МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА СИСТЕМИ КОНТРОЛЮ ЦІЛІСНОСТІ ВЕБ-ПОРТАЛУ»**

Виконав: студент 2 курсу, групи 8.1221
спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми комп'ютерні науки
(назва освітньої програми)
Я.Г. Зябров
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, доцент, к.т.н. Борю С.Ю.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент завідувач кафедри програмної інженерії, доцент, к.ф.-м.н, Лісняк А.О.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя
2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук, д.т.н., професор
_____Чопоров С.В.
(підпис)

"____" _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Зяброву Ярославу Георгійовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка системи контролю цілісності веб-порталу

Керівник роботи Борю Сергій Юрійович, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

Затверджені наказом ЗНУ від « 05 » 05 2022 р. № 500-с

2. Строк подання студентом роботи

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст роботи(перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Практична реалізація коду.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Форми для прикладу роботи програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.08.2022	
2.	Збір вихідних даних.	11.08.2022	
3.	Обробка методичних та теоретичних	20.08.2022	
	джерел.		
4.	Розробка розділів.	02.09.2022	
5.	Оформлення і нормоконтроль		
	кваліфікаційної роботи.		
6.	Захист кваліфікаційної роботи.		

Студент _____ Я.Г. Зябров
(підпис) (ініціали та прізвище)

Керівник роботи _____ С.Ю. Борю
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка системи контролю цілісності веб-порталу»: 47 с., 11 рис., 14 джерел, 1 додаток.

АВТОМАТИЗАЦІЯ ТЕСТУВАННЯ, «БИТІ» ПОСИЛАННЯ, ІНКАПСУЛЯЦІЯ, ІНТЕРФЕЙС КОРИСТУВАЧА, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, ПОЛІМОРФІЗМ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, ПРОГРАМНИЙ ПРОДУКТ, ПОЛІМОРФІЗМ, СЕРЕДОВИЩЕ РОЗРОБКИ.

Об'єктом дослідження є контроль цілісності веб-порталу.

Предметом дослідження є програмний продукт, що дозволяє контролювати цілісність веб-ресурсів.

Мета роботи – розробка системи контролю цілісності веб-порталу.

Методи дослідження: узагальнення, спостереження, експеримент та класифікація.

Одержані результати дослідження та їх реалізація дозволяють полегшити роботу ІТ-спеціалістам при тестуванні веб-продуктів різноманітного призначення. Готова робота може бути покращена в процесі супроводження програмного продукту.

SUMMARY

Master’s Qualification Thesis "Development of a web portal integrity control system": 47 pages, 11 figures, 14 sources, 1 supplement.

TEST AUTOMATION, BROKEN REFERENCES, ENCAPSULATION, USER INTERFACE, OBJECT-ORIENTED PROGRAMMING, SOFTWARE, SOFTWARE PRODUCT, POLYMORPHISM, DEVELOPMENT ENVIRONMENT.

The object of the study is the control of the integrity of the web portal.

The subject of the research is a software product that allows you to control the integrity of web resources.

Research methods: generalization, observation, experiment and classification.

The obtained research results and their implementation make it easier for IT specialists to test web products for various purposes. The finished work can be improved in the process of supporting the software product.

ЗМІСТ

ВСТУП

У створенні та підтримці сайтів існує багато підводних каменів. Одне з найслабших місць – це поява на сайті помилок «Сторінка не знайдена» або «Not Found». Відсутність сторінок при переході за посиланням дуже дратує користувачів, а для власників ресурсу така помилка є ще й серйозною брехнею, через яку може «йти» трафік. Як упоратися з такою проблемою, знайти всі биті посилання, не втративши гроші та час?

Усередині сайту між усіма сторінками існує зв’язок. Якщо хоч одну зі сторінок видалити, посилання, які до неї вели, автоматично стають неробочими.

Не слід забувати про те, що багато сторінок на сайті створюються тимчасово; коли їхня актуальність проходить, вони видаляються, отже, і посилання на них варто також підчищати.

Зовнішні биті посилання можуть з’явитися у випадку, коли сайт повністю закрився (наприклад, закрилася компанія, якій він належав). Перестали продовжувати домен – сайт автоматично припинив працювати. Саме тому варто регулярно перевіряти сайт на наявність битих посилань, щоб унеможливити такі ситуації.

У ході роботи була створена програма, яка знаходить «биті» посилання на вказаному користувачем веб-ресурсі, що допомагає тестувальникам проводити перевірку сайтів більш автоматизовано.

1 ОСОБЛИВОСТІ Й СПОСОБИ ЗАСТОСУВАННЯ СИСТЕМ КОНТРОЛЮ ЦІЛІСНОСТІ ВЕБ-ПОРТАЛУ

1.1 Аналіз предметної області програмного продукту

Биті посилання – це явище, коли гіперпосилання з часом перестають вказувати на початковий цільовий файл, веб-сторінку чи сервер через те, що цей ресурс переміщується на нову адресу або стає постійно недоступним. Посилання, яке більше не вказує на цільове посилання, яке часто називають непрацюючим або битим посиланням, є особливою формою висячого покажчика.

Швидкість утворення подібних посилань є предметом вивчення та дослідження через їх значення для здатності Інтернету зберігати інформацію. Оцінки цього показника різко відрізняються в різних дослідженнях.

В ряді досліджень вивчалась поширеність створення битих посилань у Всесвітній павутині, в академічній літературі, яка використовує URL-адреси для цитування веб-вмісту, і в цифрових бібліотеках.

Дослідження 2014 року показало, що підмножини веб-посилань (наприклад, ті, що націлені на певні типи файлів або розміщені в академічній установі) можуть мати різко різні періоди напіврозпаду [1] URL-адреси, вибрані для публікації, мають більший термін служби, ніж середня URL-адреса. Дослідження, проведене Weblock у 2015 році, проаналізувало понад 180 000 посилань із посилань у повнотекстових корпусах трьох основних видавців із відкритим доступом і виявило, що період напіврозпаду становить близько 14 років, загалом підтверджуючи дослідження 2005 року, яке виявило, що половина URL-адрес цитовані в статтях D-Lib Magazine були активними через 10 років після публікації.[2] Інші дослідження виявили вищі показники гниття ланок в академічній

літературі, але зазвичай припускають, що період напіврозпаду становить чотири роки або більше [2]. Дослідження 2013 року в BMC Bioinformatics проаналізувало майже 15 000 посилань у анотаціях з індексу цитування Web of Science Thomson Reuters і виявило, що середня тривалість життя веб-сторінок становила 9,3 року, і лише 62% були заархівовані.[3] Дослідження зовнішніх посилань у статтях New York Times у 1996–2019 роках у 2021 році показало, що 25% посилань були недоступними. Крім того, із вибірки з 4500 посилань, які все ще доступні, 13% не ведуть до оригінального вмісту, явище, яке називається дрейфом вмісту.

Дослідження 2002 року показало, що гниття посилань у цифрових бібліотеках відбувається значно повільніше, ніж у Інтернеті, виявивши, що близько 3% об'єктів більше не були доступні через рік [4] (що дорівнює періоду напіврозпаду майже 23 роки).

Загнивання ланок може виникнути внаслідок кількох випадків. Цільову веб-сторінку можна видалити. Сервер, на якому розміщено цільову сторінку, може вийти з ладу, бути видаленим з обслуговування або переміщеним на нове доменне ім'я. Реєстрація доменного імені може закінчитися чи бути передано іншій стороні. Деякі причини призведуть до того, що посилання не знайде жодної цілі та поверне помилку, як-от HTTP 404. Інші причини призведуть до того, що посилання буде націлено на вміст, відмінний від того, який передбачав автор посилання.

Інші причини непрацюючих посилань:

- реструктуризація веб-сайтів, яка спричиняє зміни в URL-адресах (наприклад, domain.net/pine_tree може бути перенесено на domain.net/tree/pine);
- переміщення раніше безкоштовного вмісту на платну основу;
- зміна архітектури сервера, що призводить до того, що код, наприклад PHP, функціонує по-іншому;
- динамічний вміст сторінки, такий як результати пошуку, який змінюється відповідно до дизайну;

- наявність інформації про користувача (наприклад, ім’я для входу) у посиланні;
- навмисне блокування фільтрами контенту або брандмауерам;
- закінчення терміну реєстрації доменного імені.

Стратегії запобігання виникненню подібних посилань можуть бути зосереджені на розміщенні вмісту там, де ймовірність його збереження є вищою, створенні посилань, які з меншою ймовірністю будуть зламані, вживанні заходів для збереження існуючих посилань або виправленні посилань, цілі яких було переміщено чи видалено.

Створення URL-адрес, які не змінюватимуться з часом, є основним методом запобігання даних ситуацій.

Стратегії, що стосуються авторства посилань, включають:

- посилання на первинні, а не на вторинні джерела та надання пріоритету стабільним сайтам ;
- уникнення посилань, які вказують на ресурси на персональних сторінках дослідників[2];
- використання чистих URL-адрес або іншим чином використання нормалізації URL-адрес або канонізації URL-адрес [5];
- уникати посилань на документи, відмінні від веб-сторінок;
- уникнення «глибоких» посилань.

1.2 Аналіз існуючого ПЗ контролю цілісності веб-порталу

Аналізуючи існуючі ПЗ, що допомагають знаходити биті посилання на веб-ресурсах, можна виділити декілька основних і розібрати їх більш детально.

Screaming Frog SEO Spider Tool [6].

Є платна та безкоштовна версії програми. У безкоштовній версії є ряд обмежень у налаштуваннях, вона сканує не більше 500 URL-адрес, але

цілком буде доречною для невеликих сайтів.

Після завантаження та встановлення програми (це десктоп-додаток), вибирається тип пошуку сторінок (рис. 1.1).

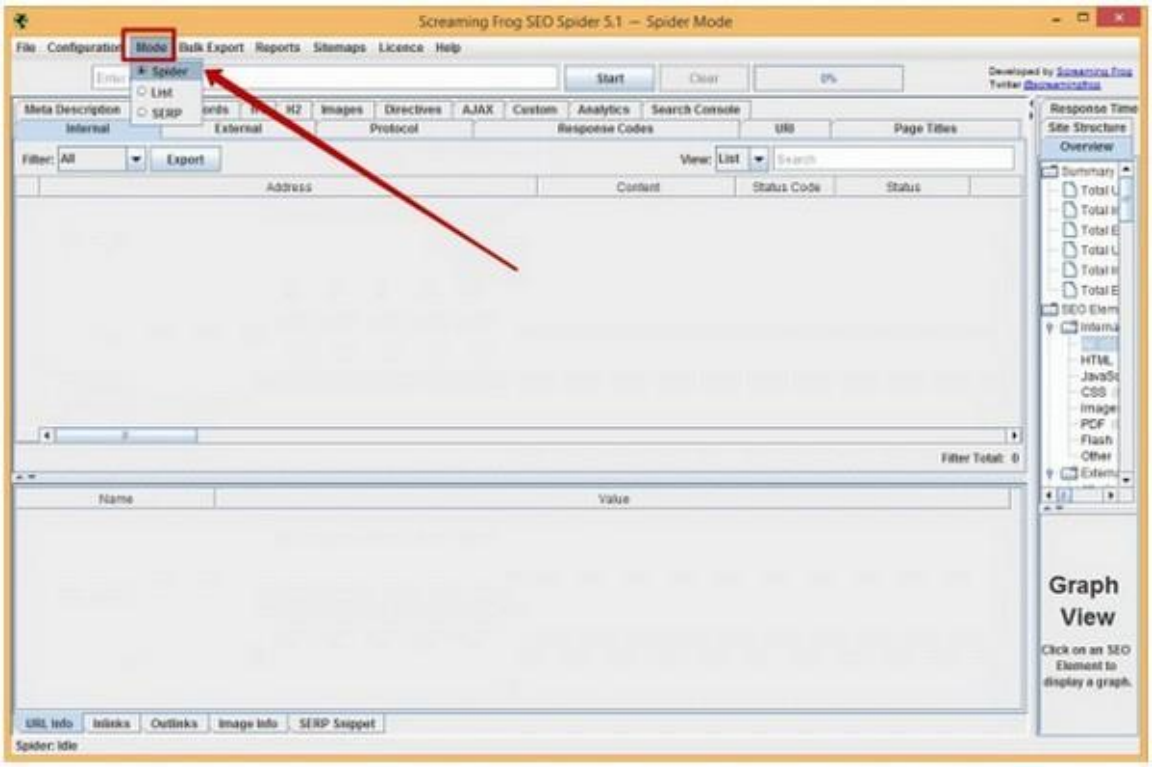

Рисунок 1.1 – ПЗ Screaming Frog SEO Spider Tool

Переваги:

– швидкість роботи (сайт із 4000 URL-адрес (сторінки, зображення тощо) просканував менш ніж за хвилину);
– великий набір налаштувань та ручних фільтрів;
– експорт битих і супутніх посилань в один із трьох форматів: xls, xlsx, csv.

Є клієнти для операційних систем Windows, Mac OS та Ubuntu

Недоліки:

– ціна ліцензії.

Netpeak Spider [6].

Netpeak Spider – десктопна програма для Windows, призначена для сканування та аналізу сайту (рисунок 1.2).

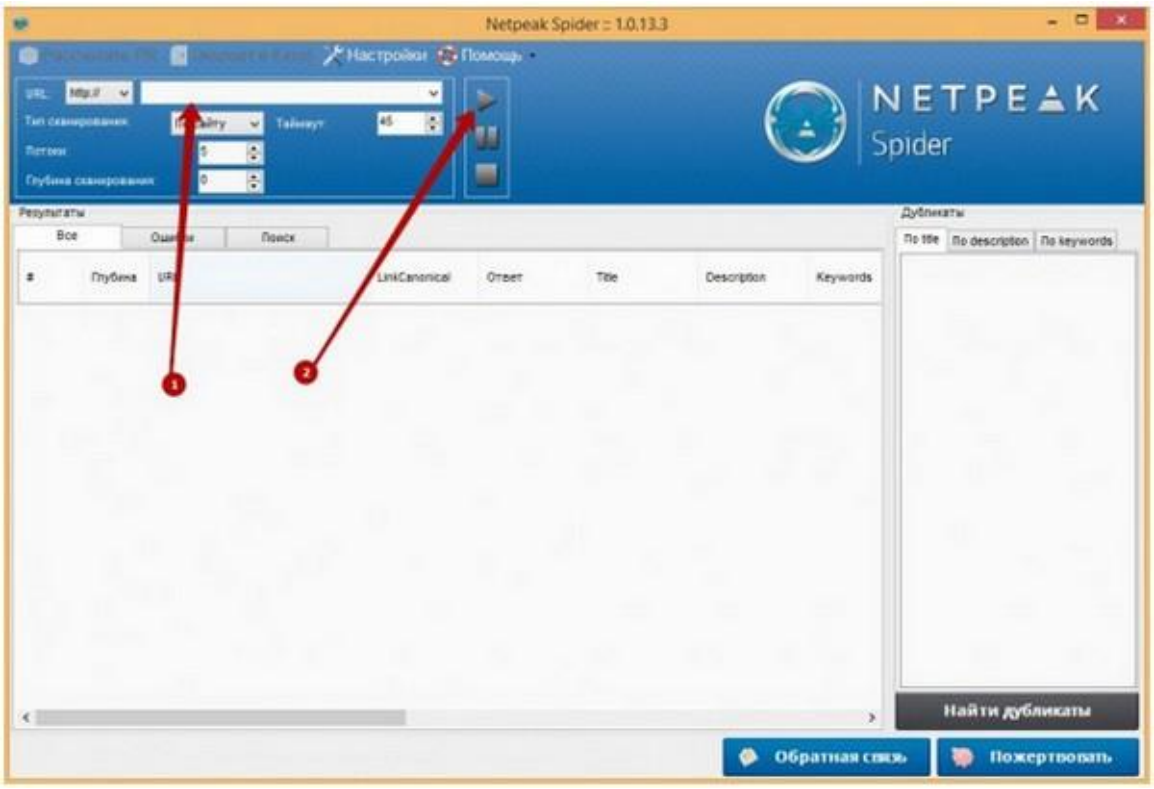

Рисунок 1.2 – ПЗ Netpeak Spider

Переваги:

– безкоштовна;
– простий інтерфейс, розроблений для звичайного користувача.

Недоліки:

– при великих обсягах сканування програма може "гальмувати";
– працює лише на Windows.

Xenu [6].

Спочатку пошук битих посилань на сайті був основним призначенням цієї програми, але згодом вона розширилася і тепер, крім пошуку битих посилань, може виконувати інші функції.

Слід зазначити, що Xenu перевіряє сайт детальніше, ніж дві попередні програми, і тому перевірка займає більше часу.

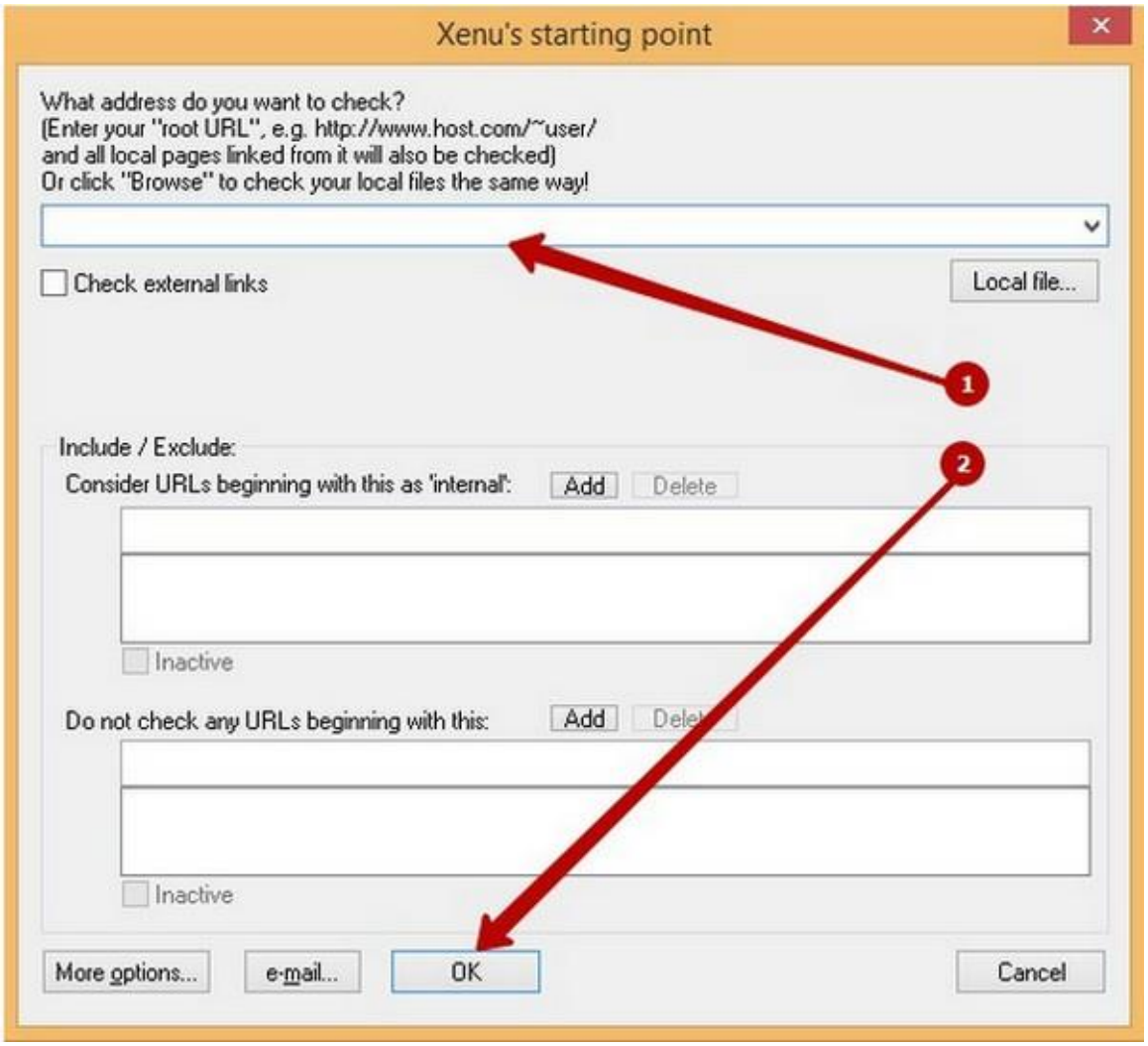

Рисунок 1.3 – ПЗ Xenu

Переваги:

– глибокий аналіз сайту.

Недоліки:

– аналіз великого сайту може забрати чимало часу;

– працює лише під Windows.

Онлайн-сервіс Siteliner [6].

Онлайн-сервіс умовно безкоштовний. Безкоштовно він перевіряє 250 найпопулярніших сторінок сайту, які визначаються на основі внутрішньої структури посилань. Для преміум-акаунту є єдине обмеження – кількість сторінок сайту, яка може перевірити сервіс, – 25 000 сторінок.

Рисунок 1.4 – Онлайн-сервіс Siteliner

Переваги:

– web-сервіс;
– швидкість перевірки вища, ніж у десктоп-додатків;
– наочно відображає бите посилання на сторінці.

Недоліки:

– не виводить список битих посилань, а виводить URL-адреси, де були знайдені биті посилання.

1.3 Постановка задачі

Метою даної роботи є розробка програмного забезпечення, що дозволяє проводити контроль цілісності веб-порталу.

Основною метою програмного забезпечення є спрощення і забезпечення швидкого й точного пошуку некоректних, застарілих та зіпсованих посилань.

Для досягнення мети в роботі вирішуються наступні задачі:

– аналіз існуючого ПЗ з пошуку битих посилань;
– розробка вимог до ПЗ;
– розробка архітектури ПЗ;
– формалізація режимів роботи ПЗ;
– розробка алгоритмів для кожного режиму та його функціоналу;
– вибір середовища розробки (вибір IDE);
– розробка програмного коду (класів, шаблонів, бібліотек);
– розробка технічної документації та інструкції по використанню;
– тестування ПЗ.

1.4 Висновки

У даному розділі була проаналізована предметна область програмного продукту, що розробляється, його актуальність та поточний стан в обраній сфері. Аналіз був побудований на даних, знайдених з публічних ресурсів, а саме більшість на наукових публікаціях, що досліджують обрану тему.

Були виокремлені переваги й недоліки кожного із обраних для порівняння програмних забезпечень, описані їх основні призначення, а також висунуті рекомендації, які б допомогли розробникам й користувачам у виконанні тих чи інших завдань.

Після проведення аналізу, був розглянутий необхідний інструментарій, що дозволяє виконати розробку програмного забезпечення для реалізації вирішення проблеми пошуку некоректних посилань на будь-яких веб-ресурсах. Далі була визначена постановка задачі, в якій описано кроки для досягнення поставленої мети на підставі проаналізованих даних.

2 РОЗРОБКА ІНТЕРФЕЙСУ ТА АРХІТЕКТУРИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Формалізація задач, що розв’язуються при розробці ПЗ

Підготовчий етап процесу розробки програмного забезпечення має для виконавця дуже просту мету – прийняти рішення, чи варто братися за реалізацію програмного продукту. З погляду замовника системи метою цього етапу є ухвалення рішення у тому, чи можна довіряти виконавцю. При успішному розвитку подій природним підсумком підготовчого етапу має стати укладання договору (або іншої угоди щодо ситуації) на створення або модифікацію системи, необхідної замовнику.

Для досягнення поставлених цілей замовнику та виконавцю спільно потрібно вирішити низку цілком певних завдань:

- на основі вихідної ідеї сформулювати цілі та завдання майбутнього проекту;
- розробити деяке вихідне бачення – концепцію проекту;
- провести аналіз затребуваності майбутнього продукту;
- провести попередню оцінку ризиків майбутнього проекту;
- на основі концепції та списку попередніх ризиків підготувати попереднє технічне рішення;
- вибрати методологію розробки та підготувати попередній план робіт;
- провести попередню оцінку трудовитрат та необхідних ресурсів;
- провести аналіз реалізованості продукту;
- провести незалежне рецензування технічного рішення;
- ухвалити рішення про те, чи варто продовжувати роботи.

Вихідна інформація буває дуже мізерною. Найчастіше, це просто деяка ідея, яка спала на думку замовнику, керівнику або комусь із членів команди. Чи перетвориться ця ідея на реальний продукт, вирішується багато в чому саме на підготовчому етапі.

Свій внесок у тривалість етапу робить, звичайно, і сама природа проекту. Якщо має бути робота на замовлення державних структур чи компанії з державною участю, то швидше за все на підготовчому етапі йтиметься про підготовку до тендеру. В цьому випадку на всіх етапах доводиться діяти з максимальним ступенем формалізації, що, звісно, не прискорює роботу. Якщо майбутній проект робиться на замовлення внутрішнього замовника, то частіше, навпаки, доводиться гасити запал надто нетерплячих менеджерів, щоб дати можливість виконавцям дати реалістичні оцінки.

При формулюванні мети та завдань проекту має бути створений документ, що дає відповіді на такі питання:

Як визначається предметна область для цього проекту? Які терміни використовуються у предметній галузі? Які бізнес-процеси, які стосуються проекту, протікають у предметній галузі?

Яка мета у проекту? Який ефект має зробити система, що розробляється на бізнес-процеси предметної області?

Які завдання потрібно вирішити, щоб досягти поставленої мети? За якими критеріями оцінюватиметься якість вирішення поставлених завдань? Які функціональні та нефункціональні показники якості системи, що розробляється?

Які обмеження на терміни виконання, ресурси, бюджет накладаються на реалізацію проекту?

Відповіді на ці питання мають дозволити виконавцям сформулювати концепцію проекту та дати оцінку його затребуваності та реалізованості. Для замовника коректність відповідей на зазначені питання важлива, тому

що ця інформація задає критерії, за якими замовник зможе вирішити, чи варто довіряти роботу тому чи іншому виконавцю.

Мета проекту задає напрямок розвитку проекту. Вона повинна коротко і ясно вказати ефект, який вплине на бізнес-процеси замовника. Якщо мета буде вказана неточно, проект розвиватиметься в неправильному напрямку. Розгорнути проект в іншому напрямку дуже складно, і чим більше буде зроблено робіт у рамках проекту, тим складніше коригуватиме напрямок його розвитку. Незабаром проект досягає стану, коли помилку в постановці мети виправити буде неможливо, і тоді доведеться все починати наново, або зовсім відмовлятися від проекту.

Для досягнення мети може знадобитися робота не лише одного виконавця. Наприклад, для створення системи може знадобитися розробка спеціального обладнання або доробка деякого програмного забезпечення. Тому постановка завдань, які необхідно виконати конкретному виконавцю у межах проекту, важливо як і формулювання мети. Виконавець повинен розуміти свою роль у проекті, межі взаємодії із замовником та іншими виконавцями та інші важливі для проекту нюанси.

При постановці завдань потрібно приділити особливе місце формулюванню критеріїв, за якими проводитиметься приймання робіт. Показники якості системи може задавати тільки замовник, і вони є важливими при оцінці реалізованості проекту. Повинні бути зазначені основні показники якості як функціональні, так і нефункціональні [7, с. 219].

Розроблений програмний продукт дозволяє виконувати пошук некоректних посилань за посиланням на веб-ресурс. Спочатку програма знаходить всі елементи сайту з тегом <a>, потім починає перевіряти кожен елемент, отримуючи атрибут <href>. Далі програма намагається відправити запит на URL з атрибутом <href>. У разі отримання відповіді з кодом 400 і вище, посилання вважається некоректним.

2.2 Розробка інформаційної моделі та алгоритму ПЗ

Існує два підходи до створення інформаційних моделей:

– об’єктно-орієнтований;

– структурний.

Об’єктно-орієнтований підхід використовує об’єктну декомпозицію, статична структура системи описується в термінах об’єктів і зв’язків між ними, а поведінка системи описується в термінах обміну повідомленнями між об’єктами. Кожен системний об’єкт має власну поведінку, яка імітує поведінку реального об’єкта.

Концептуальною основою об’єктно-орієнтованого підходу є об’єктна модель. Найважливішими елементами є:

– абстракція;

– інкапсуляція;

– модульність;

– ієрархія.

Окрім основних елементів, є три додаткові елементи, які, на відміну від основних елементів, не обов’язково є обов’язковими:

– типізація;

– паралелізм;

– стабільність.

Абстрагування – це виділення істотних ознак об’єкта, які відрізняють його від інших типів об’єктів і таким чином чітко визначають його концептуальні межі для подальшого розгляду та аналізу. Абстрагування акцентує увагу на зовнішніх ознаках об’єкта і дозволяє відокремити основні риси його поведінки від деталей реалізації. Вибір правильного набору абстракцій для даної предметної області є основним завданням об’єктно-орієнтованого проектування.

Інкапсуляція – це процес відокремлення окремих елементів об’єкта, які визначають його структуру та поведінку. Метою інкапсуляції є ізоляція інтерфейсу об’єкта, який відображає його зовнішню поведінку, від внутрішньої реалізації об’єкта. Об’єктно-орієнтований підхід передбачає, що власні ресурси класу, якими можна маніпулювати лише за допомогою методів самого класу, приховані від середовища.

Абстракція та інкапсуляція є взаємодоповнюючими операціями: абстракція привертає увагу до зовнішніх характеристик об’єкта, тоді як інкапсуляція (або інше обмеження доступу) не дозволяє об’єктам користувача бачити внутрішні елементи об’єкта.

Модульність – властивість системи, пов’язана з можливістю її декомпозиції на ряд внутрішньо пов’язаних, але слабо пов’язаних між собою модулів. Інкапсуляція та модульність створюють бар’єри між абстракціями [8, с.115].

Ієрархія – це впорядкована система абстракцій, розташованих за рівнями. Основними видами ієрархічних структур, що відносяться до складних систем, є структура класів (ієрархія за номенклатурою) і структура об’єктів (ієрархія за складом). Прикладами ієрархій класів є одиночне та множинне успадкування (клас використовує структурну або функціональну частину відповідно до одного чи кількох інших класів), а ієрархії об’єктів – це агрегація.

Типізація – це обмеження, накладене на клас об’єктів, яке запобігає (або сильно обмежує) взаємозамінність різних класів. Вводячи текст, ви можете захистити або принаймні контролювати використання об’єктів одного класу замість іншого.

Паралельність – властивість об’єктів перебувати в активному або пасивному стані та розрізняти активні та пасивні об’єкти.

Постійність – це властивість об’єкта існувати, але в часі (незалежно від процесу, який створив цей об’єкт) та/або просторово (коли об’єкт переміщується з адресного простору, у якому він був створений).

Основними поняттями об’єктно-орієнтованого підходу є об’єкт і клас. Під об’єктом розуміється відчутна реальність – об’єкт або явище, що має чітко визначену поведінку. Об’єкт має стан, поведінку та особистість; структура і поведінка подібних об’єктів визначають їх загальний клас.

Терміни «екземпляр класу» та «об’єкт» еквівалентні. Стан об’єкта позначається списком усіх можливих (статичних) властивостей цього об’єкта та поточними значеннями (динамічними) кожної з цих властивостей. Іншими словами, поведінка об’єкта цілком визначається його діями.

Індивідуальність – це властивості предмета, які відрізняють його від усіх інших предметів. У мовах операції, що виконуються над певним об’єктом, називаються методами і є частиною визначення класу.

Клас – це набір об’єктів, пов’язаних спільною структурою та поведінкою. Кожен об’єкт є екземпляром класу. Визначення класів і об’єктів є одним із найскладніших завдань об’єктно-орієнтованого проектування. Наступною групою важливих об’єктно-орієнтованих концепцій є успадкування та поліморфізм.

Поняття поліморфізму можна інтерпретувати як здатність класу належати до більш ніж одного типу.

Успадкування означає створення нових класів на основі існуючих із можливістю додавати або замінювати дані та методи. Об’єктно-орієнтована система спочатку будується з урахуванням її еволюції.

Спадкування та поліморфізм надають можливість визначати нову функціональність класів шляхом створення похідних класів - потоків базових класів. Нащадки успадковують властивості від батьківських класів, не змінюючи їх вихідний опис, додаючи власні структури даних і методи за потреби.

Визначення похідних класів, які вказують лише на відмінності або вдосконалення, заощаджує багато часу та зусиль на створення та використання специфікацій і коду.

Важливою якістю об’єктного підходу є узгодженість моделей діяльності організації та моделей проектованої системи від фази формування вимог до фази впровадження. Вимога узгодженості моделей виконується завдяки можливості використання абстракції, модульності та поліморфізму на всіх етапах розробки.

Моделі на ранній стадії можна прямо порівняти з моделями впровадження. Об’єктні моделі можна використовувати для відтворення відображення реальних сутностей предметної області, що моделюється (організації), на об’єкти та класи інформаційної системи.

Концепція роботи створеного проекту виглядає наступним чином (рис. 2.1):

Рисунок 2.1 – Принцип роботи створеного програмного продукту

Варто вказати, що при виконанні даної кваліфікаційної роботи мною був використаний об’єктно-орієнтований підхід для створення інформаційної моделі через наявність великої кількості переваг та можливостей.

2.3 Розробка користувацького інтерфейсу

2.3.1 Основи користувацького інтерфейсу

Інтерфейс користувача – це набір програмних та апаратних засобів, що забезпечують взаємодію користувача з комп’ютером. Основу такої взаємодії становлять діалоги [9, с.87].

Під діалогом у разі розуміють регламентований обмін інформацією між людиною і комп’ютером, здійснюваний у часі і спрямований на спільне рішення конкретної задачі. Кожен діалог складається з окремих процесів введення/виводу, які фізично забезпечують зв’язок користувача та комп’ютера. Обмін інформацією здійснюється передачею повідомлення.

Завдання користувача комп’ютерної програми полягають у маніпуляції з об’єктом та його властивостями – даними. На відміну від операторів, користувачі виконують професійне завдання з іншою психологічною структурою дій, іншими цілями, об’єктом праці та операціями, ресурсами, іншим соціальним середовищем взаємодії.

Різноманітність ситуацій, в яких можуть працювати інтерактивні програмні системи, ускладнює для розробника вибір цілей, які необхідно виконувати для створення вдалого інтерфейсу. Різні дослідники та організації-розробники програмного забезпечення наводять різні рекомендації, але основні з них такі:

Простота. Ця рекомендація походить від правила бритви Оккама: найкраще пояснення – найпростіше. Справді, простий інтерфейс дозволяє користувачеві швидше адаптуватися, зменшує ймовірність його помилок, та й розробнику простіше налагодити такий інтерфейс. Інтерактивна система хороша, якщо інтерфейс інтуїтивно зрозумілий, тобто відповідає предметній області та стилю мислення користувача. Інтерфейс повинен бути легким для освоєння і не створювати перед користувачем перешкоду, яку він повинен буде подолати, щоб розпочати роботу.

Дружність. Інтерфейс дружній, якщо користувач, працюючи з ним, не відчуває дискомфорту. У користувача має складатися враження, що він керує процесом. Крім того, графічний інтерфейс повинен бути побудований відповідно до ергономічних вимог: кольору екрану та елементів, їх розмір, композиція. Важливим є темп виконання операцій, який має відповідати природному темпу людини, середній час відгуку та його дисперсія. Повідомлення мають бути коректними за формою, точними та інформативними, абсолютно неприпустимі безграмотні тексти. Користувач повинен знати, на якій стадії процесу він знаходиться.

Природність інтерфейсу. Природний інтерфейс – такий, який змушує користувача істотно змінювати звичні йому способи вирішення завдання. Це, зокрема, означає, що повідомлення та результати, що видаються додатком, не повинні вимагати додаткових пояснень.

Функціональність. Хоча обчислювальна система і буває у деяких організаціях у ролі великої іграшки, але найчастіше її намагаються використовувати для справи, особливо у разі, коли виконання роботи іншими засобами менш ефективно. Функціональність системи позначає наявність значної ефективності виконання операцій, що робить її використання рентабельним. Інтерфейс повинен відображати її функціональність та давати можливість успішної роботи користувачам різної кваліфікації.

Помірна вартість. Зрозуміло, що система, що має дуже дорогий інтерфейс, але недостатню функціональність, буде, можливо, куплена, але користувач залишиться нею незадоволений: термін окупності системи залежить від функціональності. З іншого боку, економія на інтерфейсі – дуже недалекоглядна політика. Неякісний інтерфейс створює в користувачів погану думку про систему і взагалі може призвести до відмови від її використання.

Стадії проектування, як і весь процес створення інтерфейсу, дуже схожі на стадії в процесі розробки інформаційної системи в цілому:

- аналіз діяльності користувачів. Це передпроектна стадія, де визначаються завдання, процедури, уточняється характер виробництва, контингент користувачів тощо;
- формалізація результатів аналізу у вигляді схем та діаграм бізнес-процесів та сценаріїв виконання кожного завдання;
- проектування інтерфейсу для забезпечення кожного сценарію та процесу. Синтез рішення як прототипу інтерфейсу;
- тестування з користувачами прототипу чи готового інтерфейсу. Синтез рішення (малювання екранних форм) часто займає набагато менше часу, ніж етап аналізу [10, с.125].

Можна зустріти й інший, але схожий підхід, де процес проектування розбивається на 6 етапів:

- планування та оцінка;
- складання вимог до проекту;
- дизайн та проектування;
- реалізація та програмування;
- тестування та оцінка;
- випуск.

2.3.2 Типи інтерфейсів

Інтерфейси користувача бувають двох типів:

Процедурно-орієнтовані

Тут використовується традиційна модель взаємодії з користувачем, заснована на поняттях «процедура» та «операція». В рамках цієї моделі програмне забезпечення надає користувачеві можливість виконання деяких дій, для яких користувач визначає відповідність даних та наслідком

виконання яких є отримання бажаного результату. Процедурно-орієнтовані інтерфейси:

- забезпечують користувачеві функції, необхідні виконання завдань;
- акцент робиться на завдання;
- піктограми представляють додатки, вікна чи операції;
- зміст папок та довідників відображається за допомогою таблиці-списку.

Інтерфейси цього типу поділяються на:

Примітивний. Це інтерфейс, який організовує взаємодію з користувачем і використовується в консольному режимі. Єдине відхилення від послідовного процесу, що забезпечується даними, полягає в організації циклу для обробки кількох наборів даних.

Інтерфейс Меню. На відміну від примітивного інтерфейсу, дозволяє користувачеві вибирати операцію зі спеціального списку, що виводиться програмою. Ці інтерфейси передбачають реалізацію безлічі сценаріїв роботи, послідовність дій у яких визначається користувачами. Деревоподібна організація меню передбачає суворо обмежену реалізацію. При цьому можливі два варіанти організації меню: кожне вікно меню займає весь екран; на екрані одночасно присутні кілька різнорівневих меню (Windows). В умовах обмеженої навігації, незалежно від варіанта реалізації, пошук пункту більш ніж двох рівнів меню виявляється досить складним завданням.

Інтерфейс із вільною навігацією (графічний інтерфейс). Підтримує концепцію інтерактивної взаємодії з ПЗ, візуальний зворотний зв’язок з користувачем та можливість прямого маніпулювання об’єктом (кнопки, індикатори, рядки стану). На відміну від інтерфейсу Меню, інтерфейс із вільною навігацією забезпечує можливість здійснення будь-яких допустимих у конкретному стані операцій, доступ до яких можливий через різні компоненти (гарячі клавіші і т.д.). Інтерфейс з вільною навігацією

реалізується з використанням програмування подій, що передбачає застосування візуальних засобів розробки (за допомогою повідомлень) [11, с.301].

Об'єктно-орієнтований інтерфейс

Об'єктно-орієнтований інтерфейс використовує модель взаємодії з користувачем, яка орієнтована на маніпулюванні об'єктами предметної області. У рамках цієї моделі користувачеві надається можливість безпосередньо взаємодіяти з кожним об'єктом та ініціювати виконання операцій, у процесі яких взаємодіють кілька об'єктів. Завдання користувача формулюється як цілеспрямована зміна об'єкта. Об'єкт розуміється у сенсі слова модель БД, системи тощо. Такий інтерфейс передбачає, що взаємодія з користувачем здійснюється у вигляді вибору та переміщення піктограм відповідної об'єктно-орієнтованої області. Розрізняють одно-документні (SDI) та багатодокументні (MDI) інтерфейси.

2.3.3 Визначення й створення інтерфейсу

В пункті 2.2 даної роботи була розроблена концепція програмного продукту. Варто розглянути інтерфейс користувача більш детально.

На рисунку 2.2 позначені всі кнопки і поля. Треба розглянути кожен з елементів та пояснити їх призначення:

- 1 – поле, в якому користувач вводить адресу веб-ресурсу, який він бажає перевірити на наявність некоректних посилань;
- 2 – кнопка «Scan», після натискання якої програма починає проводити перевірку вказаного веб-ресурсу;
- 3 – поле з назвою «Valid links», не виконує ніяких функцій, спрямована лише для навігації користувача по результату виконання програми;

– 4 – поле, в якому перераховані коректні посилання вказаного веб-ресурсу;
– 5 – поле з назвою «Broken links», не виконує ніяких функцій, спрямована лише для навігації користувача по результату виконання програми;
– 6 – поле, в якому перераховані некоректні посилання вказаного веб-ресурсу.

Рисунок 2.2 – Інтерфейс користувача

3 РОЗРОБКА ПРОГРАМНОГО КОДУ

3.1 Вибір інструментарію й середовища розробки

IntelliJ IDEA – це інтегроване середовище розробки для мов JVM, призначене для максимального підвищення продуктивності розробника. Він виконує рутинні та повторювані завдання замість розробника ПЗ, забезпечуючи розумне завершення коду, статичний аналіз коду та рефакторинг, і дозволяє зосередитися на яскравій стороні розробки програмного забезпечення, роблячи її не лише продуктивною, але й приємною [12].

Розробка сучасних програм передбачає використання кількох мов, інструментів, фреймворків і технологій. IntelliJ IDEA розроблено як IDE для мов JVM, але численні плагіни можуть розширити його, щоб забезпечити багатомовний досвід.

Переваги обраного середовища розробки:

- функціональність;
- великий вибір інструментів для роботи з кодом;
- ергономічність;
- комфортність;
- наявність інструментів для сумісної та віддаленої роботи;

Недоліки:

- вибагливість до системних ресурсів;
- урізана функціональність безкоштовної версії;
- відсутність перекладу інтерфейсу на українську мову.

В процесі ознайомлення з середовищем розробки IntelliJ IDEA було зроблено висновок, що дане середовище найкраще підходить для програмної реалізації власного програмного продукту.

3.2 Розробка системи класів

Як було зазначено вище, для розробки модуля був обраний підхід об’єктно-орієнтованого програмування (далі ООП). Основними принципами ООП є інкапсуляція, спадкування й поліморфізм. Визначення даних принципів були висвітлені в попередніх розділах. Варто виділити основні переваги та недоліки застосування ООП в програмуванні.

До основних переваг можна віднести:

Модульність. Об’єктно-орієнтований підхід дозволяє зробити код більш структурованим, в ньому легко розібратися сторонній людині. Завдяки інкапсуляції об’єктів зменшується кількість помилок та прискорюється розробка за участю великої кількості програмістів, тому що кожен може працювати незалежно один від одного.

Гнучкість. ООП-код легко розвивати, доповнювати та змінювати. Це забезпечує незалежна модульна структура. Взаємодія з об’єктами логікою спрощує розуміння коду. Для модифікації не потрібно поринати в те, як побудовано ПЗ. Завдяки поліморфізму можна швидко адаптувати код до вимог завдання, не описуючи нові об’єкти та функції.

Економія часу. Завдяки абстракції, поліморфізму та успадкуванню можна не писати один і той же код багато разів. Це прискорює розробку нового програмного забезпечення. Інтерфейси та класи в ОВП можуть легко перетворюватися на подібність до бібліотек, які можна використовувати заново в нових проектах. Також ООП економить час за підтримки та доопрацювання програми.

Безпека. Програму складно зламати, оскільки інкапсульований код недоступний ззовні [13, с. 412].

Серед недоліків ООП виокремлюють наступні:

Складний старт. Щоб користуватися ООП, потрібно спочатку вивчити теорію та освоїти процедурний підхід, тому поріг входу високий.

Зниження продуктивності. Об’єктно-орієнтований підхід трохи знижує продуктивність коду загалом. Програми працюють дещо повільніше через особливості доступу до даних та велику кількість сутностей.

Великий розмір програми. Код, написаний з використанням ООП, зазвичай довший і займає більше місця на диску, ніж "процедурний". Це відбувається тому, що в такій програмі зберігається більше конструкцій, ніж у звичайному процедурному скрипті [14, с.356].

Програма містить наступні класи:

– public Main() – конструктор, в якому оголошуються основні параметри інтерфейсу користувача: розмір вікна, кнопки, текстові поля, тощо. В цьому класі розроблено багато методів для додавання створених об’єктів у вікно. Наприклад, add(), labelValidLinks.setAlignment(), button.addActionListener(), setTitle(), setVisible().

```
public Main () {
    setLayout(new GridLayout(3, 2));
    textField = new TextField("Enter your link to scan", 40);
    textField.setEditable(true);
    add(textField);
    button = new Button("Scan");
    add(button);
    labelValidLinks = new Label("Valid links:");
    labelValidLinks.setAlignment(Label.CENTER);
    add(labelValidLinks);
    textAreaValidLinks = new TextArea(5, 40);
    textAreaValidLinks.setForeground(Color.blue);
    textAreaValidLinks.setEditable(false);
    add(textAreaValidLinks);
    labelInvalidLinks = new Label("Broken links:");
    labelInvalidLinks.setAlignment(Label.CENTER);
```

```
        add(labelInvalidLinks);
        textAreaInvalidLinks = new TextArea(5, 40);
        textAreaInvalidLinks.setForeground(Color.red);
        textAreaInvalidLinks.setEditable(false);
        add(textAreaInvalidLinks);
        BtnCountListener listener = new BtnCountListener();
        button.addActionListener(listener);
        addWindowListener(new MyWindowListener());
        setTitle ("Link Scanner");
        setExtendedState(Frame.MAXIMIZED_BOTH);
        setVisible(true);
    }
```

– private class BtnCountListener – відповідає за дії після того, як користувач натисне на кнопку «Scan». Саме тут відбувається пошук некоректних посилань. Метод, що реалізує дану дію має назву actionPerformed() та передає параметр ActionEvent evt. Також в цьому класі використовуються методи button.setEnabled() для того, щоб зробити кнопку видимою для користувача та textField.getText(), який буде зберігати текст, введений користувачем.

```
private class BtnCountListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent evt) {
            button.setEnabled(false);
            textAreaValidLinks.setText("");
            textAreaInvalidLinks.setText("");
            ChromeOptions options = new ChromeOptions();
            options.addArguments("--headless");
            String homePage = textField.getText();
            String url = "";
```

```
HttpURLConnection huc = null;
int respCode = 200;
driver = new ChromeDriver(options);
try
{
   driver.get(homePage);
   List<WebElement> links =
driver.findElements(By.tagName("a"));
   Iterator<WebElement> it = links.iterator();
   while(it.hasNext()){
      WebElement tempLink = it.next();
      url = tempLink.getAttribute("href");
      if(url == null || url.isEmpty()){
         continue;
      }
      try {
         huc = (HttpURLConnection)(new
URL(url).openConnection());
         huc.setRequestMethod("HEAD");
         huc.connect();
         respCode = huc.getResponseCode();
         if(respCode >= 400){
            textAreaInvalidLinks.append(url + "\n");
         }
         else{
            textAreaValidLinks.append(url + "\n");
         }
      } catch (MalformedURLException e) {
         e.printStackTrace();
      } catch (IOException e) {
```

```
                    e.printStackTrace();
                }
            }
        }
        catch(Exception e)
        {
            driver.quit();
            button.setEnabled(true);
        }
        driver.quit();
        button.setEnabled(true);
    }
}
```

– private class MyWindowListener – відповідає за завершення програми у випадку, коли користувач закриває вікно програми. Тут використовуються методи windowOpened(), windowClosed(), windowIconified(), windowDeiconified(), windowActivated() та windowDeactivated() що відповідають за різноманітні стани активного вікна (закриття, деактивація, активація користувачем тощо).

```
private class MyWindowListener implements WindowListener {
    @Override
    public void windowClosing(WindowEvent evt) {
        System.exit(0);
    }
    @Override public void windowOpened(WindowEvent evt) { }
    @Override public void windowClosed(WindowEvent evt) { }
    @Override public void windowIconified(WindowEvent evt) {
System.out.println("Window Iconified"); }
```

```
    @Override public void windowDeiconified(WindowEvent evt) {
System.out.println("Window Deiconified"); }
    @Override public void windowActivated(WindowEvent evt) {
System.out.println("Window Activated"); }
    @Override public void windowDeactivated(WindowEvent evt) {
System.out.println("Window Deactivated"); }
  }
```

3.3 Тестування ПЗ

При тестуванні ПЗ були перевірені кілька десятків сайтів різноманітного призначення для знаходження в них битих посилань. Для користувача необхідно виконати єдину дію – задати посилання на веб-ресурс і натиснути на кнопку «Scan», а програма при виконанні виведе список перевірених посилань з сортуванням за різноманітними параметрами. Знайдені посилання виділяються наступними кольорами:

- синій – посилання доступне;
- червоний – знайдено некоректне посилання або посилання тимчасово недоступне.

Приклад роботи застосунку показано на рисунках: 3.1, 3.2, 3.3, 3.4, 3.5. Перевіряються веб-ресурси:

- https://www.znu.edu.ua/;
- https://twitter.com/;
- https://freesound.org/;
- https://www.mixamo.com/#/ ;
- https://www.cprogramming.com/ .

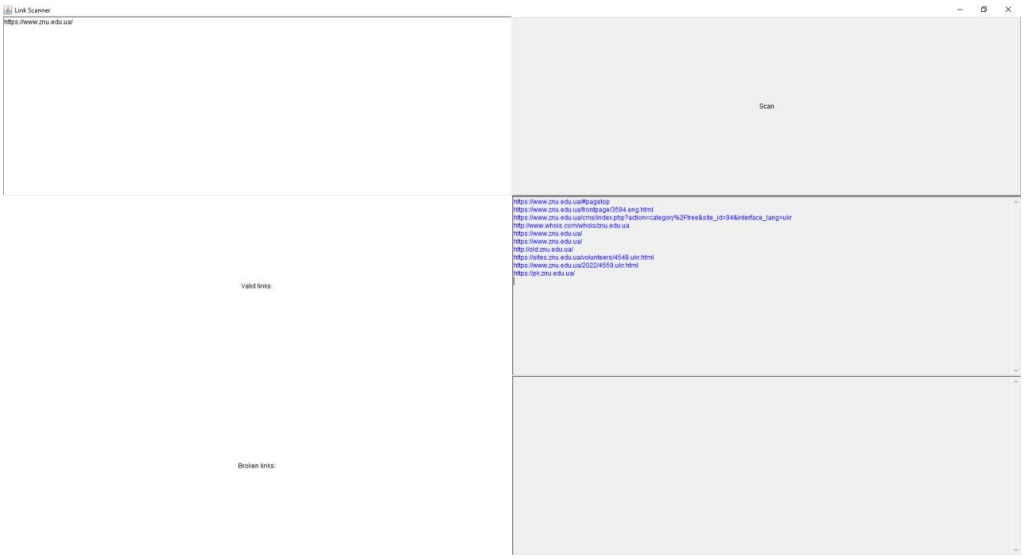

Рисунок 3.1 – Перевірка працездатності готового програмного продукту на сайті https://www.znu.edu.ua/

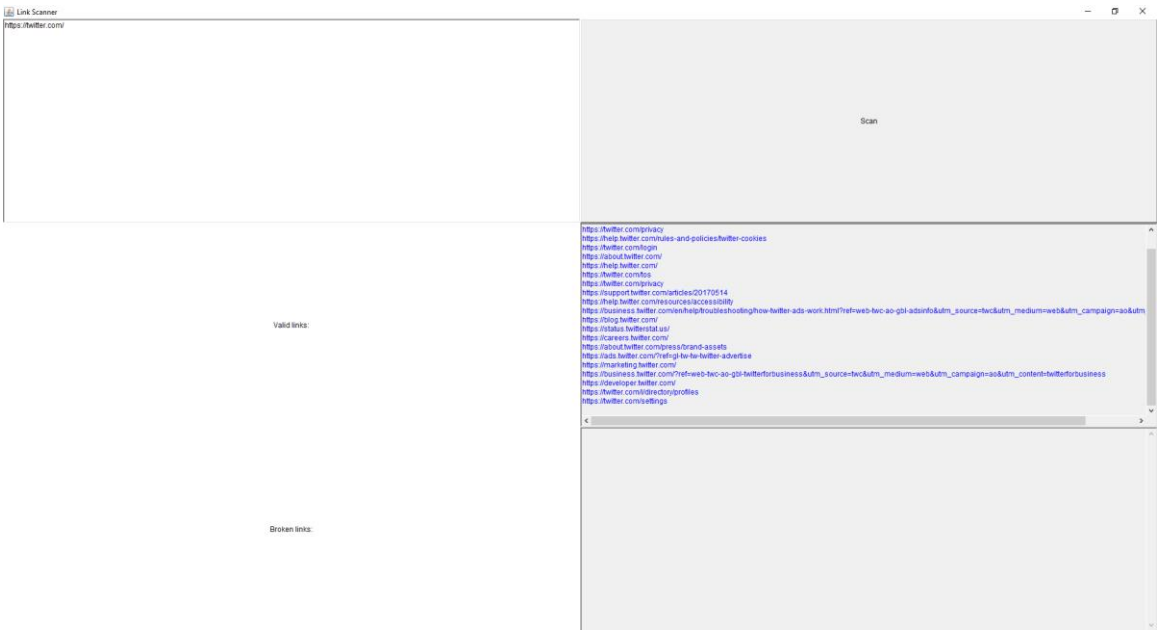

Рисунок 3.2 – Перевірка працездатності готового програмного продукту на сайті https://twitter.com/

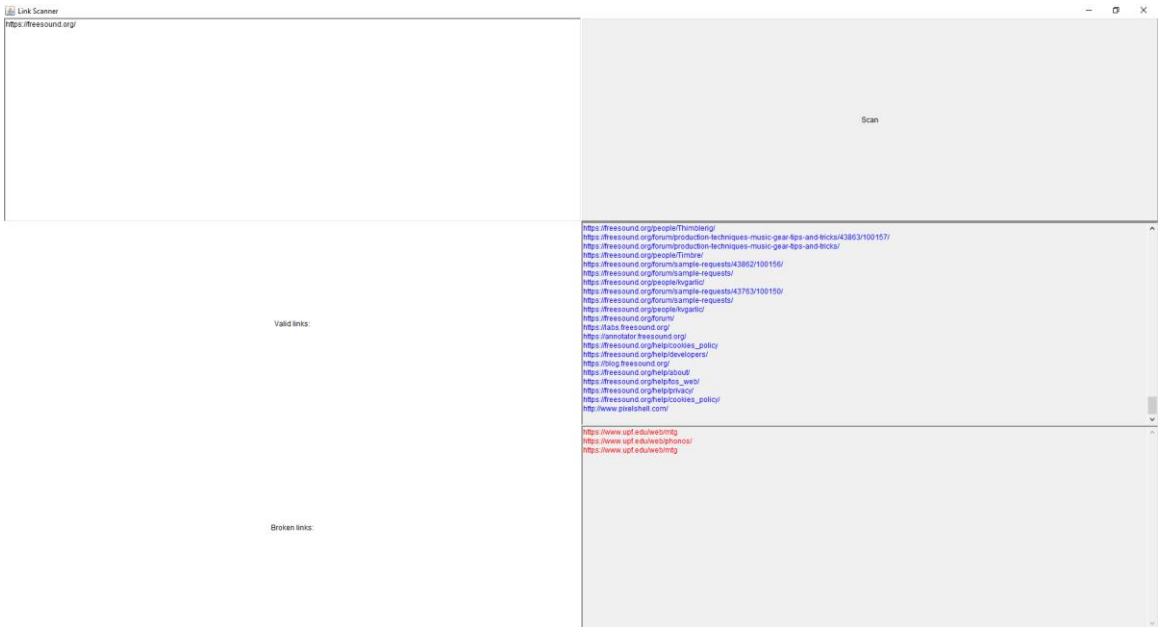

Рисунок 3.3 – Перевірка працездатності готового програмного продукту на сайті https://freesound.org/

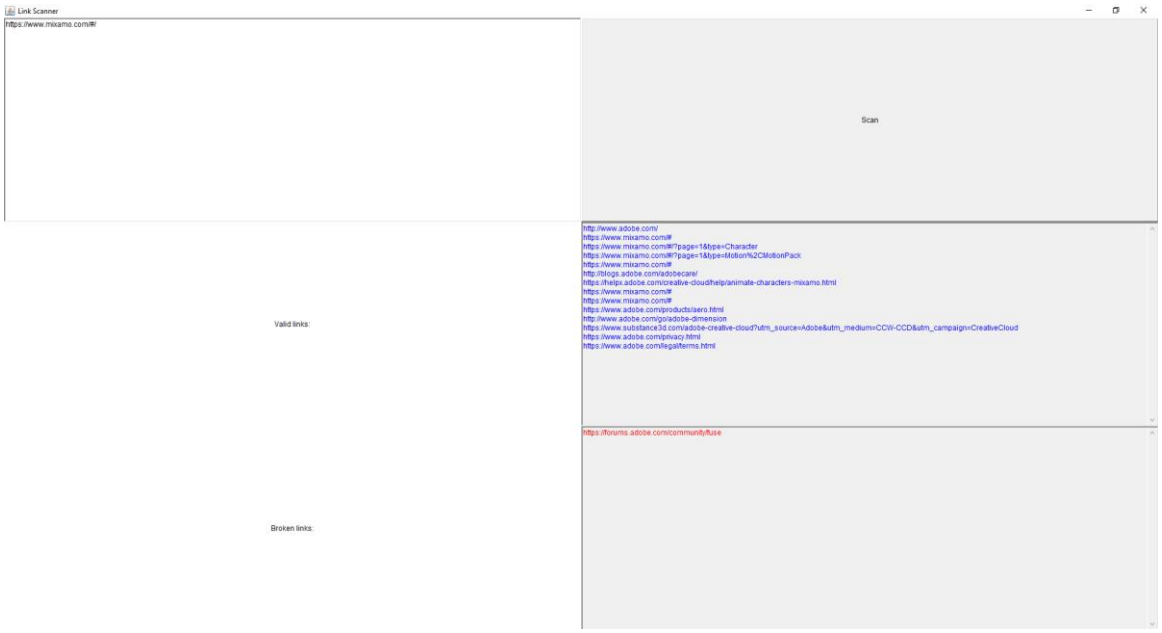

Рисунок 3.4 – Перевірка працездатності готового програмного продукту на сайті https://www.mixamo.com/#/

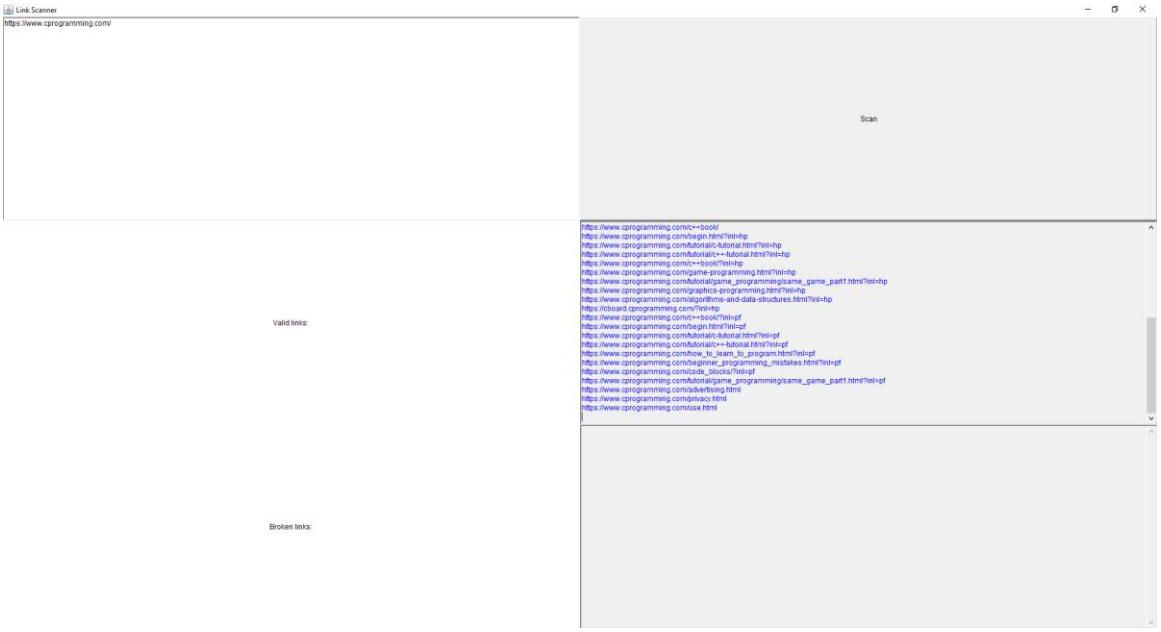

Рисунок 3.5 – Перевірка працездатності готового програмного продукту на сайті https://www.cprogramming.com/

Після перевірки даного веб-ресурсу користувач має можливість побачити некоректні посилання і виправити дану помилку.

ВИСНОВКИ

В результаті виконаної роботи було розроблено програмний продукт, що дозволяє контролювати цілісність веб-порталу. Програмний код написаний на мові програмування Java в середовищі розробки IntelliJ IDEA.

Даний програмний продукт можна використовувати у різноманітних сферах діяльності, але в особливості при тестуванні веб-ресурсів. Розроблений додаток полегшує можливість тестування цілісності веб-ресурсів, які можуть знаходитись на різних етапах розробки або навіть й подальшому супроводженні.

У результаті використання розробленного ПЗ стає можливим полегшення роботи ІТ-спеціалістів при тестуванні веб-продуктів різноманітного призначення.

ПЕРЕЛІК ПОСИЛАНЬ

1. Koehler, Wallace (2004). A longitudinal study of web pages continued: a consideration of document persistence. Information Research. URL: http://informationr.net/ir/9-2/paper174.html (дата звернення 04.11.2022).

2. McCown, Frank; Chan, Sheffan; Nelson, Michael L.; Bollen, Johan. The Availability and Persistence of Web References in D-Lib Magazine. Steve Lawrence; David M. Pennock; Gary William Flake; et al. Persistence of Web References in Scientific Research. URL:https://www.researchgate.net/publication/220364936_An_empirical_study_of_the_accessibility_of_web_references_in_two_Chinese_academic_journals (дата звернення 04.11.2022).

3. Hennessey, Jason; Xijin Ge, Steven. A Cross Disciplinary Study of Link Decay and the Effectiveness of Mitigation Techniques. BMC Bioinformatics. URL:https://www.researchgate.net/publication/258169945_A_cross_disciplinary_study_of_link_decay_and_the_effectiveness_of_mitigation_techniques (дата звернення 04.11.2022).

4. Nelson, Michael L.; Allen, B. Danette. Object Persistence and Availability in Digital Libraries. D-Lib Magazine. URL: https://www.dlib.org/dlib/january02/nelson/01nelson.html (дата звернення 04.11.2022).

5. Kille, Leighton Walter. The Growing Problem of Internet "Link Rot" and Best Practices for Media and Online Publishers. Journalist’s Resource, Harvard Kennedy School. URL: https://journalistsresource.org/media/website-linking-best-practices-media-online-publishers/ (дата звернення 04.11.2022).

6. Пошук битих посилань на сайті: огляд 4 зручних інструментів. URL: https://siteclinic.ru/blog/seo-instrumenty/kak-najti-bitye-ssylki-2/ (дата звернення 05.11.2022).

7. Миколайчук Я. М., Возна Н. Я., Пітух Я. М., Николайчук І. Р. Проектування спеціалізованих комп’ютерних систем : навч. посібник. Тернопіль : ТзОВ "Терно-граф", 2010. 392 с.

8. Бублик В.В. Об’єктно-орієнтоване програмування : підручник. Київ: ІТ-книга, 2015. 624 с.

9. Наливайко Н. Я. Інформатика: навч.посіб. Київ, 2011. 576 с.

10. Мараховський Л. Ф. Інформатика і комп’ютерна техніка: навч. посіб. Київ, 2012. 500 с.

11. Дибкова Л. М. Інформатика і комп’ютерна техніка : навч. посіб. Київ. 2012. 463 с.

12. IntelliJ IDEA. URL: https://www.wiki.uk-ua.nina.az/IntelliJ_ IDEA.html (дата звернення 05.11.2022).

13. Баженов В. А. Інформатика. Комп’ютерна техніка. Комп’ютерні технології : підручник. Київ, 2016. 592 с.

14. Бережна О. Б. Інформатика та комп’ютерна техніка. 1 частина : навч. посіб. Харків, 2017. 164 с.

ДОДАТОК А

Програмний код

```
import java.io.IOException;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Iterator;
import java.util.List;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.chrome.ChromeOptions;
import java.awt.*;        // Using AWT container and component classes
import java.awt.event.*;  // Using AWT event classes and listener interfaces

public class Main extends Frame {
   private static WebDriver driver = null;
   private Label labelValidLinks;    // Declare a Label component
   private Label labelInvalidLinks;    // Declare a Label component
   private TextField textField; // Declare a TextField component
   private Button button;   // Declare a Button component
   private TextArea textAreaValidLinks; // Multi-line TextArea to
taDisplay result
   private TextArea textAreaInvalidLinks; // Multi-line TextArea to
taDisplay result
```

```
// Constructor to setup GUI components and event handlers
public Main () {

    setLayout(new GridLayout(3, 2));

    textField = new TextField("Enter your link to scan", 40);
    textField.setEditable(true);
    add(textField);

    button = new Button("Scan");
    add(button);

    labelValidLinks = new Label("Valid links:");
    labelValidLinks.setAlignment(Label.CENTER);
    add(labelValidLinks);

    textAreaValidLinks = new TextArea(5, 40);
    textAreaValidLinks.setForeground(Color.blue);
    textAreaValidLinks.setEditable(false);
    add(textAreaValidLinks);

    labelInvalidLinks = new Label("Broken links:");
    labelInvalidLinks.setAlignment(Label.CENTER);
    add(labelInvalidLinks);

    textAreaInvalidLinks = new TextArea(5, 40);
    textAreaInvalidLinks.setForeground(Color.red);
    textAreaInvalidLinks.setEditable(false);
    add(textAreaInvalidLinks);
```

```
    BtnCountListener listener = new BtnCountListener();
    button.addActionListener(listener);

    addWindowListener(new MyWindowListener());

    setTitle("Link Scanner");
    setExtendedState(Frame.MAXIMIZED_BOTH);

    setVisible(true);
}

private class BtnCountListener implements ActionListener {
    // ActionEvent handler - Called back upon button-click.
    @Override
    public void actionPerformed(ActionEvent evt) {

        button.setEnabled(false);
        textAreaValidLinks.setText("");
        textAreaInvalidLinks.setText("");

        ChromeOptions options = new ChromeOptions();
        options.addArguments("--headless");

        String homePage = textField.getText();

        String url = "";

        HttpURLConnection huc = null;
        int respCode = 200;
```

```
driver = new ChromeDriver(options);

try
{
   driver.get(homePage);

   List<WebElement> links = driver.findElements(By.tagName("a"));

   Iterator<WebElement> it = links.iterator();

   while(it.hasNext()){

      WebElement tempLink = it.next();
      url = tempLink.getAttribute("href");

      if(url == null || url.isEmpty()){

         continue;
      }

      try {
         huc = (HttpURLConnection)(new URL(url).openConnection());

         huc.setRequestMethod("HEAD");

         huc.connect();

         respCode = huc.getResponseCode();
```

```
                    if(respCode >= 400){
                        textAreaInvalidLinks.append(url + "\n");
                    }
                    else{
                        textAreaValidLinks.append(url + "\n");
                    }
} catch (MalformedURLException e) {
                    e.printStackTrace();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        catch(Exception e)
        {
            driver.quit();
            button.setEnabled(true);
        }

        driver.quit();
        button.setEnabled(true);
    }
}

private class MyWindowListener implements WindowListener {
    // Called back upon clicking close-window button
    @Override
    public void windowClosing(WindowEvent evt) {
        System.exit(0);
```

```
            }

            // Not Used, BUT need to provide an empty body to compile.
            @Override public void windowOpened(WindowEvent evt) { }
            @Override public void windowClosed(WindowEvent evt) { }
            // For Debugging
            @Override public void windowIconified(WindowEvent evt) { System.out.println("Window Iconified"); }
            @Override public void windowDeiconified(WindowEvent evt) { System.out.println("Window Deiconified"); }
            @Override public void windowActivated(WindowEvent evt) { System.out.println("Window Activated"); }
            @Override public void windowDeactivated(WindowEvent evt) { System.out.println("Window Deactivated"); }
        }

        public static void main(String[] args) {

            System.setProperty("webdriver.chrome.driver", "src\\main\\resources\\chromedriver.exe");

            // Invoke the constructor to setup the GUI, by allocating an instance
            Main app = new Main();
        }
    }
```