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

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

ПРОГРАМНИЙ ЗАСІБ ПОШУКУ І АГРЕГАЦІЇ ПРОПОЗИЦІЙ З ОНЛАЙН ПЛАТФОРМ ОРЕНДИ НЕРУХОМОСТІ

Виконав студент 2 курсу, групи 2КІ-22м
спеціальності 123 — Комп'ютерна інженерія
Мартинов П. Г.
Керівник д.т.н., проф. каф. ОТ
Мартинюк Т. Б.
" " 2023 р.
Опонент к.т.н., доц. каф. ПЗ
Ракитянська Г. Б.
" " 2023 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф., Азаров О. Д.
"19" 12 2023 р.

ВНТУ 2023

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Рівень вищої освіти II-й (магістерський)
Галузь знань 12 — Інформаційні технології
Спеціальність 123 — «Комп'ютерна інженерія»
Освітня програма — «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
обчислювальної техніки
проф., д.т.н. О.Д. Азаров
«26»_вересня_2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

студенту Мартинову Павлу Геннадійовичу

1 Тема роботи «Програмний засіб пошуку і агрегації пропозицій з онлайн платформ оренди нерухомості», керівник роботи Мартинюк Тетяна Борисівна, к.т.н., доцент кафедри ОТ, затверджені наказом вищого навчального закладу від 18 вересня 2023р. № 247

2 Строк подання студентом роботи 8 грудня 2023 року

3 Вихідні дані до роботи: виконати розробку програмного модуля автоматичного збору даних з декількох джерел з подальшою агрегацією. Схему бази даних та лістинги програми представити в додатках до роботи.4 Зміст розрахунково-пояснювальної записки (перелік питань, які потріб-но розробити): вступ, аналіз сучасних методів розробки, варіативний вибір засо-бів для розробки програмного забезпечення, вдосконалення методу побудови архітектури підсистеми прогнозування попиту, розробка програмного модуля генерації прогнозів попиту та тарифів, програмна реалізація та тестування.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): схема роботи алгоритму програмного засобу, схема бази даних.

6 Консультанти розділів роботи представлені в таблиці1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Мартинюк Т.Б д.т.н., проф. каф. ОТ		
5	Небава М.І., к.е.н., професор кафедри ЕПВМ		

7 Дата видачі завдання 19.09.2023 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2—Календарний план

№	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Підпис
1	Аналіз завдання. Вступ	04.10.2023	
2	Аналіз літературних джерел	18.10.2023	
3	Розробка технічного завдання	01.11.2023	
4	Розробка структури програмного модуля автоматичного збору даних	15.11.2023	
5	Розробка програмного модуля автоматичного збору даних	25.11.2023	
6	Попередній захист	16.10.2023	
7	Оформлення пояснювальної записки і презентації	02.12.2023	
8	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	07.12.2023	

Студент _____ Мартинов П.Г.

Керівник роботи _____ Мартинюк Т.Б

АНОТАЦІЯ

УДК 004.42

Мартинов П.Г. Програмний засіб пошуку і агрегації пропозицій з онлайн платформ оренди нерухомості. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп'ютерна інженерія, Вінниця: ВНТУ, 2023. — 95 с. Укр. мова. Бібліогр.: 40 назв; рис.: 20 ; табл. 11.

Дану магістерську кваліфікаційну роботу присвячено створенню додатку для автоматичного збору та подальшої агрегації даних про ринок нерухомості для покращення користувацького досвіду.

В роботі був виконаний аналіз найбільш сучасних методів автоматичного збору даних, проведений огляд існуючих рішень та інструментів на ринку, розроблено програмний модуль для вирішення поставлених задач.

Перевірка працездатності системи протестована шляхом практичного тестування системи.

Ключові слова: веб-додаток, парсинг даних, оренда нерухомості.

ABSTRACT

Martinov P.G. Software Tool for Searching and Aggregating Offers from Online Real Estate Rental Platforms. Master's Qualification Thesis in the field of 123 — Computer Engineering, Vinnytsia: VNTU, 2023. — 95 p. Ukrainian language. Bibliography: 40 references; figures: 20; tables: 11.

This master's thesis is dedicated to creating an application for automatic data collection and subsequent aggregation of real estate market data to enhance user experience. The research includes an analysis of the most advanced methods of automated data collection, a review of existing solutions and tools in the market, and the development of a software module to address the defined objectives.

The functionality and performance of the system were verified through practical system testing.

Keywords: web application, data parsing, real estate rental.

ЗМІСТ

ВСТУП

У сучасному світі, де величезний обсяг інформації доступний онлайн, завдання автоматичного збору та аналізу даних з веб–ресурсів стає все більш важливим і актуальним завданням. За оцінками центру Statista [1] у 2020 році обсяг згенерованої інформації у світі досяг 47 зетабайт і це число збільшиться до 175 зетабайт до кінця 2025 року. Пошук житла, нерухомості та подібних послуг на Інтернет–платформах став доречним завданням для великої кількості користувачів. Однак, розміщення даних на численних веб–сайтах, їх фрагментація та невизначеність структури можуть створювати значні труднощі для користувачів, що шукають ідеальне житло або нерухомість, а також для аналітиків, що досліджують ринок нерухомості.

Ця магістерська робота спрямована на вирішення цих проблем шляхом застосування автоматичного збору даних з різних веб–ресурсів, аналізу цих даних та оптимізації процесу пошуку нерухомості шляхом застосування сучасних веб–технологій та фреймворків, таких як Java, JavaScript, Spring та React [2]. Основною задачею є розробка архітектури збору даних із відкритих джерел у мережі Інтернет, їх зберігання та подальшої обробки для створення системи, яка забезпечує користувачам можливість знайти оптимальну нерухомість на основі їхніх унікальних вимог та переваг.

Використання мов програмування, таких як Java та JavaScript, дозволяє розробникам створювати потужні та ефективні рішення для збору та обробки даних. Фреймворки, такі як Spring і React, надають інструменти для розробки високоякісних веб–додатків з інтуїтивними інтерфейсами та можливістю взаємодії з даними в реальному часі [3].

Ця робота також досліджує можливості поліпшення користувацького досвіду під час пошуку нерухомості шляхом інтеграції інтуїтивних інтерфейсів, використання рекомендаційних систем та впровадження штучного інтелекту для вивчення поведінки користувачів та підвищення ефективності пошуку.

У цьому контексті, дана магістерська робота представляє інноваційний підхід до автоматизованого пошуку нерухомості та аналізу даних, що може вплинути на спосіб, яким користувачі та фахівці в галузі нерухомості взаємодіють з інформацією у великих мережах Інтернету.

З кожним днем кількість доступних об'єктів для оренди зростає. За допомогою програмного засобу, який може агрегувати дані з різних платформ, користувачі можуть швидше та зручніше отримувати доступ до широкого вибору пропозицій. Споживачі хочуть мати можливість порівнювати ціни, рейтинги, відгуки та зручності різних пропозицій. Програмні засоби, які дозволяють це робити, стають надзвичайно корисними для забезпечення оптимального вибору. Завдання моніторингу та аналізу ринку оренди нерухомості України, створення нових програмних засобів, що вирішують дану проблему, а також способи роботи з аналітичним даними є **актуальною** задачею.

Метою роботи є вдосконалення методу автоматичного збору та аналізу даних про ринок оренди нерухомості шляхом застосування сучасних веб-технологій.

Задачі дослідження:

— провести аналіз існуючих методів та засобів автоматичного збору, зберігання та менеджменту даних із інтернет-ресурсів;

— виконати проєктування архітектури системи збору даних з відкритих інтернет-джерел на основі методів та технік предметно-орієнтованого проєктування;

— розробити систему збору та обробки даних, яка дозволяє користувачам приймати рішення щодо вибору оптимальної пропозиції за їх критеріями;

— виконати обґрунтування доцільності виконання нового наукового рішення, розрахувати економічні витрати для створення програмних засобів автоматичного збору даних із інтернет-ресурсів про ринок оренди нерухомості.

Об’єктом дослідження є процес отримання даних про ринок оренди нерухомості шляхом автоматичного моніторингу та аналізу інтернет-публікацій.

Предметом дослідження є методи та засоби моніторингу та аналізу даних із інтернет-публікацій про ринок оренди нерухомості.

Новизна полягає в удосконалені методу автоматичного збору та аналізу даних про ринок оренди нерухомості, який відрізняється від відомих підходів у поєднанні засобів автоматичного пошуку та агрегації даних із використанням селекторів зображень, таблиць, атрибутів елементів та спливаючих вікон, що дозволило покращити доступ до даних рядовому користувачу.

Практична значимість результатів — створено зручний засіб пошуку об’єктів оренди нерухомості, який може використовуватись звичайними користувачами для пошуку об’єкту оренди у модулі моніторингу та відображення актуальних даних, а також використовуватися аналітиками і компаніями у модулі аналізу для отримання інформації про стан ринку оренди нерухомості у певний момент часу.

Апробація результатів магістерської роботи — зроблено доповідь на Всеукраїнській науково-практичній інтернет конференція «Молодь в науці: дослідження, проблеми, перспективи (МН-2024).

За результати магістерської роботи **опубліковано** 1 наукову працю [4].

1 АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ АВТОМАТИЧНОГО ЗБОРУ ДАНИХ

Аналіз методів та засобів автоматичного збору даних є важливим кроком при розробці системи для збору та аналізу інформації з різних веб–ресурсів. Автоматичний збір даних, також відомий як веб-скрапінг або веб-кравлінг, є процесом автоматизованого збору інформації з різних джерел в Інтернеті. Цей процес може бути використаний для отримання даних з веб–сайтів, соціальних мереж, форумів, новинних сайтів та багатьох інших джерел.

1.1 Огляд методів автоматичного збору даних

Наразі відомі та широко використовуються наступні методи збору даних.

Веб-скрапінг — це процес видобування інформації з HTML-коду сторінок веб-сайтів. Цей метод дозволяє отримувати текстовий контент, зображення, таблиці та інші дані з веб-ресурсів. Для здійснення веб-скрапінгу потрібні скрипти або спеціальні інструменти. Початковим етапом є отримання HTML-коду веб-сторінки, з якої потрібно зібрати інформацію. Для цього використовуються бібліотеки або інструменти для HTTP-запитів, такі як Requests в Python. Отриманий HTML-код аналізується для виділення необхідних даних. Для цього використовуються парсери, наприклад, BeautifulSoup (Python), які допомагають ефективно розбирати структуру сторінки та виділяти потрібні елементи. Після аналізу HTML-коду можна видобути потрібну інформацію, таку як текст, URL-адреси, дати, зображення і т. д. Отримані дані можна зберегти у вигляді структурованого тексту або занести у базу даних. Їх також можна зберігати у різних форматах, таких як CSV, JSON або у базі даних, для подальшого використання або аналізу. Якщо потрібно оновлювати дані регулярно, можна налаштувати веб-скрапер на автоматичне виконання операцій оновлення з певною періодичністю. Веб-скрапінг є потужним інструментом для отримання інформації з Інтернету, але при цьому важливо дотримуватися етичних норм і відповідального використання цього методу. Перед використанням веб-скрапінгу для конкретної мети важливо

ретельно вивчити юридичні та етичні аспекти, а також можливі обмеження, які можуть бути встановлені власниками веб-сайтів. На сьогоднішній день одним з найбільш повних досліджень сучасних інструментів веб-скрапінгу є робота Уміля Перссона з Королівського технологічного університету у Стокгольмі [2]. У цій роботі детально розглянуті інструменти для збору даних за допомогою методів веб-скрапінгу, такі як Puppeteer, Nightmare.js, Scrapy, Selenium (у різних варіаціях використання), HtmlUnit, rvest. Також проведена оцінка їхньої ефективності з точки зору швидкодії, використання обчислювальних ресурсів, функціональності, надійності, простоти використання та наявності документації для цих продуктів. За висновком автора, будь-який з вищезгаданих інструментів є придатним та ефективним для практичного використання. Однак Puppeteer виділяється як найкращий інструмент, оскільки він має найширший функціонал і вважається одним з найшвидших та надійних з точки зору виконання робіт.

API (Application Programming Interface) — це набір правил й стандартів, що дозволяють різним компонентам і складовим програм (зазвичай розробленими різними розробниками) взаємодіяти між собою. Визначаючи доступні функції та операції, API регулює обмін даними між програмами. Багато веб-сервісів надають API для отримання даних у структурованому форматі. Це включає визначення доступних функцій, їх параметрів і результатів. API також встановлює протоколи та формати, за допомогою яких програми можуть обмінюватися інформацією, наприклад, через HTTP для взаємодії через Інтернет або інші протоколи, такі як WebSocket для реального часу. Багато API потребують авторизації для перевірки прав доступу користувача до конкретних ресурсів чи функцій. Це може включати використання ключів API, токенів або інших методів аутентифікації. Використання API дозволяє отримати доступ до інформації, яку надають різні сервіси, такі як Twitter, Facebook, Google Maps і багато інших. Використання API часто передбачає реєстрацію та отримання ключів доступу.

Веб-кравлінг представляє собою ітеративний процес, який включає автоматичний обхід веб-сайтів та видобування інформації з різних їх сторінок. Цей

метод дозволяє програмам (веб-кравлерам) автоматично переходити між сторінками веб-сайту з метою збору різноманітних даних, таких як текст, URL-адреси, метадані та інше. Веб-кравлери розпочинають свій процес з одного чи кількох вихідних URL-адрес, які є початковими точками для сканування. Вони переходять за посиланнями на веб-сторінках та аналізують їх зміст, витягуючи необхідні дані, такі як текст, зображення, посилання тощо. Отримана інформація індексується та зберігається у базі даних для подальшого використання, зокрема для пошукових процесів. Важливим аспектом є дотримання веб-кравлерами правил, встановлених власниками веб-сайтів, оскільки деякі сайти можуть накладати обмеження на швидкість кравлінгу або блокувати доступ з певних IP-адрес. Оновлення індексів є ключовим, оскільки вміст веб-сайтів постійно змінюється, і регулярне оновлення допомагає забезпечити актуальність інформації для користувачів.

Сканування документів: Якщо дані розміщені в електронних документах, таких як PDF, Word або Excel, можна використовувати програмні засоби для автоматичного збору даних з цих документів. Наприклад, бібліотека Python, які Tika або PyPDF2, може бути використана для видобування текстового контенту з PDF–файлів.

Соціальний медіа–моніторинг — це процес активного слідкування, аналізу та реакції на думки, обговорення та відгуки, які користувачі висловлюють у соціальних медіа та інших онлайн-спільнотах. Ця діяльність має значення для розуміння громадської думки, реакції на події або товари, а також для виявлення тенденцій та популярних тем в інтернеті. Соціальний медіа–моніторинг корисний як для підприємств (наприклад, для контролю репутації бренду), так і для досліджень у галузях маркетингу, політики, громадської думки та інших.

Моніторинг розпочинається з відслідковування різних соціальних медіа–платформ, таких як Twitter, Facebook, Instagram, LinkedIn, форуми, блоги та інші. Це виконується за допомогою спеціальних інструментів моніторингу соціальних медіа та застосунків API цих платформ. Збираються повідомлення, коментарі,

пости, теги та інші відгуки користувачів. Отримані дані піддаються аналізу, включаючи текстовий аналіз, визначення настрою (sentiment analysis), ідентифікацію ключових слів та тематичний аналіз, а також інші аналітичні методи.

Важливо визначити осіб або користувачів, що мають значний вплив у соціальних мережах, оскільки це допоможе у керуванні взаємодією з впливовими зацікавленими сторонами. Результати аналізу даних узагальнюються у вигляді звітів та рекомендацій, які можуть використовуватися для прийняття рішень у різних галузях, від маркетингу до політики. На основі результатів моніторингу можуть бути прийняті заходи щодо реагування на негативні відгуки або підтримки позитивних відгуків у соціальних медіа.

Машинне навчання і обробка природної мови (NLP) представляють собою два напрямки, які можна використовувати для автоматизованого аналізу та обробки текстової інформації з різних джерел. Моделі машинного навчання можуть автоматично розпізнавати та систематизувати текст, видобувати деталі чи визначати структуру даних. Поєднання цих двох областей може сприяти автоматизації збору текстової інформації, наприклад, шляхом використання моделей машинного навчання для витягування тексту з веб-сторінок. Наприклад, модель може визначати та виділяти заголовки, текст статей, дати тощо. Отримані дані можна піддати обробці за допомогою методів NLP, які, у свою чергу, допомагають у визначенні тематики тексту, розпізнаванні іменованих сутностей (наприклад, імена, місця, події) або аналізі сентименту (визначення емоційного забарвлення тексту). Моделі машинного навчання можуть також автоматично фільтрувати та сортувати зібрані дані, вибираючи лише ті, що відповідають певним критеріям чи ключовим словам. Такі моделі можуть створювати пошукові системи або агрегатори даних, що спрощує процес пошуку інформації на веб-сайтах. Крім того, машинне навчання може застосовуватися для автоматичного оновлення даних з регулярністю, виявляючи нову інформацію та оновлюючи дані відповідно. Використання цих методів дозволяє індивідуалізувати збір даних та надавати

користувачам рекомендації на основі їхніх інтересів та попередньої взаємодії з інформацією.

З розглянутих вище методів автоматичного збору даних для виконання поставлених задач в даній роботі найбільше підходять веб–скрапінг та використання API інтернет ресурсів, кожний з яких має свої особливості, переваги та недоліки використання .

Порівняння методів збору даних наведено у таблиці 1.1

Таблиця 1.1 — Методики збору даних із джерел у мережі Інтернет

Характеристика	Вебскрапінг	Використання відкритих API
Структура даних	Дані представлені у придатному для людини форматі, структура визначається у момент збору даних.	Дані порівняно добре структуровані, формат формально визначений розробником API.
Складність розробки	Порівняно низька, підходи мало відрізняються в залежності від джерела, існує багато інструментів які реалізують цей підхід	Порівняно висока через різноманітність варіантів реалізації API та потребує індивідуальної розробки для кожного джерела. В випадках використання API свідомо ускладняється зі сторони джерела.
Швидкість роботи	Від 0.5 до 5 секунд на кожен запит, сильно залежить від швидкодії сервера збору даних	До 0.1 секунди на запит

Якщо порівнювати розглянуті вище методології збору інформації з відкритих джерел мережі Інтернет, можемо отримати наступні висновки.

Вебскрапінг є ефективною технологією з великим різноманіттям інструментів, що конкурують між собою.

На відміну від нього, використання відкритих API є менш формалізованим та потребує індивідуального підходу до кожного джерела даних та потребує досвідчених спеціалістів. Також треба зазначити, що не всі API є відкритими та іноді потребують додаткової авторизації.

Проте основною перевагою другої методики є на порядок вища швидкодія запитів до API у порівнянні з повноцінним запитами веббраузеру та аналізом сторінки після цього. Це дозволяє використовувати другий підхід, якщо важлива швидкодія та є потреба у зборі надвеликих обсягів інформації.

1.2 Програмні засоби автоматичного збору даних

Існує багато програмних інструментів, які допомагають автоматизувати процес збору даних з різних джерел в Інтернеті. Ось декілька популярних програмних засобів для автоматичного збору даних:

Scrapy — це потужний фреймворк для веб–скрапінгу на Python, він надає ряд інструментів для створення та налаштування веб–скраперів і дозволяє витягувати дані з веб–сайтів у структурованому форматі.

Beautiful Soup — це бібліотека для Python, яка допомагає парсити HTML–код веб–сторінок та виділяти з нього інформацію. Вона добре поєднується з іншими бібліотеками для веб–скрапінгу.

Selenium — це інструмент для автоматизації браузера. Він дозволяє взаємодіяти з веб–сторінками, заповнювати форми, натискати кнопки та витягувати дані з веб–сайтів, які використовують JavaScript для завантаження контенту.

Octoparse — це веб–скрапінг інструмент, який має графічний інтерфейс для налаштування правил збору даних. Він призначений для користувачів з різними рівнями навичок та дозволяє витягувати дані з веб–сайтів без програмування.

Apify — це хмарна платформа для веб–скрапінгу та автоматизації завдань на веб–сайтах. Вона надає різноманітні інструменти для створення та запуску веб–скраперів та збору даних.

WebHarvy — це інструмент для видобування даних з веб–сторінок з графічним інтерфейсом. Він дозволяє створювати правила для видобування даних з веб–сайтів та зберігання їх у різних форматах.

Import.io — це інструмент для автоматичного збору даних з веб–сайтів з подальшим імпортом їх до архівних файлів або баз даних.

ParseHub — це інтуїтивно зрозумілий веб–скрапінг–інструмент, який дозволяє користувачам витягувати дані з веб–сайтів з графічним інтерфейсом.

Mozenda — це хмарний сервіс для автоматичного збору даних з веб–сторінок, який надає інструменти для створення та запуску скраперів.

Ці інструменти різняться за функціональністю, складністю та ціною. Вибір залежить від конкретних потреб і ресурсів, які ви готові витратити на автоматичний збір даних.

1.3 Методології та інструменти зберігання великих обсягів інформації

Зберігання даних для їх подальшого використання є значним підрозділом комп’ютерних наук з часу її відокремлення від математики та являє собою важливу теоретичну та практичну задачу. Методика та технологія зберігання може суттєво відрізнятись з залежності від зовнішніх умов: обсягу інформації, що зберігається, структурованості даних, вимогам к швидкості доступу та обробки інформації.

Розглянемо ці методи від найбільш простих до найсучасніших рішень з найвищими вимогами.

Простим методом зберігання структурованої та неструктурованої інформації є використання файлів, таких як CSV для табличних даних та JSON для об'єктів з різною структурою. Ці файли широко використовуються для обміну інформацією і не потребують складних інструментів для зберігання та обробки. Наприклад, запити HTML часто використовують JSON для представлення даних. Проте

основним обмеженням файлів є ускладнений доступ до них та обмежена можливість швидкої агрегації інформації.

На вищому рівні в ієрархії методів зберігання даних розташовані системи управління базами даних (СУБД). Вони є більш розширеним інструментом, який виконує не лише функції зберігання даних, а й надають різноманітні можливості для оптимізації роботи з даними. СУБД ведуть журнали роботи з даними, забезпечують оптимальний доступ до них, а також виконують резервне копіювання та відновлення даних у разі виникнення неполадок [3]. В залежності від моделі обробки даних можна виділити основні типи систем управління базами даних:

— реляційні забезпечують найшвидший доступ до добре структурованих даних та гарантують відповідність даних наперед заданій схемі за допомогою математичного апарату реляційної логіки;

— об’єктні в них робота ведеться з окремими об’єктами, кожен з яких має свої властивості та використовує методи взаємодії з іншими об’єктами;

— типу «ключ–значення», в яких об’єкти зводяться до деякого набору даних доступного по мітці;

— графові оптимізовані під дані, що мають графову структуру, наприклад соціальні або семантичні зв’язки.

СУБД можуть задовольняти усі потреби користувачів щодо зберігання та доступу до даних, навіть для найбільших організацій, є добре налаштованими та надійними інструментами. Проте СУБД не призначені для обробки неструктурованих даних та не забезпечують повний цикл роботи з даними: їх обробку як з технічної, так і з бізнес–логіки організації, для цього були розроблені 2 наступні концепції.

У великих компаніях та урядових установах традиційно використовується консервативний підхід до організації даних, де використовується сховище даних, що об'єднує систему управління базами даних і систему підтримки прийняття рішень. Цей підхід передбачає розділення даних на структуровану вхідну інформацію, яка зберігається у первісному вигляді, і слабоструктуровані аналітичні

висновки, що ґрунтуються на цих даних. Сховище даних є корпоративним стандартом у багатьох великих українських компаніях на сьогоднішній день. Однак основним недоліком цього методу є велика централізація даних та спрямованість на роботу з історичною інформацією, що не завжди відповідає потребам сучасної організації.

Озеро даних виступає як конкурент сховищу даних, надаючи більш гнучкий і децентралізований доступ до інформації всередині організації. Цей підхід об’єднує всі дані в одному просторі, що зазвичай має відкритий доступ до всіх його елементів, включаючи різнорідні дані з різних джерел: текстові файли, фото- та відеоматеріали, різні бази даних, що спеціалізуються на різних типах інформації. Оскільки ці дані є слабоструктурованими, ключовим елементом для таких систем є побудова додаткових метаданих, які описують та надають контекст інформації. Одним з часто використовуваних підходів до створення метаданих є MEDAL, теоретична база якого розробляється колективом авторів з Ліонського університету. Завдяки гнучкості цього підходу можна забезпечити швидкий пошук по усім джерелам та обробленій інформації за допомогою спеціалізованих інструментів, що потенційно може значно прискорити роботу аналітиків та прийняття рішень у великих організаціях.

Підсумовуючи методики роботи з даними потрібно ще раз відмітити їх різноманіття та адаптованість під різні сценарії використання: від індивідуальних досліджень до інституцій будь–якого розміру. Тому вибір інструменту зберігання даних повинен залежати від багатьох факторів, найважливішими з яких є масштаб організації, природа даних, які будуть зберігатися, та потреба в їх аналізі.

1.4 Огляд актуальних програмних засобів ринку нерухомості

Перелік популярних сайтів пошуку нерухомості в Україні:

— olx є одним з найпопулярніших сайтів для пошуку житла та різних товарів в Україні;

— центральна база нерухомості (cian.com.ua): Сайт, спеціалізований на нерухомості, де можна знайти оголошення про продаж та оренду житла;

— flatfy популярний сайт для пошуку житла в Україні, який пропонує широкий вибір пропозицій;

— lun.ua спеціалізований сайт для пошуку нерухомості, який також надає корисну інформацію про ринок нерухомості в Україні;

— dom.ria.ua сайт також спеціалізується на нерухомості і надає різноманітні оголошення про продаж та оренду житла.

Далі описані готові рішення автоматичного збору даних в сфері нерухомості.

DATACOL — є провідним постачальником рішень у сфері збору та обробки даних. Вони спеціалізуються на розробці програмних продуктів і інструментів, призначених для автоматизованого збору даних з різних джерел в Інтернеті. DATACOL відома своєю високою ефективністю та надійністю в сфері веб–скрапінгу та автоматизації процесів збору даних. їхні рішення знаходять застосування в різних галузях, включаючи маркетинг, дослідження ринку і аналіз даних.

CENTUM — це інноваційна компанія, що спеціалізується на розробці рішень для збору та аналізу даних у реальному часі. Вони пропонують комплексні системи збору даних, які дозволяють корпораціям та підприємствам отримувати оперативну інформацію для прийняття рішень. CENTUM відома своєю технологічною експертизою та інноваціями у сфері інтернету речей (IoT) та великих даних (Big Data).

A–PARSER — це компанія, що спеціалізується на розробці програмного забезпечення для веб–скрапінгу та автоматизації процесів збору даних з Інтернету. Вони пропонують інструменти, які допомагають підприємствам та дослідникам отримувати доступ до важливої інформації з веб–сайтів та інших джерел даних. A–PARSER відома своєю потужною та гнучкою платформою для роботи з текстовими даними.

AVADA–MEDIA — це компанія, що надає послуги у сфері розробки програмного забезпечення та рішень для обробки та аналізу даних. Вони спеціалізуються на створенні високоефективних та інноваційних програм, які допомагають клієнтам вирішувати складні завдання з обробки та збору даних. AVADA–MEDIA відома своєю здатністю надавати індивідуальні рішення, адаптовані до потреб кожного клієнта.

1.5 Методи візуалізації даних

Ефективний аналіз даних про нерухомість має вирішальне значення для того, щоб покупці могли зрозуміти ринок нерухомості та шукати відповідні об’єкти для проживання чи оренди. Розглянемо типові методи візуалізації даних та їх особливості.

Кругова діаграма є одним з найбільш поширених та простих графічних методів для візуалізації кількісних даних. Ці діаграми легко зрозуміти, оскільки вони наглядно відображають відношення часток до цілого. Вони ідеально підходять для швидкого відображення пропорційного розподілу даних. Наприклад, можуть бути використані для візуалізації обсягів викидів CO2 за різними секторами економіки [8]. На рисунку 1.1 наведено приклад кругової діаграми.

Рисунок 1.1 — Кругова діаграма

Стовпчикова діаграма чи гістограма – ще один популярний спосіб візуалізації даних для швидкого сприйняття інформації. Гістограми перетворюють набір

даних у прямокутні стовпці, висоти або довжини яких пропорційні величинам, які вони відображають. Такі діаграми значно полегшують порівняння порівнянних даних однієї категорії в рамках обмеженого періоду часу. Приклад стовпчикової діаграми показано на рисунку 1.2.

Рисунок 1.2 — Стовпчикова діаграма

Лінійні графіки, так само як і стовпчикові діаграми, служать для візуалізації даних у зручному і точному форматі, що сприяє легкому сприйняттю інформації та виявленню тенденцій або відношень між показниками (за умови використання кількох ліній). Ці графіки використовуються для відображення результатів даних щодо неперервної змінної — це найбільш звичайний випадок. Важливо використовувати різні кольори у цьому типі візуалізації для полегшення аналізу інформації. Приклад лінійного графіку наведено на рисунку 1.3.

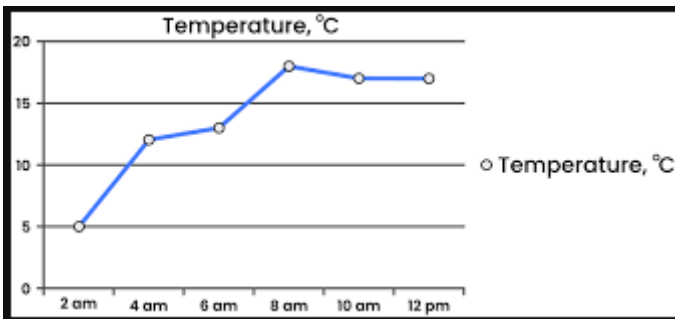


Рисунок 1.3 — Лінійна діаграма

Радіальна діаграма, також відома як павукоподібна або пелюсткова діаграма, корисна для порівняння кількох кількісних змінних. Ці діаграми спрощують порівняння декількох показників одночасно і дозволяють виявляти суттєві відмінності між ними, що можуть потребувати подальшого уваги. Зразок радіальної діаграми можна побачити на рисунку 1.4.

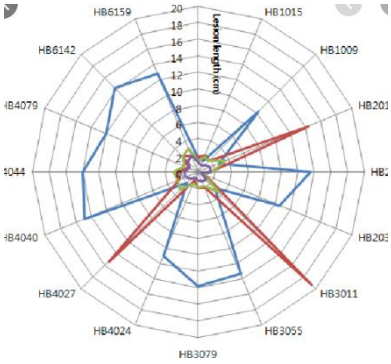


Рисунок 1.4 — Радіальна діаграма

Цей метод візуалізації показує частотність зустрічання слів у тексті, відображаючи розмір кожного слова відповідно до його частотності. Усі слова представлені у вигляді кластера або "хмари слів". Як альтернативу, слова можуть бути організовані у будь-якому форматі, наприклад, розміщені у вигляді горизонтальних ліній, колонок або в інших структурах. Приклад хмарки тегів представлено на рисунку 1.5.

Рисунок 1.5 — Хмарка тегів

Геовізуалізація є однією з найбільш широко використовуваних форм візуалізації даних про ринок нерухомості. Вона допомагає відобразити дані, які складно або непрактично представити у текстовому форматі. Існує кілька способів відображення інформації на карті [9].

За допомогою бульбашкових діаграм можна відобразити дані у вигляді коліс, розмір яких пропорційний значенню даних у певних географічних регіонах. Ці діаграми зручні для порівняння пропорційних значень, що характерні для конкретних географічних областей. Вони уникують проблеми, пов'язаної з розмірами територій, яка часто виникає при використанні картограм. Однак основним недоліком бульбашкових діаграм є ймовірність накладання бульбашок, якщо значення, які вони відображають, є занадто великими, що може призвести до виходу за межі регіонів, зображених на карті. Цю особливість слід враховувати при їх використанні [10]. На рисунку 1.6 показано приклад бульбашкової діаграми.

Рисунок 1.6 — Бульбашкова діаграма

Фонова картограма є зображенням, на якому поділ географічних областей або регіонів відображений різними кольорами, відтінками або малюнками в залежності від змінної інформації. Ця особливість дозволяє створювати візуалізацію даних, які властиві різним географічним областям, та відображати патерни або варіативність у межах певного місця [11]. На рисунку 1.7 показано приклад фонової картограми.

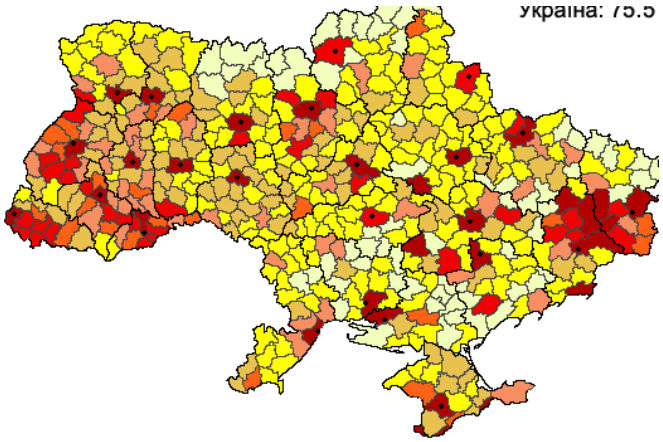


Рисунок 1.7 — Фонова картограма

Точкова картограма, також відома як точкова карта густини розподілу, представляє собою метод візуалізації географічних патернів або розподілу даних на карті. На такій карті показуються точки, що мають однаковий розмір та розміщені в межах географічного регіону, який зображений на карті. Цей тип візуалізації часто використовується для аналізу характеру розподілу об’єктів у певному регіоні та для виявлення патернів, які характеризуються групуванням точок на карті. Точкові карти зручно аналізувати і корисно використовувати для огляду даних. Проте, для відображення конкретних числових даних цей метод може бути менш ефективним [12]. На рисунку 1.8 наведено приклад точкової картограми.

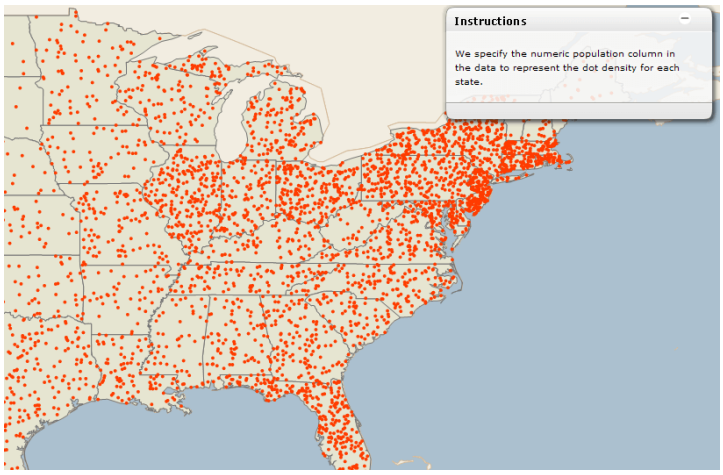


Рисунок 1.8 — Точкова картограма

З аналізу існуючих методів представлення даних про ринок нерухомості виокремлюються декілька ключових проблем. По-перше, існуючі системи обмежені лише властивостями об'єктів, тоді як важлива інформація, яка є вирішальною для користувачів, така як регіональні особливості, освітня і транспортна інфраструктура, часто не враховується. По-друге, ці системи надають лише географічну інформацію про розташування об'єктів на карті та не дозволяють порівнювати властивості за кількома аспектами. Нарешті, комерційні платформи обмежені в доступі до детальних знань про нерухомість та можливостей глибокого вивчення даних ринку нерухомості [13].

Важливим аспектом при візуалізації даних є також визначення найбільш критичних характеристик при перегляді таких даних. За результатами дослідження, при пошуку нерухомості, користувачі частіше всього звертають увагу на ціну об’єкту, його розташування та площу [14].

1.6 Висновки з аналізу сучасного стану методів та технологій збору, зберігання та менеджменту даними

У першому розділі було проведено аналіз наукової літератури яка стосується методів збору відкритих даних з мережі Інтернет, зберігання та обробки великих обсягів інформації на предмет використання технологій, які будуть використані для створення системи збору даних з відкритих джерел, їх агрегації та категоризації.

По–перше, було розглянуті архітектури та технології побудови систем збору інформації з відкритих джерел у мережі Інтернет, основними з яких є вебскрапінг та використання прикладних програмних інтерфейсів (API). У порівнянні зі застосуванням API вебскрапінг є більш гнучким та розповсюдженим інструментом, який дозволяє використовувати універсальний інтерфейс веббраузеру для збору інформації та є простішим у використанні. Проте його використання обмежено з точки зору використання обчислювальних ресурсів та швидкості збору інформації. На відміну від нього, використання програмних інтерфейсів дозволяє отримувати більш структуровану інформацію за менший інтервал часу. Останні дві

характеристики є критичними з точку зору архітектури майбутньої системи, тому на мою думку для цілей дипломного проєктування використання API є більш пріоритетним.

По–друге, було проаналізовано існуючий ринок готових рішень та їх оцінка в результаті чого стає ясно, що готових рішень, доступних рядовому користувачу, на даний момент не має. Через що розробка програмного засобу в даній магістерській роботі є актуальною задачею.

По–третє, було досліджено різноманітні варіанти зберігання та обробки великих обсягів інформації: зберігання у файлах, використання систем управління базами даних з різними моделями та методологій роботи з даними data warehouse, data lake. Перший варіант може використовуватися в роботі системи, проте лише для тимчасового зберігання даних. На етапі проєктування використання СУБД є найбільш раціональною опцією роботи з даними оскільки є компромісом між трудомісткістю та зручністю зберігання та обробки інформації. В подальшому, за умови розвитку проєкту після реалізації цілей виконання магістерської роботи, найбільш раціональним варіантом організації системи роботи з даними є модель data lake, яка є більш гнучкою та якнайкраще відповідає предметній галузі системи.

2 МЕТОДИ ТА ЗАСОБИ СИСТЕМИ МОНІТОРИНГУ ТА АНАЛІЗУ ДАНИХ ПРО РИНОК НЕРУХОМОСТІ

В даному розділі описуються методи та засоби створення системи моніторингу та аналізу даних.

2.1 Компонент збору та первісної обробки даних

Для збору та первісної обробки даних використовується збирач даних, обробник даних, модуль зберігання даних.

В роботі збирача даних, який є парсером, виконується процес надсилання HTTP-запитів до зовнішніх джерел інформації, таких як інтернет-магазини, для отримання даних. Для отримання цих даних парсер може використовувати зовнішні API інтернет-магазинів або системи аналізу вмісту веб-сторінок. Після успішного збору необхідної інформації, отриманий набір даних передається до модуля зберігання у системі для подальшого зберігання та обробки.

Модуль зберігання даних — це компонент системи, що відповідає за зберігання всієї необхідної для системи інформації. Цей модуль працює з різноманітними форматами даних, включаючи як чітко структуровані формати, наприклад, JSON та табличні дані, так і менш структуровані дані, такі як фотографії товарів.

У своїй роботі модуль зберігання даних використовує інтерфейси, що надаються збирачем даних і обробником інформації про товари. Він також забезпечує інтерфейс для внутрішніх модулів, наприклад, обробника даних, а також для зовнішніх компонентів системи.

Цей модуль забезпечує зручний доступ до даних, які зберігаються в ньому, іншим компонентам системи через визначені інтерфейси, спрощуючи взаємодію та обробку даних в системі.

Обробник даних та класифікатор товарів виконують важливу роль у системі. Основна мета цих модулів полягає у перетворенні різноманітних

слабоструктурованих даних, отриманих із різних джерел, в однорідний формат, що дозволить подальше зручне використання цих даних.

Окрім цього, обробник даних та класифікатор товарів займаються ідентифікацією однакових товарів у різних магазинах з метою спрощення порівняння їх цінових пропозицій. Для виконання цих завдань вони використовують необроблені дані, які зберігаються в data lake (озері даних), та після їх аналізу створюють структуровану інформацію про товари. Такий підхід дозволяє системі зручно та ефективно опрацьовувати дані, роблячи їх доступними для подальшого використання у єдиному форматі.

Модуль геокодінгу потрібен для обробки інформації про адреси фізичних магазинів та оцінки варіантів доставки або самовивозу з них. Для своєї роботи він використовує публічні геокодінгові сервіси та записує отриману інформацію в озеро даних.

2.2 Порівняння методів зберігання даних: data lake та data warehouse

Під час проєктування системи для автоматичного збору даних з веб-сайтів було визначено, що одним з ключових компонентів є модуль зберігання та обробки інформації. Тому важливо ще на етапі концептуалізації програмного продукту визначити, як саме дані будуть зберігатися у системі, як вони будуть використовуватися та кому буде надано доступ до них.

Давайте детальніше порівняємо два основні підходи у системах зберігання даних: вже перевірений й традиційний data warehouse й новішу концепцію data lake. Розглянемо їхні відмінності й переваги, щоб визначити найбільш практичний варіант підходу для використання в системах збору даних із відкритих джерел у мережі Інтернет.

Сховища даних [15] були рекомендовані як ефективний та стійкий метод зберігання та управління інформацією в розмірах великих організацій, оскільки вони використовують швидкодію роботи реляційних баз даних та стандартизовані процедури обробки об'ємних даних.

Спочатку дані з різних джерел збираються у тимчасові бази даних. Далі вони пройшовши процес ETL (витягнення, трансформація, завантаження) обробляються та приводяться до відповідності заздалегідь визначеним схемам зберігання даних у основному сховищі. Для користування даними кінцевими користувачами, такими як аналітики або інші співробітники організації, створюються вітрини даних (data mart), які містять певну частину інформації та спеціалізуються на вирішенні конкретних завдань. На заключному етапі аналітики створюють необхідні звіти з даних, що містяться у вітрині.

На відміну від сховища даних, озеро даних [16], має менш складну структуру, відображену на рисунку. Дані також поступають з різних джерел, але відсутність необхідності у їх обробці перед записом у озеро даних є ключовою особливістю. Таким чином, уся вхідна інформація зберігається у вихідному форматі. Після цього вона може використовуватися для створення проміжних даних або для кінцевого використання у різних сферах діяльності будь-якими співробітниками організації, такими як аналітики, дослідники даних, менеджери та працівники на всіх рівнях.

Data warehouse найбільш відповідає потребам компаній, де дані є передбачуваними та використовуються переважно бізнес-користувачами, такими як банки, роздрібна торгівля та логістика. У той час як data lake є більш вигідним варіантом для організацій, де набори даних різноманітні та менш передбачувані, а також коли майже всі співробітники використовують дані компанії та мають відповідні навички для їх роботи.

2.3 Збереження даних за допомогою веб–скрапера

Веб-скрапінг — це метод збору даних, який просканує визначену веб-сторінку або набір сторінок та отримує потрібну інформацію за певними попередньо встановленими правилами. Зазвичай ці правила визначаються за допомогою XPath або CSS селекторів. Для створення веб-скрапера потрібні значні обчислювальні ресурси від сервера. Ще однією проблемою розробки веб-скрапера є необхідність обходу блокування, яке встановлюється власниками веб-сайтів або

адміністраторами. Для цього зазвичай використовують проксі-сервери, які дозволяють трафіку, що надходить від роботів, виглядати як звичайна діяльність користувача або від багатьох звичайних користувачів.

2.3.1 Селектори об’єктів

Існує кілька типів об'єктів на веб-сторінці, які можна вибрати для більш точного вилучення необхідних даних. Розглянемо основні елементи, які використовуються для скрапінгу.

Селектор тексту використовується для виділення тексту з визначеного елемента та його дочірніх елементів. Цей селектор вилучає текст, виключаючи HTML, та ігнорує тексти в тегах <script> і <style>. Теги нового рядка
 замінюються символами нового рядка. Для додаткового форматування тексту можна використовувати регулярні вирази для вилучення певних частин тексту.

При роботі з селектором тексту потрібно вказати CSS селектор елемента, з якого будуть вилучені дані, а також регулярний вираз для виділення певної підстроки. Наведена нижче таблиця містить приклади використання регулярних виразів для форматування даних під час скрапінгу [18].

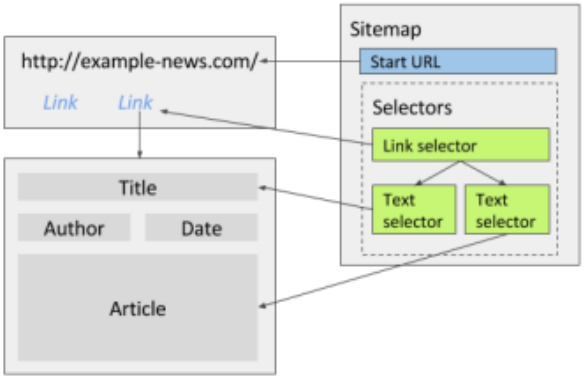


Рисунок 2.1 — Селектор тексту

Селектор посилань використовується для вилучення посилань та навігації по веб-сайту. При використанні селектора посилань без дочірніх селекторів, він

отримує посилання та атрибут href цих посилань. Якщо до селектора посилань додати дочірні селектори, вони використовуватимуться на сторінці, на яку вказує це посилання. Також, якщо відзначити відповідний чекбокс, можна здійснювати перехід на кілька посилань одночасно.

Селектор посилань працює лише з тегами з атрибутом href [19]. Якщо селектор посилань не працює, потрібно перевірити чи виконуються наступні умови. Можливо, змінюється посилання в рядку URL після натискання елемента. Якщо посилання не зміниться, можливо, сайт використовує ajax для завантаження даних. Замість використання селектора посилань слід використовувати селектор натискання елемента. Якщо сайт відкриває спливаюче вікно, слід використовувати селектор спливаючих вікон з посиланнями. Можливо, сайт використовує JavaScript window.location для зміни URL–адреси. Веб–скрапери наразі не можуть впоратися з такою навігацією.

Для роботи з селектором посилань необхідно обрати CSS селектор бажаного посилання, а також відмітити чекбокс, що вказує чи необхідно здійснювати перехід по деякій кількості посилань, чи лише за одним посиланням.

Наприклад, якщо ви працюєте з сайтом електронної комерції, який має багаторівневу структуру навігації з категоріями і підкатегоріями, ви можете створити два різних селектори для посилань. Один селектор буде вибирати посилання на категорії, а інший на підкатегорії, доступні в межах цих категорій. Селектор посилань для підкатегорій має бути підпорядкований селектору посилань категорій. Крім цього, селектори для вилучення даних зі сторінок підкатегорій мають бути створені як дочірні до селектора підкатегорій [20].

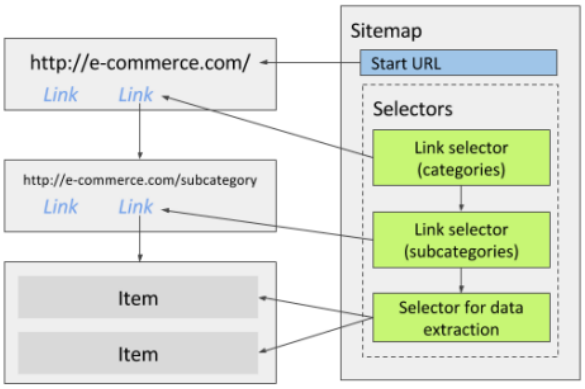


Рисунок 2.2 — Селектор посилань

Селектор посилань Sitemap.xml є корисним інструментом для переходу по всьому веб-сайту без необхідності встановлювати окремі селектори для розбиття сторінок або іншої навігації. Цей селектор витягує URL-адреси з файлів sitemap.xml, які публікують веб-сайти для полегшення роботи сканерів пошукових систем. Зазвичай, ці файли містять URL-адреси всіх важливих сторінок на сайті, допомагаючи скраперам та іншим інструментам сканувати веб-сайт більш ефективно [21].

Веб-скрапер підтримує стандартний формат sitemap.xml і може обробляти стиснуті файли sitemap.xml.gz. Якщо основний файл sitemap.xml містить посилання на інші файли sitemap.xml, селектор буде працювати рекурсивно, щоб знайти всі URL-адреси в цих підфайлах sitemap.xml. Проте веб-скрапер має обмеження на розмір завантаження, тому використання кількох URL-адрес sitemap.xml може призвести до перевищення ліміту. Щоб уникнути цього, рекомендується розділити карту сайту на декілька карт сайту, кожна з яких містить лише один файл sitemap.xml [12].

Для використання вказаного селектору, необхідно вказати список URL-адрес файлів sitemap.xml, які потрібно обробити. Можна додати кілька URL-адрес. Також є можливість використання регулярних виразів для відповідності підрядку з URL-адрес. Якщо налаштовано, веб-скрапер видалятиме лише URL-адреси з

sitemap.xml, які відповідають вказаному регулярному виразу. Також можна задати мінімальний пріоритет URL-адрес, які підлягають вилученню.

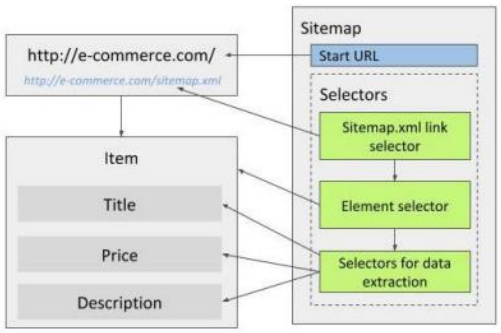


Рисунок 2.3 — Селектор мапи сайту

Селектор спливаючих вікон діє аналогічно до селектора посилань і використовується для вилучення URL-адрес і навігації по сайту. Основна відмінність полягає в тому, що селектор спливаючих вікон використовується у випадках, коли натискання на посилання призводить до відкриття нового вікна (спливаючого) замість завантаження URL-адреси в поточній вкладці або відкривання її в новій вкладці. Цей селектор перехоплює подію створення спливаючого вікна та виділяє URL-адресу. Якщо сайт використовує візуальні спливаючі вікна, але це не справжні вікна, рекомендується спробувати використати селектор натискання на елемент.

Вибираючи ці елементи посилання, можна навести курсор миші на елемент й натиснути літеру «S», щоб вибрати його та запобігти відкриванню спливаючого вікна на зображенні.

Селектор зображень використовується для вилучення атрибуту src (URL) зображень. При виборі CSS-селектора для знаходження зображень на сайті, всі зображення переміщуються вгору. Веб-скрапер дозволяє виконувати скрипт, який шукає URL-адреси зображень у файлі csv, зібрані за допомогою селектора зображень, та завантажує їх. Він спробує завантажити зображення з усіх URL-адрес, які зберігаються у стовпцях, назви яких закінчуються на "-src". [23].

Селектор таблиць призначений для виділення даних з таблиць. Він має налаштовувані CSS-селектори: один для вибору самої таблиці та інші для рядка заголовка та рядків даних. Після вибору селектора для таблиці, програма спробує визначити селектори для рядка заголовка та даних. Для перевірки правильності роботи селекторів, ви можете скористатися функцією "Попередній перегляд елемента". Селектор рядка заголовка використовується для ідентифікації стовпців у таблиці, особливо коли дані збираються з різних сторінок. Також є можливість перейменувати стовпці таблиці. На рисунку 2.4 показано, які елементи слід обирати під час вилучення даних з таблиці.

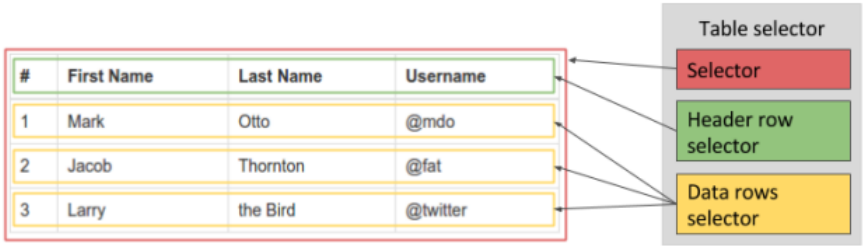


Рисунок 2.4 — Селектор таблиць

Селектор атрибутів елемента може витягувати значення атрибутів елемента HTML. Наприклад, можна використовувати цей селектор, щоб отримати атрибут title з цього посилання: <a href="#" title="my title">link<a>.

Селектор HTML може витягувати HTML і текст у вибраному елементі. Буде витягнутий лише внутрішній HTML елемента.

Згрупований селектор може групувати текстові дані з кількох елементів в один запис. Видобуті дані зберігатимуться як JSON.

Селектор елементів призначений для виділення елементів, що містять у собі кілька під-елементів даних на веб-сторінці. Наприклад, цей селектор можна використовувати для виділення списку товарів на сайті електронної комерції. Кожен елемент, що вибирається селектором, стає батьківським для своїх дочірніх

селекторів. Дочірні селектори здійснюють виділення даних тільки з елементу, який був вибраний селектором елементів. [24].

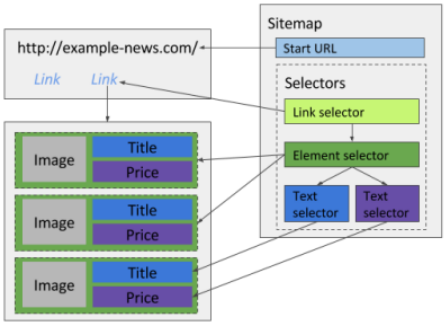


Рисунок 2.5 — Селектор елементів

Селектор елементів працює подібно до селектора елементів, але додатково прокручує сторінку вниз кілька разів, щоб знайти нові елементи, які з'являються при прокручуванні. Можна встановити затримку між прокручуванням сторінки та пошуком елементів за допомогою атрибуту затримки. Процес прокручування припиняється, коли нові елементи більше не з'являються. Проте, якщо сторінка може прокручуватися безмежно, цей селектор може застрягти в безкінечному циклі. [25].

Селектор натискання елемента працює на подібних принципах, що і селектор елемента. Він призначений для вибору елементів, які можуть бути передані як батьківські елементи своїм дочірнім селекторам. Однак він має здатність взаємодіяти з веб-сторінкою, може спрацьовувати натисканням кнопок для завантаження нових елементів. Наприклад, це може виявитися корисним, коли сторінка використовує JavaScript або AJAX для переходу на інші сторінки або завантаження нового вмісту.

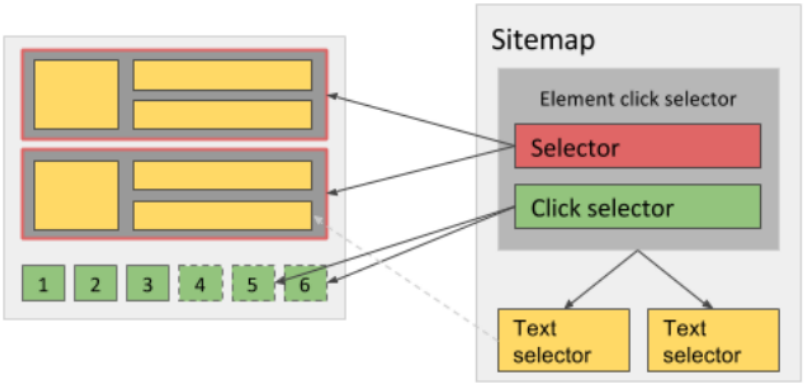


Рисунок 2.6 — Селектор натискання

Селектор розбиття на сторінки використовується для переміщення між різними сторінками або для завантаження всіх доступних елементів, використовуючи кнопку "Завантажити більше". Цей селектор завжди працює у вигляді рекурсії, що дозволяє відкрити всі сторінки розбиття на сторінки. Для отримання даних зі сторінок розбиття на сторінки, селектори, які вилучають дані, повинні бути встановлені як дочірні елементи для селектора розбиття на сторінки [22].

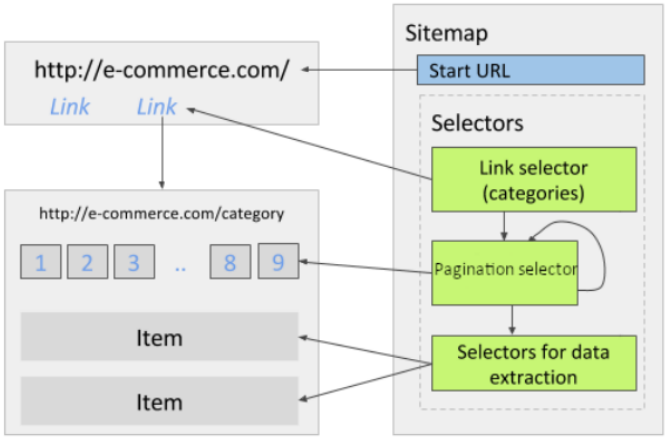


Рисунок 2.7 — Селектор розбиття на сторінки

Як висновок можна відзначити, що сервіс WebScraper надає широкий вибір інструментів для автоматичного збору будь–якої інформації з вебсайтів і є не складним в опануванні.

2.3.2 Додаткові функції

Проксі використовується для уникнення блокування скрапера цільовим веб-сайтом або для отримання доступу до сайту з різних місць без обмежень. За замовчуванням проксі використовує IP-адреси, які знаходяться в США. Для використання інших місцезнаходжень IP-адрес можна звернутися до служби підтримки. Також можна інтегрувати сторонніх постачальників проксі-серверів та в разі потреби налаштовувати їх місце розташування [24].

Коли проксі–сервер увімкнено, скрапер буде змінювати IP–адреси кожні 5 хвилин. Якщо сторінку не вдається завантажити з увімкненим проксі–сервером, скрапер змінює IP–адресу та повторює копію сторінки.

Кількість паралельних завдань визначає, скільки процесів скрапінгу можна виконати одночасно. Якщо всі доступні процеси використані і нове завдання скрапінгу починається, воно буде заплановано та додано до черги, поки не звільниться якийсь процес. Завдання скрапінгу можна вручну призупинити або відновити, щоб звільнити паралельні процеси або змінити їх порядок виконання.

Існує два типи драйверів для виконання завдань скрапінгу:

— повний драйвер – завантажує сторінку так само, як і розширення для браузера Web Scraper, усі активи завантажуються, і JavaScript виконується до початку вилучення даних;

— швидкий драйвер не виконує JavaScript на сторінці, дані витягуються з необробленого HTML, сторінка завантажується швидше [24].

Веб–скрапер має функціональні можливості для контролю якості просканованих даних. Кожна карта сайту має окрему конфігурацію контролю якості даних. Якість даних можна визначити за кількома критеріями:

— мінімальна кількість записів;

— максимальний відсоток невдалих сторінок;

— максимальний відсоток пустих сторінок;

— мінімальний відсоток полів, які необхідно заповнити.

Планувальник — це функція, яка дозволяє автоматизувати скрапінг на певний час. Якщо поточне скрапінгове завдання триває або виконується довше, ніж заплановано, нове заплановане завдання розпочнеться лише після завершення попереднього. Щоденний планувальник дозволяє вибрати дні тижня та час, коли потрібно розпочати скрапінг. Інтервал можна встановити в годинах або хвилинах, і якщо використовується інтервал по годинах, планувальник розпочне завдання за круглими годинами.

Парсер — це функція, що автоматизує обробку даних після їхнього отримання. Зазвичай вона використовується за допомогою певних сценаріїв, які записуються користувачем або програмним забезпеченням для електронних таблиць. Ця модульна структура дозволяє створювати та налаштовувати кілька синтаксичних аналізаторів для кожного стовпця, щоб ефективно обробляти дані. Це означає можливість створювати широкий спектр методів обробки, починаючи від простих і закінчуючи більш складними. [22].

2.3.3 Схема бази даних

Перед тим як приступити до розробки структури кубів для аналізу ринку оренди нерухомості, потрібно спершу створити схему бази даних, у якій накопичуються мета-дані, та на основі якої створюються куби. Для збереження даних про об’єкти оренди нерухомості створимо чотири таблиці.

У першій таблиці, структура якої наведена у таблиці 2.1, зберігаються дані про міста, у яких знаходяться об’єкти нерухомості.

Таблиця 2.1 – Опис таблиці cities

Назва поля	Опис	Тип
id	Ідентифікатор об’єкта	integer

Продовження таблиці 2.1

name	Назва містаа	text
lat	Довгота для зображення об’єкта на мапі	double
lng	Широта для зображення об’єкта на мапі	double
district	Область, у якій знаходиться місто	text
population	Населення міста	integer

У другій таблиці, структура якої наведена у таблиці 2.2, зберігаються дані про будівлі, у яких знаходяться квартири.

Таблиця 2.2 — Опис таблиці buildings

Назва поля	Опис	Тип
id	Ідентифікатор об’єкта	integer
buildType	Тип будинку	text
buildYear	Рік побудови будинку	timestamp
city_id	Ідентифікатор міста, у якому знаходиться будинок	integer

У третій таблиці, структура якої наведена у таблиці 2.3, зберігаються дані про квартири, які є об'єктами для оренди.

Таблиця 2.3 – Опис таблиці apartment

Назва поля	Опис	Тип
id	Ідентифікатор об’єкта	integer
rooms	Кількість кімнат	integer
Total_area	Загальна площа	integer
Living_area	Житлова площа	integer
rennovation	Тип ремонту, що зроблено у квартирі	text
Building_id	Ідентифікатор будівлі	integer

У четвертій таблиці, структура якої наведена у таблиці 2.5, зберігаються дані про усі операції, що проводились над певним об'єктом оренди.

Таблиця 2.4– Опис таблиці offers

Назва поля	Опис	Тип
id	Ідентифікатор об’єкта	integer
Created_at	Дата створення оголошення	integer
Active_for	Тривалість активності оголошення	integer
price	Ціна операції	integer
Property_id	Ідентифікатор об'єкту оренди, з яким була проведена операція	integer

2.4 Аналітична обробка у реальному часі

Аналітична обробка даних у реальному часі (On–Line Analytical Processing) — це такий процес аналізу даних, що відбувається майже миттєво або ж із мінімальною затримкою після отримання бажаної вхідної інформації. Цей факт означає, що отримані дані аналізуються, обробляються й інтерпретуються практично у масштабі реального часу, без значної затримки.

Аналітична обробка даних у реальному часі — це такий метод аналізу даних, що спрямований на оперативний доступ, оброблення та аналіз великих обсягів інформації із метою виявлення тенденцій, зв'язків і шаблонів. Шляхом використання OLAP користувачі можуть взаємодіяти з даними через різноманітні виміри і перспективи, що сприяє більш глибокому розумінню даних.

Основні характеристики технології OLAP:

— мультидименсійність, в якій дані представлені у вигляді "кубів" (тривимірні структури даних) з різними вимірами, які дозволяють аналізувати дані з різних перспектив;

— швидкий доступ до даних, OLAP базується на попередньо агрегованих та оптимізованих даних, що дозволяє швидкий доступ до інформації для аналізу;

— підтримка аналітичних операцій в системі OLAP дозволяє проводити аналіз даних, включаючи сумування, розрахунки середніх значень, відсотків тощо.

— динамічність та інтерактивність, завдяки чому користувачі можуть змінювати точки зору та параметри аналізу даних в реальному часі;

— підтримка бізнес–процесів: OLAP допомагає приймати управлінські рішення, аналізувати результати та планувати стратегії на основі отриманих даних.

Усі OLAP-продукти повинні дотримуватись наступним характеристикам, в яких полягає ідеал технології:

"Fast" або швидкий — це характеристика системи, яка має здатність надавати більшість відповідей користувачам протягом приблизно п'яти секунд. Навіть якщо система передбачає, що процес може зайняти більше часу, користувачі можуть втратити інтерес і зосередження, що може негативно вплинути на якість аналізу.

Досягнення такої швидкості стає складнішим із великим обсягом даних, особливо коли потрібні складні обчислення у режимі реального часу. Постачальники послуг використовують різноманітні методи для досягнення цієї мети, такі як спеціалізовані форми зберігання даних, попередні обчислення або підвищені вимоги до обладнання. Проте, на сьогоднішній день немає повної оптимізації у всіх 29 аспектах. Здається дивним, що хоча отримання звіту за хвилину, що раніше потребувало днів, може здаватися швидким, користувачі швидко можуть втрачати інтерес під час очікування. Це може призвести до менш успішного проекту, навіть якщо звіт буде менш деталізованим або швидким.

Analysis (аналіз) — це здатність системи виконувати різноманітний логічний та статистичний аналіз, що є характерним для певної програми, і зберігати цей аналіз у доступній формі для кінцевого користувача. Користувач має мати можливість створювати нові спеціальні обчислення без потреби у програмуванні. Усі функціональні можливості аналізу повинні бути доступні та інтуїтивно зрозумілі для кінцевих користувачів. Опції аналізу можуть включати різні процедури, такі як аналіз часових рядів, розподіл витрат, валютні конвертації, пошук цілей і т. п. Ці можливості різняться в широкому спектрі продуктів залежно від їхньої спрямованості.

Shared (розділяється) — це характеристика системи, що дозволяє виконувати всі вимоги забезпечення безпеки даних та надає розподілений та одночасний доступ до даних для різних рівнів користувачів. Система має здатність обробляти численні зміни даних вчасно та в безпечному режимі. Це є однією з головних проблем багатьох продуктів OLAP, оскільки вони часто припускають, що для всіх додатків OLAP необхідне лише читання та пропонують спрощені засоби захисту.

Багатовимірний (Multidimensional) - ключовий елемент системи OLAP. Ця вимога полягає у забезпеченні багатовимірного концептуального представлення даних, включаючи повну підтримку ієрархій та множинних ієрархій. Це розуміння даних в багатовимірних контекстах визначає найбільш логічний метод аналізу бізнес-інформації. Кількість вимірів, які система може обробляти, не має строгих

обмежень, оскільки вона залежить від конкретної програми та вимог ринку, на який вона спрямована. Найважливіше — це забезпечення дійсно багатовимірного концептуального розуміння даних, без прив'язки до певної технології баз даних. Ця характеристика є ключовою складовою для систем OLAP.

Інформація повинна бути доступною там, де її потрібно, незалежно від обсягу або місця зберігання. Проте багато в чому залежить від конкретної програми. Спроможність різних продуктів оцінюється за їхньою здатністю обробляти великий обсяг вхідних даних, а не обов'язково за їхньою потужністю зберігання. Різноманітні OLAP-системи можуть працювати з величезними обсягами даних, в тисячі разів перевищуючи найменші системи. Враховуючи це, важливо враховувати різні фактори, такі як резервне копіювання даних, необхідний обсяг оперативної пам'яті, використання місця на диску, продуктивність, можливості інтеграції з іншими сховищами даних та інші [25].

В основі OLAP–технологій лежить подання інформації у вигляді OLAP–кубів.

OLAP–куби містять бізнес–показники, що використовуються для аналізу та прийняття управлінських рішень, наприклад: прибуток, рентабельність продукції, сукупні кошти (активи), власні кошти, позикові кошти тощо.

Бізнес–показники зберігаються у кубах у вигляді простих таблиць, як і звичайних системах обліку чи бухгалтерських програмах, а розрізах, що становлять основні бізнес–категорії діяльності організації: товари, магазини, клієнти, час продажу тощо.

OLAP-куби дозволяють проводити оперативний аналіз та генерувати звіти з вибраною глибиною деталізації та в різних аспектах. Ці звіти зазвичай формуються для менеджерів, аналітиків, фінансистів та керівників різних підрозділів. Особливістю цих звітів є можливість взаємодії, що дозволяє швидко отримати відповіді на потрібні питання та приймати рішення, ґрунтуючись на результаті проведеного аналізу [26].

2.5 Моделювання OLAP кубів

У цьому розділі більш докладно розглядається створення OLAP-куба, його складові та пояснюються найкращі практики для створення високоякісної системи. При розробці OLAP-куба варто враховувати кілька ключових аспектів, що сприяють функціональності та продуктивності куба. Перше, що слід відзначити, це використання надійних джерел даних і постачальників даних. В добре спроектованому OLAP-кубі дизайн вимірів вважається однією з ключових складових, і важливо переконатися, що атрибути, відносини та ієрархії відповідають потребам кінцевого користувача і адекватно відображають основні дані [27].

2.5.1 Таблиця фактів

Таблиця фактів у системі OLAP є одним із важливих компонентів для зберігання та аналізу даних. Вона містить конкретну інформацію, числові дані, такі як обсяги продажів або виробництва, призначені для аналізу. Ця таблиця взаємодіє з іншими таблицями у базі даних, має зв'язки з таблицями розмірів та іншою важливою інформацією. У таблиці фактів містяться виміри та показники, які можуть бути проаналізовані з різних точок зору, а також вона оптимізована для швидкого доступу під час виконання аналітичних операцій та створення звітів. Це є основною складовою для аналізу даних та прийняття бізнес-рішень.

Групи даних або вимірів, які розміщуються у таблиці фактів, відомі як групи мір. Якщо у кубі присутні кілька таблиць фактів, але вони зазвичай не використовуються одночасно, може бути корисно створити кілька окремих кубів, кожен з яких має свою власну тематику. Це випливає з того, що кілька груп вимірів може негативно впливати на продуктивність запиту, навіть якщо запитується лише одна група вимірів. Якщо рідко використовуваний запит вимагає використання більше, ніж одного куба, можна створити спеціальний куб, який об'єднує пов'язані групи показників [28].

Якщо кілька груп показників мають спільний вимір та однаковий рівень деталізації, можливо, їх варто об'єднати у єдину групу мір, яка включатиме кілька розділів. Це підвищує ефективність та спрощує користування для користувачів. Однак у випадку чіткого розрахунку цей підхід може варто виключити. Щоб покращити продуктивність, такі групи мають бути розміщені в окремій групі вимірів. Наприклад, якщо компанія має різних клієнтів, кожен з яких купує декілька товарів, для розрахунку кількості клієнтів можна використовувати різні методи підрахунку. Для зменшення обсягу даних у кубі важливо враховувати, що числові типи даних відрізняються за обсягом, тому доцільно використовувати найменший числовий тип даних для кожного показника [28].

2.5.2 Осі для організації даних у кубах

Осі в технології OLAP (Online Analytical Processing) використовуються для організації даних в многовимірних кубах. Осі визначають різні виміри та атрибути, за допомогою яких дані можуть бути аналізовані з різних перспектив.

В основі многовимірного куба лежить концепція вимірів та їх організації в просторі. Кожна ось у кубі відповідає певному виміру, такому як час, місце, продукт або категорія клієнтів. Ці осі утворюють різні напрямки аналізу даних, а додавання нових вимірів до куба може розширити аналітичні можливості.

Осі в многовимірному кубі дозволяють користувачеві вибирати конкретні атрибути для аналізу, дозволяючи розглядати дані з різних точок зору. Наприклад, якщо осі включають виміри часу, продукту та регіону, користувач може аналізувати дані про продажі за певний період часу в конкретній географічній області.

Це дає можливість виконувати аналіз з різних поглядів та використовувати OLAP для отримання значущої інформації з даних, що допомагає в управлінні бізнес–процесами та ухваленні управлінських рішень.

Під час створення вимірів важливо правильно налаштувати зв'язки між їх атрибутами. Це є вирішальним для коректного відображення даних. Щодо ієрархій,

вони є ключовою частиною вимірів. Кожен раз, коли вимір має пов'язані атрибути, варто розглядати можливість створення ієрархій. Усі виміри повинні мати принаймні одну ієрархію, за винятком "батьківсько-дочірніх", оскільки вони мають свій унікальний тип ієрархії. Порядок атрибутів у ієрархії має значення, оскільки, якщо атрибути на нижчому рівні мають менше елементів, ніж на вищому, це може призвести до непорозумінь. У відношенні до ієрархій важливо, щоб усі атрибути були пов'язані з вищим рівнем, що допомагає оптимізувати роботу сервера[39].

2.6 Схема роботи додатку

Структурну схему застосунку зображено на рисунку 2.8.

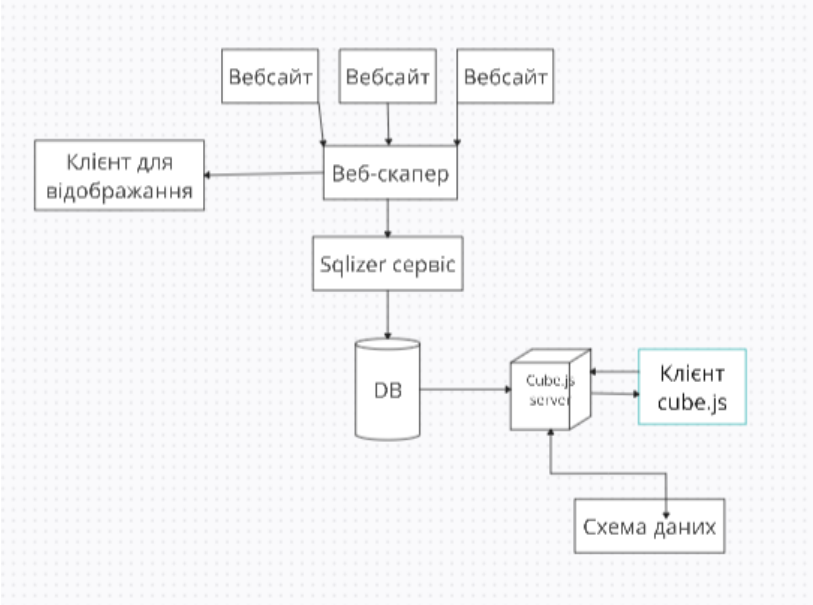


Рисунок 2.8 – Архітектура програмного засобу

3 РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ МОНІТОРИНГУ ТА СТАТИСТИЧНОГО АНАЛІЗУ ДАНИХ

В результаті розробки було створено програмний продукт призначений для автоматичного збору даних та аналізу ринку оренди нерухомості. Його функціональність включає збір даних через веб–скрапінг, обробку цих даних з внесенням до бази даних MySQL. Він також відображає актуальні оголошення, створює OLAP куби для аналізу даних та візуалізує статистичні дані з використанням цих кубів. Цей програмний засіб спрямований на забезпечення користувачів зручним та інформативним способом отримання інформації про ринок оренди нерухомості для подальшого аналізу та прийняття рішень.

3.1 Опис програмних засобів що використовуються

При розробці програмного застосунку були використані наступні програмні засоби:

Spring — популярний фреймворк для розробки програмного забезпечення в середовищі мови програмування Java. Він надає комплексний набір інструментів та бібліотек, які спрощують розробку веб–додатків, сервісів, додатків для підприємств та інших програмних рішень. В Spring вбудовані інструменти для розробки веб–додатків, такі як Spring MVC, що дозволяє обробляти HTTP–запити, створювати веб–сервіси, має модульну структуру, що дозволяє розробити додаток за принципами модульності для полегшення розширення та обслуговування коду. Фреймворк Spring сприяє створенню тестирувальних сценаріїв та підтримці тестування, що допомагає забезпечити високу якість коду та функціональності програмного забезпечення.

Бібліотека Jsoup — це потужний інструмент у Java для парсингу HTML та обробки даних на основі HTML–документів. Вона дозволяє легко отримувати доступ до вмісту веб–сторінок, видобувати потрібну інформацію та здійснювати маніпуляції з даними, які потрібні для подальшого аналізу або використання в програмах. Jsoup було обрано для розробки парсера даних з кількох причин. Вона

має простий та легкий у використанні API, що спрощує витягування інформації з HTML–сторінок. Також вона дозволяє легко маніпулювати структурою та елементами HTML–документу через уніфікований DOM–інтерфейс. Крім цього, Jsoup надає можливість вибирати елементи з HTML–документу за допомогою CSS–подібного синтаксису, що робить процес витягування даних зручнішим. Jsoup також вміє ефективно обробляти неповні або пошкоджені HTML–сторінки, намагаючись виправити помилки та забезпечуючи здатність парсингу навіть у таких випадках.

XAMPP — набір програмного забезпечення, який включає в себе всі необхідні компоненти для створення локального веб–сервера або локального сервера для розробки веб–додатків. В ньому використовується сервер Apache , який використовується для обробки HTTP–запитів та відповідей на них.

MySQL — Система управління базами даних (СУБД), яка дозволяє зберігати, керувати та маніпулювати даними. MySQL широко використовується для зберігання даних для веб–додатків, блогів, електронних комерційних сайтів та інших типів додатків.

JPA — Використання JPA спрощує роботу з базами даних у Java–програмах, уникнення необхідності написання багато коду для взаємодії з БД та забезпечення стандартизованого підходу до роботи з об'єктно–реляційним відображенням даних. Таким чином, використання JPA в проекті веб–парсера дозволяє ефективно управляти отриманими даними, забезпечуючи їх збереження та обробку в базі даних і спрощуючи взаємодію з цими даними у програмі.

3.2 Створення інтерфейсу користувача для відображення та представлення аналітичної інформації у вигляді візуальних елементів.

В цьому підрозділі описано створення інтерфейсу користувача, який дозволить представити аналітичну інформацію таким чином, щоб можна було врахувати вказані налаштування та зручно переглядати дані.

Інтерфейс користувача для відображення аналітичної інформації використовується для того, щоб зробити дані зрозумілими та доступними для користувачів через візуальні елементи. Основна мета цього інтерфейсу — це надати можливість користувачам легко аналізувати дані, робити висновки та приймати рішення на основі цієї аналітики.

Користувачеві доступні різноманітні можливості для аналізу ринку нерухомості через створення власних залежностей та графіків, використовуючи дані, що надаються за допомогою cube.js. Це дозволяє користувачеві вибирати потрібні значення, одиниці виміру та способи відображення інформації. Крім того, функція фільтрації дозволяє вибирати дані за певним показником. На прикладі рисунку 3.1 демонструється лінійний графік, створений користувачем, що ілюструє зміну середньої ціни об'єкту нерухомості кожного місяця протягом останнього року.

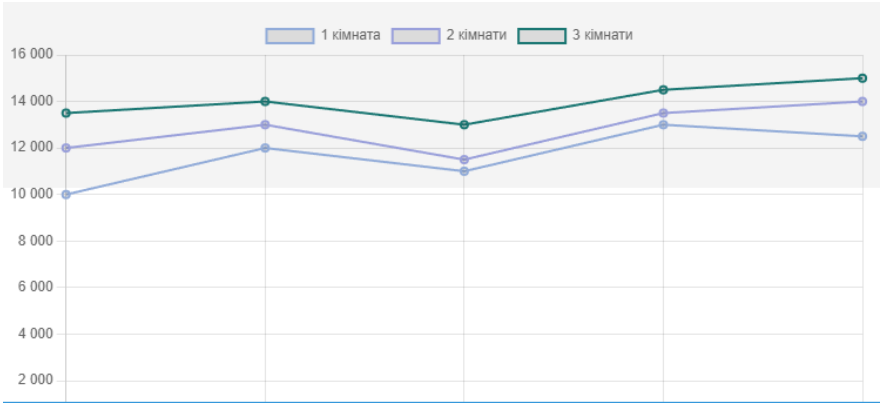


Рисунок 3.1 — Створення діаграми

Натискання на кнопки з кількістю кімнат змінює візуалізацію та відображає дані у вигляді графіку.

3.3 Генерація тестових даних

Для ілюстрації аналізу даних відображення в графіках за допомогою технології OLAP та потрібно мати значну кількість даних. Створення такого обсягу даних вимагає часу, оскільки веб–скрапер оперує лише поточними

об'єктами, кількість яких, як правило, не перевищує кількох тисяч. Зі зростанням часу можна накопичувати більше даних, але для тестування роботи додатку ми вирішили створити 1,5 мільйони записів про нерухомість вручну, вибираючи значення, які максимально наближені до реальних.

Для створення даних ми використовували сервіс json–generator, який надає зручний інтерфейс для створення даних у форматі JSON. Цей сервіс дозволяє налаштовувати формат та значення для кожного поля окремо. На рисунку наведено приклад генерації двох записів про операції оренди.

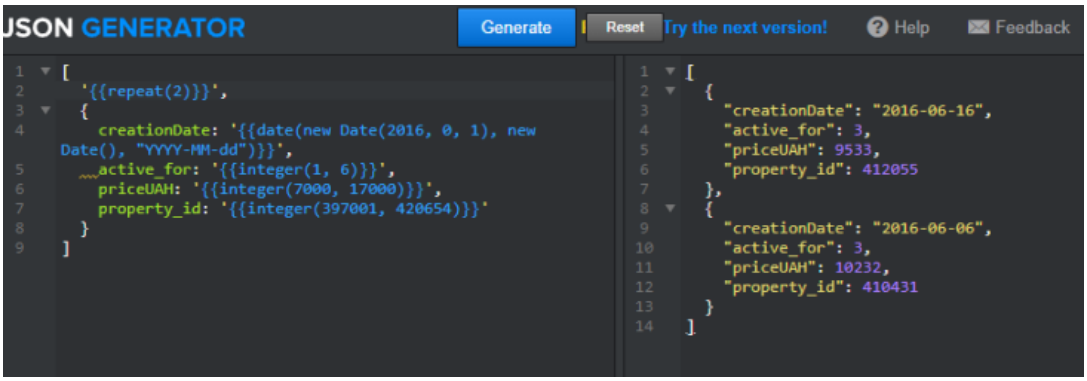
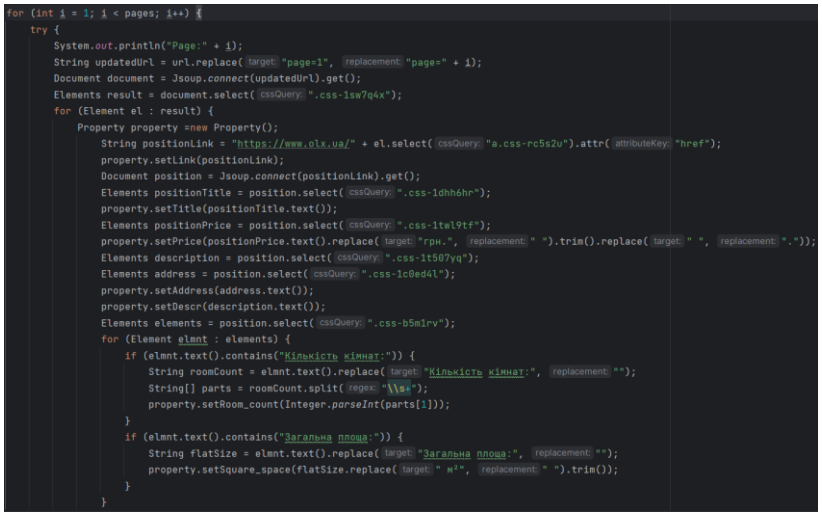


Рисунок 3.2 — Генерація даних

3.3 Розробка веб–парсера для збору інформації

Збір інформації за допомогою веб–скрапера є ключовим етапом у процесі накопичення даних для подальшої обробки та аналізу. Для отримання інформації про останні об'єкти нерухомості, які доступні на вибраних дошках оголошень, розробимо веб–парсер за допомогою фрейморвку Spring та з використанням бібліотеки Jsoup.

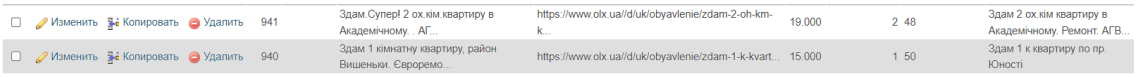
Алгоритм проходження веб–скраперу наступний. Спочатку парсер перевіряє наявну кількість сторінок з оголошеннями і через цикл збирає необхідні метадані з html сторінки. Частину коду парсера представлено на рисунку



```
for (int i = 1; i < pages; i++) {
    try {
        System.out.println("Page:" + i);
        String updatedUrl = url.replace( target: "page=1",  replacement: "page=" + i);
        Document document = Jsoup.connect(updatedUrl).get();
        Elements result = document.select( cssQuery: ".css-1sw7q4x");
        for (Element el : result) {
            Property property =new Property();
                String positionLink = "https://www.olx.ua/" + el.select( cssQuery: "a.css-rc5s2u").attr( attributeKey: "href");
                property.setLink(positionLink);
                Document position = Jsoup.connect(positionLink).get();
                Elements positionTitle = position.select( cssQuery: ".css-1dhh6hr");
                property.setTitle(positionTitle.text());
                Elements positionPrice = position.select( cssQuery: ".css-1twl9tf");
                property.setPrice(positionPrice.text().replace( target: "грн.",  replacement: " ").trim().replace( target: " ",  replacement: "."));
                Elements description = position.select( cssQuery: ".css-1t507yq");
                Elements address = position.select( cssQuery: ".css-1c0ed4l");
                property.setAddress(address.text());
                property.setDescr(description.text());
                Elements elements = position.select( cssQuery: ".css-b5m1rv");
                for (Element elmnt : elements) {
                    if (elmnt.text().contains("Кількість кімнат:")) {
                        String roomCount = elmnt.text().replace( target: "Кількість кімнат:",  replacement: "");
                        String[] parts = roomCount.split( regex: "\\s+");
                        property.setRoom_count(Integer.parseInt(parts[1]));
                    }
                    if (elmnt.text().contains("Загальна площа:")) {
                        String flatSize = elmnt.text().replace( target: "Загальна площа:",  replacement: "");
                        property.setSquare_space(flatSize.replace( target: " м²",  replacement: " ").trim());
                    }
                }
```

Рисунок 3.2 — Код парсера

Так, веб–скрапер автоматично переглядає кожну сторінку з оголошеннями, збирає загальну інформацію про об'єкти, а потім переходить до кожного окремого об'єкту, щоб доповнити отримані записи більш деталізованою інформацією, такою як ціна, площа жилого приміщення, кількість кімнат тощо. По завершенню роботи, отримаємо набір даних, який буде збережений в базі даних в наступному форматі як показано на рисунку 3.3.



☐	Изменить	Копировать	Удалить	941	Здам.Супер! 2 ох.кім.квартиру в Академічному. . АГ...	https://www.olx.ua//d/uk/obyavlenie/zdam-2-oh-km-k...	19.000	2 48	Здам 2 ох.кім квартиру в Академічному. Ремонт. АГВ...
☐	Изменить	Копировать	Удалить	940	Здам 1 кімнатну квартиру, район Вишеньки. Євроремо...	https://www.olx.ua//d/uk/obyavlenie/zdam-1-k-kvart...	15.000	1 50	Здам 1 к квартиру по пр. Юності

Рисунок 3.3 — Представлення даних, отриманих парсером, в базі даних.

Також при об'яві класу, який визиває парсер була використана анотація @Component , що в свою чергу дозволило використання анотації @Scheduled для автоматичного виклику функції парсера завжди тільки після його завершення для забезпечення наявності завжди актуальних даних в обраній базі даних. Приклад анотації наведено на рисунку 3.4.

```java
@Scheduled(fixedDelay = 1000)
public void ScheduledDataParsing(){
    System.out.println("Scheduled task begins=========");
    olxParserService.startParsing();
    System.out.println("Scheduled task ends=========");
}
```

Рисунок 3.4 – Постійний виклик функції через 1 секунду після завершення

4. ТЕСТУВАННЯ ПРОГРАМНОГО МОДУЛЯ ДЛЯ ПОШУКУ І АГРЕГАЦІЇ ПРОПОЗИЦІЙ НЕРУХОМОСТІ

Тестування програмного модуля пошуку і агрегації пропозицій з онлайн платформ оренди нерухомості було важливою частиною розробки. Цей розділ присвячений опису процесу тестування та аналізу отриманих результатів.

4.1 Тестування програмного модуля

Тестування програмного модуля — це процес перевірки функціональності, продуктивності та надійності програмного забезпечення з метою забезпечення його коректної роботи.

Тестування програмного модуля спрямоване на перевірку функціональності, надійності, продуктивності та відповідності вимогам розробленого програмного забезпечення. Цілі тестування полягають у визначенні та підтвердженні:

— відповідності вимогам,перевірка відповідності функціональності програми зазначеним вимогам і специфікаціям;

— функціональності, перевірка коректності роботи функцій програмного модуля за різних умов використання;

— надійності, визначення стабільності та надійності програми під час її використання;

— продуктивності, оцінка швидкості та ефективності виконання програмних процесів;

Види тестування програмного модуля:

— модульні тести, тестування окремих компонентів або функцій модуля для перевірки їх коректності;

— інтеграційні тести, перевірка взаємодії між різними компонентами програмного забезпечення;

— функціональні тести, тестування функціональності програмного модуля згідно зі специфікаціями;

— стрес-тести, перевірка роботи модуля при великому обсязі даних або в тяжких умовах навантаження;

— системне тестування, перевірка програми як єдиного функціонального блоку для переконання в тому, що вона відповідає специфікації.

Кількість тестів для навіть простих програмних компонентів може бути практично нескінченною. Однак, стратегія тестування полягає у проведенні лише обов'язкових тестів з урахуванням обмежених часових та ресурсних можливостей. Таким чином, програмні засоби перевіряються стандартним запуском програми для виявлення помилок, дефектів або інших недоліків. В нашому випадку будуть використовуватись системне тестування програмного додатку.

Системне тестування — це процес перевірки поведінки всієї системи як цілого, щоб забезпечити відповідність вимогам та очікуванням клієнтів. Цей вид тестування включає в себе тестування всіх компонентів програмного забезпечення разом з їх взаємодією для перевірки працездатності та коректності роботи системи в цілому.

Системне тестування орієнтоване на перевірку функціональності, продуктивності, надійності та інших аспектів системи перед її випуском на ринок або впровадженням у виробництво. Це може включати тестування інтеграції, тестування відмовостійкості, тестування производительности і тестування відновлення після відмов.

Основна мета системного тестування - виявлення проблем та дефектів, які можуть виникнути при реальному використанні системи та забезпечення її готовності до випуску або впровадження.

Методи тестування білої та чорної скриньки - це два різних підходи до тестування програмного забезпечення, які базуються на різних рівнях доступу до внутрішньої структури програми та знання її коду.

Метод тестування білої скриньки базується на знанні внутрішньої структури програми, коду, алгоритмів та інших технічних деталей. Виконується аналіз коду для створення тестів, які перевіряють правильність функціонування програми на

основі її внутрішніх механізмів. Мета полягає у тестуванні усіх можливих шляхів в коді для підтвердження коректності роботи програми та виявлення потенційних помилок.

При методі тестування чорної скриньки програма розглядається як чорний ящик, тобто тестувальник не має доступу до внутрішніх деталей чи коду програми. Тестування здійснюється на основі зовнішнього інтерфейсу програми та функціональності без знання її внутрішньої реалізації. Тестування чорної скриньки орієнтоване на перевірку зовнішньої поведінки, вхідних та вихідних даних для перевірки відповідності очікуваним результатам.

Результати проведення тестів методом чорної скриньки представлено у таблиці 4.1

Таблиця 4.1

<table>
<tr><td rowspan="3">Код тест-кейса</td><td colspan="3">Опис тест-кейса</td></tr>
<tr><td colspan="2">Хід тестування</td><td>Очікуваний результат</td></tr>
<tr><td>Дата тестування</td><td>Результат</td><td>Примітка</td></tr>
<tr><td rowspan="3">001</td><td colspan="3">Перевірка коректного відображення оголошень на стартовій сторінці</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Перевірити відображення результатів пошуку</td><td>Коректне відображення</td></tr>
<tr><td>7.12.2023</td><td>Пройдено</td><td>-</td></tr>
<tr><td rowspan="3">002</td><td colspan="3">Сортування за ціною</td></tr>
<tr><td colspan="2">1. Запустити додаток
2. Вибрати фільтр сортування ціни за зростанням
3. Перевірити відображення результатів пошуку</td><td>Список оголошень відображається коректно</td></tr>
<tr><td>7.12.2023</td><td>Пройдено</td><td>-</td></tr>
</table>

Продовження таблиці 4.1

003	Сортування за ціною		
	1. Запустити додаток 2. Вибрати фільтр сортування ціни за спаданням 3. Перевірити відображення результатів пошуку	Список оголошень відображається коректно	
	7.12.2023	Пройдено	-
004	Сортування за кількістю кімнат		
	1. Запустити додаток 2. Вибрати фільтр сортування кількості кімнат за спаданням 3. Перевірити відображення результатів пошуку	Список оголошень відображається коректно	
	7.12.2023	Пройдено	-
005	Сортування за кількістю кімнат		
	1. Запустити додаток 2. Вибрати фільтр сортування кількості кімнат за спаданням 3. Перевірити відображення результатів пошуку	Список оголошень відображається коректно	
	7.12.2023	Пройдено	-

4.2 Інструкція користувача

Програмний засіб розроблено для спрощення та полегшення пошуку оренди нерухомості через онлайн платформи. Ця інструкція призначена для надання допомоги користувачеві під час використання програми.

Для початку роботи необхідно запустити додаток. Користувачеві буде представлений інтерфейс, що складається з фільтрів пошуку, графіку середніх цін за місяць та самих оголошень. Є можливість відсортувати оголошення за ціною та кількістю кімнат за спаданням. Після натискання кнопки Пошук з'являться 9 оголошень на кожній сторінці. Натиснувши на фільтр сортування, відкриється вікно, де ви можете редагувати параметри сортування. Після вибору всіх параметрів натисніть кнопку "Пошук". Результати пошуку відобразяться на сторінці в розділі "Оголошення". Кожне оголошення включає:

— заголовок;

— фотографію об'єкту нерухомості;

— ціну;

— опис;

— місцезнаходження.

Для отримання детальної інформації або перегляду більше даних про об'єкт нерухомості, натисніть на заголовок оголошення.

На нижній частині сторінки є пагінація, де ви можете обрати бажану сторінку для подальшого перегляду інших результатів.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному випадку нового програмного засобу пошуку і агрегації пропозицій з онлайн платформ оренди нерухомості . Особливістю програми є інноваційний підхід до автоматизованого пошуку нерухомості та аналізу даних, можливість поліпшення користувацького досвіду під час пошуку нерухомості шляхом інтеграції інтуїтивних інтерфейсів. Новизна полягає в у вдосконаленні засобів візуалізації стану ринку та в удосконалені методу автоматичного збору та аналізі даних ринку оренди нерухомості.

Аналогом може бути WebScraper 100$/міс., т.б. 46500 грн/рік.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 5.1 — Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Крите-рій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку

Продовження табл. 5.1

Ринкові переваги					
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	4
Наявність аналогів на ринку	4	4	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	4
Супровідна документація	4	3	4
Сума	44	42	44
Середньоарифметична сума балів	(44+42+44) / 3 = 43,33		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 - 10	Низький
11-20	Нижче середнього
21- 30	Середній
31- 40	Вище середнього
41 -48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок того, що програмний продукт є новим програмним засобом організації та оптимізації часуі виробничого процесу з інтеграцією сервісів від Google та Microsoft. Особливістю програми є можливість інтеграції аналогів для об'єднання даних. Привабливість для споживачів полягає в об'єднанні декількох сучаснихвикористовуваних систем у одному зручному додатку.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

5.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_{о} = \frac{M}{T_{р}} \cdot t, \qquad (5.1)$$

де М — місячний посадовий оклад конкретного розробника, грн.;

$T_р$ — число робочих днів в місяці, 21 день;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	28386	1013	52	52676,52
Програміст	25389	906	52	47112,84
Всього				99788,36

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

5.2.2 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 13,3% від основної заробітної плати розробників та робітників:

$$Зд = Зо \cdot 13{,}3\ \% / 100\ \% \qquad (5.2)$$

$$Зд = (99788{,}36 \cdot 13{,}3\ \% / 100\ \%) = 13271{,}76\ (грн.)$$

5.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Нз = (Зо + Зд) \cdot 22\ \% / 100\% \qquad (4.3)$$

$$Нз = (99788{,}36 + 13271{,}76) \cdot 22\ \% / 100\ \% = 24\ 873{,}23\ (грн.)$$

5.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

5.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$\frac{Ц}{T_в} \frac{t_{вик}}{12} \text{ [грн.]} \qquad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ — термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на комп’ютер балансова вартість якого становить 21000 грн., термін його корисного використання згідно податкового законодавства — 2 роки, а термін його фактичного використання — 1,72 міс.

$$A_{обл} = \frac{21000}{2} \times \frac{1.72}{12} = 30005 \text{ грн}$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення.

Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн. (IntelliJ IDEA Ultimate 2 276 грн/міс, використовувався 1,72 місяці), то даний нематеріальний актив не амортизується, а

його вартість включається у вартість розробки повністю, Внем.ак.= 13062 грн Розрахунки заносимо до таблиці 5.5.

Таблиця 5.5 — Амортизаційні витрати

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	21000	2	1,72	30005
Офісне обладнення (меблі)	24000	4	1,72	859,8
Приміщення	1204651	20	1,72	8640
Всього				11004,57

5.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п , \quad (5.5)$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,3 кВт;

Ф — фактична кількість годин роботи обладнання, годин;

Кп — коефіцієнт використання потужності, Кп = 0,9.

$$Ве = 0,9 \cdot 0,3 \cdot 8 \cdot 40 \cdot 6,2 = 535,68 \text{ (грн.)}$$

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені бзпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_e = (з_О + з_Р) \times \frac{Н_{інв}}{100\%} \quad (5.6)$$

де Нів — норма нарахування за статтею «Інші витрати».

Ів = 99788,36* 57% / 100% = 56879,36 (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково- технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…3000% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \times \frac{Н_{нзв}}{100\%}, \quad (5.7)$$

де Ннзв — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

Ннзв = 99788,36* 110 % / 100 % = 109 767,196 (грн.)

5.2.8 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$Взаг = 99788,36+13271,76 +24\ 873,23 +13062+11004,57+544,32+56879,36 +$$

$$+109\ 767,196 = 329\ 190,796 \text{ грн.}$$

5.2.9 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати η на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються ЗВ, визначається за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн)}, \qquad (5.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 329\ 190,796 / 0,5 = 658\ 381,592 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

5.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_{\text{і}} = (\pm \Delta Цо \times N + Цо \times \Delta N)_{i} \times \lambda \times р \times (1 - \frac{\vartheta}{100}), \quad (5.10)$$

де ±ΔЦо — зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

Цо — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, Цо = Цб ± ΔЦо;

Цб — вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість і ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333;

р — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2023 році ϑ =18%.

Припустимо, що при прогнозованій ціні 60 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на 40 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року — на 10000 шт., протягом другого року — на 30000 шт., протягом третього року на 70000 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$$\Delta П1 = (0*40 + (60 + 40)* 10000* 0{,}8333* 0{,}27) * (1 - 0{,}18) = 185237 \text{ грн.}$$

$$\Delta П2 = (0*40 + (60 + 40)*(10000+30000)* 0{,}8333* 0{,}27) * (1 - 0{,}18) = 738003.28 \text{ грн.}$$

$$\Delta П3 = (0*40 + (60 + 40)*(10000+30000+70000)* 0{,}8333* 0{,}27) * (1 - 0{,}18) = 2049999.97 \text{ грн.}$$

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 994214 грн.

5.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження $\sum$та комерціалізації науково-технічної розробки:

$$ПП = \sum_1^T \times \frac{\Delta П_i}{(1+\tau)^t} \quad , \qquad (5.11)$$

де ΔП збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, = 0,05…0,15;

t — період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= (185237/(1+0,1)1)+(738003.28 /(1+0,1)2)+(2049999.97/(1+0,1)3)

= 168397.2727+609915.0661+1540088.6872= 2319401.025 грн.

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.12)$$

де kінв — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію і це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай kінв=2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 2 * 658\,381{,}592 = 1{,}316{,}763\ грн$$

Тоді абсолютний економічний ефект Еабс або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \qquad (5.13)$$

Еабс = 2319401.025 – 1,316,763 = 1,002,638грн.

Оскільки Еабс $>$ 0 то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розрахувати внутрішню економічну дохідність або показник внутрішньої норми дохідності (IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково- технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій Ев . Для цього використаємо формулу:

$$E_{в} = \sqrt[Т_Ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.14)$$

Тж − життєвий цикл наукової розробки, роки.

$$Ев = \sqrt[3]{(1 + 1{,}002{,}638/1{,}316{,}763)} - 1 = 0{,}448$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \qquad (5.15)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d = (0,09…0,14);

f—показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{min} = 0{,}14 + 0{,}1 = 0{,}24.$$

Так як Ев > τmin, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

$E_в$

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$\text{Ток} = \frac{1}{\text{Е}_\text{в}}, \qquad (5.16)$$

Ток = 1 / 0,448 = 2,23 р

Оскільки Ток < 3-х років, а саме термін окупності рівний 2,23 роки, то фінансування даної наукової розробки є доцільним.

ВИСНОВКИ

Під час виконання магістерської роботи було досліджено процес автоматичного збору даних, розроблено програмний модуль автоматичного збору даних з різних ресурсів з їх подальшою агрегацією з метою покращення користувацького досліду при виборі пропозицій оренди нерухомості.

В першому розділі було проведено аналіз відомих методів та засобів для створення програмного засобу автоматичного збору даних з різних ресурсів з їх подальшою агрегацією. Було проаналізовано методи та засоби автоматичного збору даних, методи візуалізації даних, розглянуті конкретні методології та інструменти зберігання інформації та проведено огляд існуючих готових рішень на ринку нерухомості.

В другому розділі на основі розглянутих методів та засобів систем моніторингу й аналізу даних ринку нерухомості виконано порівняння популярних методів зберігання даних та розроблено програмний засіб автоматичного збору даних з описанням принципу його роботи та детальним оглядом його функціональних можливостей, розроблено схему бази даних і представлено схему роботи створеної програми.

В третьому розділі магістерської роботи була представлена розробка програмних засобів моніторингу даних. Приведено список програмних інструментів, що використовуються, створення інтерфейсу користувача, розробку веб-парсера автоматичного збору інформації.

В четвертому розділі було розглянуто основні методи тестування, виконано тестування програмного модуля методом «чорної скриньки», розроблено список тестових випадків для тестування, розроблено керівництво для користувача.

У п'ятому розділі магістерської роботи проведено оцінку економічної цілеспрямованості розробки програмного модуля для автоматичного збору даних ринку нерухомості. Також був проведений технологічний аудит науково-технічної розробки, в ході якого були розраховані необхідні економічні витрати. Ці витрати включали оплату праці, витрати на матеріали, амортизацію обладнання, програмні

засоби та витрати на електроенергію, необхідні для успішної реалізації програмного додатку.

Створено програмний засіб пошуку об’єктів оренди нерухомості, який може використовуватись звичайними користувачами для пошуку об’єкту оренди у модулі моніторингу й відображення даних про фактичний стан об’єктів оренди нерухомості, а також використовуватися аналітиками й компаніями для аналізу даних про стан ринку оренди нерухомості.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Chart: Global Data Creation is About to Explode | Statista [Електронний ресурс], – Режим доступу: URL: https://www.statista.com/chart/17727/global–data–creation–forecasts/.

2. Emil Persson. Evaluating tools and techniques for web scraping – KTH Royal Institute of Technology, 2019.

3. Web Scraping with Python: A Comprehensive Guide by Ryan Mitchell(https://www.oreilly.com/library/view/web–scraping–with/9781491985564/)

4. Мартинов П.Г., Очкуров М.А. «Програмний засіб пошуку та відбору даних з онлайн-платформ оренди нерухомості». Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2024)», 2024 р. [Електронний ресурс] — https://conferences.vntu.edu.ua/index.php/mn/mn2024/paper/view/19782.

5. A Gentle Introduction to Web Scraping (https://towardsdatascience.com/a–gentle–introduction–to–web–scraping–with–python–b914a64b2fb8)

6. The Ultimate Guide to Web Scraping with Python(https://www.learndatasci.com/tutorials/ultimate–guide–web–scraping–w–python–requests–and–beautifulsoup/)

7. Web Scraping using Beautiful Soup and Selenium for dynamic page (https://medium.com/ymedialabs–innovation/web–scraping–using–beautiful–soup–and–selenium–for–dynamic–page–2f8ad15efe25)

8. What is an API? In English, please (https://www.freecodecamp.org/news/what–is–an–api–in–english–please–b880a3214a82/)

9. RESTful API with Spring Boot and Spring Data JPA"

10. Офіційна документація Swagger/OpenAPI (https://swagger.io/specification/)

11. Natural Language Processing in Action Lane, Howard і Hapke(https://www.manning.com/books/natural–language–processing–in–action).

12. Practical Natural Language Processing Stanford Online(https://web.stanford.edu/class/cs224n/)

13. Jsoup(https://jsoup.org/)

14. HtmlUnit (https://habr.com/en/sandbox/19174/)

15. Crawler4j(https://github.com/yasserg/crawler4j)

16. Dedić, Nedim and STANIER, Clare (2016) An evaluation of the challenges of Multilingualism in Data Warehouse development. In: ICEIS 2016, 25–28 April 2016, Rome, Italy.

17. The Data Lake Is A Design Pattern [Електронний ресурс], – Режим доступу: URL: https://medium.com/data–ops/the–data–lake–is–a–design–pattern–888323323c66

18. Bhatt D. Focused Web Crawler / D. Bhatt, V. Daiwat // Data Warehouses / D. Bhatt, V. Daiwat. – Boston, 2020. – P. 101–110.

19. Web Scraping vs. API: What's the Best Way to Extract Data? [Електронний ресурс]. – Режим доступу: https://www.makeuseof.com/webscraping–vs–api/.

20. Dahiwale P. Intelligent web crawler / P. Dahiwale, A. Mokhade, M. Raghuwanshi // Emerging Trends in Technology / – Mumbai, 2010. – P. 26–27.

21. Sun, Yang & Councill, Isaac & Giles, C.. (2010). The Ethicality of Web Crawlers. Proceedings – 2010 IEEE/WIC/ACM International Conference on Web Intelligence, WI 2010. 1. 668 – 675.

22. Chaudhari S. Review of web crawlers / Sangita Chaudhari // Int. J. Knowledge and Web Intelligence / Sangita Chaudhari. – Mumbai, 2014. – P. 49.

23. Patel J. Advanced Web Crawlers / Jay Patel // Getting Structured Data from the Internet / Jay Patel. – Berkeley, 2020. – P. 371–393

24. Machlis S. Simple Web scraping / Sharon Machlis // Practical R for Mass Communication and Journalism / Sharon Machlis. – New York, 2018. – P. 203–208.

25. Patel J. Introduction to Web Scraping / Jay Patel // Getting Structured Data from the Internet / Jay Patel. – Berkeley, 2020. – P. 1–30.

26. Bhatt D. Focused Web Crawler / D. Bhatt, V. Daiwat // Data Warehouses / D. Bhatt, V. Daiwat. – Boston, 2020. – P. 101–110.

27. Chaudhari S. Review of web crawlers / Sangita Chaudhari // Int. J. Knowledge and Web Intelligence / Sangita Chaudhari. – Mumbai, 2014. – P. 49.

28. Visualization–Aided Exploration of the Real Estate Data / M.Li, T. Sellis, S. Yan, Z. Bao // Databases Theory and Applications / M.Li, T. Sellis, S. Yan, Z. Bao. – Cham, 2016. – P. 453–459.

29. Home Price Visualization [Електронний ресурс]. – Режим доступу: https://www.verachen.me/home–price–visualization (дата звернення: 22.11.2023).

30. Методи визуалізації та їх сприйняття [Електронний ресурс]. – Режим доступу: https://webometr.kpi.ua/files/visual-data.pdf (дата звернення: 22.11.2023).

31. Aparicio M. Data visualization / M. Aparicio, C. Costa // Communication Design Quarterly / M. Aparicio, C. Costa. – Lisboa, 2014. – P. 7–11

32. Maximiano M. Data visualization techniques for real–time information — A custom and dynamic dashboard for analyzing surveys' results / M. Maximiano, C. Reis, D. Guevara // Data Visualization Techniques / M. Maximiano, C. Reis, D. Guevara. – Caceres, 2018. – P. 1–7.

33. Jung H. Study on the Visualization Elements of Web Information Services: Focused on Researcher Network and Graphic Chart / H. Jung, M. Lee // Information Visualization / H. Jung, M. Lee. – Las Vegas, 2011. – P. 459–463

34. What is the best way to make sense of property data? [Електронний ресурс]. – Режим доступу: https://assetti.pro/property–data–visualization/ (дата звернення: 22.11.2023).

35. Dharmendra B. An overview of data warehousing and OLAP technology / Biswas Dharmendra // Data Warehouses and OLAP: Concepts, Architectures, and Solutions / Biswas Dharmendra. – Mumbai, 2020. – P. 65–74

36. What is OLAP? [Електронний ресурс]. – Режим доступу: https://www.ibm.com/cloud/learn/olap (дата звернення: 21.11.2023).

37. Wang Q. Reduced Quotient Cube: Maximize Query Answering Capacity in OLAP / Q. Wang, B. Zou, J. You // IEEE Access / Q. Wang, B. Zou, J. You. – New York, 2021. – P. 21.

38. Chen Z. An Approach for Developing Platform of OLAP / Zhuo Chen // Transactions on Computer Science and Engineering / Zhuo Chen. – Hong Kong, 2019. – P. 543–545

39. OLTP and OLAP: a practical comparison [Електронний ресурс]. – Режим доступу: https://ua.stitchdata.com/resources/oltp-vs-olap/ (дата звернення: 21.11.2023).

40. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

ДОДАТОК А

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
_____проф., д.т.н. О. Д. Азаров
«___» _____2023р

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи
«Програмний засіб пошуку і агрегації пропозицій з онлайн платформ оренди нерухомості»
08-54.КМКР.035.00.000 ПЗ

Науковий керівник: д. т. н., проф. каф ОТ
_____ Мартинюк Т.Б.
Магістрант групи 2КІ-21м
_____ Мартинов П.Г.

Вінниця 2023

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність дослідження пов’язана з необхідністю підвищення доступності та зручності отримання інформації рядовими користувачами про ринок нерухомості.

1.2 Наказ про затвердження теми магістерської кваліфікаційної роботи.

2 Мета і призначення МКР

2.1 Мета магістерської роботи полягає у підвищенні якості корстувацького досвіду шляхом розробки програмного модуля автоматичного збору.

2.2 Призначення розробки — виконання магістерської кваліфікаційної роботи.

3 Вихідні дані для виконання МКР

Виконати розробку програмного модуля автоматичного збору даних з декількох джерел з подальшою агрегацією. Схему бази даних та лістинги програми представити в додатках до роботи.

4 Вимоги до виконання МКР

МКР повинна задовольняти такі вимоги:

— запропонувати програмний модуль автоматичного збору інформації;

— розробити алгоритм та структуру для програмного модуля автоматичного збору даних;

— розробити структуру програмного модуля;

— забезпечення зручного та легкого для користувача графічного інтерфесу;

— результат роботи, а саме програмний модуль автоматичного збору даних.

5 Етапи МКР та очікувані результати в таблиці А.1

Таблиця А.1 — Етапи виконання роботи

№	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз завдання. Вступ	01.09.23	02.09.23	Вступ
2	Аналіз літературних джерел	03.09.23	04.09.23	Розділ 1
3	Розробка технічного завдання	4.09.2023	24.09.23	Технічне завдання
3	Розробка структури програмного модуля автоматичного збору даних	25.09.23	08.10.23	Розділ 2, розробка структури
4	Розробка програмного модуля автоматичного збору даних	11.10.23	29.10.23	Розділ 3, розробка програми
5	Практична реалізація, результати.	01.11.23	14.11.23	Розділ 3
	Тестування програмного модуля	15.11.23	30.11.23	Розділ 4
6	Розробка економічної частини	02.12.23	7.12.23	Розділ 4
7	Оформлення пояснювальної записки	02.12.23	15.12.23	ПЗ, презентація

6 Матеріали, що подаються до захисту МКР

До захисту МКР подаються: пояснювальна записка МКР, ілюстративні та графічні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук рецензента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

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

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

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

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

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

— Методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія». Кафедра обчислювальної техніки ВНТУ 2022;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

Результати роботи програми

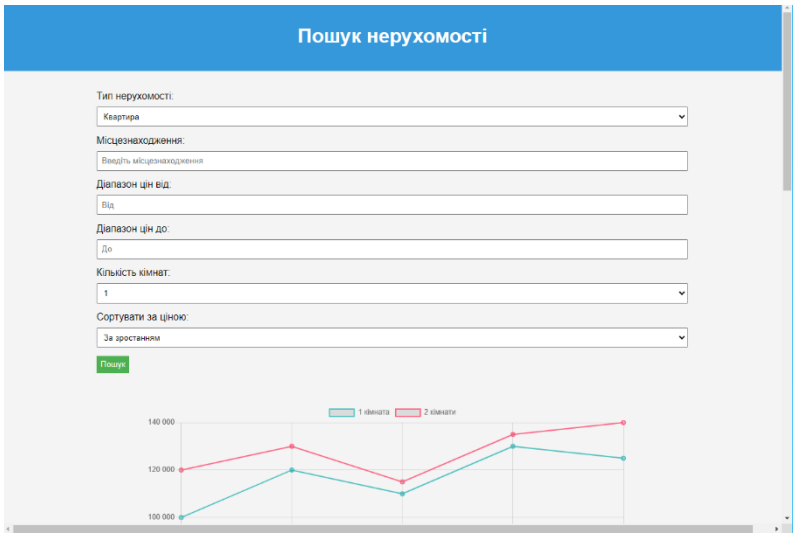


Рисунок Б.1 — Фільтри на основній сторінці додатку.

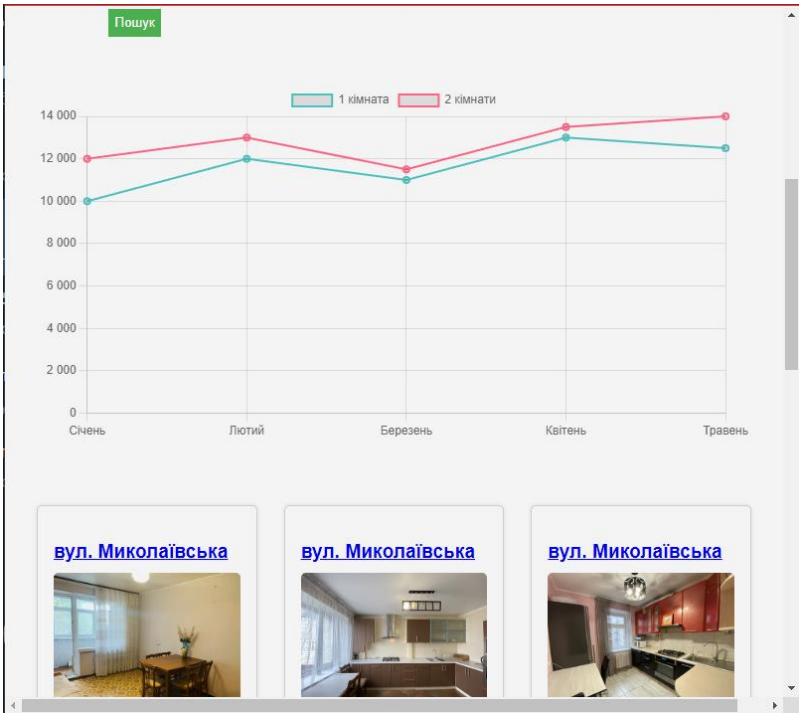


Рисунок Б.2 — Графік цін на основній сторінці додатку.

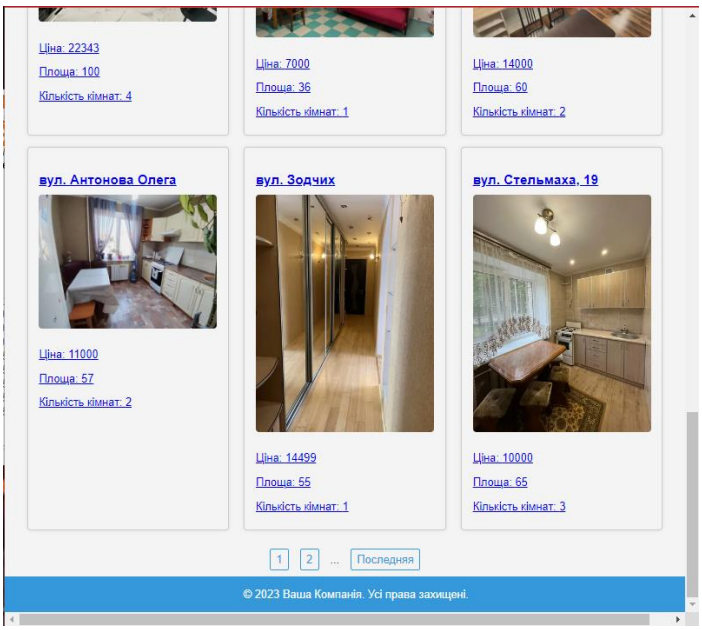


Рисунок Б.3 — Пагінація та оголошення на основній сторінці додатку.

ДОДАТОК В

Лістинг класу збору даних

Фрагмент лістингу класу автоматичного збору даних

```
package com.ProPropertyPortal.demo.parser;
import com.ProPropertyPortal.demo.model.Property;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;
import org.springframework.stereotype.Component;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
@Component
public class DomRiaParser {
    private static int max_pages;
    public static List<Property> parseData(int pages, String url){
        List<Property> list = new ArrayList<>();
        for(int i =1;i<pages; i++){
            try{
                System.out.println("Page:"+i);
                String updatedUrl=url.replace("page=1","page="+i);
                Document document= Jsoup.connect(updatedUrl).get();
                Elements result=document.select(".realty-item");
                for(Element el:result){
                    Property property =new Property();
                    String price=el.select("b.size22").text().replace("грн"," ");
```

```
                property.setPrice(Integer.parseInt(price.replaceAll("[^\\d]", "")));
                property.setAddress(el.select("a.realty-link ").text());
                property.setDescr(el.select("div.mt-15").text());
                property.setTitle(property.getDescr());
                //System.out.println("Location  "+el.select("span.mb-5").text());
                if(!el.select("span.point-before").isEmpty()){
                    String rooms=el.select("span.point-before").get(0).text();
                    if(rooms.contains("кімнат")){
                        String[] parts = rooms.split("\\s+");
                        property.setRoom_count(Integer.parseInt(parts[0]));
                    }
                    property.setSquare_space(Integer.parseInt(el.select("span.point-
before").get(1).text().replace("м²"," ").trim()));
                    //System.out.println("Floor                    "+el.select("span.point-
before").get(2).text());
                }
                property.setLink("https://dom.ria.com/"+el.select("a.realty-
link").attr("href"));
                list.add(property);
            }
        }  catch (IOException e) {
            System.out.println("error");
            e.printStackTrace();
        }
    }
    return list;
}
public static int getPages(String  url){
```

```
        try{
            Document document= Jsoup.connect(url).get();
            Elements result = document.select(".page-item");
            max_pages= Integer.parseInt(result.get(4).text());
            System.out.println("max_pages = "+max_pages);
        }
        catch (IOException e) {
            System.out.println("error");
            e.printStackTrace();
        }
        return max_pages;
    }
}
```

ДОДАТОК Г

Лістинг головного комнонента

Фрагмент лістингу комнонента App.js

```
import  ct, { useState, useEffect } from 'react';
import axios from 'axios';
import './App.css';
import Header from './components/Header';
import FilterForm from './components/FilterForm';
import ChartComponent from './components/ChartComponent';
import RentalItem from './components/RentalItem';
import Footer from './components/Footer';
const API_ENDPOINT = 'http://localhost:8080/home';
const chartData = {
  labels: ['Січень', 'Лютий', 'Березень', 'Квітень', 'Травень'],
  datasets: [
    {
      label: '1 кімната',
      data: [10000, 12000, 11000, 13000, 12500],
      borderColor: 'rgba(75, 192, 192, 1)',
      borderWidth: 2,
      fill: false
    },
    {
      label: '2 кімнати',
      data: [12000, 13000, 11500, 13500, 14000],
      borderColor: 'rgba(255, 99, 132, 1)',
      borderWidth: 2,
      fill: false
    },
```

```
  ]
};
function App() {
  const [rentalItems, setRentalItems] = useState([]);
  const fetchData = async (queryParams) => {
    try {
      const response = await axios.get(API_ENDPOINT, { params: queryParams });
      setRentalItems(response.data); // Assuming the response contains an array of rental
items
    } catch (error) {
      console.error('Error fetching data:', error);
    }
  };
  useEffect(() => {
    // Fetch data initially without any query params
    fetchData({});
  }, []);
  const handleFormSubmit = (formData) => {
    // Call the API with form data as query parameters
    fetchData(formData);
  };
  return (
    <body>
      <Header />
      <section>
        <FilterForm onSubmit={handleFormSubmit} />
        <ChartComponent chartData={chartData} />
      </section>
      <div className="rental-info">
```

```
      {rentalItems.map((item, index) => (
        <RentalItem
          key={index}
          title={item.address}
          imageSrc={item.imageSrc}
          price={item.price}
          square_space={item.square_space}
          room_count={item.room_count}
          link={item.link}
        />
      ))}
    </div>
    <ul className="pagination">
      <li><a href="#">1</a></li>
      <li><a href="#">2</a></li>
      <li><span>...</span></li>
      <li><a href="#">Последняя</a></li>
    </ul>
    <Footer />
  </body>
 );
}
export default App;
```

ДОДАТОК Д

Лістинг компоненту оголошення

Фрагмент коду комнонета оголошення веб-додатка

```
import React, { useRef, useEffect } from 'react';
import Chart from 'chart.js/auto';
import '../styles/ChartComponent.css';
const ChartComponent = ({ chartData }) => {
  const chartContainer = useRef(null);
  const chartInstance = useRef(null);
  useEffect(() => {
    if (chartContainer && chartContainer.current) {
      if (chartInstance.current) {
        chartInstance.current.destroy();
      }
      const ctx = chartContainer.current.getContext('2d');
      chartInstance.current =  new Chart(ctx, {
        type: 'line',
        data: chartData,
        options: {
          scales: {
            y: {
              beginAtZero: true
            }
          }
        }
      });
    }

    return () => {
```

```
    // Clean up: Destroy the chart when the component unmounts
    if (chartInstance.current) {
      chartInstance.current.destroy();
    }
  };
}, [chartData]); // Empty dependency array means this effect runs once, similar to componentDidMount
  return <canvas ref={chartContainer} id="chartContainer" />;
};
export default ChartComponent;
```

ДОДАТОК Е

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Програмний засіб пошуку і агрегації пропозицій з онлайн платформ оренди нерухомості

Тип роботи: магістерська кваліфікаційна робота
(БДР, МКР)

Підрозділ кафедра обчислювальної техніки
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність 90% Схожість 10%

Аналіз звіту подібності (відмітити потрібне):

✓ Запозичення, виявлення у роботі, оформлені коректно і не містять ознак плагіату.

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її виконання автором. Роботу направити на розгляд експертної комісії кафедри.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

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

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

Автор роботи _____ Мартинов П. Г.
(підпис) (прізвище, ініціали)

Керівник роботи _____ Мартинюк Т. Б.
(підпис) (прізвище, ініціали)