МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему **Комп'ютерна система для автоматизованого збору і аналізу інформації з Веб-ресурсів**

Виконав: студент 2 курсу, групи 8.1211-2іпз
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Інженерія програмного забезпечення
(код і назва освітньої програми)
Д.В. Царьков
(ініціали та прізвище)
Керівник доцент, к.ф.-м.н., І.А.Скрипник
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел» П.О.Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти другий (магістерський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)
Освітня програма Інженерія програмного забезпечення
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри Т.В. Критська
"_12_" __вересня_____2022_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Царьков Денис Володимирович
(прізвище, ім'я, по батькові)

1.Тема роботи Комп'ютерна система для автоматизованого збору і аналізу інформації з Веб-ресурсів

керівник роботи Скрипник Ірина Анатоліївна, доцент, к.ф.-м.н.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 02.06.2022 р. №597-с

2. Строк подання студентом кваліфікаційної роботи 1 грудня 2022 р.
3. Вихідні дані магістерської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми використання синтаксичних аналізаторів для аналізу веб-ресурсів, ступінь релевантності їх застосування;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
слайдів презентації

6. Консультанти розділів магістерської роботи

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

7. Дата видачі завдання 12.09.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	17.09.2021 – 21.09.2021 р.	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	22.09.2021 р. – 13.10.2021 р.	виконано
3	Аналіз існуючих рішень	15.10.2021 р. – 29.10.2021 р.	виконано
4	Дослідження процесу взаємодії користувача з веб-ресурсами	30.10.2021 р. – 26.11.2021 р.	виконано
5	Дослідження етапів роботи систем автоматизованого збору та аналізу інформації	28.11.2021 р. – 14.12.2021 р.	виконано
6	Узгодження подальших дій з науковим керівником	15.12.2021 р. – 25.02.2021 р.	виконано
7	Реалізація статичної комп'ютерної системи автоматизованого збору та аналізу інформації	26.02.2022 р. – 23.04.2022 р.	виконано
8	Реалізація динамічної комп'ютерної системи автоматизованого збору та аналізу інформації	24.04.2022 р. – 08.07.2022 р.	виконано
9	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	08.07.2022 р. – 14.08.2022 р.	виконано
10	Усунення додаткових проблем та «багів»	15.08.2022 р. – 16.10.2022 р.	виконано
11	Перевірка працездатності створеного програмного застосунку	16.10.2022 р. – 30.10.2022 р.	виконано
12	Порівняння створеного програмного застосунку з існуючими рішеннями автоматизованого збору та аналізу інформації	01.11.2022 р. – 15.11.2022 р.	виконано
13	Оформлення звіту	16.11.2022 р. – 30.11.2022 р.	виконано

Студент _____ Д.В.Царьков
(підпис) (ініціали та прізвище)

Керівник роботи _____ І.А. Скрипник
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ І.А. Скрипник
(підпис) (ініціали та прізвище)

АНОТАЦІЯ

Сторінок: 70

Рисунків: 38

Джерел: 17

Царьков Д.В. Комп’ютерна система для автоматизованого збору і аналізу інформації з Веб-ресурсів: кваліфікаційна робота магістра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник Скрипник. І. А. Запоріжжя : ЗНУ, 2022. 70с.

Мета і завдання дослідження полягають у розробці системи автоматичного збору і аналізу інформації. Знайти аналоги на ринку, визначити їх недоліки. Дослідити варіанти використання синтаксичних аналізаторів для аналізу веб-ресурсів, ступінь релевантності їх застосування у такому вигляді. Дослідити етапи роботи аналогічних систем на ринку та досягти певної універсальності завдяки тому, щоб не надавати застосунок, а конструктор запитів, цілеспрямований на звичайних користувачів, які не мають професійних навичок.

У процесі дослідження була розглянута проблема, що стосується визначення переваг та недоліків програмних застосунків, які реалізують аналогічний функціонал. Як результат, був розроблений програмний застосунок, основна ідея якого полягає в максимальній універсальності та багатофунціональності, щоб задовольнити якомога більше запитів звичайних користувачів. Програмний застосунок призначений надавати користувачу можливість створювати поетапно запит, який у подальшій роботі програмного застосунку буде в автоматизованому вигляді виконувати пошук, аналіз та зберігання даних у вказаному користувачем вигляді з цільового веб-ресурсу.

Ключові слова: *Синтаксичний аналіз, бази даних, HTTP, веб-сайт, веб-сервер, системи управління бази даних, інтерфейс користувача, уніфікований локатор ресурсів, XPath*.

SUMMARY

Pages: 70

Figures: 38

Sources: 17

Tsarkov. D.V/ Development and research of a system for automatic collection and analysis of information from web resources: qualification work of the master of specialty 121 "Software Engineering" / Science head Skrypnyk. I. A. Zaporizhzhia: ZNU, 2022. 70 p.

The purpose and objectives of the study are to develop a system of automatic collection and analysis of information. Find analogues in the market, identify their shortcomings. Investigate the options for using parsers to analyze web resources, evaluate the relevance of their use in this form. Investigate the stages of similar systems on the market, and achieve a certain versatility, by not providing an application, and a query designer that is targeted at ordinary users who do not have professional skills.

In the course of the research, the problem of determining the advantages and disadvantages of software applications that implement similar functionality was considered. As a result, a software application was developed, the main idea of which is maximum versatility and multifunctionality

to meet as many requests of ordinary users. This software application is designed to allow the user to create a step-by-step query, which in the future of the software application will automatically perform, search, analyze, and store data in the form specified by the user, from the target web resource.

Keywords: *Parsing, database, HTTP, website, web-server, database management system, user interface, URL, XPath.*

ЗМІСТ

ВСТУП

Актуальність теми

У наш час, коли мережа інтернет стала невід'ємною частиною нашого життя, та коли майже у кожному куточку світу є можливість приєднатися до цієї мережі, виникає все більше нових проблем, які потребують рішень. Однією з таких проблем постало питання збору та аналізу інформації. Оскільки мережа інтернет містить гігантські обсяги інформації, постає питання у тому, як отримати певну відсортовану та проаналізовану, навіть поверхнево, інформацію, не долучаючи для розв'язання цього питання людину.

Рішення дійсно з'явились, але, на жаль, основні принципи побудови програмних застосунків, які вирішують питання автоматичного збору та аналізу інформації з веб-ресурсів, маю певний недолік. А саме: вони не гнучкі, негнучкість проявляться у тому вигляді, що коли веб-ресурс змінює структуру своєї веб-сторінки, процес автоматичного збору та аналізу зупиняється. Але не тільки у цьому полягає їх негнучкість. Навіть коли розробники переконують, що їх програмний застосунок універсальний, у більшості випадків вони вводять в оману довірливих користувачів. Такі програмні застосунки —це просто великий набір синтаксичних аналізаторів, які спрямовані на певну групу веб-ресурсів. А коли у користувача виникає потреба проаналізувати інформацію з якогось мало відомого ресурсу, він стикається з тим, що у цьому програмному застосунку його не має, і проаналізувати його неможливо без залучення розробників цієї системи. Вони можуть вирішити це питання шляхом створення нового парсеру, хоч він і буде виглядати як частина “універсальної” системи.

У тих випадках, коли програмний застосунок універсальний, він являє собою лише каркас парсеру: цей програмний за стосунок, у більшості випадків, не може надати користувачу можливість створення запитів, які можуть задовольнити його потреби без залучення фахівців або розробників

універсального парсеру, які також я у першому випадку напишуть синтаксичний аналізатор чи запит у своєму програмному застосунку.

Тобто рішення на ринку є, але вони не універсальні або не надають змогу користувачу самому, без фахівців, створювати запити до тих веб-ресурсів, які його цікавлять.

Мета і завдання дослідження

Мета і завдання дослідження полягають в тому, щоб дослідити такий тип програмного забезпечення, як комп'ютерна система автоматичного аналізу та збору інформації з веб-ресурсів. Дослідити комп'ютерні системи автоматичного аналізу і збору інформації з веб-ресурсів як явище. Проаналізувати основні етапи розробки програмного забезпечення такого типу. Проаналізувати, як взаємодіє програмне забезпечення комп'ютерного аналізу з веб-ресурсом. Виокремити етапи, які можна автоматизувати, якщо залучити до процесу конструювання запитів користувача, враховуючи те, що користувачі не повинні мати поглиблених знань з роботи парсерів. На основі отриманих під час дослідження даних сконструювати програмне забезпечення комп'ютерного аналізу й обробки даних з веб-ресурсів, яке буде спроможне генерувати запити без застосування послуг програміста. Виконати тестування та порівняння програмного забезпечення, яке було отримано під час дослідження, з типовим представником програмного забезпечення типу комп'ютерна система автоматичного аналізу і обробки інформації з веб-ресурсів.

Сформувати висновки про мінуси та плюси такого програмного забезпечення за результатами досліджень.

Об'єкт дослідження

Об'єктом дослідження являється система автоматизованого збору та аналізу інформації з веб-ресурсів. А саме: процес побудови системи автоматизованого збору та аналізу інформації; процеси, які проходять під час взаємодії програмного застосунку з веб-ресурсом; як веб-ресурс взаємодіє з веб-

сервером; які варіанти надання інформації користувачу використовують веб-ресурси; яку інформацію від користувача потребує веб-ресурс для коректної роботи та для надання послуг; які є сценарії, при яких веб-ресурс може не надати користувачу потрібну йому інформацію.

Предмет дослідження

Предметом дослідження являється визначення характеристик програмного забезпечення, сформованого автоматизованим конструктором запитів та типовим представником програмного забезпечення для автоматичного збору та аналізу інформації з веб-ресурсів.

Методи дослідження

Для вирішення поставленої задачі використовуються наступні методи дослідження.

1. Аналіз існуючих рішень та їх особливостей.
2. Аналіз переваг та недоліків існуючих рішень.
3. Аналіз різноманітних бібліотек, призначених для реалізації таких програмних застосунків.
4. Аналіз різних інструментів, які призначені для створення таких програмних застосунків.
5. Експериментування з різними інструментами, які призначені для нагляду за статусом роботи систем для автоматичного збору та аналізу даних з веб-ресурсів.
6. Аналіз принципів роботи веб-ресурсів для покращення автоматизованого збору та аналізу інформації.

Наукова новизна одержаних результатів

Наукова новизна одержаних результатів дослідження полягає у тому, що під час розробки системи автоматизованого збору та аналізу інформації ми досягаємо реальної універсальності цього аналізатору у тому вигляді, що

алгоритм роботи програмного застосунку дозволяє гнучко підлаштовуватись під різні веб-ресурси.

Практичне значення одержаних результатів

Практичне значення одержаних результатів дослідження полягає у тому, що було розроблено програмний застосунок автоматизованого збору та аналізу даних з веб-ресурсів. Даний програмний застосунок дозволяє вирішити основну проблеми існуючих програмних рішень на ринку, а саме те, що ці програмні застосунки за своїм походженням є вузько направленими рішеннями, у яких відсутня будь яка універсальність. І коли на цільових веб-ресурсах змінюється HTML розмітки, користувач втрачає можливість отримувати проаналізовану інформацію, втрачає можливість вирішення цього питання без залучення розробників програмного застосунку.

Апробація одержаних результатів

Результати дослідження були представлені на науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених Запорізького національного університету «Молода наука-2021» [1], на ІІ всеукраїнській науково-практичній конференції за участю молодих науковців «Актуальні питання сталого науково-технічного та соціально-економічного розвитку регіонів України» [2].

Глосарій

Git — розподілена система контролю версій, відстеження змін у файлів, також використовується під час спільної розробки.

Java Script —

*Архітектура (*англ. *Architecture)* — структура компонентів, їх взаємозв’язки, вказівки та принцип, що керують проектуванням та еволюцією.

AJAX (Asynchronous JavaScript і XML) — набір методів веб-розробки, який застосовує різні веб-технології на стороні клієнта, для виконання

асинхронних запитів до веб-додатків. За допомогою Ajax веб-застосунки можуть надсилати та отримувати дані з серверу асинхронно.

Uniform Resource Locator URL — це посилання на веб-ресурс, який визначає його місцезнаходження в комп'ютерній мережі та механізм його отримання.

URL-адреса — це певний тип уніфікованого ідентифікатора ресурсу. URL-адреси найчастіше зустрічаються для посилань на веб-сторінки, але також використовуються для передачі файлів, електронної пошти доступу до бази даних та багатьох інших програм.

HTTP — протокол прикладного рівня в моделі пакету інтернет-протоколів є основою передачі даних для Всесвітньої павутини.

Status code — видаються сервером у відповідь на запит клієнта, зроблений на сервер. Він включає коди з IETF (RFC),

HTML — є стандартною для документів, призначених для відображення у веб-браузері.

Веб-застосунок — програмне забезпечення, доступ здійснюється за допомогою веб-браузеру, поставляється програмне забезпечення через всесвітню мережу інтернет користувачам з активним інтернет з’єднанням.

Веб-сторінка — це гіпертекстовий документ у всесвітній павутині. Вебсторінки доставляються веб-сервером користувачеві і відображаються у веббраузері.

База даних — це організована сукупність даних, що зберігаються і до яких здійснюється доступ в електронному вигляді.

Веб-сервер — комп’ютерне програмне забезпечення та основне обладнання, яке приймає запити через протокол HTTP або його захищений варіант HTTPS.

Веб-браузер — прикладне програмне забезпечення для доступу до вебсайтів. Коли користувач запитує веб-сторінку з веб-сайту, веб-браузер отримує файли від веб-сервера, а потім відображає сторінку на екрані користувача.

Браузери використовуються на різних пристроях, таких як: комп’ютери, планшети, смартфони.

Протокол — це система правил, яка дозволяє двом або більше суб'єктам системи зв'язку передавати інформацію за допомогою будь-якого роду варіацій .

MPA — це багатосторінкові програми, які працюють як наші звичайні веб-сайти. Вони відправляють запит на сервер і повністю оновлюють сторінку при виконанні з нею будь-яких дій (перехід на іншу сторінку, введення і зміна даних).

SPA — веб-додаток або веб-сайт, який взаємодіє з користувачем шляхом динамічного переписування поточної веб-сторінки новими даними з веб-сервера, замість методу за замовчуванням веб-браузера, що завантажує цілі нові сторінки. Мета полягає в швидших переходах, які роблять веб-сайт більш схожим на рідний додаток.

*Domain Name System DN*S — ієрархічна і розподілена система іменування комп'ютерів, служб та інших ресурсів в мережі Інтернет або інших мережах інтернет-протоколу. Він пов'язує різну інформацію з доменними іменами, присвоєними кожному з пов'язаних суб'єктів. Найпомітніше, що він легко запам'ятовує доменні імена на числові, необхідні для пошуку та ідентифікації комп'ютерних служб та пристроїв за допомогою базових . Система доменних імен є важливим компонентом функціональності Інтернету з 1985 року.

Парсер — програмний застосунок, або його частина, призначена для аналізу рядка символів, або в природній мові, комп'ютерних мовах або структурах даних, що відповідають правилам формальної граматики.

Краулер (англ. *Web crawler*) — програмний застосунок, або його частина, призначений до систематичного обходу веб-сторінок, та збереження вмісту сторінки для подальшого використання.

Клієнт (англ. *Client*) — це апаратний чи програмний компонент обчислювальної системи, що посилає запити серверу, взаємодіє із сервером, використовуючи певний протокол.

Windows Presentation Foundation WPF — спочатку розроблена для рендеринга в додатках на базі Windows.

XML Path Language XPATH — мова виразів, призначена для підтримки запиту або перетворення XML-документів. Він був визначений консорціумом Всесвітньої павутини W3C і може бути використаний для обчислення значень з вмісту XML-документа. Підтримка XPath існує в додатках, що підтримують XML, таких як веб-браузери і багато мов програмування.

Copyright — вид інтелектуальної власності, який надає її власнику виключне право копіювати, поширювати, адаптувати, демонструвати та виконувати творчу роботу, як правило, протягом обмеженого часу.

Cache — це апаратний або програмний компонент, який зберігає дані, щоб майбутні запити на ці дані могли бути обслужені швидше, Дані, що зберігаються в кеші, можуть бути результатом попередніх обчислень або копією даних, що зберігаються в іншому місці. Попадання кешу відбувається, коли запитувані дані можна знайти в кеші, тоді як пропуск кешу відбувається, коли він не може.

РОЗДІЛ 1 ДОСЛІДЖЕННЯ ПРОБЛЕМИ КОМП'ЮТЕРНИХ СИСТЕМ ДЛЯ АВТОМАТИЗОВАНОГО ЗБОРУ ТА АНАЛІЗУ ІНФОРМАЦІЇ З ВЕБ-РЕСУРСІВ

1.1 Огляд проблеми автоматизованого збору та аналізу інформації з веб-ресурсів

Програмні застосунки для автоматизованого збору та аналізу інформації з веб-ресурсів — програмні застосунки, які мають на меті забезпечити користувачу можливість отримати відформатовані дані з певного веб-ресурсу. Для цього програмний застосунок імітує поведінку користувача, робить запити з певними атрибутами, перевіряє статуси відповідей від веб-ресурсу та проводить синтаксичний аналіз результатів запитів. Для того, щоб розуміти, які процеси проходять під час роботи програмного застосунку, треба дослідити те, як, що і яким чином отримує користувач. У тому числі потрібно проаналізувати роль веб-браузера у взаємодії користувача з веб-ресурсом.

1.2 Огляд процесу взаємодії користувача з веб-ресурсом

1.2.1 Взаємодія користувача з веб-ресурсом

Щоб отримати дані від веб-ресурсу користувачі застосовують веб-браузер. Користувач передає у рядок пошуку адресу ресурсу, який їх цікавить, та натискають кнопку підтвердити. У цей час відбувається наступне (рис.1). Клієнт, у нашому випадку це веб-браузер, надсилає HTTP-запит по адресі з певними параметрами, отримує відповідь, проводить розбір відповіді та форматує її у прийнятний для користувача вигляд.

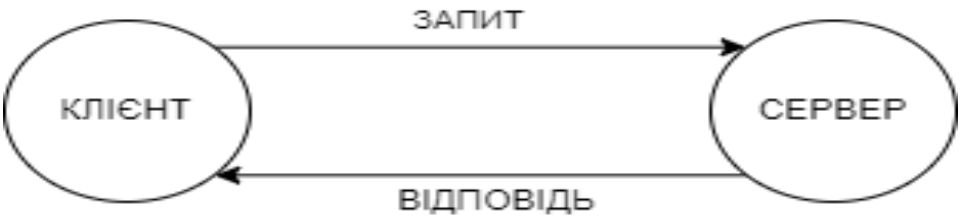


Рис.*1 Зображення базового представлення взаємодії користувача з веб-ресурсом*

1.2.2 URL-адреса та взаємодія веб-браузера з нею

URL (Uniform Resource Locator) — це адреса, яка надана певному унікальному ресурсу у мережі інтернет. Такими URL-адресами (рис.2) можуть слугувати HTML-сторінки, JS-скріпти, чи файли CSS. В наш час існує вірогідність, що URL-адреса може вести на ті ресурси, які більше не існують або були перенесені.

https://moodle.znu.edu.ua/

https://www.google.com/

https://prozorro.gov.ua/search/plan

Рис.*2 Приклади URL-адрес*

URL-адреса складається з декількох части, деякі є обов’язковими. Розглянемо їх на прикладах:

Протокол

https:// — це протокол (рис.3). Він вказує те, який саме протокол передачі даних повинен використовувати веб-браузер. Зазвичай застосовують HTTP-протокол та HTTPS-протокол, модифіковану версію, зосереджену на безпеці даних. В наш час все більше веб-ресурсів застосовують захищену версію протоколу. Веб-браузери підтримують обидва протокли за замовчуванням, тому, коли користувач не вказує проткол у адресному рядку, браузер підставить протокл автоматично.

Рис.*3 Частина Url-адреси, яка вказує протокол*

Доменне ім’я

Доменне ім’я (рис.4) слугує для того, щоб користувач міг шукати веб-ресурс, не запам’ятовуючи ip адресу веб-ресурсу. Замість цього він вводить зручну для запам’ятововування назву ресуру, за якою веб-браузер буде звертатись до DNS серверу, отрмавши від нього ip-адресу.

Рис.*4 Частина URL, що містить доменне ім’я*

Порт

Протоколи працюють в парі з портами. HTTP-протокол використовує 80 порт за замовчуванням, HTTPS-протокол застосовує 443 (рис.5). Ці значення є стандартими, тому в більшості випадків користувачу не потрібно прописувати порти самому, але є веб-ресурси, які використовують інші порти. У таких випадках користувач повинен прописувати його для того, щоб мати змогу отримати потрібну йому інформацію.

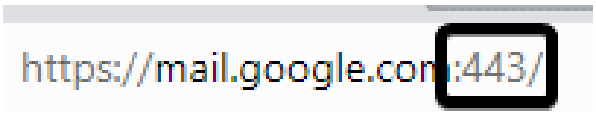


Рис.*5 Частина URL, що містить порт*

Шлях до сторінки

Це частина URL-адреси (рис. 6) після доменного ім’я. Вона відповідає за те, щоб користувач міг потрапити на потрібну йому сторінку веб-ресурсу. У простих реалізаціях веб-ресурсів шляхи до сторінок є відображенням

реального розташування html- чи css-файлів на веб-сервері. Головна сторінка зазвичай вказується просто як протокол та доменне ім’я.

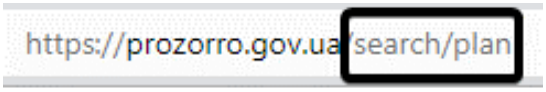


Рис. *6 Частина URL, що містить шлях до сторінки*

Параметри

Це дані, які передає веб-браузер до веб-ресурсу. Вони слугують для того, щоб повідомити про те, які додаткові дії повинен провести веб-ресурс перед тим, як надавати відповідь.

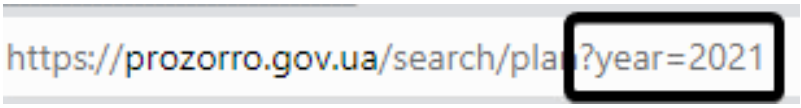


Рис. *7 Частина URL, яка містить параметри запиту*

Типи URL-адрес

URL адреси поділяють на два види: статичні та динамічні.

Статична URL-адреса – це постійна адреса, яка не змінюється, що вказує на певну сторінку. Зазвичай такі адреси не містять параметрів.

Динамічні адреси сторінок формують контент у відповідь до заданих у запиті параметрів. Адреси такого типу можна зустріти на сайтах, які побудовані на CMS. Вміст сайту зберігається у базі даних і при потребі виводиться для відображення на сторінках за запитом. Також динамічні адреси формуються, коли ви використовуєте фільтри чи пошук на сайті.

Формати URL

На даний момент URL-адреси можна поділити на такі формати:

- Латиниця.
- Кирилиця.
- Транслітеровані.

- Зрозумілі людині адреси.

Процес роботи браузера з URL

Коли користувач ввів у адресний рядок URL-адресу. Веб-браузер виконує запит до DNS-серверу з метою отримати для вказаного користувачем доменного імені ip-адресу серверу, на якому знаходиться потрібний користувачу веб-ресурс. Після того, як DNS сервер надавав веб-браузеру ip-адресу, веб-браузер формує запит, надсилає його до сервера та очікує відповідь (рис. 8).

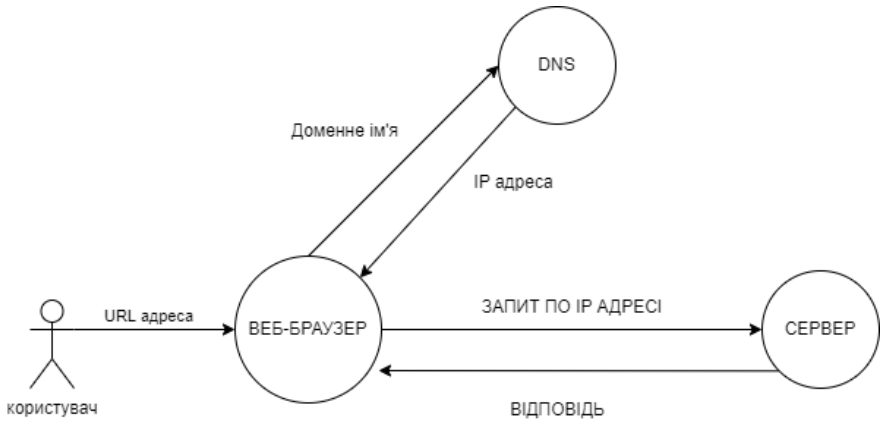


Рис.*8 Процес роботи веб-браузеру з URL*

1.2.3 HTTP-запити

HTTP — протокол передачі гіпертекстових документів, основою протоколу є взаємодія клієнту та серверу. Клієнт ініціює з’єднання та посилає запит до серверу. Сервер в свою чергу очікує на з’єднання щоб отримати запит клієнту, проводить необхідні дії згідно з запитом, та надсилає клієнту повідомлення з результатом. Основними запитами, які робить звичайний користувач являються GET та POST запити.

1.2.3.1 Get-запит

Get-запит (рис. 9) спрямований на отримання інформації з веб-ресурсу. Він не повинен вносити будь яких змін у структуру, данні веб-ресурсу.

Рис.9 Приклад Get запиту

1.2.3.2 Post-запит

Post запит (рис. 10) спрямований на те щоб відправляти дані на сервер. Зазвичай Post відправляється через HTML форми, та приводить до змін на сервері.

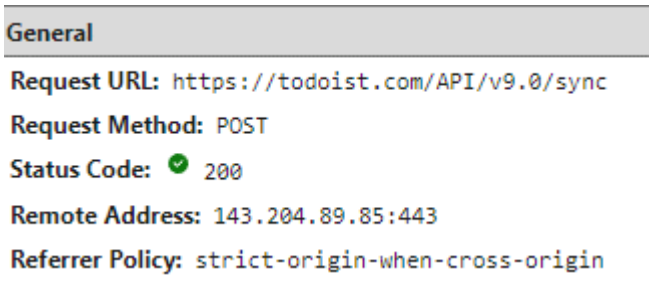


Рис.10 Приклад Post запиту

1.2.4 HTTP заголовки

Заголовки дозволяють клієнту та серверу додавати додаткову інформацію до HTTP запиту або відповіді. У заголовках інформація зберігається у такому вигляді: не чутливе до регістру ім’я поля ,а після роздільника “:” значення поля запиту. Заголовки супроводжують процес обміну даними по протоколу HTTP. Вони можуть містити інформацію, описи даних, необхідну інформацію для взаємодії між клієнтом та сервером.

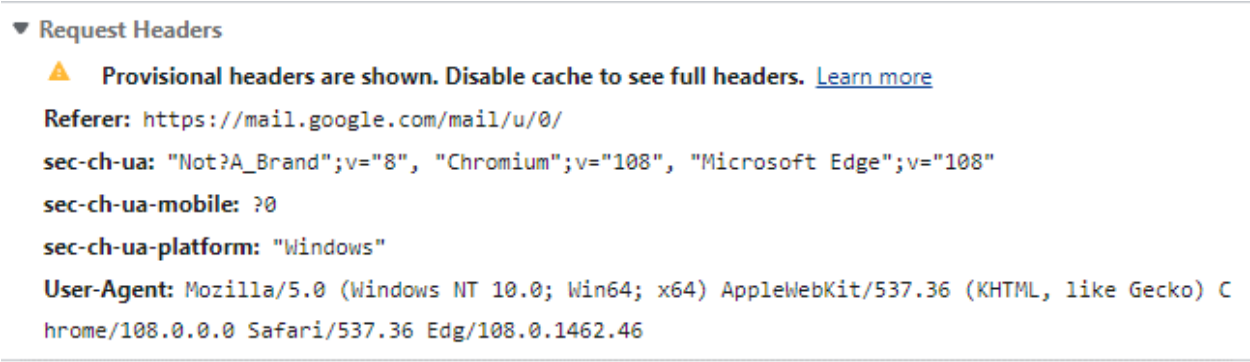


Рис. 11 Приклад HTTP заголовка

Http заголовки можна згрупувати за наступним контекстом:

- Основні заголовки, Заголовки які застосовуються, як до запитів, так і до відповідей, вони не мають відношення до даних, які передаються у тілі запиту.
- Заголовки запиту містять інформацію про клієнта, о ресурсі до якого робиться запити, або більше інформації о ресурсі.
- Заголовки відповідей містить інформацію про відповідь на запит, наприклад інформація про місцезнаходження даних, або веб-серверу.
- Заголовки про тіло ресурсу, містять інформацію про довжину тексту, мову контенту, або його кодування.

1.2.5 HTTP-статус-коди

Протокол надає перелік кодів відповіді (статусу), які демонструють те, чи був виконаний HTTP-запит. Коди згруповані на п’ять класів:

1. 100-199 — Інформаційні, чи був запит отриманий, та валідний. Обробка запиту продовжується.
2. 200-299 — Успішні, запит був прийнятий, оброблений, та виконаний.
3. 300-399 — Перенаправлення. Для продовження виконання запиту, користувач, повинен виконати подальші дії.
4. 400-499 — Помилка клієнта. Запит не може бути виконаним або не зрозумілий синтаксис.

5. 500-599 — Серверні помилки.

Рис.12 Приклад статус коду

Можливо, що сервер надасть код, який не є у цьому переліку, у цьому разі код є нестандартизованим кодом відповіді.

1.2.6 HTTP-відповіді

Веб-ресурси, обробивши запит клієнта, надають йому відповідь. Відповідь складається з наступних елементів:

1. Status line – це перший рядок відповіді HTTP, він містить інформацію про те, яка версія протоколу була використана, код стану запиту та текстовий варіант пояснення статус-коду.
2. HTTP-заголовки, вони мають аналогічну структуру побудови, як і у випадку HTTP-запиту.
3. Тіло відповіді, не всі HTTP-відповіді мають тіло, наприклад, відповідь зі статус-кодом 204 (Немає контенту).

1.2.7 Структура HTML

Кожний HTML-документ починається з декларації типу документа. Він вказується для того, щоб веб-браузер міг отримати версію HTML та вірно відобразити сторінку.

```
<!DOCTYPE html>
<html lang="ua">
<head>
    <title>Головна | Prozorro </title>

    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width,initial-scale=1,maximum-scale=1,user-scalable=no">
    <meta name="csrf-token" content="vxYctbHqaxL2nEbLV6P7uRt59iKMolHRjeJuCjqN">

    <link rel="preload" href="//fonts.googleapis.com" as="font">
    <link rel='shortcut icon' type='image/x-icon' href='https://prozorro.gov.ua/favicon.ico' />
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <link rel="dns-prefetch" href="//fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,300;0,400;0,600;0,700;0,800;1,300;1,400;1,
    <link href="https://fonts.googleapis.com/css?family=Nunito&display=swap" rel="stylesheet" type="text/css">
    <link href="/css/app.css?id=92f762d6ba910843405b" rel="stylesheet">

        <script>
        (function (w, d, s, l, i) {
            w[l] = w[l] || [];
            w[l].push(
                {'gtm.start': new Date().getTime(), event: 'gtm.js'}
            );
            var f = d.getElementsByTagName(s)[0],
                j = d.createElement(s), dl = l != 'dataLayer' ? '&l=' + l : '';
            j.async = true;
            j.src = 'https://www.googletagmanager.com/gtm.js?id=' + i + dl;
            f.parentNode.insertBefore(j, f);
        })(window, document, 'script', 'dataLayer', 'GTM-WQ46WMM');
    </script>


</head>
<body>
```

Рис.*13* Приклад HTML документа

У примітивному вигляді HTML-сторінка складається, як мінімум, з трьох HTML-тегів, а саме: `<html>`, `<head>`, `<body>`.

Тег `head` містить заголовок сторінки, ключові слова, опис сторінки та службові дані. Саме у цьому тегу підключаються зовнішні ресурси, такі як стилі, js-скрипти. Вміст цього тегу не відображається користувачу на сторінці після того, як веб-браузер проаналізував та сконструював сторінку.

Тег `body` містить у собі зміст сторінки, який бачить користувач у вікні веб-браузера.

Для того щоб підключати CSS стилі у HTML існує тег `<link>`. Цей тег має атрибут `href`, який містить адресу, за якою веб-браузер повинен отримати `css` файл, а також атрибут `rel` із значенням `stylesheet`, саме завдяки ньому браузер розуміє, що даний тег з цією адресою вказує на `css` файл.

Зазвичай у тегу `body` містяться такі теги як: <header>, <main>, <footer>.

Тег `header` містить вступну частину html сторінки, містить меню, лого-тип.

Тег `main` призначений для основної частини вмісту веб-сторінки. Основний контент складається з контенту, який безпосередньо відноситься до основної теми html-сторінки.

Тег `footer` містить кінцеву частину html-сторінки, зазвичай, саме у ньому на сайті вказують права власності на інформацію, яка знаходиться на ресурсі, контактні дані, посилання на опис інформації про власника веб-ресурсу, посилання на умови оплати, доставки.

1.2.8 Процес конвертації HTML у DOM

DOM (Document Object Model). Згідно з моделлю кожен HTML-тег є об’єктом. Вони мають ієрархічну структуру, а саме: кожен тег, який вкладений у інший тег, є «дитиною» батьківського елементу, як і батьківський елемент, він також є об’єктом.

Веб-ресурс у відповідь на запит надає веб-браузеру HTML, вона приходить у вигляді байтів. Для того, щоб побудувати сторінку, веб-браузер конвертує байти у текст. Далі цей текст перетворюється на вузли, а вузли перетворюються на об’єкти. Після цього будується DOM-дерево (рис. 14, 15).

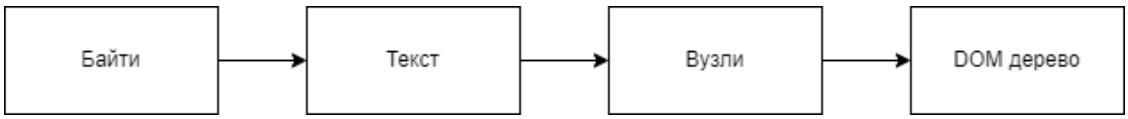


Рис. *14 Етапи перетворення відповіді у DOM*

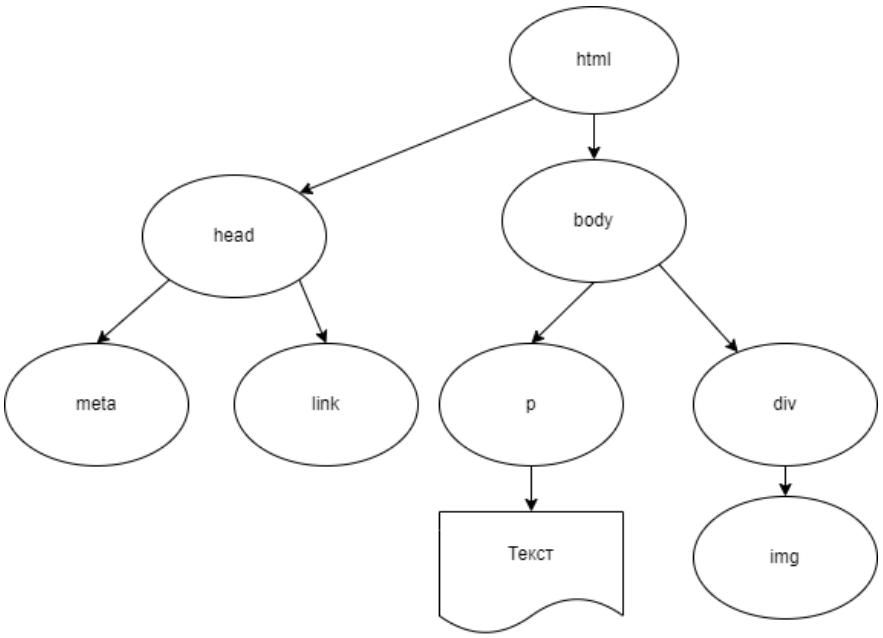


Рис. *15 Зображення структури DOM дерева.*

Ми можемо, застосувавши інструменти розробника, які в наш час надають веб-браузери за замовченням, переглянути, скільки часу займає перетворення байтової відповіді у документ моделі об’єкта DOM.

1.3 Огляд видів веб-застосунків та пагінації веб-сторінок

1.3.1 Види веб-застосунків

Будь який веб-застосунок у мережі інтернет можливо віднести до одного з трьох типів веб-застосунків, а саме багато сторінкові MPA, односторінкові SPA, та прогресивні PWA.

Односторінкові веб-застосунки. SPA або Single Page Application – це односторінковий веб-застосунок (рис. 16), який надає веб-браузеру одну HTML-сторінку. Завдяки динамічному оновленню за допомогою JavaScript, а саме AJAX, під час використання веб-застосунку, веб-браузеру не потрібно постійно перезавантажувати або завантажувати додаткові сторінки. На практиці це означає, що коли користувач бачить контент веб-сторінки, та коли він пролистує або переходить на інші сторінки, замість повної перебудови

сторінки потрібні елементи підвантажуються через наданий AJAX-функціонал та додаються до існуючого відображення сторінки у веб-браузері.

Такий підхід дозволяє надати користувачу більш приємний досвід користування веб-ресурсом, тому що під час взаємодії з таким типом веб-застосунку може скластись враження, що користувач взаємодіє з декстопним програмним застосунком, на підставі того, що веб-застосунок миттєво реагує на дії користувача, без затримок.

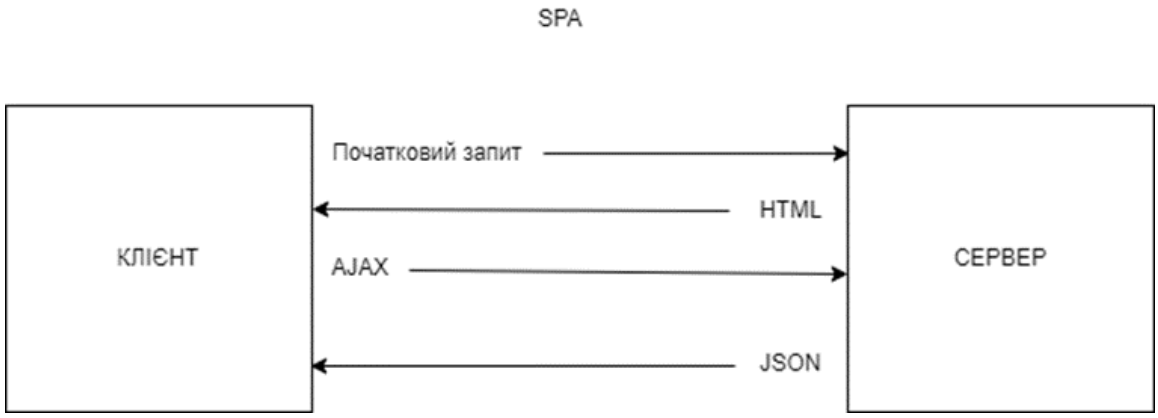


Рис.*16 Зображення процесу роботи SPA застосунку*

Перевагами такого типу веб-застосунків є:

- Висока швидкість реагування на дії користувача – всі ресурси завантажуються за одну сесію, а під час дій на сторінці данні просто змінюються, що зберігає час користувача.
- Гнучкість користувацького інтерфейсу – завдяки тому, що веб-сторінка всього одна, розробнику легше керувати та змінювати зовнішній вигляд та структуру сторінки, зберігати дані про сенс, керувати станами і анімаціями.
- Кешування даних – веб-застосунок, надавши одну відповідь, зберігає дані та після цього може функціонувати без під’єднання до мережі інтернет.

Недоліками такого типу веб-застосунків є:

- Навантаження на веб-браузер – з причини того, що такі веб-застосунки мають великий розмір, це займає суттєві проміжки часу. У деяких випадках це займає більше 500 мс, що може негативно відбитися на враженнях користувача від користування веб-застосунком.
- Потрібна підтримка Java Script – без JS неможливо повноцінно використовувати функціонал веб-застосунку.
- Витіки пам’яті у Java Script.

Багатосторінкові веб-застосунки. MPA або Multi Page Application – це багатосторінковий веб-застосунок (рис. 17), він працює по традиційній схемі. Тому у тих випадках, коли користувач або застосунок вносить зміни у дані, або при завантаженні нової сторінки веб-браузер перебудовує сторінку.

Перевагою такого типу застосунків є те, що вони звичні для користувача завдяки простому інтерфейсу та класичній навігації по веб-ресурсу.

Недоліками такого типу веб-застосунків є:

- Тісні взаємовідносини між back end, та front end.
- Складності при розробці.

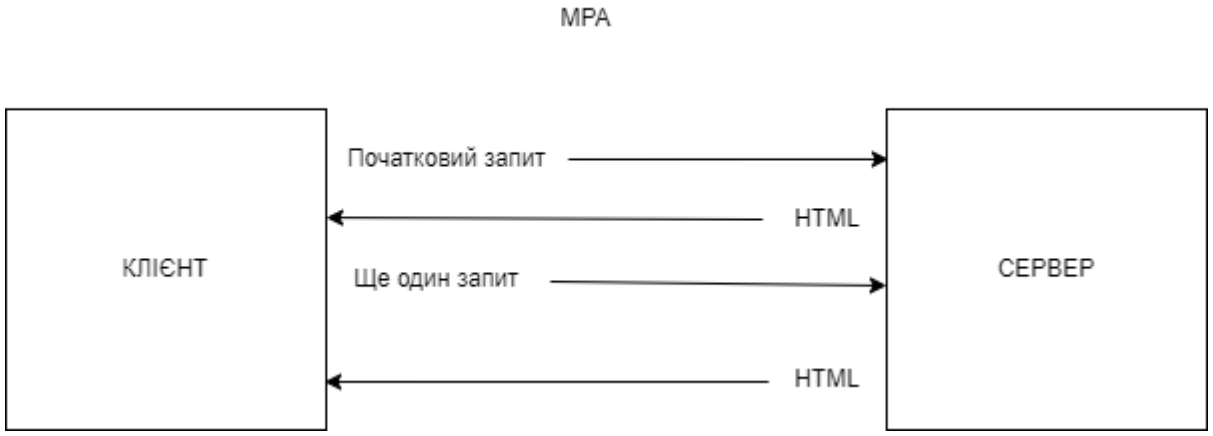


Рис.*17 Зображення процесу роботи MPA застосунку*

Прогресивні веб-застосунки. PWA або Progressive Web Application взаємодіють з користувачем, як програмний застосунок. Такий тип веб-застосунків може встановлюватись на пристрої користувача, надсилати йому push-

повідомлення, працювати у офлайн режимі. Прикладом такого веб-застосунка для смартфону є GMAIL.

1.3.3 Види надання контенту користувачу

Для того, щоб надати користувачу згруповані дані, та реалізувати це у розумних рамках часових, та ресурсних витрат, застосовують два варіанти поділу та надання порції елементів веб-ресурсу. Такими варіантами є пагінація, та нескінченний скролінг. В обох варіантах розв'язання питання надання порції даних є як свої сильні, так і слабкі сторони.

1.3.3.1 Пагінація

Пагінація (англ. Pagination) – це порядкова нумерація сторінок (рис. 18), яка розміщується зверху або знизу, або інколи з двох сторін сторінки веб-ресурсу.

Рис. *18 Приклад пагінації*

У нашому випадку пагінацією є структурування даних, які були поділені на певні сторінки, наприклад по 20 товарів на сторінку. Це робиться, як для того, щоб зробити меншими обсяги інформації, яку треба обробляти веб-браузеру, або користувачу.

Зустрічається посторінкова пагінація майже на усіх сайтах, але найчастіше в інтернет-магазинах. У випадку магазину застосування поділу всього обсягу інформації є доволі зрозумілим, бо користувачу не буде зручним процес гортання сотні або тисячі товарів, алей ще по тій з причин, що коли користувач у випадку того, як він запам'ятав певний товар він може вказати, конкретну

сторінку де знаходився той чи інший товар не повторюючи пошук, серед всього обсягу товарів.

Також причиною поділу даних на сторінки є прискорення надання відповіді веб-серверу на запит користувача. Хоч HTML сторінки може містити будь-яку кількість інформації, але якщо вона буде містити, наприклад 1000 товарів одразу, це суттєво впливає на час передавання даних.

Види пагінації у представленні користувачу:

- Пряма порядкова нумерація 1-2-3-4.
- Сторінки вказані у певному діапазоні 1-10, 10, 20, 30.

Види пагінації у реалізації:

Серверна пагінація – це коли веб-сервер, на якому розміщенні дані, повертає лише підмножину, яка відповідає критеріям наданим клієнтом. Разом з підмножиною сервер, повертає загальну кількість результатів, що відповідають критеріям запиту, та розташування, в якому знаходиться клієнт в межах запиту. Тобто сервер надає дані які запитував клієнт, скільки результатів і де у списку ми отримуємо наші дані.

Пагінація на стороні клієнта – це коли веб-сервер при виконанні запиту повертає клієнту одразу всі дані одним великим шматком. Нумерація сторінок на стороні клієнту дозволяє, гортати сторінки з даними, майже блискавично, оскільки клієнту не потрібно звертатись до сервера, для того, щоб він нам надав нову підмножину.

1.3.3.2 Нескінченний скролінг

Інструмент який дозволяє користувачу передивлятись результати запиту розміщенні на веб-сторінці, у повному обсязі. Контент відображається динамічно, як тільки користувач прокручує сторінку до самого кінця, або натискає кнопку «Показати ще». Реалізація такого варіанту надання даних на запит користувача, використовується в односторінкових веб-застосунках.

1.4 Огляд теоретичної комп'ютерної системи автоматизованого збору та аналізу інформації з веб-ресурсів

Скрейпинг – це процес збору структурованих даних автоматизованим способом (рис. 19). Він також відомий як інтелектуальний аналіз даних. Як правило, інтелектуальний аналіз веб-даних використовується окремими особами та компаніями, які хочуть використовувати загальнодоступні веб-дані для прийняття більш обґрунтованих рішень.

Якщо ви коли-небудь копіювали та вставляли інформацію з веб-сайту, ви виконуєте ту саму функцію, що й будь-який парсинг веб-ресурсів, за винятком того, що ви збираєте дані вручну.

На відміну від виснажливого процесу ручного вилучення даних, парсер веб-ресурсів використовує автоматизацію для вилучення сотень, мільйонів або навіть мільярдів одиниць структурованих даних із, здавалося б, нескінченних кордонів Інтернету.

Вилучення веб-даних – має великий спектр застосувань, так і форм реалізації.

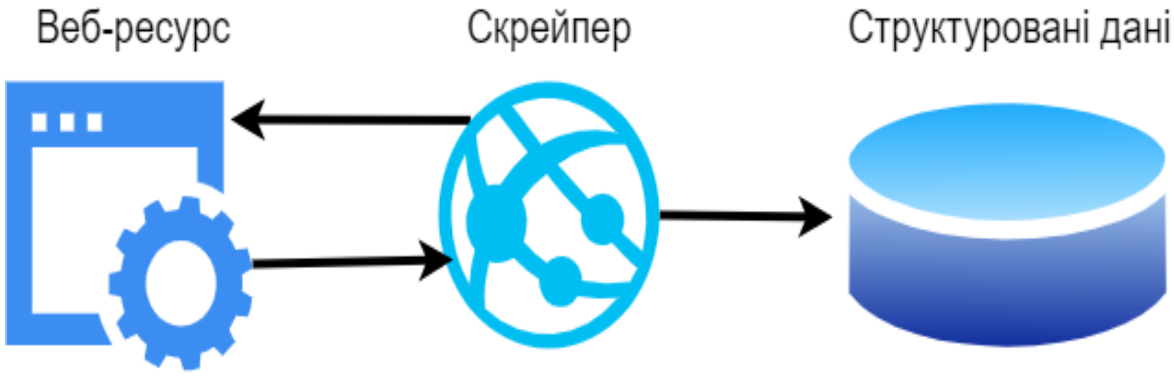


Рис.*19 Робота скрейперу*

1.4.1 Огляд процесу роботи системи

Процес роботи програмного застосунку для збору та аналізу інформації з веб-ресурсів, можна поділити на такі етапи:

1. Отримання переліку адрес, які підлягають обробці. В цьому етапі, програмний застосунок може використовувати заздалегідь підготовлений користувачем, або розробником перелік URL адрес, та додаткову інформацію про тип таких сторінок. Не завжди адресами слугують веб-сторінки, ними можуть слугувати, і Ajax точки взаємодії веб-сервера з клієнтом.

2. Формування запитів, до веб-сервера. На основі заздалегідь розроблених форм запитів, програмний застосунок використовує ці форми потрібного типу, для отриманих адрес запитів.

3. Отримання, та зберігання відповідей. Програмний застосунок на цьому етапі, надсилає до веб-ресурсу запит, перевіряє статус коди, та у тому випадку якщо все добре зберігає відповідь для подальшої роботи.

4. Процес аналізу, на цьому етапі програмний застосунок проводить дії над наданою веб-ресурсом відповіддю. А саме перетворює HTML строку у потрібний йому об'єкт, та застосовує регулярні вирази, Xpath тощо... Та формує об'єкти результату обробки для зберігання.

5. Процес зберігання, програмний застосунок, відповідно до заздалегідь продуманої логіки зберігає відформатовані дані до файлу, бази даних.

1.4.2 Аналіз впливу типу веб-застосунку на роботу системи

У залежності від типу веб-застосунку, треба враховувати, те що принципи отримання даних системою збору будуть відрізнятись. Наприклад для того, щоб отримати інформацію з HTML сторінки у застосунку типу WPA ми надсилаємо Get-запит. Здебільшого багатосторінкові веб-застосунки, надають одразу повну сторінку, з усією інформацією, але є випадки коли у таких застосунках використовуються AJAX, тоді розробник програмного застосунку буде вимушений аналізувати увесь трафік який генерують веб-браузер, та веб-ресурс під час своєї взаємодії. Метою аналізу є виокремлення певних етапів взаємодії, для подальшої імітації програмним застосунком дії користувача, а саме порядок запитів, їх атрибути. Бо на превеликий жаль багато веб-застосунків,

при виявленні підозрілої активності можуть заблокувати роботу програмного застосунку.

Веб-застосунки типу SPA використовують AJAX, тому для них етап аналізу трафіка, та виокремлення важливих етапів є обов'язковим.

1.4.3 Огляд процесу побудови системи

аналізу інформації виглядає наступним чином:

1. Знайти веб-ресурс з якого ми збираємося отримати дані. Цей процес можна поділити на додаткові, такі як аналіз структури Url адрес ресурсу, дізнатись про те як фільтри та їх значення впливають на структуру запиту.
2. Проаналізувати вміст сторінки або сторінок, на предмет інформації, яка нас цікавить. Дані зазвичай вкладені в теги. Отже, поглянемо на сторінку та побачимо, під яким тегом містяться дані, які ми хочемо отримати.
3. Знайти дані для вилучення у HTML (рис. 20).
4. Визначитись зі структурою проаналізованих даних.
5. Визначитись з варіантом зберігання даних. Формат зберігання залежить від потреб користувача.

Рис.*20 Поле даних, у структурі HTML.*

1.5 Огляд існуючих рішень

1.5.1 Parserok

Parserok.ru – реалізован у якості надбудови над MS EXCEL (рис. 21). Це рішення надає користувачу можливості формувати запити до веб-ресурсів із наданого переліку, має можливість додавання нових веб-ресурсів, але для цього користувачу потрібно буде звертатись до розробника, ця послуга коштує додаткову плату, також це призводить до додаткових часових витрати, в очікуванні створення нового сценарію роботи для парсеру. Результати роботи зберігаються у якості exel-таблиць.

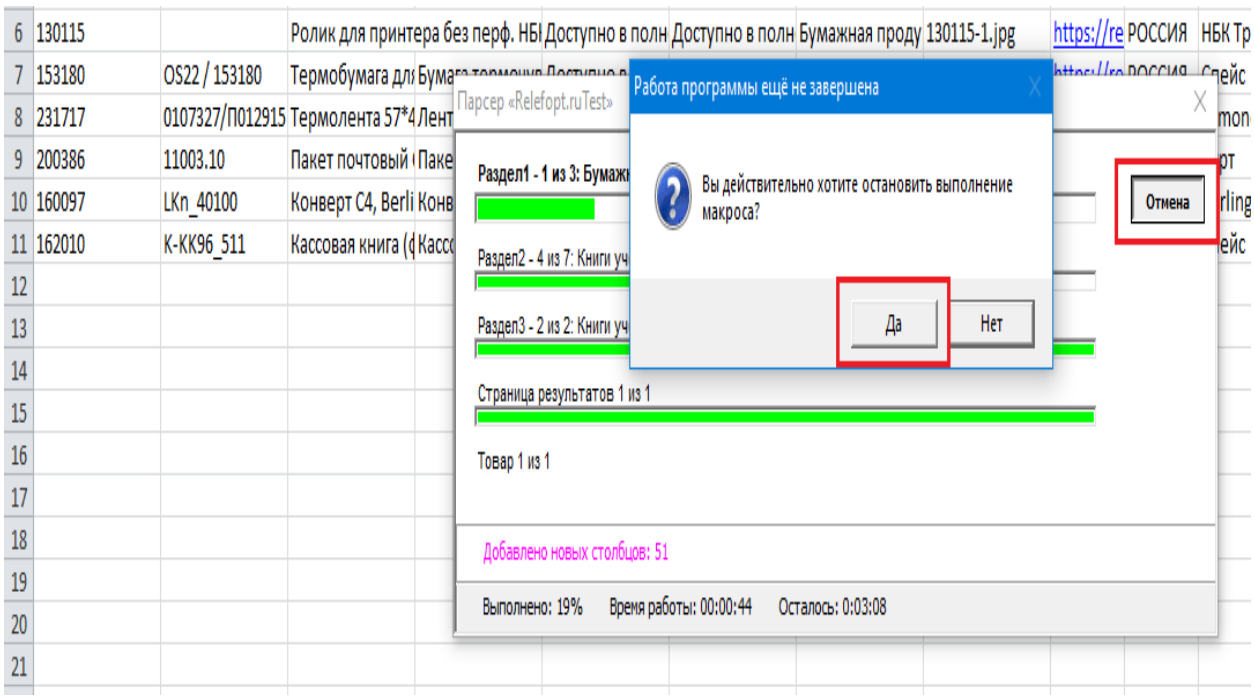


Рис. *21 Інтерфейс Parserok*

1.5.2 Datacol

Datacol – десктопний програмний застосунок, розроблений в м. Запоріжжі компанією ООО “Интернет-Автоматизация” (рис. 22). Має великий перелік ресурсів, з якими може працювати та надавати структуровані результати роботи автоматизованої системи аналізу, але, на жаль, попри ствердження розробників, цей програмний застосунок не є універсальним, а лише має певну кількість заздалегідь розроблених сценаріїв роботи їх програмного застосунку, для кожного з представлених у списку веб-ресурсів. Також є підстави вважати, що ця комп’ютерна система не матиме змоги аналізувати інформацію з веб-ресурсу, якщо той внесе зміни до структури сторінок до того часу, як розробники не перероблять сценарій і не опублікують оновлення. У тих випадках, коли користувачу потрібен певний ресурс, якого немає у переліку, потрібно буде замовляти у них розробку нового сценарію роботи, за окрему додаткову платню, та в такому випадку вони не вказують, чи будуть вони підтримувати працездатність сценарію роботи програмного застосунку у разі його зміни.

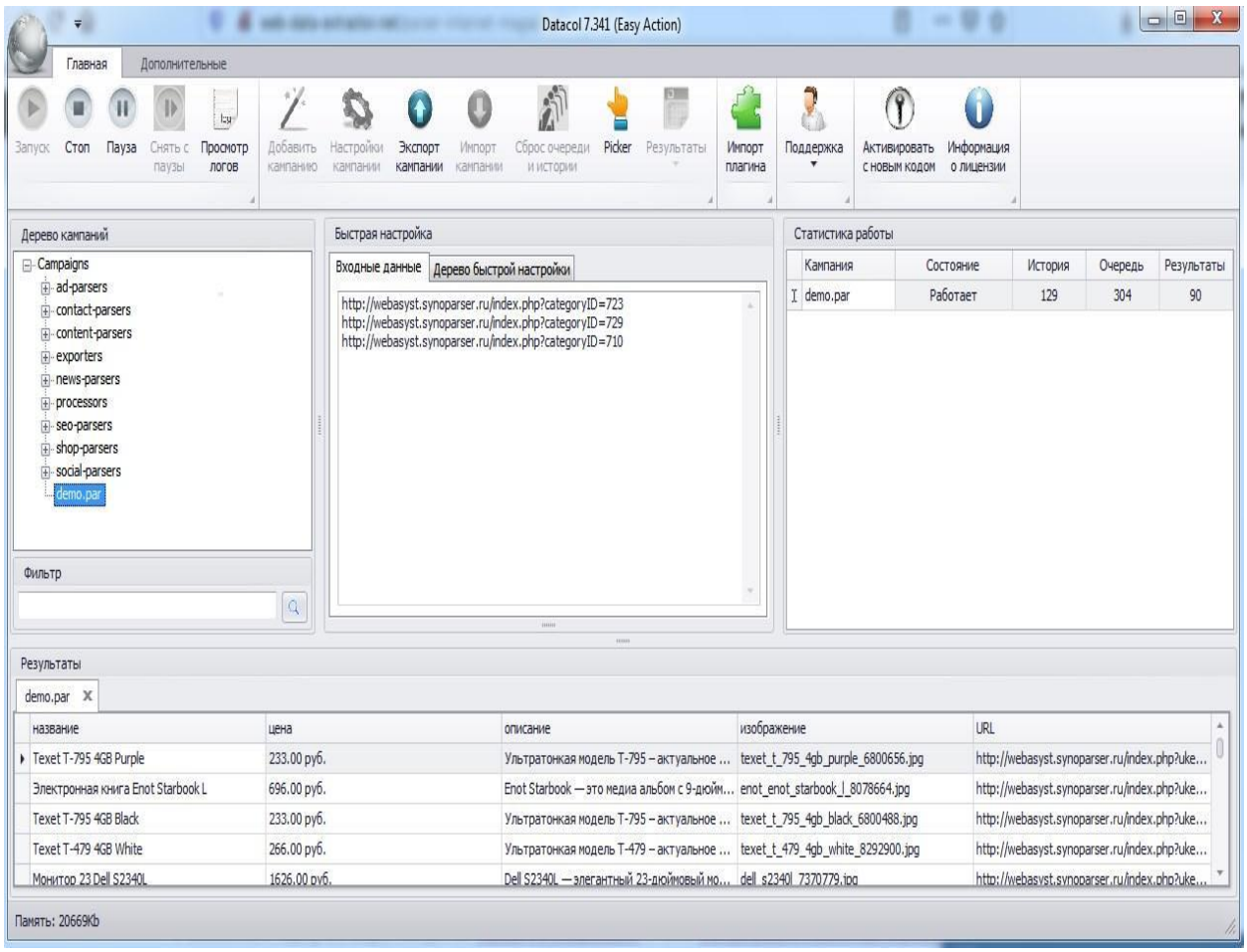


Рис. *22 Інтерфейс Datacol*

1.6 Висновки з розділу 1

В даному розділі було проведено огляд проблеми комп’ютерної системи для автоматизованого збору та аналізу інформації. Досліджено взаємодію користувача та веб-браузера з веб-ресурсом, яку він імітує під час збору даних. Детально розібрано основні етапи, поняття та принципи, які використовуються для взаємодії з веб-ресурсом. Було визначено важливі деталі, які впливають на роботу програмного застосунка, які забезпечує автоматизований збір та аналіз інформації з веб-ресурсів. Проаналізовані існуючі рішення. На основі цього, можна зробити та сформулювати наступні висновки:

1. Визначення ролі програмного застосунку для збору і аналізу відіграє важливу роль в розробці програмного забезпечення. Даний аспект впливає на успішність процесу розробки програмного забезпечення.

2. Досліджено взаємодію користувача з веб-ресурсом, детально розглянуто кожний аспект, які будуть використовуватись під час розробки програмного застосунку.

3. Проаналізовано етапи роботи веб-браузеру та веб-ресурсу.

4. Проаналізовано і сформовано етапи роботи програмного застосунку автоматизованого збору та аналізу даних, який імітує взаємодію веб-браузера та веб-серверу.

5. Детально розглянуто те, яку інформацію надає веб-ресурс на запит, які є можливі варіанти відповідей, і те, яку інформацію вони можуть містити.

6. Розглянуто різні варіанти веб-ресурсів, такі як динамічні та статичні, та сформовано розуміння того, як ці відмінності впливатимуть на роботу програмного застосунку.

7. Проаналізовано готові рішення, які реалізують автоматизований збір та аналіз веб-ресурсів.

РОЗДІЛ 2 ДОСЛІДЖЕННЯ ЗАСОБІВ АВТОМАТИЗОВАНОГО ЗБОРУ ТА АНАЛІЗУ ІНФОРМАЦІЇ З ВЕБ-РЕСУРСІВ

2.1 Python та .NET C# — інструменти для побудови програмних застосунків автоматизованого збору та аналізу даних

.Net C# надає користувачу певний перелік інструментів та бібліотек, які ми можемо використовувати під час побудови такого типу програмного забезпечення:

Для виконання запитів існує декілька варіантів рішення:

HttpClient являє собою клас для надсилання HTTP-запитів і отримання HTTP-відповідей від ресурсу, вказаного за допомогою URL. Має методи для виконання асинхронних запитів. Може виконувати запити та повертати відповіді веб-ресурсу у вигляді потоків, або строк. Екземпляр цього класу призначений для багаторазового використання упродовж всього часу роботи програмного застосунку. Якщо замість повторного використання екземпляра, ми будемо створювати новий для кожного запиту до сервера, кількість сокетів буде вичерпано, що у свою чергу призведе до помилок. Недоліком роботи класу є швидкість завантаження сторінок.

HttpWebRequest представляє пов’язану з Http реалізацію класу WebRequest. Має такі методи GetResponse, який робить синхронний запит до веб-ресурсу, та повертає відповідь у вигляді HttpWebResponse, який містить об’єкт відповіді. З міркувань безпеки файли cookie за замовчуванням вимкнені, але за потреби є можливість їх увімкнути.

RestSharp – призначення NuGet пакету, здійснювати синхронні та асинхронні запити до веб-ресурсів через HTTP протокол. Основною аудиторією є розробники, які використовують REST API. Окрім цього API RestSharp може викликати і інші через HTTP, за умови що в нас є URL та параметри запита, який ми збираємося надіслати, відповідають стандартам W3C HTTP.

Для розбору HTML існують наступні рішення:

HtmlAgilityPack

HtmlAgilityPack є однією з найбільш (якщо не найбільш) відомих бібліотек аналізу HTML у світі .NET. В результаті про неї написано безліч статей. Коротше кажучи, це швидка, відносно зручна бібліотека для роботи з HTML (за умови, що запити XPath прості). Ця бібліотека розбору буде зручною, якщо завдання типове і добре описано виразом XPath.:

CsQuery – порт JQuery для .NET 4. У ньому реалізовані всі селектори CSS2 та CSS3, всі методи маніпуляції DOM JQuery. На даний момент підтримка цього пакету розробником припинена, замість неї рекомендується використовувати AngleSharp.

AngleSharp — це бібліотека .NET, яка дає вам можливість аналізувати HTML, SVG. Важливим аспектом AngleSharp є те, що CSS також можна аналізувати. Включений аналізатор побудований на основі офіційної специфікації W3C. До особливості бібліотеки можна віднести:

- Портативність.
- Розширюваний.
- Повністю функціональній DOM.

Regex — однин з перших підходів вирішення задачі розбору HTML. Він дозволяє виконувати аналіз сторінок набагато швидше, ніж бібліотеки для побудови DOM.

Також для повної імітації запитів користувача є віртуальні браузери:

Headless Chrome — це чудовий інструмент для автоматизованого тестування та серверних середовищ, де вам не потрібна видима оболонка інтерфейсу користувача. Наприклад, ви можете запустити деякі тести на реальній веб-сторінці, створити її PDF-файл або просто перевірити, як браузер візуалізує URL-адресу.

Інструменти розробки на мові програмування Python — відмінний виборі для розробки систем збору та аналізу веб-ресурсів, оскільки він включає

нативні бібліотеки спеціально створені для процесу аналізу і обробки веб-ресурсів.

Ось чому Python є ідеальним варіантом для розробки такого типу програмних застосунків:

- Легкий у розумінні — процес читання коду Python схоже з читанням англійської мови, що робить синтаксис Python простим у вивченні. Він легкий для розуміння, читабельний.
- Менші часові витрати — програмні застосунки для автоматизованого збору і аналізу даних націлені на збереження часу, тому часові витрати на написання програмного застосунку теж мають значення. Python дозволяє писати короткі і прості фрагменти коду для комплексної задачі, та це зберігає час розробника.
- Велика кількість бібліотек —Python має величезну екосистему бібліотек, які містять інструменти та сервіси для різних цілей. Процес збору та аналізу даних з веб-ресурсів не є тому виключенням.
- Широка підтримка спільноти — Ми всі звертаємося за допомогою в певний момент часу, працюючи з великими обсягами даних. Python поставляється з однією з найпотужніших і найбільших спільнот, тому вам не потрібно турбуватися про усунення несправностей під час розробки будь-якого коду.

Існує кілька популярних бібліотек Python, доступних для ефективного процесу збору та аналізу даних з веб-ресурсів:

- Scrapy — бібліотека для збору інформації з веб-ресурсів, та вилучення її у структурованому вигляді. Бібліотеку можна використовувати для моніторингу, автоматизованого тестування. Недоліком бібліотеки є погана взаємодія з веб-застосунками основаними на javaScript.
- Selenium — безкоштовний, з відкритими кодом веб-драйвер, головним недоліком є те, що системи відстеження трафіку, швидко ідентифікують його коли ми використовуємо Webdriver для

відкриття великої кількості веб-сторінок. У такому випадку власникам веб-ресурсів навіть не потрібно розгортати комплексні системи захисту від автоматизованого збору.

- BeautifulSoup — одна з найкорисніших Python бібліотек для аналізу HTML і XML документів у деревоподібну структуру для ідентифікації та вилучення даних.

2.2 Бібліотеки та інструменти використані у системі автоматизованого збору та аналізу даних

Для реалізації комп'ютерних систем автоматизованих систем збору та аналізу інформації з веб-ресурсів, використовувалось наступні інструменти, які можуть покращити та полегшити розробку програмного застосунку.

1. HttpWebRequest — надав можливість робити швидкі запити, з параметрами до веб-ресурсів.

2 HtmlAgilityPack — пакет використовувався для побудови DOM об'єкту.

3. WPF — система для побудови клієнтських додатків, з візуально привабливим можливостями взаємодії з користувачем.

2.3 Висновки з розділу 2

В даному розділі був проведений аналіз інструментів, технології, бібліотек, які використовуються для побудови комп'ютерних систем автоматизованого збору та аналізу інформації з веб-ресурсів. На основі проведеного аналізу можна сформулювати такі висновки.

1. Python — оптимальний вибір для зручної, та швидкої розробки систем автоматизованого збору і аналізу, бібліотеки надають готову реалізацію процесів, які допомагають, без складнощів побудувати застосунок.

2. Для побудови десктопних застосунків автоматизованого збору та аналізу даних доречно використовувати мову C#, так як вона надає необхідні засоби, для побудови користувацького інтерфейсу, має реалізований функціонал для базових процесів взаємодії з веб-ресурсами.

4. Щоб полегшити розробку програмного забезпечення, краще використовувати готові бібліотеки та інструменти.

3. Ознайомлено та вивчено основні інструменти, які надає .NET, для побудови базового функціоналу програмного застосунку збору та аналізу даних з веб-ресурсів.

РОЗДІЛ 3 РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗОВАНОГО ЗБОРУ І АНАЛІЗУ ІНФОРМАЦІЇ З ВЕБ-РЕСУРСІВ

3.1 Призначення розробки

Побудовані програмні застосунки автоматизованого збору та аналізу даних призначенні для ознайомлення з процесом побудови програмних застосунків такого типу, аналізом того, як можливо автоматизувати певні процеси побудови такого типу програмного забезпечення. Визначити переваги та недоліки обох варіантів систем для автоматизованого збору та аналізу, та сформувати певне представлення у яких випадках вигідніше використовувати один з варіантів для поставлених майбутніх задач збору та обробки інформації. Виконати тестування для кожного застосунку та порівняти їх за отриманими показниками.

3.2 Функціональні та не функціональні вимоги

3.2.1 Функціональні вимоги

3.2.1.1 Функціональні вимоги статичної автоматизованої системи збору та аналізу даних з веб-ресурсів

Розроблений в ході виконання дослідження програмний застосунок для автоматизованого аналізу і оброки інформації повинен надавати наступні функціональні вимоги:

1. Користувач повинен мати змогу запустити програмний застосунок.
2. Користувач повинен мати змогу закрити програмний за стосунок.
3. Користувач може переглядати результати роботи програмного застосунку у наданій йому таблиці.
4. Програмний застосунок повинен виконувати Get запити за https.

5. Програмний застосунок повинен аналізувати отримані від веб-ресурсу дані, та формувати їх за певними критеріями.

3.2.1.2 Функціональні вимоги динамічної автоматизованої системи збору та аналізу даних з веб-ресурсів

Розроблене в ході виконання дослідження програмне забезпечення повинно надавати наступні функціональні вимоги:

1. Користувач повинен мати змогу обирати директорію для зберігання інформації яка генерується під час роботи програмного застосунку.
2. Користувач повинен мати можливість для створення нових запитів.
3. Користувач повинен мати можливість передивлятись перелік створених їм запитів.
4. Користувач повинен мати можливість створювати нові поля для запиту, які будуть використовуватись під час аналізу.
5. Користувач повинен мати змогу передивлятись створені ним поля для запитів.
6. Користувач повинен мати змогу редагувати поля запитів, та оновлювати Xpath для них.
7. Користувач повинен мати можливість запускати процес збору і аналізу для окремого запиту.
8. Користувач повинен мати можливість отримувати результати аналізу у вигляді файлу з структурованими даними.

3.2.2 Не функціональні вимоги

1. Зручний та зрозумілий інтерфейс програмного застосунку.
2. Уся взаємодія з програмним застосунком, відбувається через користувацький інтерфейс.
3. Просто у використанні програмного застосунку.

4. Швидке виконання збору та аналізу даних, та надання їх користувачу
5. Забезпечити можливість відновлення роботи програмного застосунку, після певних збоїв у його роботі.
6. Забезпечити користувачу зручну та зрозумілу можливість обирати директорію, де зберігати інформацію, яка потрібна для роботи програмного застосунку.

3.3 Діаграми використання

3.3.1 Діаграми використання у статичній автоматизованій системі збору та аналізу даних з веб-ресурсів

Відповідно до специфікації UML діаграма випадків використання - це діаграма, яка показує взаємозв'язки між учасниками та випадками використання в системі.

Дійовою особою є користувач — користувач якому потрібно отримати проаналізовані дані з певного веб-ресурсу.

Виходячи з потреб дійових осіб, виділяються наступні варіанти використання програмного застосунку (рис. 23):

1. Відкрити програмний застосунок.
2. Закрити програмний застосунок.
3. Переглянути результат обробки інформації з веб-ресурсу.
4. Почати процес автоматизованого збору та аналізу даних.

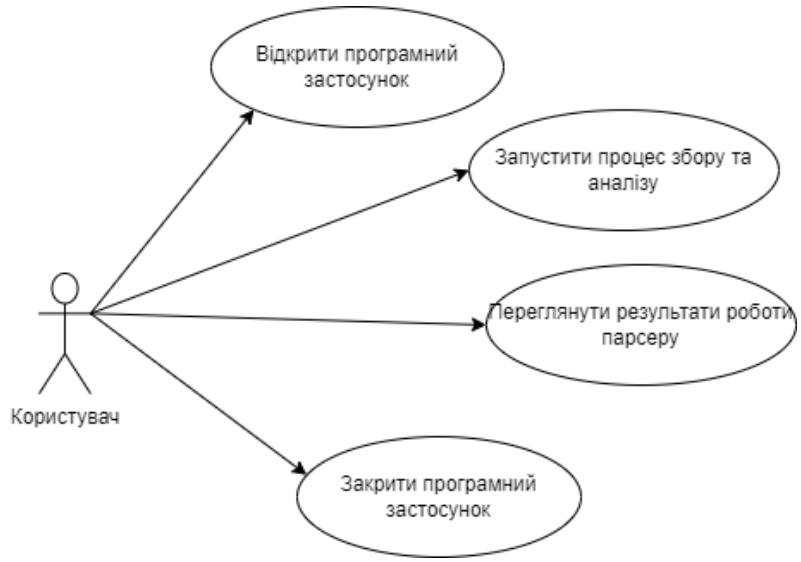


Рис. *23 Діаграма використання статичного програмного застосунку для збору і аналізу інформації з веб-ресурсу.*

3.3.2 Діаграми використання у динамічний автоматизованій системі збору та аналізу даних з веб-ресурсів

Дійовою особою у нижче наведених діаграмах використання є користувач. Користувач — людина, яка взаємодіє з програмним застосунком, з метою формування запитів, та отримання проаналізованих даних.

1. Для основного функціоналу програмного застосунку виділяються наступні варіанти використання користувачем(рис. 24):

- Запуск програмного застосунку.
- Закриття програмного застосунку.
- Обирання директорії де зберігаються дані потрібні для роботи програмного застосунку.
- Робота з запитами.

Рис. *24 Діаграма використання, для основного функціоналу*

2. Для взаємодії користувача з запитами виділяються наступні варіанти використання (рис. 25):

- Переглянути існуючі запити.
- Створити новий запит.
- Редагувати ім’я запиту.
- Редагувати текст посилання на елемент того, що цікавить на веб-ресурсі.
- Згенерувати новий Xpath на елемент.
- Запустити процес збору та аналізу запиту.
- Робота з полями запиту.
- Видалити запит.

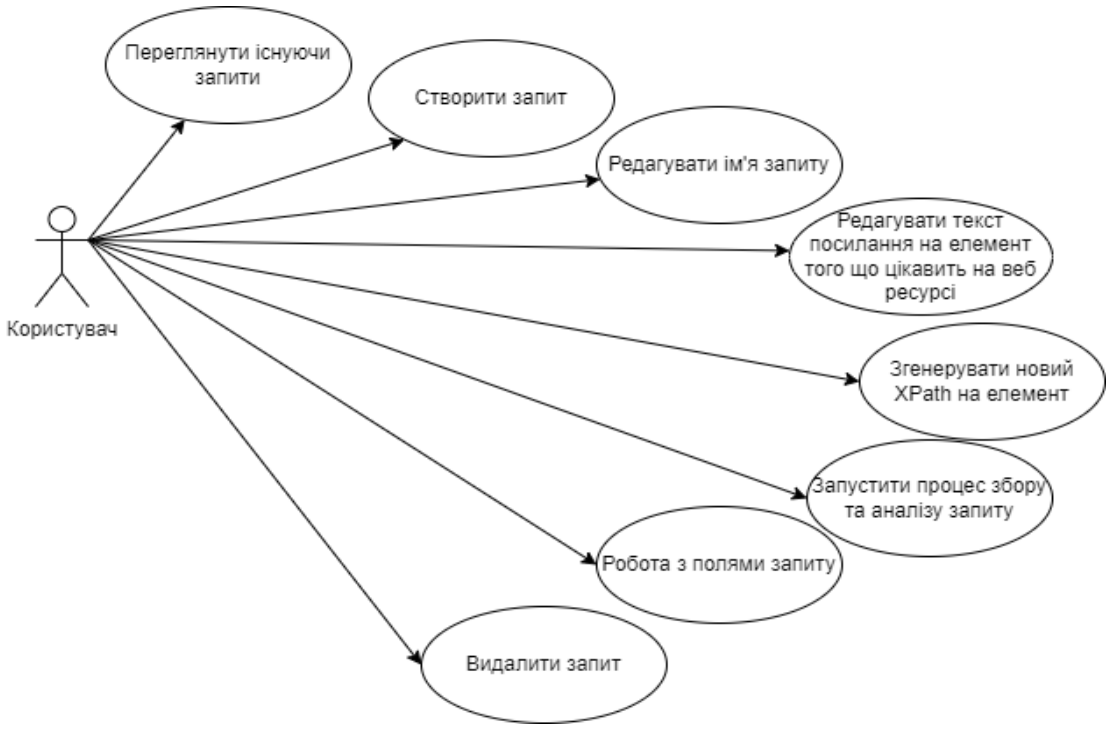


Рис.*25 Діаграма використання для вікна створення або редагування запитів користувачем.*

3. Для взаємодії користувача з полями запиту виділяються наступні варіанти використання (рис. 26):

- Переглянути існуючі поля запиту.
- Створити нове поле запиту.
- Редагувати ім’я поля запиту, яке використовується, під час форматування даних.
- Редагувати текстове поле по якому відбувається пошук відповідного поле у html сторінці.
- Оновити Xpath поля.
- Видалити поле запиту.

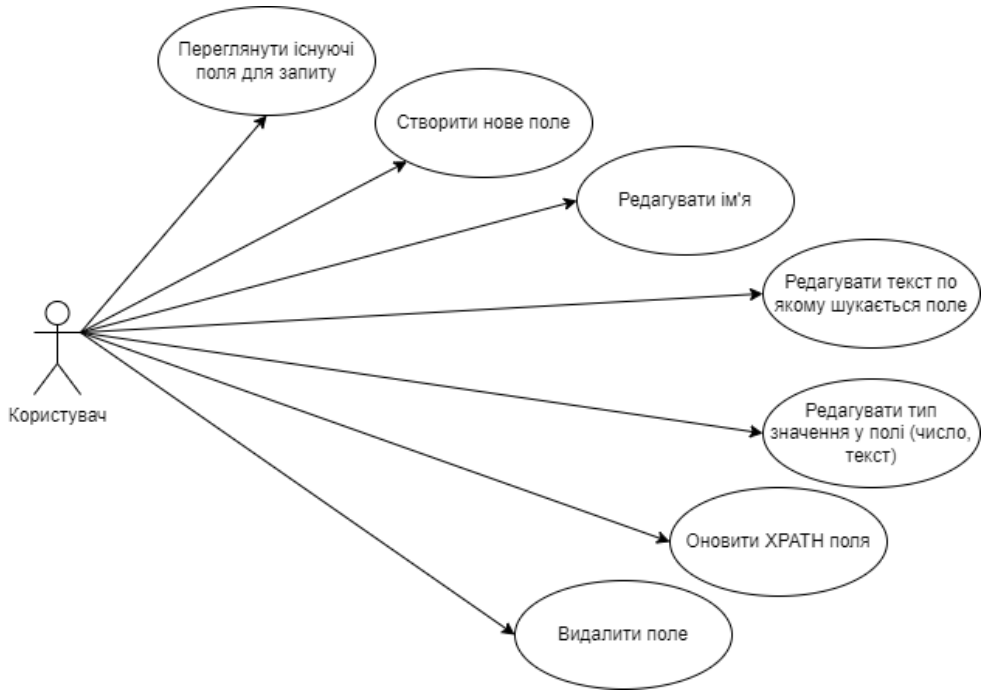


Рис.*26 Діаграма використання для вікна створення та редагування полів запиту.*

3.4 Вимоги до інтерфейсу

На основі аналізу функціоналу який повинен забезпечувати програмний застосунок, та вимог до проектування інтерфейсної частини програмного застосунку, було сформовано перелік рекомендацій для створення зрозумілого, та зручного для користувача інтерфейсу. Можна виділити наступні з них:

1. Головне вікно програмного забезпечення, воно повинно надавати користувачу перелік створених ним запитів, також вона повинна містити елементи, які дозволять вказати в якій директорії зберігаються запити, елементи які ініціюють процес створення нового запиту, та збереження поточного переліку запитів. Також треба реалізувати можливість переглядати поля запиту натискаючи два рази по рядку з запитом.
2. Вікно створення запиту, воно повинно містити поля, які користувач повинен заповнити, для створення нового запиту, а саме, та елемент для підтвердження створення нового запиту.

3. Вікно перегляду полів запита повинно містити перелік існуючих на даний проміжок часу полів запиту, надавати можливість ініціювати процес створення нового поля запиту, також зважаючи, на те що дане вікно являє собою запит, та його деталі, користувач у цьому вікні повинен мати можливість почати процес обробки та аналізу даних
4. Вікно створення або редагування поля запиту, повинно містити поля даних, та зрозумілі користувачу підписи до них, які потрібні для створення нового поля даних, в ньому користувач повинен мати можливість окрім створення, редагувати вже існуюче поле, та зберігати зміни.

3.5 Діаграми послідовності

3.5.1 Діаграма послідовностей у статичній автоматизованій системах збору та аналізу даних з веб-ресурсів

Користувач запускає програмний застосунок. Програмний застосунок робить екземпляр класу HTTPClient, та через нього відправляє Get запит до ресурсу. Після отримання відповіді від ресурсу, та перевірки статус коду, відсилає контент відповіді серверу до HTTP Agility pack там рядок конвертується до DOM документа, та надсилається до XPATH. Після отримання бажаних даних, по XPATH, зберігаємо результати, та записуємо у файл, після виконання викликаємо користувацький інтерфейс та відображаємо результати парсингу.

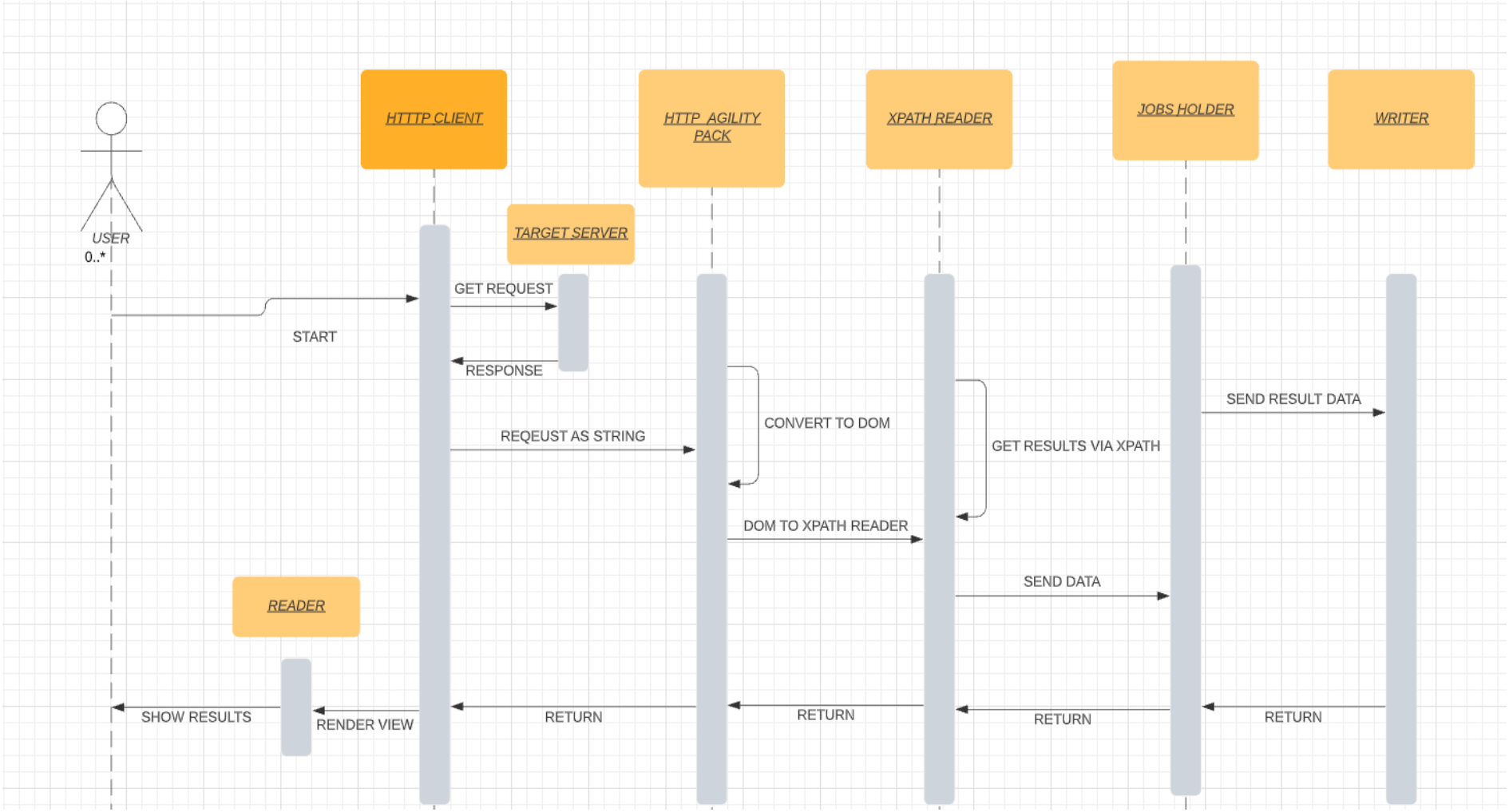


Рис. *27 Діаграма послідовностей для статичної системи автоматизованого збору та аналізу даних*

3.5.2 Діаграми послідовностей у динамічній автоматизованій системі збору та аналізу даних з веб-ресурсів

Діаграма послідовності для варіанта використання «Користувач обирає директорію для роботи програмного застосунку»

Користувач натискає на кнопку «Open folder», перед ним з’являється діалогове вікно системного провідника, він обирає де саме будуть зберігатись, або зберігаються дані, та натискає кнопку «обрати». Програмний застосунок отримає шлях до вказаного користувачем місця, та перевіряє чи існують папки та файли потрібні для роботи якщо файлів немає, то створює їх. Після цього програмний застосунок виводить основну інформацію про існуючі запити на екран.

Рис. 28 Користувач обирає директорію для роботи застосунку

Діаграма послідовності для варіанта використання «Користувач створює новий запит»

Користувач натискає кнопку «Create new request», перед ним з'являється нове вікно, де йому потрібно вказати:

1.Назву запиту

2. Головну URL адресу веб-ресурсу.

3. URL адресу, яка містить список інформації та посилання на сторінки з нею.

4. Url посилання на сторінку з інформацією.

5. Текстовий допис який супроводжує посилання на сторінку з інформацією.

Після того як користувач вказав ці дані, та натиснув кнопку «Save request», програмний застосунок записує інформацію у файл запитів, додає туди згенерований Xpath, який формується по текстовому допису. Створює файл, який буде містити поля запиту, з назвою запиту яку вказав користувач.

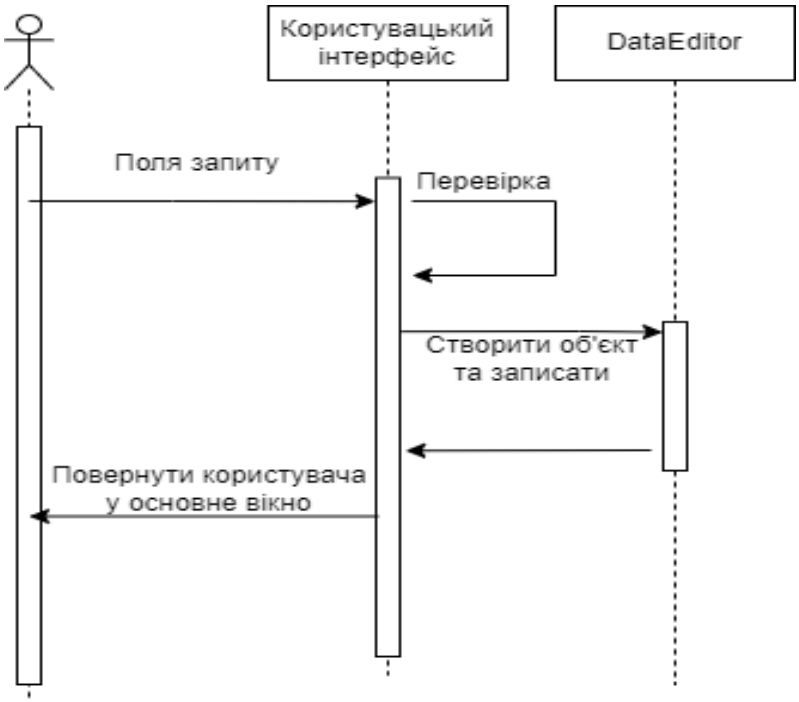


Рис. *29 Користувач створює запит*

Діаграма послідовності для варіанта використання «Користувач відкриває інформацію про запит»

Користувач натискає два рази на рядок з запитом, перед ним з'являється нове вікно, де він може:

1. Передивитись список полів запиту.
2. Створити нове поле запиту
3. Зберегти зміни.
4. Запустити процес збору та аналізу веб-ресурсу.

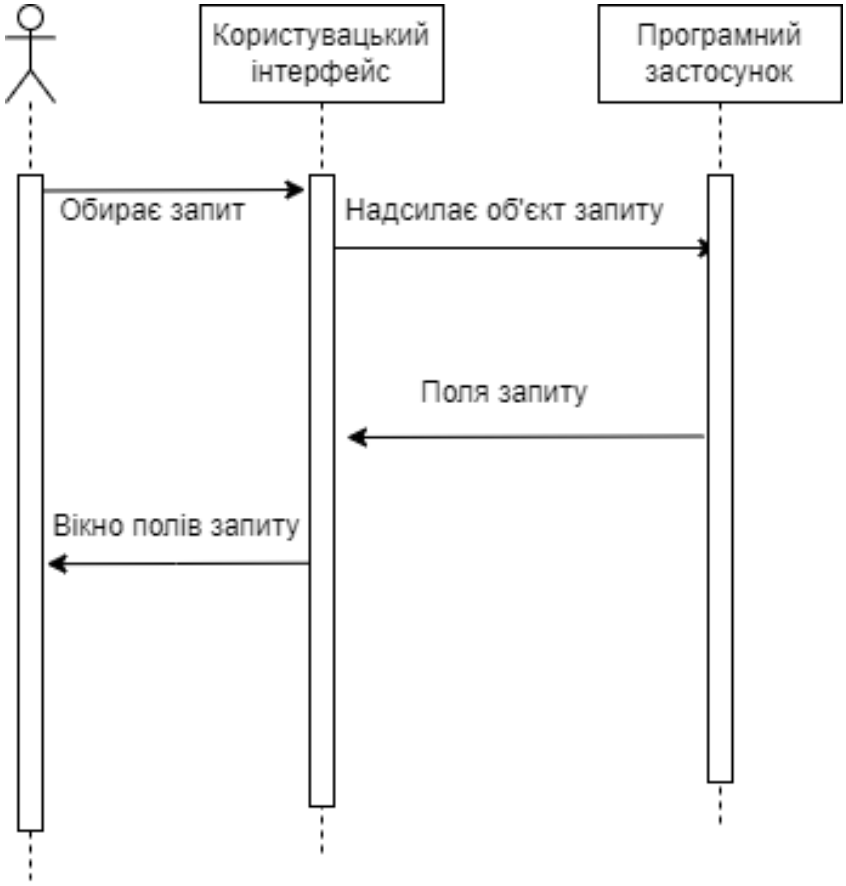


Рис. *30 Користувач переглядає запити*

3.6 Діаграми класів

3.6.1 Діаграма класів у статичній автоматизованій системі збору та аналізу даних з веб-ресурсів

На діаграмі (рис. 31), представленій нижче, зображено відношення між створеними класами у статичній системі автоматизованого збору та інформації.

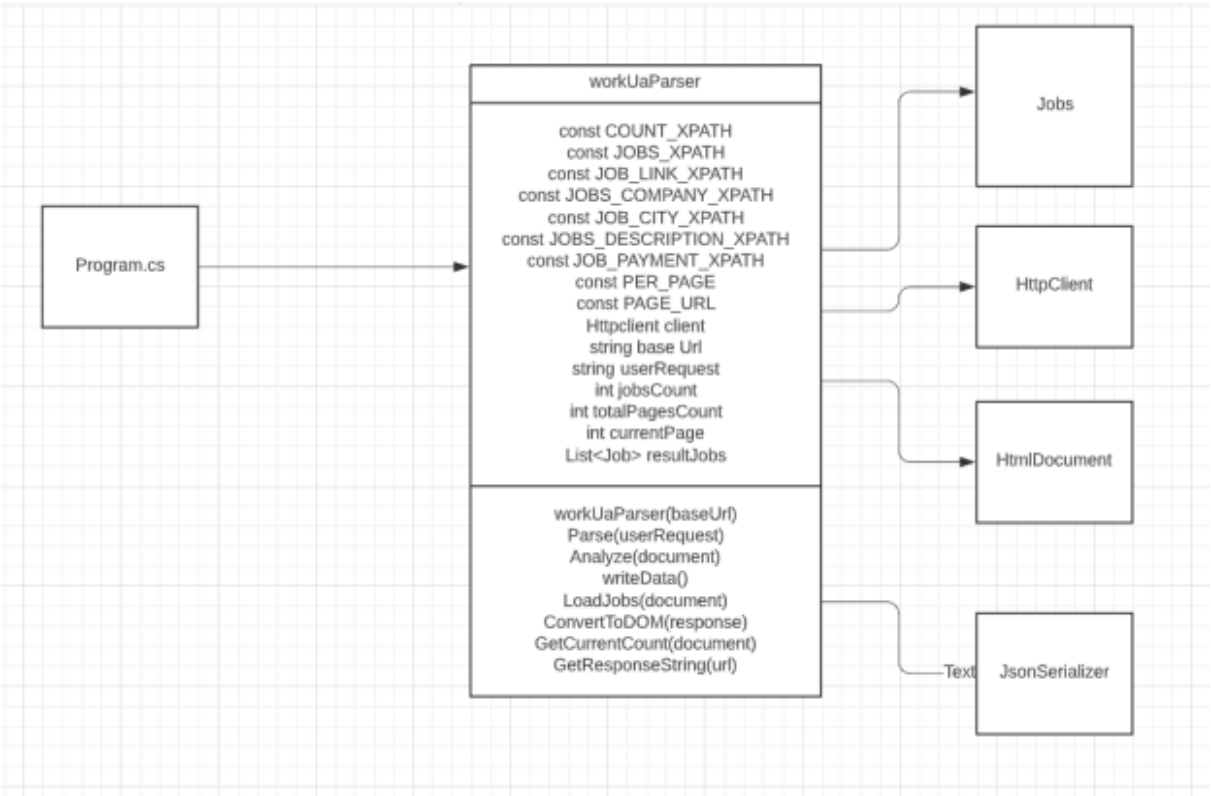


Рис. *31 діаграма класів статичної системи.*

3.6.2 Діаграми класів у динамічній автоматизованій системі збору та аналізу даних

Діаграма класів (рис. 32), програмного застосунку динамічного автоматизованого збору та аналізу інформації.

Клас DataEditor містить методи запису та читання файлів типу Json.

Клас DomAnalyze містить методи взаємодії з пакетом HtmlAgilityPack для побудови Dom моделі документу, та проведення аналізу.

Клас WebClient містить методи для виконання запитів, та повернення відповіді від веб-серверу у вигляді html рядка.

Клас UserRequest містить основну інформацію для побудови запиту користувача.

Клас UserRequestField містить основну інформацію про поля запиту користувача.

Клас Parser основний клас, відповідає за процес запитів до серверу, аналізу інформації, та запису отриманих результатів до файлів.

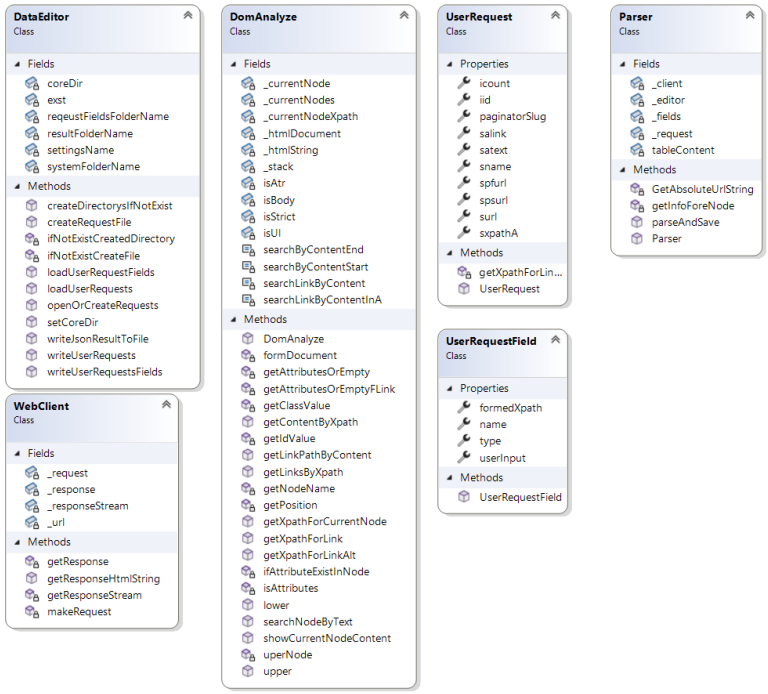


Рис. *32 діаграма класів у динамічній системі*

3.7 Засоби реалізації

Побудовані програмні застосунки були виконані у середовищі розробки Visual Studio 2022 Community Edition, об’єктно орієнтованою мовою C#, для побудови користувацького інтерфейсу використовувався WPF. Для запитів до веб-серверу використовувався стандартний клас .NET WebClient. Для побудови DOM об’єкту та взаємодії з ним використовувався HtmlAgilityPack.

3.8 Модулі і алгоритми

3.8.1 Модулі і алгоритми у статичній автоматизованій системі збору та аналізу даних з веб-ресурсів

В лістингу 1 відображено константи у парсері для XPATH, по перше ми виключаємо можливість помилки розробника у майбутньому при зміні логіки програми, також зробивши по константі на елементи які нас цікавлять, ми зможемо швидко змінити їх у разі змін на сайти. Змінна Client яку ми одразу ініціалізуємо надає нам можливість робити запити на сервер, та переглядати їх. BaseUrl основна частина URL сайту, userRequest строкова змінна, містить дані які ввів користувач.

Лістинг *1 Поля класу workUaParser*

```
    private const string COUNT_XPATH = "//div[con-
tains(@class,'card')]/div[contains(@class,'add-
top')]";
            private const string JOBS_XPATH =
«//div[contains(@class,'job-link')]»;
            private const string JOB_LINK_XPATH =
«.//h2/a»;
            private const string JOB_COMPANY_XPATH =
«.//div/span/b»;
            private const string JOB_CITY_XPATH =
«.//div/span[3]»;
            private const string JOB_DESCRIPION_XPATH =
«.//p[contains(@class,'cut-bottom')]»;
            private const string JOB_PAYMENT_XPATH =
«.//div/b»;
            private const int PER_PAGE = 14;
            private const string PAGE_URL = «?page=»;
            private HttpClient client = new HttpCli-
ent();
            private string baseUrl;
            private string userRequest;
            private int jobsCount = 0;
            private int totalPagesCount = 0;
            private int currentPage = 1;
            private List<Job> resultJobs = new
List<Job>();
```

Лістинг 2 відображає метод Parse викликається ззовні, починає процес обробки та аналізу даних.

Лістинг *2 Метод Parse*

```
Public async Task Parse(string userRequest)
        {
            this.userRequest = userRequest;
            string response = await
GetResponseString(userRequest);
        this.Analyze(ConvertToHtmlDocument(re-
sponse));
        }
```

В лістингу 3 відображено метод Analyze який перевіряє кількість існуючих елементів, а саме у нашому випадку вакансії та підраховує скільки сторінок треба буде ще запитати у сервера. Робить запит до сервера для кожної окремої вакансії та передає її в у метод LoadJobs.

Лістинг *3. Метод Analyze*

```
         private async void Analyze(HtmlAgili-
tyPack.HtmlDocument document)
        {
            this.jobsCount = this.GetCur-
rentCount(document);
            this.totalPagesCount = this.jobsCount /
PER_PAGE + (this.jobsCount % PER_PAGE);
            LoadJobs(document);
            do
            {
                this.currentPage++;
                string nextUrl =  this.userRequest
+ PAGE_URL + currentPage.ToString();

                Debug.WriteLine(nextUrl);
            }
            while (this.currentPage <= this.to-
talPagesCount);
            WriteData();
        }
```

В лістингу 4 відображено метод LoadJobs він приймає DOM документ та проводить аналіз за регулярними виразами, та додає результати обробки у список вакансії

Лістинг *4. Метод LoadJobs*

```
            private void LoadJobs(HtmlAgili-
tyPack.HtmlDocument document)
            {
                HtmlNodeCollection jobs = document.Doc-
umentNode.SelectNodes(JOBS_XPATH);
                foreach (HtmlNode job in jobs)
                {
                    var    titleAndLink    =    job.Se-
lectSingleNode(JOB_LINK_XPATH);
                    string url = this.baseUrl + ti-
tleAndLink.Attributes["href"].Value;
                    string title = titleAndLink.Attrib-
utes["title"].Value;
                    string company = job.SelectSingle-
Node(JOB_COMPANY_XPATH).InnerText;
                    string  city  =  job.SelectSingle-
Node(JOB_CITY_XPATH).InnerText;
                    string   description   =   job.Se-
lectSingleNode(JOB_DESCRIPION_XPATH).InnerText;
                    var   prom   =   job.SelectSingle-
Node(JOB_PAYMENT_XPATH);
                    string payment ="";
                    if(prom != null)
                    {
                        payment = prom.InnerText;
                    }
                    resultJobs.Add(new Job()
                    {
                        url = url,
                        title = title,
                        company = company,
                        city = city,
                        description = description,
                        payment = payment
                    });
                }
                return;
            }
```

3.8.2 Модулі і алгоритми у динамічній автоматизованій системі збору та аналізу даних з веб-ресурсів

В лістингу 5 відображено методи класу DataEditor для отримання та запису інформації про запити користувача у Json файл.

Лістинг 5. *Методи класу DataEditor для запису та отримання інформації*

```
       public void writeJsonResultToFile(string name, string
json)
            {
                string file = Path.Combine(coreDir,
resultFolderName, name + exst);
                using (StreamWriter outputFile = new
StreamWriter(file))
                {
                    outputFile.WriteLine(json);
                }
            }

            public BindingList<UserRequest>
openOrCreateRequests()
            {
                BindingList<UserRequest> requests = new
BindingList<UserRequest>();
                string file = Path.Combine(coreDir,
systemFolderName, settingsName + exst);

                ifNotExistCreateFile(file);
                using (StreamReader r = new
StreamReader(file))
                {
                    string json = r.ReadToEnd();
                    if (!string.IsNullOrEmpty(json))
                        requests =
JsonSerializer.Deserialize<BindingList<UserRequest>>(json);
                }
                return requests;
            }
```

На лістингу 6 відображено метод parseAndSave класу Parse він відповідає за процес надсилання запитів до веб-серверу, та подальшого виокремлення інформації згідно з сформованими полями запиту.

Лістинг 6. *Метод parseAndSave*

```
      string currentLink = _request.spfurl;
                    string htmlResponse =
_client.getResponseHtmlString(currentLink);
                    DomAnalyze dom = new
DomAnalyze(htmlResponse);
                    var response =
dom.getLinksByXpath(_request.sxpathA);
                    foreach (HtmlNode node in response)
                    {
                        string link =
GetAbsoluteUrlString(_request.surl,
node.Attributes["href"].Value);
                        if(node.Attributes["href"].Value !=
"#")
                        {
                            getInfoForeNode(link);
                        }
                    }
                    JsonSerializerOptions options = new
JsonSerializerOptions
                    {
                        Encoder =
JavaScriptEncoder.Create(UnicodeRanges.All)
                    };
                    string json =
JsonSerializer.Serialize(tableContent,options);

_editor.writeJsonResultToFile(_request.sname, json);
```

На лістингу 7 відображено метод getAttributesOrEmptyFLink класу DomAnalyze для формування Xpath згідно з наявними у елемента атрибутами.

Лістинг *7. Метод getAttributesOrEmptyFLink*

```
    string attributes = string.Empty;
                string nodeName = getNodeName(node);
                string nodeId = getIdValue(node);
                string nodeClass = getClassValue(node);
                if (isAttributes(nodeClass, nodeId) is
false)
                {
                    attributes += nodeName ;
                }
                else
```

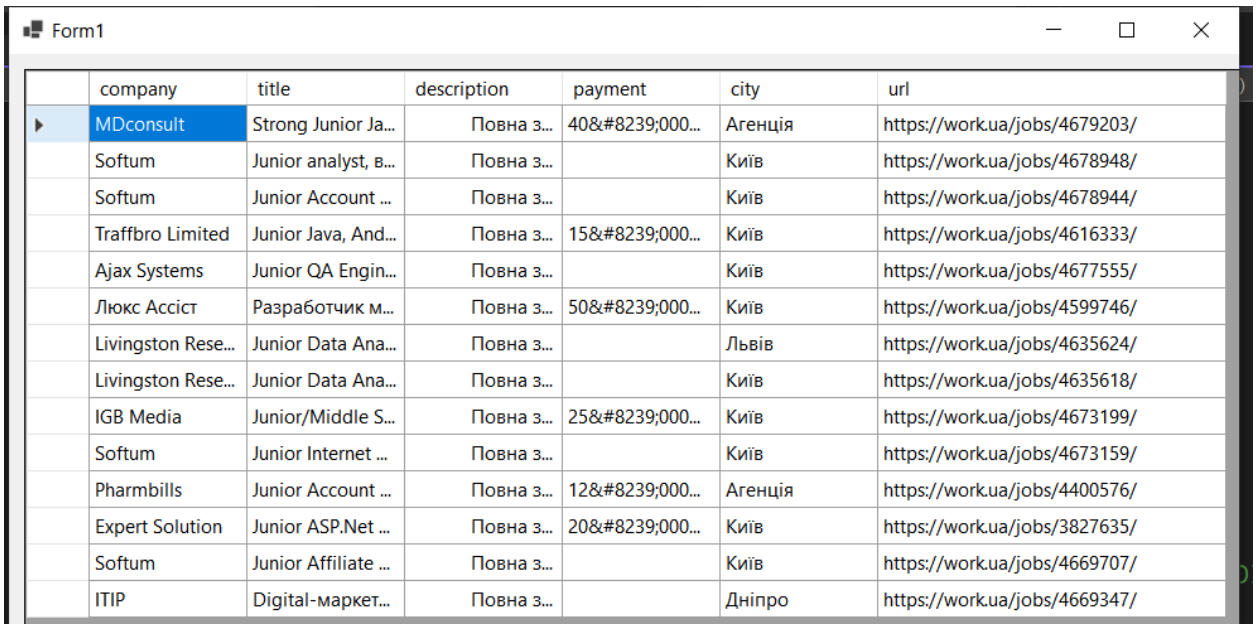
```
                {
                    isAtr = true;
                    if (!String.IsNullOrEmpty(nodeId) &&
!String.IsNullOrEmpty(nodeClass))
                    {
                        attributes += nodeName + "[" +
nodeId + " and " + nodeClass + "]";
                    }
                    else if
(!String.IsNullOrEmpty(nodeClass))
                    {
                        attributes += nodeName + "[" +
nodeClass + "]";
                    }
                    else
                    {
                        attributes += nodeName + "[" +
nodeId + "]";
                    }
                }
                return attributes;
```

3.9 Проєкт інтерфейсу

3.9.1 Проєкт інтерфейсу статичної системи автоматизованого збору і аналізу інформації

Головним критерієм під час розробки інтерфейсу програмного застосунку було надати користувачу зручну, просту у розумінні форму відображення отриманих під час обробки даних. Інтерфейс був побудований з урахуванням того, що користувачу немає потреби взаємодіяти з процесом збору та аналізу даних.

Ілюстрація наведена нижче (Рис.33) демонструє інтерфейс програмного застосунку збору та аналізу інформації.



company	title	description	payment	city	url
MDconsult	Strong Junior Ja...	Повна з...	40 000...	Агенція	https://work.ua/jobs/4679203/
Softum	Junior analyst, в...	Повна з...		Київ	https://work.ua/jobs/4678948/
Softum	Junior Account ...	Повна з...		Київ	https://work.ua/jobs/4678944/
Traffbro Limited	Junior Java, And...	Повна з...	15 000...	Київ	https://work.ua/jobs/4616333/
Ajax Systems	Junior QA Engin...	Повна з...		Київ	https://work.ua/jobs/4677555/
Люкс Асciст	Разработчик м...	Повна з...	50 000...	Київ	https://work.ua/jobs/4599746/
Livingston Rese...	Junior Data Ana...	Повна з...		Львів	https://work.ua/jobs/4635624/
Livingston Rese...	Junior Data Ana...	Повна з...		Київ	https://work.ua/jobs/4635618/
IGB Media	Junior/Middle S...	Повна з...	25 000...	Київ	https://work.ua/jobs/4673199/
Softum	Junior Internet ...	Повна з...		Київ	https://work.ua/jobs/4673159/
Pharmbills	Junior Account ...	Повна з...	12 000...	Агенція	https://work.ua/jobs/4400576/
Expert Solution	Junior ASP.Net ...	Повна з...	20 000...	Київ	https://work.ua/jobs/3827635/
Softum	Junior Affiliate ...	Повна з...		Київ	https://work.ua/jobs/4669707/
ITIP	Digital-маркет...	Повна з...		Дніпро	https://work.ua/jobs/4669347/

Рис.*33 Інтерфейс статичного застосунку.*

3.9.2 Проект інтерфейсу динамічної системи автоматизованого збору і аналізу інформації

Головними критеріями під час розробки інтерфейсу програмного застосунку були простота та легкість у використанні, тільки у такому випадку користувач зможе зручно використовувати програмний застосунок, та в такому випадку програмний застосунок може отримати популярність, та багато нових користувачів. Інтерфейс програмного застосунку повинен враховувати, що користувач не повинен мати додаткових знань для роботи з програмним застосунком.

Ілюстрації представлені нижче (Рис. 34-37), демонструють інтерфейс програмного застосунку який був побудований за допомогою WPF.

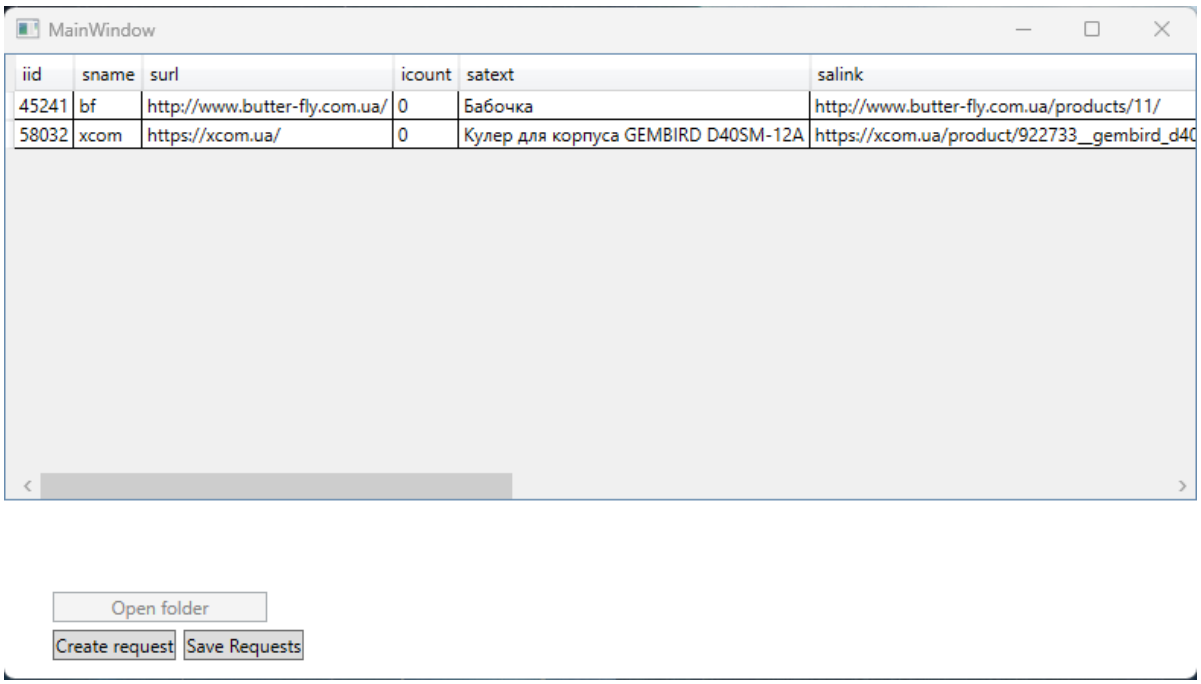


Рис. *34 Головне вікно програмного застосунку*

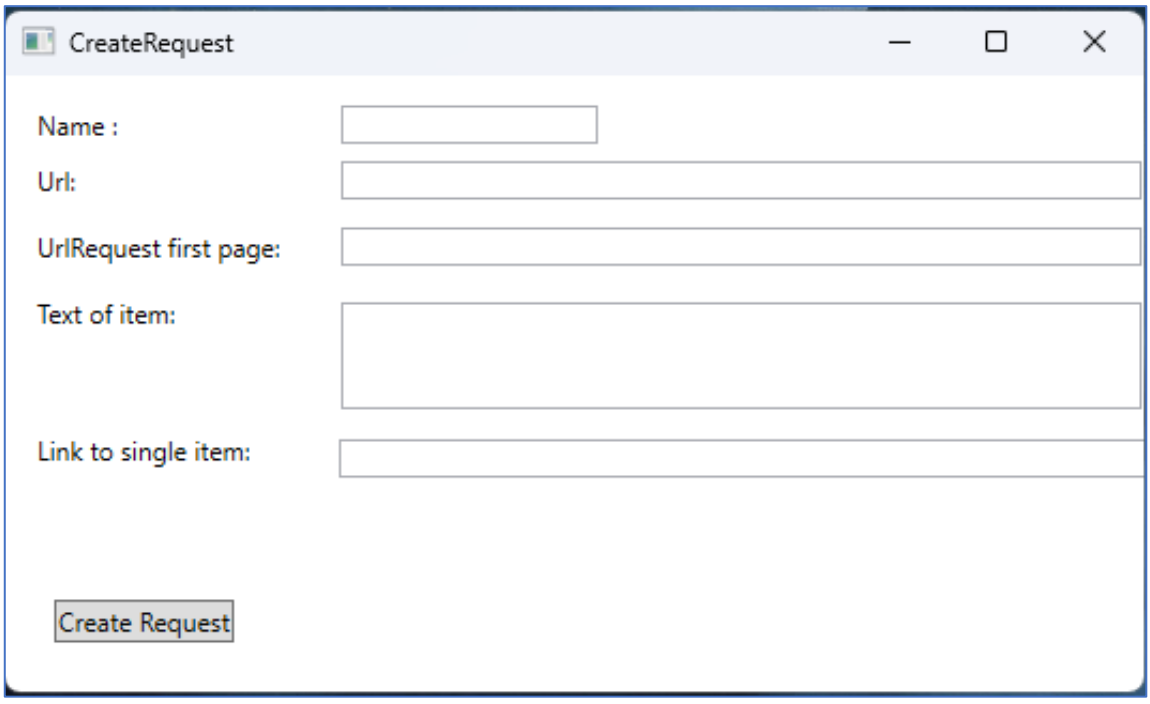


Рис. *35 Вікно створення запиту*

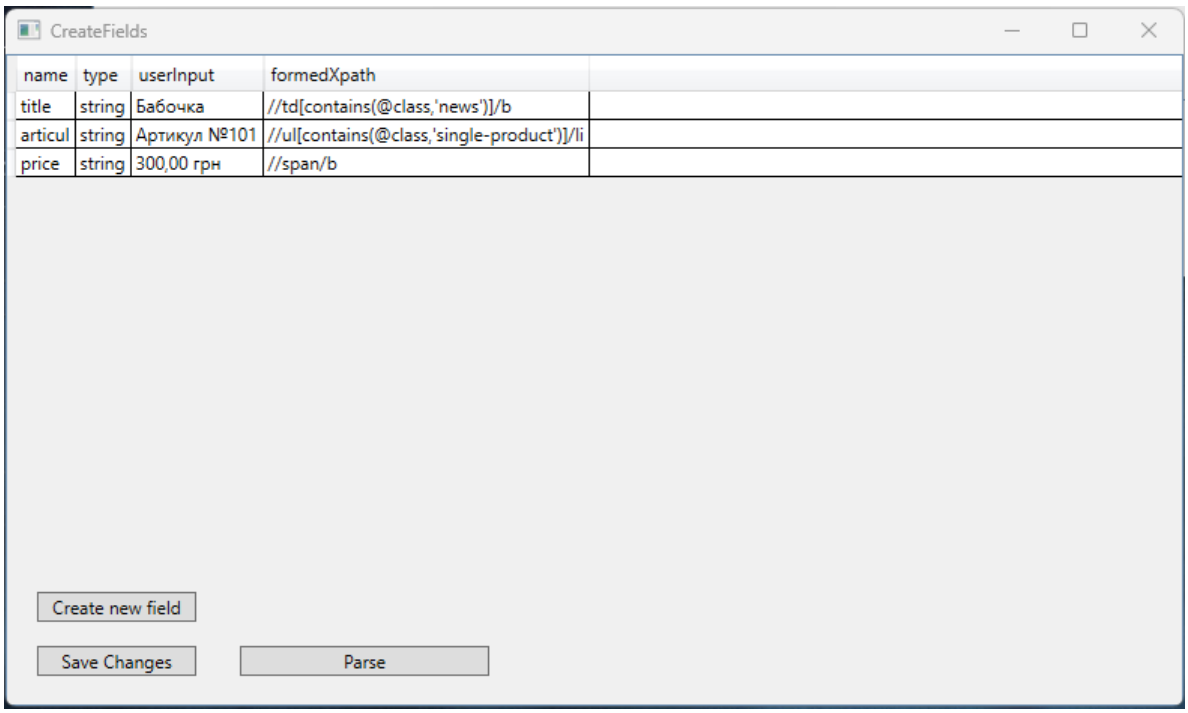


Рис. *36 Вікно полів запиту*

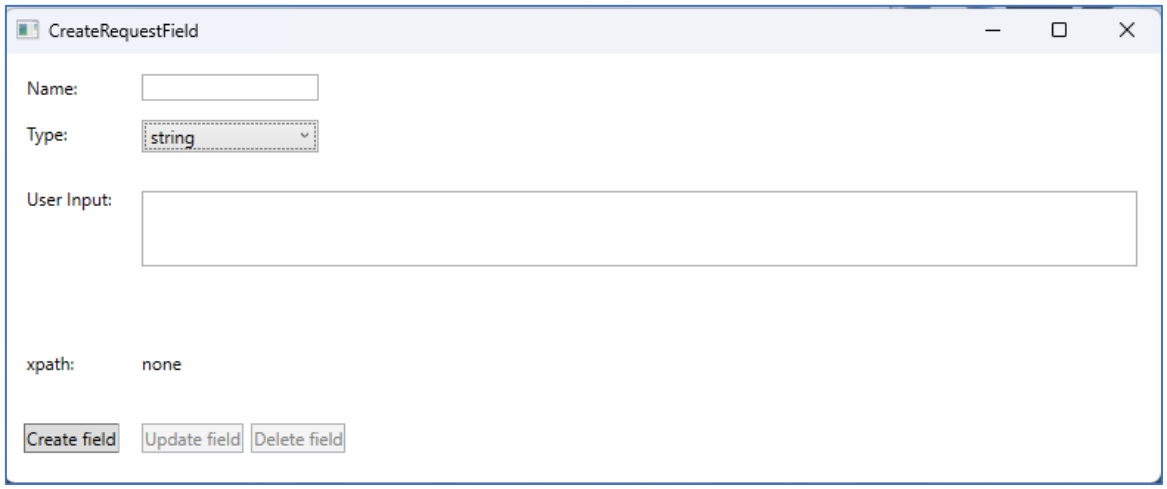


Рис.*37 Вікно створення поля запиту.*

3.10 Висновки з розділу 3

В даному розділі було реалізовано програмні застосунки статичного та динамічного автоматизованого збору і аналізу інформації з веб-ресурсів, проведено аналіз функціональних та нефункціональних вимог до програмних застосунків, побудовано діаграми класів, діаграми використання, діаграми послідовностей, тощо. На основі цих дій можна сформувати наступні висновки:

1. Було реалізовано програмні застосунки статичного та динамічного автоматизованого збору і аналізу.

2. Побудовано та надано опис діаграм класів застосунків статичного та динамічного автоматизованого збору і аналізу.

3. Використовувався пакет HtmlAgilityPack для побудови DOM об’єкта та взаємодії з ним.

4. Програмний застосунок надає можливість побудови запитів.

5. Програмний застосунок має простий та зрозумілий у використанні користувацький інтерфейс.

6. Програмний застосунок використовує Json файл для зберігання даних.

РОЗДІЛ 4 ДОСЛІДЖЕННЯ РЕЗУЛЬТАТІВ РОБОТИ КОМП'ЮТЕРНОЇ СИСТЕМИ ДЛЯ АВТОМАТИЗОВАНОГО ЗБОРУ І АНАЛІЗУ ІНФОМОРАЦІЇ З ВЕБ-РЕСУРСІВ

4.1 Перевірка працездатності програмного застосунку динамічного автоматизованого збору і аналізу інформації з веб-ресурсів

Програмний застосунок динамічного автоматизованого збору і аналізу інформації з веб-ресурсів, було перевірено шляхом формування запитів до декількох веб-ресурсів. На рис. 38 відображено результати роботи програмного застосунку для сформованого запиту. Було виявлено, що варіант побудови шляхів по Xpath має декілька недоліків, а саме:

1. Можливість пошуку по тексту можлива тільки в тих випадках коли текст на html сторінці, та у запитів сходяться за регістром.

2. Проблеми з кодуванням спеціальних символів таких як нерозривного пробілу, в таких випадках користувачу, щоб сформувати поле запиту потрібно вказувати кодування елементу « ».

3. Дублювання інформації на веб-сторінці. Відповідно до того, що ми формуємо запит згідно до текстового виразу можливі випадки, коли буде сформовано Xpath на інший елемент сторінки.

4. Дублювання css селекторів на сторінці, в деяких веб-ресурсах структура документа має таку побудову селекторів, що, наприклад технічні характеристики, та детальний опис можуть мати однаковий шлях, якщо формувати Xpath по селекторам.


```
{
   "url":"http://www.butter-fly.com.ua/products/14/",
   "name":"Солнышки",
   "articul":"Артикул №104 \t\t\t\t\t\t\t"
},
```


Рис. *38 Приклад результату обробки*

Також було виявлено вимоги, до формату веб-ресурсів, інформація для якої ми формуємо запит повинна містить на кожній сторінці (вакансія, товар, послуга тощо). Для можливості поліпшити ці вади було надано можливість вносити зміні напряму у збудовані Xpath, внесені зміни застосовуються з часу натискання користувачем кнопки «Save».

Програмний застосунок розрахований на веб-застосунки типу WPA.

4.2 Варіанти покращення програмного застосунку

На основі проведеного дослідження, аналізу взаємодії користувача з веб-ресурсом та процесу аналізу результатів роботи програмного застосунку можливі такі покращення:

1. Реалізація переходів між знайденими по текстовій інформації елементам, та побудова шляху саме до них.

2. Реалізація переходу до батьківського елементу.

3. Реалізація більш гнучкого налаштування URL адрес та взаємодії з ними.

4. Застосування проксі для запобігання блокування запитів веб-ресурсом.

4.3 Висновки з розділу 4

В даному розділі було проведено дослідження різних факторів, які впливають на процес роботи програмного застосунку. Для покращення результатів роботи програмного застосунку було надано декілька варіантів подальшого розвитку.

ВИСНОВКИ

В ході виконання дипломної роботи було запроєктовано та реалізовано програмний застосунок автоматизованого збору та аналізу інформації з веб-ресурсів. Тому, можна зробити наступні висновки:

1. Досліджено проблеми побудови програмних застосунків автоматизованого збору та аналізу інформації.

2. Проведено теоретичні дослідження у галузі систем автоматизованого збору та аналізу інформації.

3. Досліджено засоби, бібліотеки, інструменти для вирішення поставленої проблеми.

4. Реалізовано та проведено перевірку працездатності програмного застосунку автоматизованого збору та аналізу інформації з веб-ресурсів.

5. Визначено переваги та недоліки побудованого рішення для універсального автоматизованого збору та аналізу інформації.

6. Сформовано варіанти покращення програмного застосунку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Буч Г. Об'єктно-орієнтовне проектування з прикладами застосування. - Київ : Видавничий центр «Академія», 2002.
2. Введение в Razor Pages. URL : https://metanit.com/sharp/aspnet5/29.1 (дата звернення 01.12.2022).
3. Гайдамакин Н.А. Автоматизовані інформаційні системи, бази і банки даних. Ввідний курс. Київ : Гелиос АРВ, 2002.
4. Гайна Г.А. Основи проектування баз даних: Навчальний посібник. Київ : Кондор, 2008.
5. Entity Framework Core. URL: https://www.entityframeworktutorial.net/efcore/entity-framework-core (дата звернення 01.12.2022).
6. Офіційна документація Microsoft з ASP.Net Core: Introduction to ASP.NET Core. URL: https://docs.microsoft.com/en-us/aspnet/core/ (дата звернення 01.12.2022).
7. Пасічник В.В., Резніченко В.А. Організація баз даних та знань. . Київ : Видавнича група BHV, 2006.
8. Фрімен А. ASP.NET Core MVC с примерами на C# для профессионалов – Вильямс, 2017.
9. Hypertext Transfer Protocol (Http/1.1): Semantics and Content RFC 7231 . URL: https://datatracker.ietf.org/doc/rfc7231/ (дата звернення 01.12.2022).
10. Інформаційний документ що містить технічну специфікацію Hypertext Markup Language 2.0. URL: https://www.rfc-editor.org/rfc/rfc1866 (дата звернення 01.12.2022).
11. Інформаційний документ, що містить технічні специфікації Https. URL: https://www.rfc-editor.org/rfc/rfc2818 (дата звернення 01.12.2022).

12. Інформаційний документ що містить технічну специфікацію TLS протоколу. URL: https://www.rfc-editor.org/rfc/rfc2246 (дата звернення 01.12.2022).
13. Офіційний ресурс про HTML специфікацію W3C. URL: https://www.w3.org/TR/ (дата звернення 01.12.2022).
14. Офіційний ресурс про HTML специфікацію WHATWG. URL: https://html.spec.whatwg.org/multipage/ (дата звернення 01.12.2022).
15. API Gateway. URL: https://www.redhat.com/en/topics/api/what-does-an-api-gateway-do. (дата звернення 01.12.2022).
16. Parsing of HTML Document Pranit C. Patil , Pramila M. Chawan , Prithviraj M. Chauhan . URL: https://www.researchgate.net/profile/Pramila-Chawan/publication/316510678_Parsing_of_HTML_Document/links/590191270f7e9bcf6546a9a4/Parsing-of-HTML-Document.pdf (дата звернення 01.12.2022).
17. The Universal Parser Compiler and Its Application to a Speech Translation System Masaru Tomita, Marion Kee, Hiroaki Saito, Teruko Mitamura and Hideto Tomabechi. URL: https://aclanthology.org/1988.tmi-1.9/ (дата звернення 01.12.2022).

Декларація

академічної доброчесності

здобувача ступеня вищої освіти ЗНУ

Я, Царьков Денис Володимирович, студент 2 курсу, форми навчання денної, Інженерного навчально-наукового інституту ім. Ю.М. Потебні, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz17bd-35@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Комп'ютерна система для автоматизованого збору і аналізу інформації з Веб-ресурсів**» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 30.11.2022 _____ Царьков Денис Володимирович
(студент)

Дата 30.11.2022 _____ Скрипник Ірина Анатоліївна
(науковий керівник)