційних систем;
— огляд існуючих систем автоматизованого тестування;
— проведення варіантного аналізу засобів для розроби сайтів, на основі якої створено веб-додаток, що дозволяє бронювати номери у готелях.Перелік матеріалів, що подаються до захисту БДР

4 Перелік матеріалів, що подаються до захисту

До захисту подається: пояснювальна записка БДР, графічні і ілюстративні матеріали, протокол попереднього захисту БДР на кафедрі, відгук наукового керівника, відгук рецензента, анотації до БДР українською та іноземною мовами, довідка про відповідність оформлення БДР діючим вимогам.

5 Порядок контролю виконня та захисту БДР

5.1 Робота виконується в три етапи, таблиця А.1

Таблиця А.1 — Етапи виконання роботи

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз інформаційних мережевих систем, огляд інуючих систем автоматизованого тестування та вибір засобів для розробки	12.02.2022	12.03.2022	Розділи 1 та 2
2	Підготовка матеріалів та опис розробки	22.03.2022	22.03.2022	Чернетки матеріалів
3	Оформлення появнювальної записки та ілюстративного матеріалу	23.04.2022	17.05.2022	Пояснювальна записка

6 Обов’язковий ілюстративний матеріал

Перелік обов’язкового ілюстративного матеріалу:

До обов’язкового ілюстративного матеріалу належать класифікація готелів за функціональним призначенням, приклад системи тестування автоматизованого тестування

7 Порядок контролю та прийому

Виконання етапів графічної та розрахункової документації БДР контролюється науковим керівником згідно зі встановленими термінами. Захист БДР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання БДР

При оформлюванні БДР використовуються:

— ДСТУ 3008 : 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302 : 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— документами на які посилаються у вище вказаних.

Технічне завдання до виконання отримав _____ Щербань. К.П

ДОДАТОК Б

Чинники становлення та розвитку засобів розміщення

Рисунок Б.1 — Чинники становлення та розвитку засобів розміщення

ДОДАТОК В

Класифікація готелів за функціональним призначенням

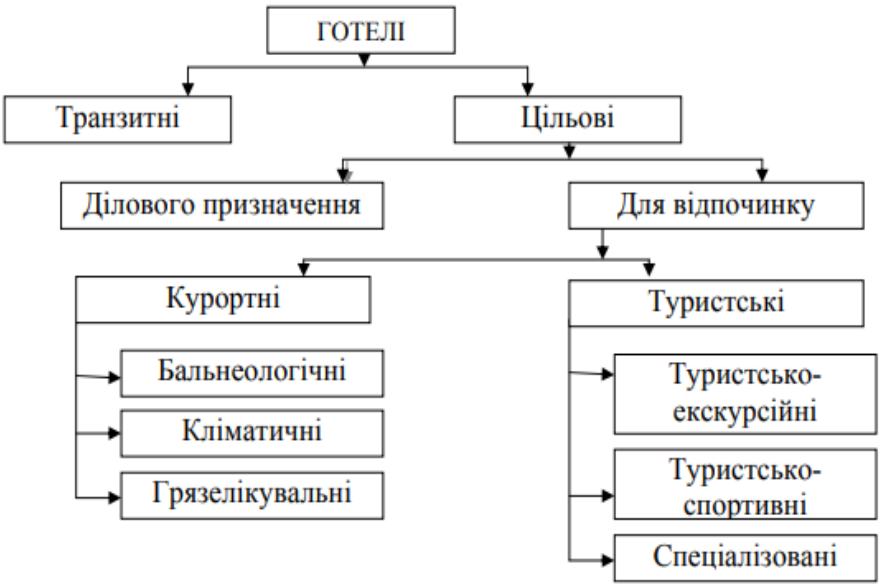

Рисунок В.1 — Класифікація готелів за функціональним призначенням

ДОДАТОК Г

Приклади систем автоматизованого тестування

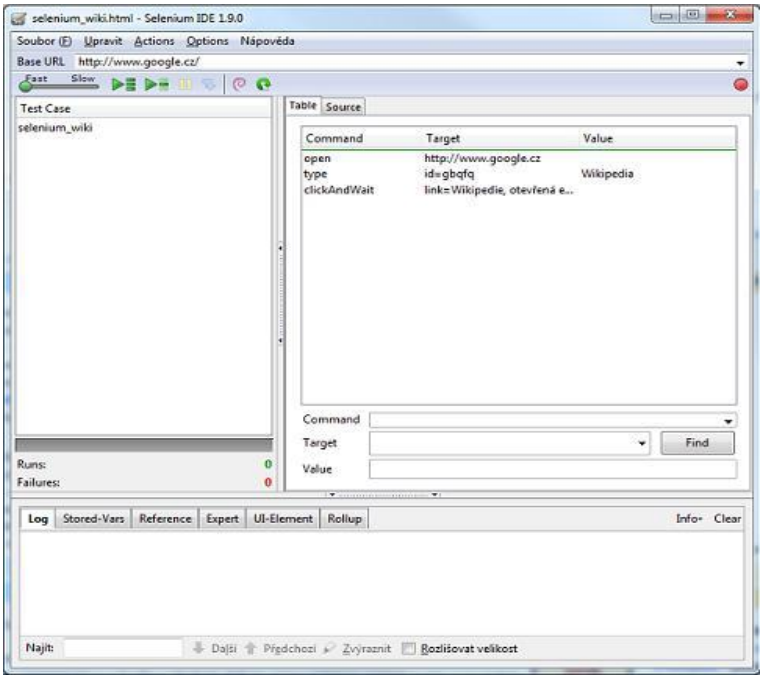

Рисунок Г.1 — Інтерфейс Selenium

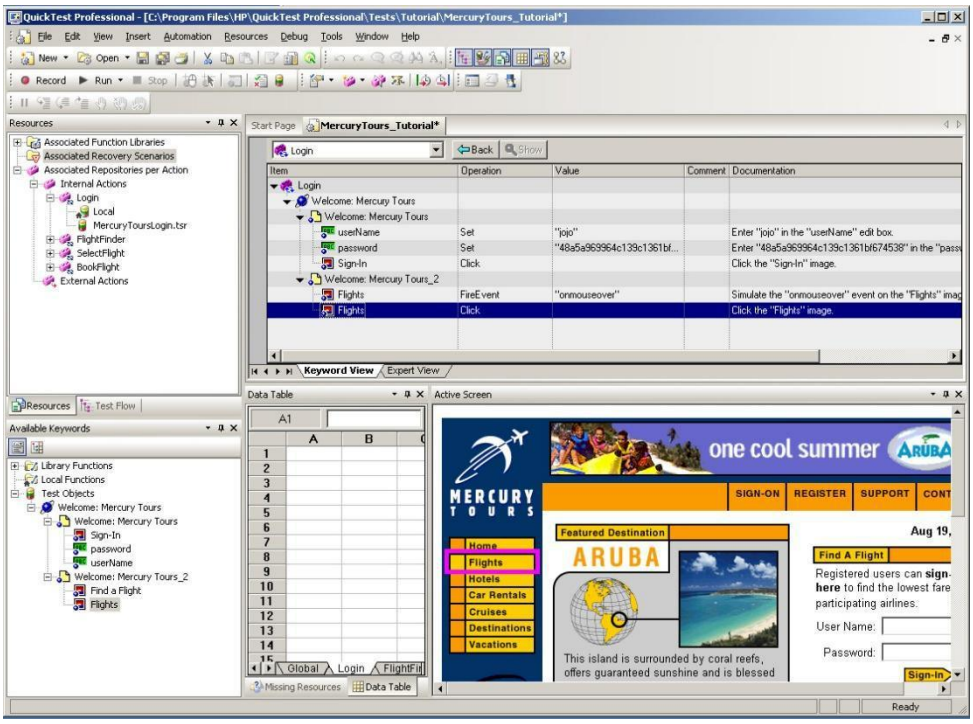

Рисунок Г.2 — Інтерфейс HP QuckTest Professional

ДОДАТОК Д

Архітектура системи

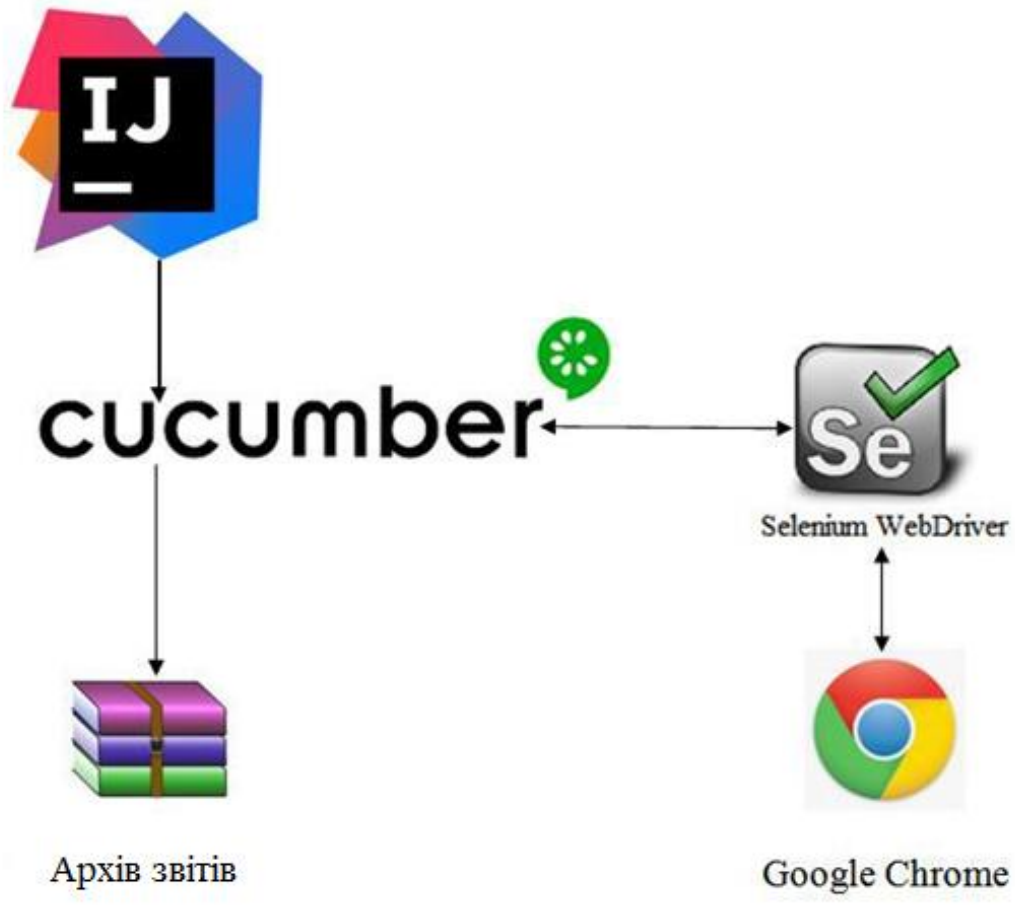

Рисунок Д.1 — Архітектура системи

ДОДАТОК Е

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ НА
НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: _____

_____

_____

Тип роботи: бакалаврська дипломна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 89,2% Схожість 10,8%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.
- □ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- □ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ _____
(підпис) (прізвище, ініціали)

Керівник роботи _____ _____
(підпис) (прізвище, ініціали)