Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій

УДК *На правах рукопису*

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: Інтелектуальна система оцінки нерухомості на основі системного аналізу

Спеціальність 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

(позначення)

Студент групи ІПЗ-42
_____Антон ГУНЬКО
(підпис) (розшифровка підпису) (дата)

Науковий керівник
к.т.н., доц. _____ Олексій БИЧКОВ
(посада) (підпис)(розшифровка підпису) (дата)

Консультант з питань нормконтролю
фахіваць _____ Тамара ЧАПОВСЬКА

Допускається до захисту
з питань нормоконтролю

Завідувач кафедри

к.т.н., доц._____ Олексій БИЧКОВ
(посада) (підпис)(розшифровка підпису) (дата)

2021

Рішенням Екзаменаційної комісії

випускна кваліфікаційна робота студента

Гунько Антона

захищена з оцінкою

_____

Голова екзаменаційної комісії

кандидат фізико-математичних наук Бондарчук А.П.

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖУЮ:
Завідувач кафедри програмних систем і технологій
_____
(Олексій БИЧКОВ)
“__”_____ 2021 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ
Гуньку Антону Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема випускної кваліфікаційної бакалаврської роботи “Інтелектуальна система оцінки нерухомості на основі системного аналізу ”
затверджена наказом вищого навчального закладу від „11” листопада 2020 р. № 6

2. Строк здачі студентом закінченої роботи: _____
3. Вихідні дані до роботи:
Теоретичні концепції та формальні моделі побудови оцінки нерухомості на базі системного аналізу.
4. Зміст пояснювальної записки (перелік питань, що їх належить розробити):
1. Аналіз концепцій збору інформації об’єктів нерухомості за допомогою комп’ютерних технологій.
2. Пошук можливостей зображення зібраної інформації на інтерактивній карті.
3. Вивчення категоріальних концепцій в алгоритмах оцінки нерухомості.
4. Формування повного циклу роботи програмного забезпечення

5. Перелік графічного матеріалу (з точним забезпеченням обов’язкових креслень)
Рис 1.1. центральний український реєстр нерухомості
Рис 2.1. Приклад сайту (самостійне відкриття сайту)
Рис 2.2. Структура сайту (пошук необхідних посилань)
Рис 2.3. Приклад сайту (відкриття сайту за доп. Selenium WebDriver)
Рис 2.4. Зібрані сторінки об’яв
Рис 2.5. Структура сайту
Рис 2.6. Результати роботи програми
Рис 2.7. Приклад сторінки (відкриття сторінки за доп. Selenium WebDriver)
Рис 3.1. API
Рис 3.2. Головна сторінка веб-додатку

Консультанти розділів проекту

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Бичков О.С		
2	Бичков О.С		
3	Бичков О.С		

7. **Дата видачі завдання** 13 жовтня 2019 р.

Керівник _____ (Олексій БИЧКОВ) (підпис)

Завдання прийняв до виконання _____ (Антон ГУНЬКО) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір і вивчення літератури	25.10.2020	виконано
2	Пошук можливостей зображення зібраної інформації на інтерактивній карті	10.11.2020	виконано
3	Вивчення категоризаційних концепцій в алгоритмах оцінки нерухомості	30.11.2020	виконано
4	Розробка алгоритмічної моделі	15.12.2020	виконано
5	Опис розробленого алгоритму	25.12.2020	виконано
6	Програмна реалізація методу категоризації	15.01.2021	виконано
7	Затвердження пояснювальної записки роботи завідувачем кафедри	25.01.2021	виконано

Студент – бакалавр _____(Антон ГУНЬКО)
Керівник роботи _____ (Олексій БИЧКОВ)

ЗМІСТ

Жирній убрать

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ПЗ - програмне забезпечення

БД - база даних

API - Application Programming Interface

REST - Representational State Transfer

JS - Java Script

TS - TYPE Script

HTML - Hypertext Markup Language

CSS - Cascading Style Sheets

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 55 с., 16 рис., 4 додат., 8 джерел.

Тема: Інтелектуальна система оцінки нерухомості на основі системного аналізу.

Об’єкт дослідження: оцінка житлової нерухомості що базується на системному аналізі.

Мета роботи: розробка web-додатку, за допомогою якого користувач може побачити всі об’яви об’єктів нерухомості та оцінити власне житлове майно.

Предмет дослідження: алгоритм оцінки вартості житлової нерухомості, взаємодія користувача з додатком.

Результати дослідження:

Досліджено методи алгоритмів оцінки вартості об’єктів нерухомості, що базуються на ситемному аналізі. Створено повноцінний web-додаток з інтерактивною картою.

Висновок

В результаті роботи над дипломним проєктом було розроблено та розгорнуто повноцінну інтелектуальну систему оцінки вартості нерухомості.

WEB-ДОДАТОК, СЕРВЕР, КЛІЄНТ, TYPESCRIPT, JAVASCRIPT, ANGULAR, PYTHON, КЛАСТЕРИЗАЦІЯ, ВЕБ-СКРАПІНГ, CSV.

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 55 с., 16 рис., 4 доп., 8 источника.

Тема: Интеллектуальная система оценки недвижимости на основе системного анализа.

Объект исследования: оценка жилой недвижимости основанный на системном анализе.

Цель работы: разработка web-приложения, с помощью которого пользователь может увидеть все объявления объектов недвижимости и оценить собственное жилое имущество.

Предмет исследования: алгоритм оценки стоимости жилой недвижимости, взаимодействие пользователя с приложением.

Результаты исследования:

Исследованы методы интеллектуального подбора рекомендаций на основе фильтрации содержимого и колаборативних фильтрации. Создано полноценное мобильное приложение, готовое к релизу.

Вывод:

В результате работы над дипломным проектом была разработана и развернута полноценная интеллектуальная система оценки стоимости недвижимости.

WEB-ПРИЛОЖЕНИЕ, СЕРВЕР, КЛИЕНТ, TYPESCRIPT, JAVASCRIPT, ANGULAR, PYTHON, КЛАСТЕРИЗАЦИЯ, ВЕБ-СКРАПИНГ, CSV.

SUMMARY

Final qualifying bachelor's thesis: 55 pages, 16 figures, 5 appendices, 8 sources.

Topic: Intelligent system for realty estimate using/via system analysis

Object of research: assessment of residential properties based on a system analysis.

Purpose: development of a web application where user can see all advertisements for the sale of real estate and estimate the value of his own apartment

Results of the research:

Methods of intelligent selection of recommendations based on content filtering and collaborative filtering have been investigated. A full-fledged mobile application has been created, ready for release

Conclusion

As a result of work on the graduation project, a full-fledged intellectual system for real estate valuation was developed and deployed.

WEB-APPLICATION, SERVER, CLIENT, TYPESCRIPT, JAVASCRIPT, ANGULAR, PYTHON, CLUSTERING, WEB-SCRAPING, CSV.

ВСТУП

Мета і задачі дослідження

Метою бакалаврської роботи є розробка інтелектуальної системи оцінки нерухомості на базі системного аналізу.

Проєкт, що розроблюється повинен включати в себе серверну частину з базою даних та клієнтський додаток, що у свою чергу має поділятися на відтворення даних із БД на інтерактивній карті та форму для заповнення необхідних параметрів житла для подальшої його оцінці. Бажаними вимогами являються швидка робота додатку та серверу та об’єктивна оцінка користувацького майна.

Досягнення мети включало розв’язання таких **задач**:

1) огляд існуючих ресурсів, формування єдиної БД;
2) пошук можливостей відображення даних на інтерактивній карті ;
3) вибір об’єктивного алгоритму оцінки нерухомості та обгрунтування доцільності його використання;
4) реалізація додатку;

Об’єктом дослідження є метод оцінки вартості нерухомості на основі системного аналізу.

Предметом дослідження є система, що надає близьку до об’єктивної оцінку вартості нерухомості та зображує схожі за параметрами квартири.

Методи дослідження

Для розробки сервісу досліджено існуючі рєстри об’яв продажу нерухомості, методи та рішення щодо фомування єдиної бази даних, проаналізовано технології, за допомогою яких можливо зобразити інформацію на інтерактивній карті, їхні позитивні та негативні сторони. Також було досліджено рішення в галузі системного

аналізу, що дозволяє швидко знаходити подібні за параметрами квартири, робити приблизьну оцінку введеної користувачем моделі нерухомості.

Наукова новизна отриманих результатів

Запропоновано рішення, яке може стати зручним інструментом для оцінки вартості власної нерухомості.

Практичне значення одержаних результатів

Одержано веб-додаток, що дозволяє переглянути об’яви продажів нерухомості, зібраних із різних відкритих ресурсів у інтернеті. Також даний додаток дозволяє виконати оцінку майна, використовуючи методи системного аналізу.

Особистий внесок студента

Особистим результатом є:

1. запропонований підхід до формування єдиної бази даних об’єктів нерухомості.
2. повноцінний розроблений веб-додаток, що виконує відображення об’єктів нерухомості на інтерактивній карті та оцінку вартості нерухомості, параметри якого введені користувачем.

Структура та обсяг роботи

Робота викладена на 55 сторінках друкованого тексту, який складається із вступу, трьох розділів, висновків, списку використаних джерел (8 найменування). Робота містить 16 малюнків та 4 додатки.

РОЗДІЛ 1

АНАЛІЗ ІСНУЮЧИХ РЕСУРСІВ ДАНИХ ПРО НЕРУХОМІСТЬ

1.1 Пошук та аналіз ресурсів

Джерела інформації, що використовуються для наповнення бази даних пропозицій на ринку, можуть істотно відрізнятися за структурою даних про виставлені об'єкти нерухомості, використовувати різну термінологію і поняття. Тому з метою подальшої обробки вся зібрана інформація повинна бути приведена до єдиного уніфікованого (стандартизованого) структурованого вигляду:

- по атрибутах опису об'єкта;
- по термінології (з дотриманням недвозначного розуміння наведених понять);
- по форматам зберігання даних.

Саме тому було вирішено зосередитися на двох існуючих методах збору інформації:

1. Використання єдиної централізованої бази даних (якщо таке джерело існує).
2. Самостійний збір інформації та наповнення бази даних, використовуючи програмні засоби.

Вдалося віднайти центральнізований український реєстр об’єктів житловох нерухомостей, що були продані в минулому (http://www.spfu.gov.ua/). Даний реєстр включав в себе .zip архіви проданих житлових нерухомостей України в минулому.

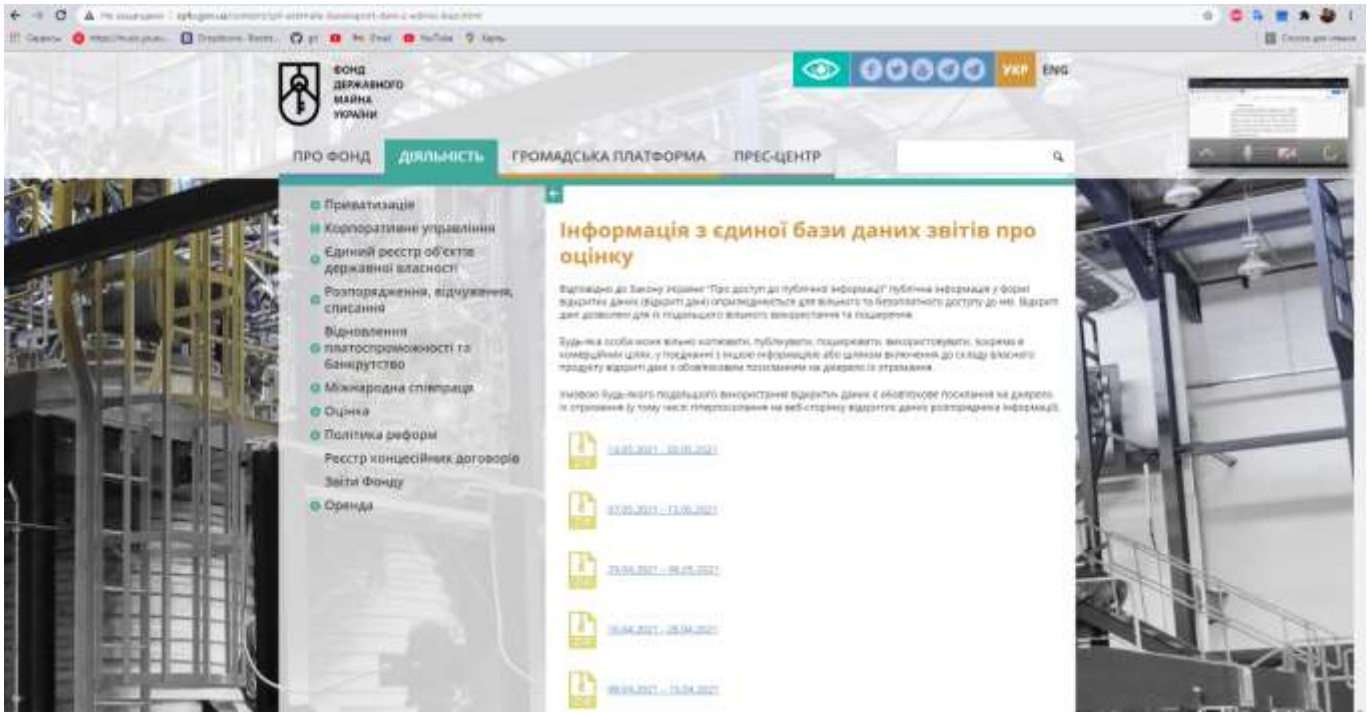

Рис. 1.1. http://www.spfu.gov.ua/ - центральний український реєстр нерухомості

Рєстр включав наступні атрибути

1. Вид об’єкта оцінки.
2. Дата оцінки.
3. Дата реєстрації звіту.
4. Дата складання звіту.
5. Оціночна вартість об’єкта оцінки, грн.
6. Статус звіту.
7. Регіон об’єкта оцінювання.
8. Населений пункт об’єкта оцінювання.
9. Тип вулиці.
10. Вулиця.
11. Вид об’єкта нерухомості.
12. Тип будинку.
13. Клас будинку.
14. Рік введення в експлуатацію.

15. Загальна площа, кв.м.
16. Площа житлових приміщень, кв. м.
17. Площа допоміжних/підсобних приміщень, кв. м.
18. Поверх у будівлі (для квартири)/Поверховість будівлі.
19. Матеріал стін.
20. Матеріал перекриття.
21. Інженерне обладнання.
22. Наявність вбудованого паркінгу.

Даний реєстр відповідає всім вимогам:

1. Відсутність дублікатів.
2. Повнота інформації.
3. Відсутність інформації про неіснуючі об’єкти.
4. Чіткість та однозначність в найменуванні атрибутів об’єкта.
5. Постійне оновлення даних.

Проте, в цьому реєстрі знайшлися 2 великих недоліка, які спричинили до відмови від даного рішення:

1. Відсутність конкретної адреси (назва вулиці + номер будинку + поверх). Даний факт робив би неприпустиму похибку в майбутньому навчанні нейронної мережі, адже, без чіткого розуміння розташування нерухомості складно навіть людині робити припущення щодо її вартості.
2. Інформація була поділена на безліч проміжків дат, коли відбувалися акти продажі нерухомості. Це робило незручну та складноавтоматизовану роботу з цим ресурсом.

Тому, проаналізувавши можливі методи пошуку інформації, було прийнято рішення щодо самостійного збору даних, використовуючи при цьому програмні засоби.

1.2 Висновки до розділу

У даному розділі було показано процесс пошуку інформації, пов’язаної з продажем нерухомості.

Вдалося віднайти централізований український реєстр об’єктів житловох нерухомостей, що були продані в минулому (http://www.spfu.gov.ua/). Проте, в цьому реєстрі знайшлися 2 великих недоліка, які спричинили до відмови від даного рішення. Тому, проаналізувавши можливі методи пошуку інформації, було прийнято рішення щодо самостійного збору даних, використовуючи при цьому програмні засоби.

РОЗДІЛ 2

ФОРМУВАННЯ БАЗИ ДАНИХ ОБ’ЄКТІВ НЕРУХОМОСТІ

2.1 Вибір найбільш ефективного методу формування бази даних з веб-ресурсів

Найпоширенішим способом програмного збору інформації у веб-просторі виявився веб-скрепінг.

Веб-скрепінг (від англ. scraping — «вишкрібання», веб-збирання або витягнення веб-даних) — перетворення у структуровані дані інформації з веб-сторінок, які призначені для перегляду людиною за допомогою браузера.

Як правило, виконується за допомогою комп'ютерних програм, що імітують поведінку людини в інтернеті, або з'єднуючись з веб-сервером напряму по протоколу HTTP, або управляючи повноцінним веб-браузером. Але буває і скрапінг за допомогою копіювання даних людиною. Це форма копіювання, в якій конкретні дані збираються та копіюються з інтернету, як правило, в базу даних або електронну таблицю для подальшого пошуку чи аналізу.

Веб-скрапінг включає в себе завантаження та вилучення. Спочатку завантажується сторінка (що робить браузер, коли ви переглядаєте сторінку), після цього можна здійснювати добувати потрібну інформацію. Зміст сторінки може бути проаналізовано, переформатовано, його дані скопійовані в електронну таблицю тощо. Веб-скрапери, як правило, беруть щось із сторінки, щоб використати це для інших цілей деінде. Прикладом може бути пошук і копіювання імен та телефонних номерів або компаній та їх URL-адрес до списку (контактне сканування).

Існує декілька рішень для скрпаінга веб-сайтів. Серед них:

- Окремі сервіси, які працюють через API або мають веб-інтерфейс (Embedly, DiffBot і ін.).
- Проекти з відкритим кодом, на різних мовах програмування (Goose, Scrapy - Python; Goutte - PHP; Readability, Morph - Ruby).
- Самостійне побудування програми збору інформації (при цьому якість отриманної інформації значно підвищується – ви самі вирішуєте що та як ваша система буде збирати з веб-сторінок)

Веб-сторінки побудовані за допомогою текстових мов розмітки (HTML та XHTML) і часто містять велику кількість корисних даних у текстовій формі. Однак більшість веб-сторінок призначені для кінцевих користувачів, а не для зручності автоматичного використання. Через це були створені набори інструментів, які «збирають» веб-вміст. Веб-скрапери — це прикладний програмний інтерфейс для вилучення даних з веб-сайту.

Існують методи, які деякі веб-сайти використовують для запобігання веб-скрапінгу. Наприклад, виявлення та заборона ботів від сканування (перегляду) своїх сторінок. Виявляти таких ботів сайти можуть по-різному. Однією з основних ознак веб-скрепінгу, яку майже кожен веб-ресурс бере до уваги – надто швидке та багатокількісне надсилання напитів на отримення ресурсів сайту (по-іншому DDoS-атаки). У відповідь на це існують веб-скрапінгові системи, які спираються на використання методів аналізу об'єктної моделі документа, комп'ютерного бачення та обробку тексту природною мовою, що імітує пошук людини, щоб дозволити збирати вміст веб-сторінок для автономного синтаксичного аналізу. Прикладом роботи такої

веб-скрапінгової системи буде будь-яка система, що використовує технологію Selenium (або схожі.

Selenium Webdriver — це популярний інструмент управління браузером, який максимально найближче імітує дії користувача.

Selenium Webdriver безпосередньо викликає команди браузера, використовуючи рідне для кожного конкретного браузера API. Як відбуваються ці виклики і які функції вони виконують залежить від конкретного браузера. В результаті, після запуску програми, де використовується Selenium Webdriver, на комп'ютері користувача відкривается сайт за доступом до будь-якої інформації.

Краулінг

Веб-скрапінг це акт вилучення інформації з сайту. Іноді програмі доводиться переходити від одного посилання до іншого щоб зібрати всю необхідну інформацію - це називається краулінг (crawling). Ми знайдемо цікаві для нас URL і опрацюємо їх дані, що будуть знаходитись за цими адресами

2.2 Реалізація збору інформації

Була створена програма, що спочатку збирає посилання на сторінки реальних об'яв продажу квартир певного веб-сайту. Наступним етапом був перехід за цими посиланнями та збір потрібної інформації. Алгоритм программи:

1. Вибір сайту (веб-ресурсу).
2. Аналіз сайту: знаходження рішень для оптимальної реалізації збору інформації даного сайту.
3. Перехід на сайт.
4. Збір посилань на сторінки об'яв зі списку квартир сайту в окремий txt файл.
5. Вибір наступного сайту (веб-ресурсу) та повернення до кроку 2.

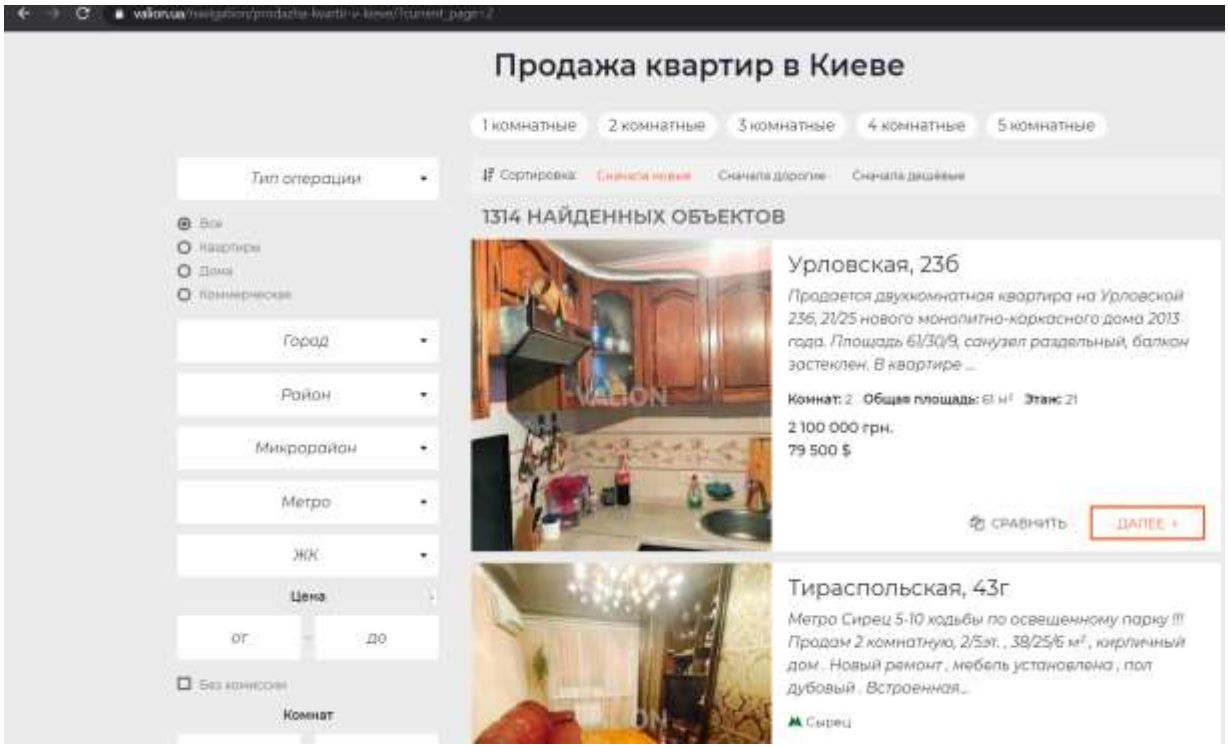

Рис. 2.1. Приклад сайту (самостійне відкриття сайту)

Як видно з рис. 2.1. поточний сайт складався зі сторінок – списків міні-об’яв нерухомості. Якщо натиснути кнопку «Далее», ми перейдемо на сторінку, де саме знаходиться інформація про нерухомість. Отже потрібно виконати краулінг, тобто первинний збір адрес сторінок із сайту, а вже потім взяття інформації з кожної сторінки окремо.

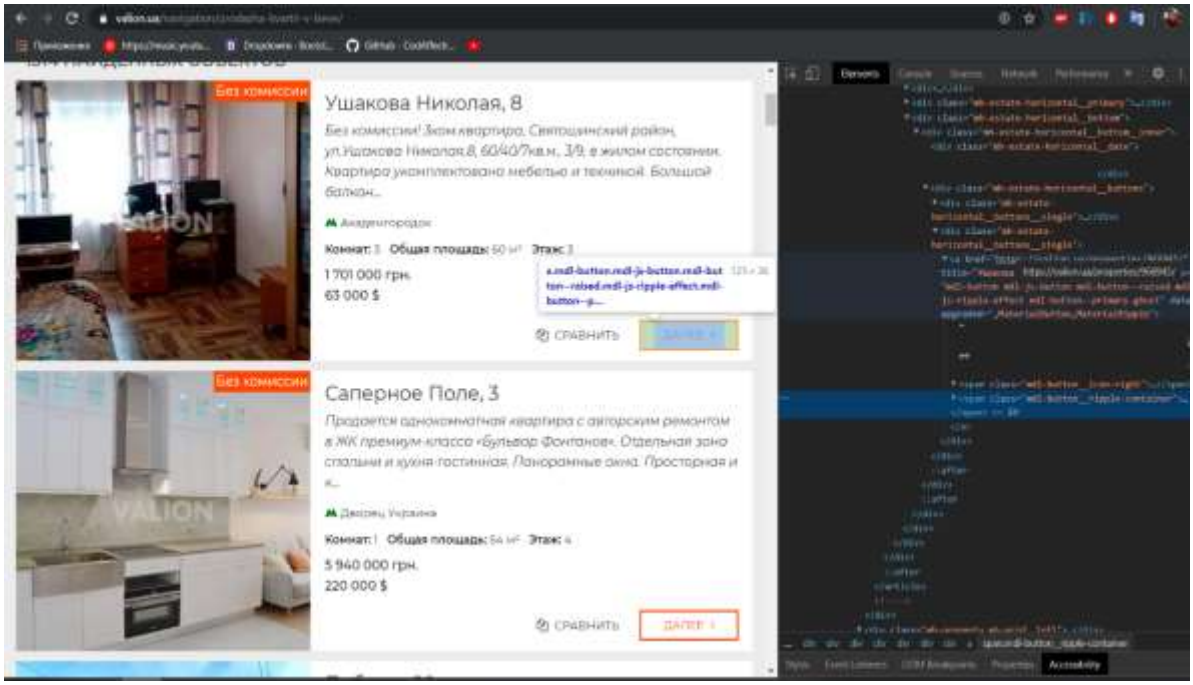

Рис. 2.2. Структура сайту (пошук необхідних посилань).

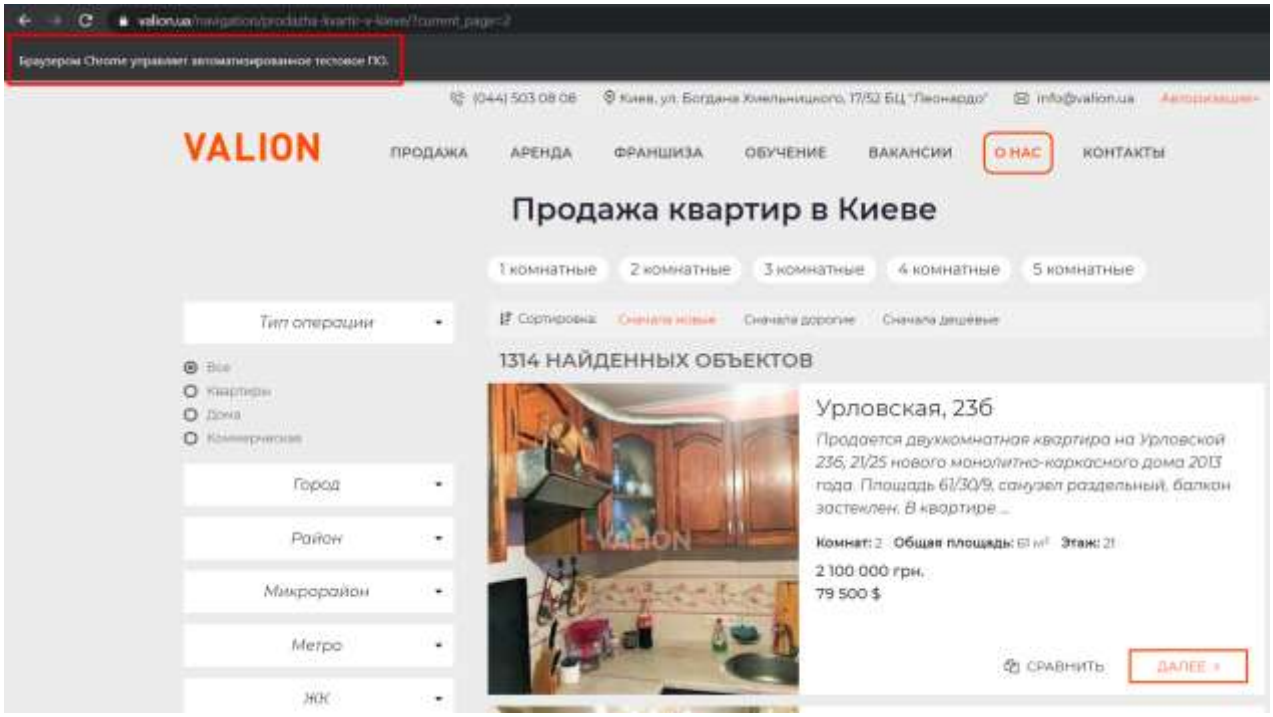

Рис. 2.3. Приклад сайту (відкриття сайту за доп. Selenium WebDriver)

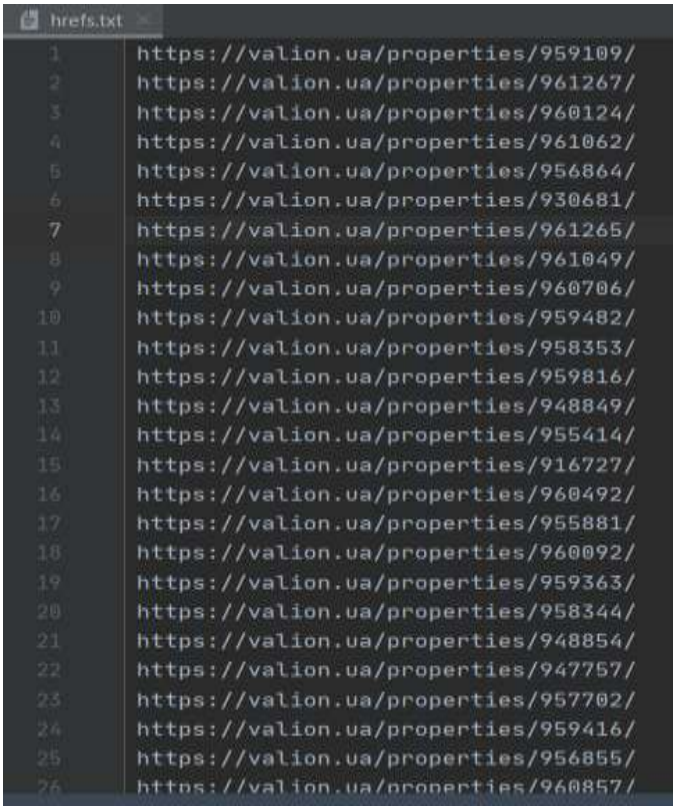

Рис.2.4. Зібрані сторінки об’яв

Вибір формату даних:

Формат CSV є найбільш часто використовуваним форматом імпорту та експорту для баз даних і електронних таблиць.

Файл CSV (значення, розділені комами) дозволяє зберігати дані в табличній структурі з розширенням .csv. CSV-файли широко використовуються в додатках електронної комерції, оскільки їх дуже легко обробляти. Деякі з областей, де вони були використані, включають в себе:

1. імпорт і експорт даних клієнтів
2. імпорт і експорт продукції
3. експорт замовлень
4. експорт аналітичних звітів по електронній комерції

Основна задача розробленого алгоритму програми:

1. Перехід на сторінку об’яви нерухомості.
2. Отримання коду сторінки HTML.
3. Знаходження в структурі коду необхідні блоки з корисною інформацією.
4. Збереження потрібної інформації в локальні змінні. Формування списку значень атрибутів однієї об’яви.
5. Занесення запису до бази даних.
6. Взяття наступного посилання та повернення до п. 1.

Код програми:

Додаток А.

Була використана бібліотека BeautifulSoup, що значно спрощує роботу з текстом, а саме кодом HTML. За допомогою її функціоналу було легко знайти необхідний контейнер з класом та взяти потрібну інформацію з нього.

Створена програма, що збирає наступні характеристичні атрибути зі сторінок продажу нерухомості:

1. Місто.
2. Тип будинку.
3. Район.
4. Мікрорайон
5. Вулиця.
6. Номер будівлі.
7. Кількість кімнат.
8. Найближче метро.
9. Загальна площа.
10. Житлова площа.
11. Площа кухні.

12. Поверх.
13. Кількість поверхів.
14. Ціна продажу.

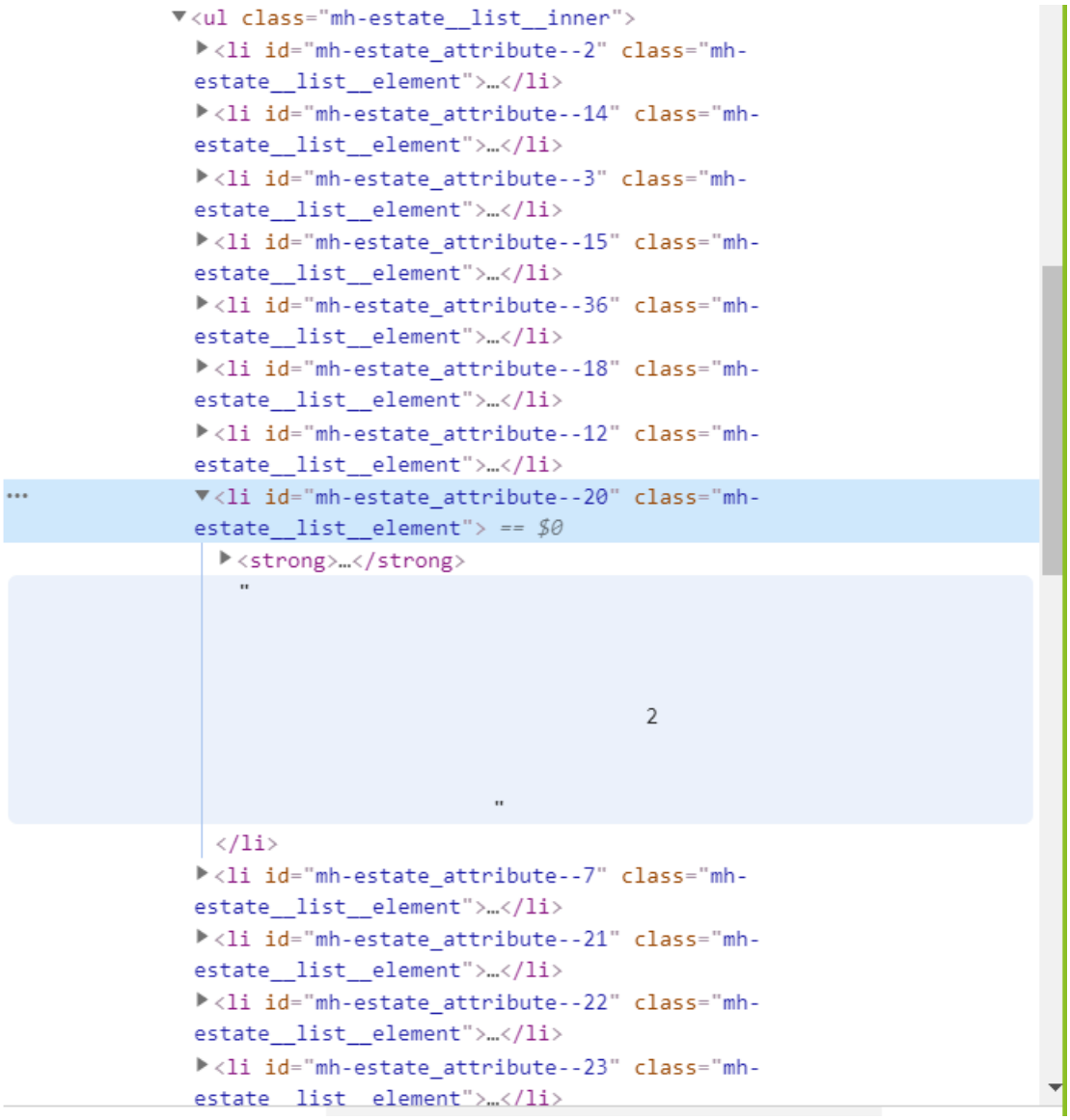

Рис.2.5. Структура сайту

Також проводився підрахунок часу, який потрібен для зчитування даних з однієї сторінки. В середньому це складало 1,5 секунди.

Якщо якась із сторінок не мала достатньої інформації – така сторінка пропускалась на даному этапі (виводилось повідомлення «Something went wrong while reading» + посилання сторінки).

Рис.2.6. Результати роботи програми

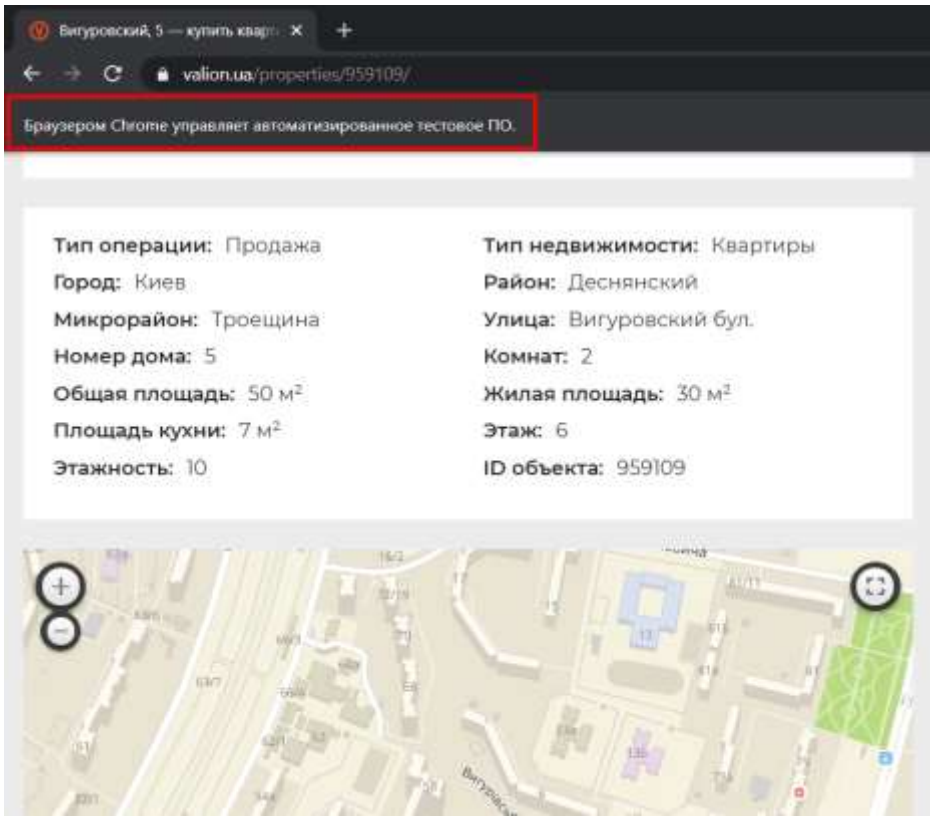

Рис.2.7. Приклад сторінки (відкриття сторінки за доп. Selenium WebDriver)

Проблему наявності повторюваних оголошень можна було легко вирішити, використовуючи бібілотеку Pandas в мові програмування Python.

Pandas - це бібліотека програмного забезпечення Python для обробки та аналізу даних. Робота з даними панд будується на підлозі бібліотеки NumPy, що є інструментом низького рівня. Надає спеціальні структури даних та операції для маніпулювання числовими таблицями та часовими рядами.

Видалення дублікатів записів супроводжувалося збіганням даних в ключових колонках, таких як: «вулиця», «номер будинку», «кількість кімнат», «загальна площа», «поверх».

Код програми:

Додаток Б.

2.3 Висновки до розділу

У розділі була сформована база даних об’яв продажу об’єктів нерухомості у місті Київ методом веб-скрапінгу, що налічувала понад 10тис. записів. Використовувались такі технології або бібліотеки, як **Python HTTP Client**, **Selenium Webdriver, Pandas.** Формат зібраних даних – csv – найбільш зручний для роботи в даній галузі.

Також було проведення видалення дублікатів записів з БД по основним колонкам, а саме: «вулиця», «номер будинку», «кількість кімнат», «загальна площа», «поверх».

РОЗДІЛ 3

РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ

3.1 Реалізація серверної частини додатку

Для створення серверної частини додатку, було прийняте рішення використовувати мову програмування Python.

Python (частина читання - Python, назва передбачена у британському шоу Monty Python) - інтерпретована в добре організованому способі програмування з високою роздільною здатністю із суворим динамічним набором тексту. Зламаний у 1990 році Рочі Гідо ван Россумом. Структурування з високою роздільною здатністю відразу для динамічної обробки та динамічної обробки програмного забезпечення, а також демпферних компонентів компонентів. Python забезпечить модульні та модульні пакети для модульності та повторного переліку коду. Інтерпретатор Python та стандартні бібліотеки бібліотек, доступні для компіляції та перегляду на всіх основних платформах. Нове програмування на Python прийняло низку парадигм програмування, включаючи об’єктно-орієнтовані, процедурні, функціональні та аспектно-орієнтовані.

Серед основних переваг цієї мови програмування можна назвати наступні:

- чистий синтаксис (для виділення блоків слід використовувати відступи)
- переносимість програм (що характерно для більшості інтерпретованих мов)
- стандартний дистрибутив має велику кількість корисних модулів (включаючи модуль для розробки графічного інтерфейсу)
- можливість використання Python в Інтернеті (дуже корисно для експериментів та вирішення простих задач)

• стандартний дистрибутив має просте, але в той же час досить потужне середовище розробки, яке називається IDLE і яке написане на Python;

• зручний для розв'язування математичних задач (має засоби для роботи з комплексними числами, може працювати з цілими числами довільного розміру, в діалоговому режимі може використовуватися як потужний калькулятор)

• відкритий код (можливість редагувати його іншими користувачами).

Стандартна бібліотека Python дуже обширна і пропонує широкий спектр можливостей. Бібліотека містить вбудовані модулі (написані на мові C), які забезпечують доступ до функціональних можливостей системи, таких як введення / виведення файлів, які інакше були б недоступні програмістам Python, а також модулі, написані на Python, що забезпечують стандартизовані рішення для багатьох проблем, що виникають у щоденне програмування. Деякі з цих модулів явно розроблені для заохочення та підвищення портативності програм Python шляхом абстрагування специфіки платформи в нейтральні для платформи API.

Інсталятори Python для платформи Windows зазвичай включають всю стандартну бібліотеку і часто також містять багато додаткових компонентів. Для Unix-подібних операційних систем Python, як правило, надається як колекція пакетів, тому може знадобитися використовувати пакувальні засоби, що постачаються з операційною системою, для отримання деяких або всіх додаткових компонентів.

На додаток до стандартної бібліотеки, зростає колекція з декількох тисяч компонентів (від окремих програм та модулів до пакетів та цілих платформ розробки додатків), доступних з індексу пакетів Python.

Інтерпретатор Python та багату стандартну бібліотеку (як джерела, так і двійкові дистрибутиви для всіх основних операційних систем) можна отримати на веб-сайті Python www.python.org і можна безкоштовно розповсюджувати. Цей же сайт має

дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор мови Python може бути розширений функціями та типами даних, розробленими на C або C ++ (або іншою мовою, яку можна викликати з C). Python також зручна як мова розширення для додатків, що вимагають подальшого налагодження.

Можливості мови програмування Python:

- Інтерактивний режим

У режимі відладки інтерпретатора Python має інтерактивний режим, при якому висловлюються, вводяться з клавіатурою, що не використовується, і результат відображається на екрані. Цей режим цікавий не тільки новинками, але і досвідченими програмами, які можуть бути в режимі онлайн протестувати будь-який фрагмент коди, перш ніж використовувати його в основній програмі, або просто використовувати його як калькулятор з більшим набором функцій.

- Об'єктно-орієнтоване програмування

Дизайн мови Python побудований на об'єктно-орієнтованій моделі програмування. Реалізація ООП у Python елегантна, потужна і добре продумана, але в цей час досить специфічна за порівнянням з іншими об'єктно-орієнтованими мовами.

- Функціональне програмування

Python підтримує парадигму функціонального програмування, зокрема:

1. функція є об'єктом;
2. функції вищих порядків;
3. рекурсія;
4. розвинена обробка списків (спискові вирази, операції над послідовностями, ітератори);
5. аналог замикань (closures);

6. часткове застосування функції;
7. можливість реалізації інших засобів на самій мові (наприклад, каррінг).

- Модулі та пакети

Програмне забезпечення (застосунок або бібліотека) на Python оформлюється у вигляді модулів, які у свою чергу можуть бути зібрані в пакунки. Модулі можуть розташовуватися як у каталогах, так і в ZIP-архівах. Модулі можуть бути двох типів за своїм походженням: модулі, написані на «чистому» Python, і модулі розширення (extension modules), написані на інших мовах програмування. Наприклад, в стандартній бібліотеці є «чистий» модуль pickle і його аналог на Ci: cPickle. Модуль оформляється у вигляді окремого файлу, а пакет — у вигляді окремого каталогу. Підключення модуля до програми здійснюється оператором import. Після імпорту модуль представлений окремим об'єктом, що дає доступ до простору імен модуля. У ході виконання програми модуль можна перезавантажити функцією reload().

- Інтроспекція

Python підтримує повну інтроспекцію часу виконання. Це означає, що для будь-якого об'єкта можна отримати всю інформацію про його внутрішню структуру.

Застосування інтроспекції (метапрограмування) є важливою частиною того, що називають «pythonic style», і широко застосовується в бібліотеках і фреймворках Python, таких як PyRO, Pyro[en], PLY, CherryPy, Django та інших, заощаджуючи час програміста, що ними користується.

При розробці серверної частини системи, було створено API, що дозволяло діставати дані з базі даних, використовуючи бібліотеку Flask.

API - це абревіатура від Application Programming Interface (Інтерфейс програмування програм), який є посередником програмного забезпечення, що дозволяє двом додаткам спілкуватися між собою. Кожного разу, коли

використовується такий додаток, як наприклад Facebook, надсилається миттєве повідомлення або перевіряється погода на телефоні, ви використовуєте API.

Протягом багатьох років те, що таке «API», часто описує будь-який загальний інтерфейс підключення до програми. Однак зовсім недавно сучасний API набув деяких характеристик, які роблять їх надзвичайно цінними та корисними:

Сучасні API дотримуються стандартів (як правило, HTTP та REST), зручних для розробників, легкодоступних та широко зрозумілих

- До них ставляться більше як до продуктів, ніж до коду. Вони розроблені для споживання для конкретної аудиторії (наприклад, для розробників мобільних пристроїв), вони задокументовані та створені таким чином, щоб користувачі могли мати певні очікування щодо його обслуговування та життєвого циклу.
- Оскільки вони набагато стандартизовані, вони мають набагато сильнішу дисципліну щодо безпеки та управління, а також контролюються та управляються за результатами та масштабами
- Як і будь-яка інша частина виробничого програмного забезпечення, сучасний API має власний життєвий цикл розробки програмного забезпечення (SDLC) проектування, тестування, побудови, управління та управління версіями. Також сучасні API добре задокументовані для споживання та встановлення версій.

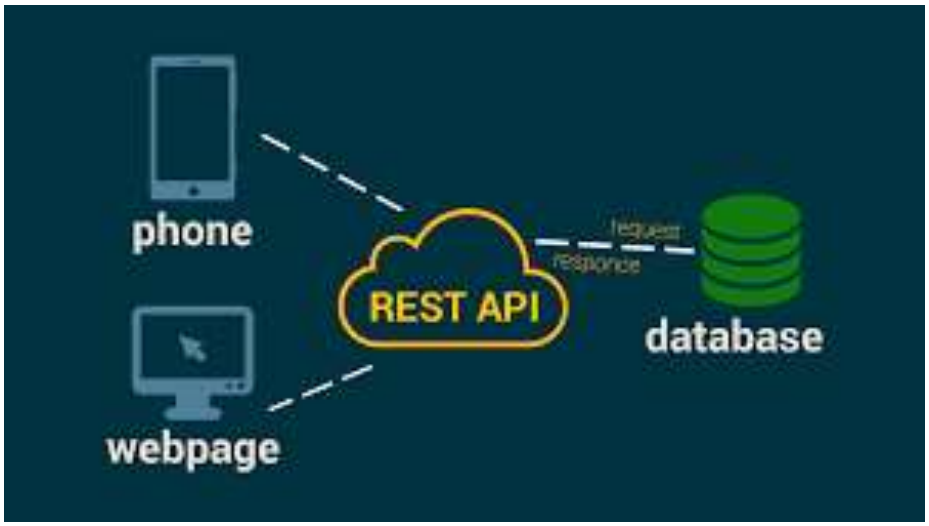

Рис.3.1. API

Властивості використаної бібліотеки Flask:

- Містить сервер для розробки та відлагоджувач
- Вбудована підтримка юніт-тестів
- Управління запитами RESTful
- Використовує шаблони Jinja2
- Має підтримку безпечних куків (сесії на стороні клієнта)
- 100% відповідність WSGI 1.0
- Підтримка Unicode
- Докладна документація
- Сумісність з Google App Engine
- Наявність розширень для забезпечення бажаної поведінки

Код програми:

Додаток В.

Методи API:

1. /allBuildings – повертає всі записи з бази даних об’єктів нерухомості, відфільтрувавши по поточним координатам інтерактивної карти.
2. /calculateAnalitics – повертає найбільш подібні до моделі квартири користувача записи з бази даних об’єктів нерухомості

3.2 Реалізація кліентської частини додатку

Для створення клієнтської частини додатку, було прийняте рішення використовувати фреймворк Angular та мову програмування Typescript.

Angular - це фреймворк JavaScript з відкритим кодом. Призначений для розробки односторінкових додатків. Його мета - розширити браузерні програми на основі MVC та спростити тестування та розробку.

Фреймворк працює з HTML, який містить додаткові атрибути, які описуються директивами, і прив'язує вхід або вихід області сторінки до моделі, що є звичайними змінними JavaScript. Значення цих змінних встановлюються вручну або отримуються із статичних або динамічних даних JSON.

Angular розроблений з переконанням, що декларативне програмування найкраще підходить для побудови користувацьких інтерфейсів та опису програмних компонентів, тоді як імперативне програмування чудово підходить для опису ділової логіки. Фреймворк адаптує та розширює традиційний HTML, забезпечуючи двостороннє прив'язку даних для динамічного вмісту, дозволяючи моделі та поданню синхронізуватися автоматично. Як результат, Angular зменшує роль маніпуляції DOM та покращує учасника тестування.

Цілі розробки:

- Відділення DOM-маніпуляції від логіки додатки, що покращує тестованих коду.
- Ставлення до тестування як до важливої частини розробки. Складність тестування безпосередньо залежить від структурованості коду.
- Поділ клієнтської і серверної сторони, що дозволяє вести розробку паралельно.
- Проведення розробника через весь шлях створення програми: від проектування призначеного для користувача інтерфейсу, через написання бізнес-логіки, до тестування

Порівняння з Backbone.js:

Схожими можливостями володіє Backbone.js - JavaScript-бібліотека, заснована на шаблоні проектування Model-View-Presenter (MVP), призначена для розробки веб-

додатків з підтримкою RESTful JSON інтерфейсу. Backbone - дуже легка бібліотека (упакована і gzip-стисла за величиною ~ 6.3 КБ), але для роботи необхідна бібліотека Underscore.js, а для підтримки REST API і роботи з DOM елементами рекомендується підключити jQuery-подібну бібліотеку: jQuery або Zepto. Backbone.js створений Джеремі Ашкенасом, який відомий також як творець CoffeeScript.

Однак, є і суттєві відмінності:

- Зв'язування даних

Найбільш характерною особливістю, яка розділяє бібліотеки, є спосіб синхронізації моделі та подання. У той час як AngularJS підтримує двостороннє зв'язування даних, Backbone.js, щоб зв'язати модель і уявлення, в значній мірі спирається на шаблонний код.

- REST

Backbone.js добре підтримує RESTful-бекенда. У AngularJS також дуже легко працювати з RESTful API за допомогою сервісу $ resource. У той же час в AngularJS є більш гнучкий сервіс $ http, який підключається до віддалених серверів за допомогою браузерного об'єкта XMLHttpRequest або через JSONP.

- Шаблони

Як шаблон Angular використовує комбінацію настроюються HTML-тегів і виразів. Backbone.js використовує різні шаблонизатор, такі як Underscore.js.

TypeScript - це надмножина JavaScript, тобто, будь-який код на JS є правильним з точки зору TypeScript. Однак, TypeScript володіє деякими додатковими можливостями, які не входять до JavaScript. Серед них - строга типізація (тобто, вказівка типу змінної при її оголошенні, що дозволяє зробити поведінку коду більш передбачуваним і спростити налагодження), механізми об'єктно-орієнтованого програмування і багато іншого. Браузери не підтримують TypeScript безпосередньо, тому код на TS треба транспілювать в JavaScript.

Для побудови самої карти найдоцільнішим методом стало використання плагіну AGM – Angular Google Maps (раніше відомий як angular2-google-maps)

Його будова складається з 4 пакетів:

- @ agm / core

Основний модуль angular-google-maps. Містить усі директиви / послуги / труби основного модуля. Щоб використовувати даний пакет, треба додати AgmCoreModule.forRoot () у своєму модулі програми.

- @ agm / snazzy-info-window

Надає рішення для стильових / настроюваних інформаційних вікон за допомогою «Snazzy Info Window».

- @ agm / markerclusterer Кластеризовані маркери з markerclustererplus

Пакет використовує markerclustererplus для додавання підтримки кластеризації до AGM.

- @ agm / drawing

Пакет додає підтримку малювання до AGM.

В результаті був отриманий веб додаток з необхідними для аналізу полями та картою:

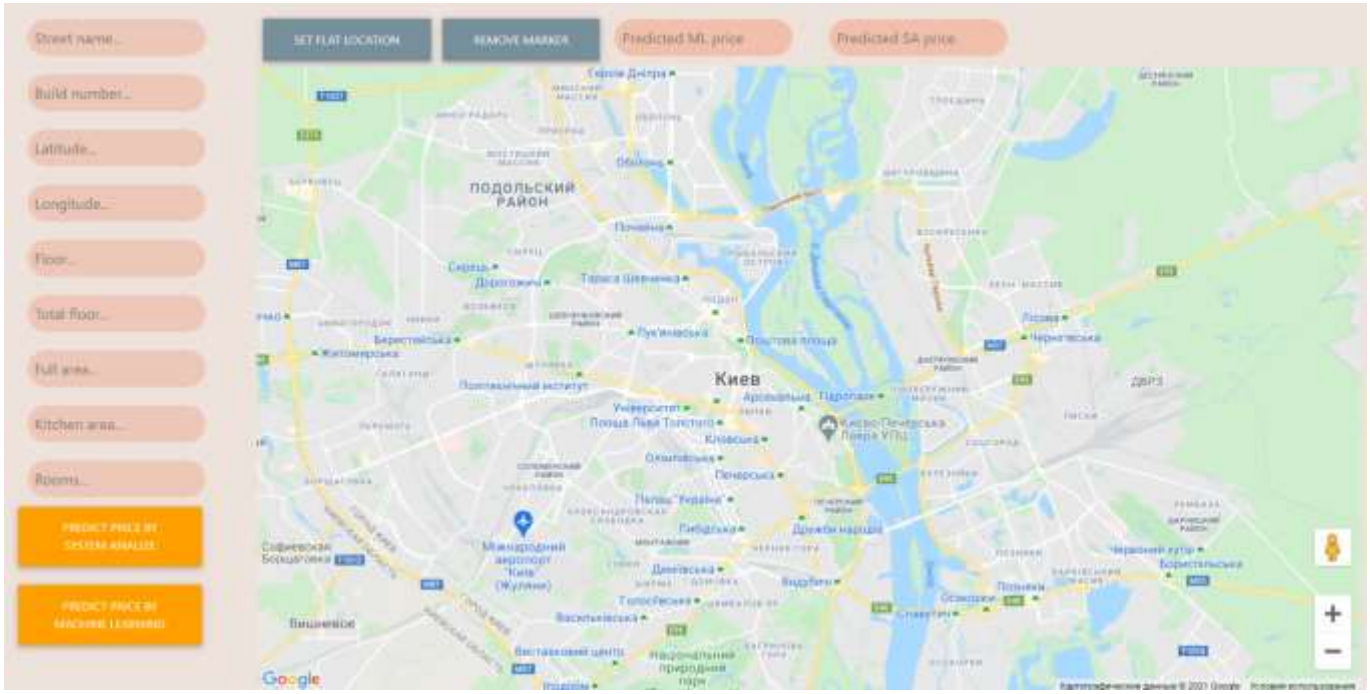

Рис.3.2. Головна сторінка веб-додатку

Код програми:

Додаток Г

Для покращення інтерфейсу карти під час зображення на ній точок об’єктів нерухомості було прийняте рішення щодо кластеризації маркерів за принципом їх місцезнаходження.

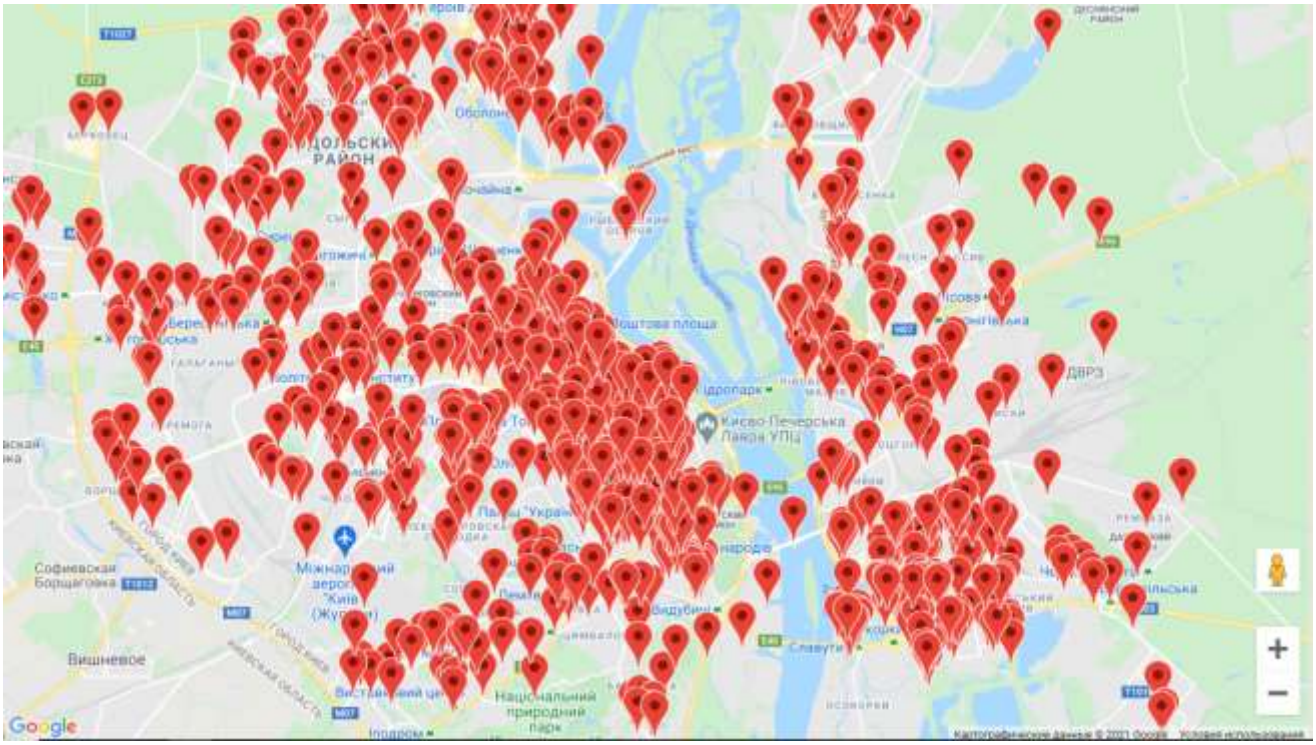

Рис.3.3. сторінка карти до кластеризації

Рис. 3.4. Сторінка карти після кластеризації

При наведенні курсору на один із маркерів, можна побачити вартісь квартири та оригінальне посилання на продаж об’єкту нерухомості.

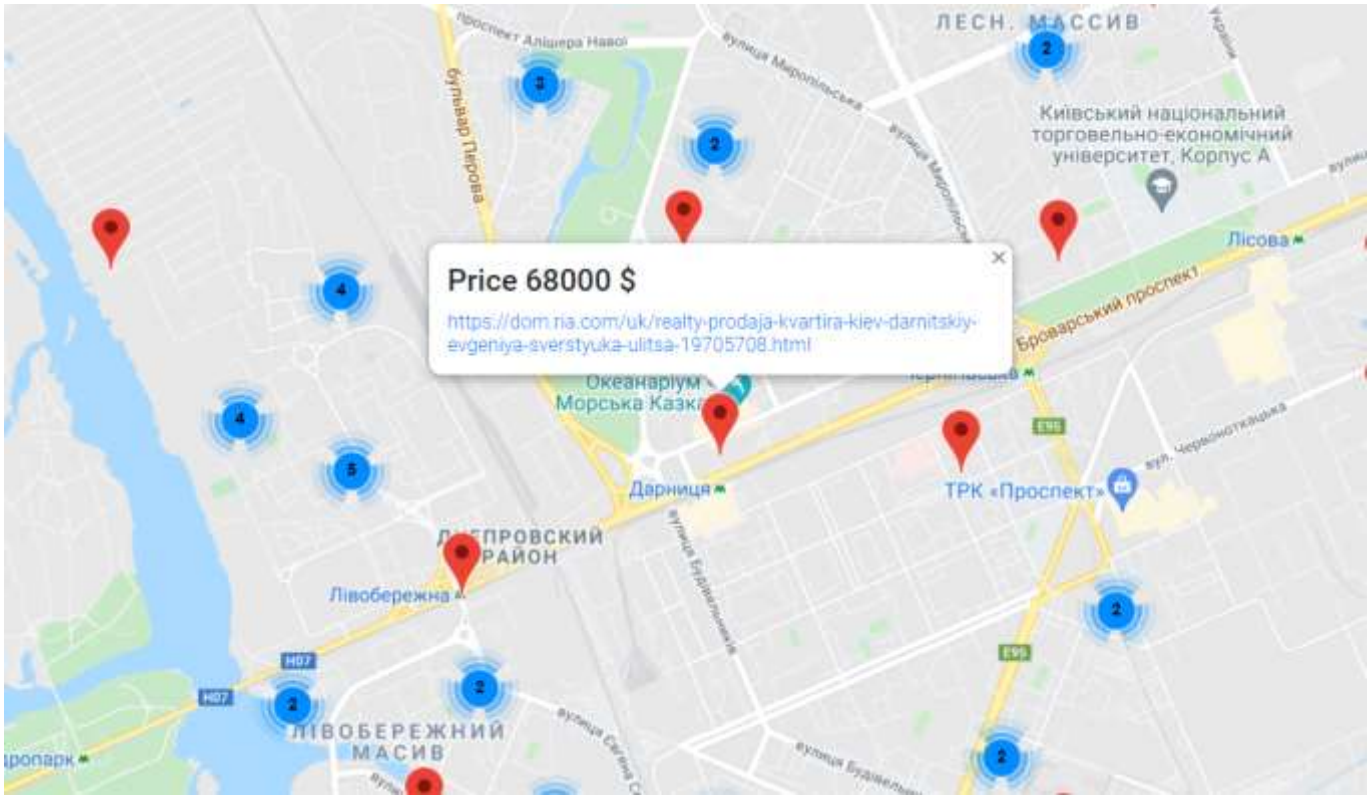

Рис. 3.5. Маркер

Також за допомогою кнопки «Set flat location» можна зафіксувати координати користувацької квартири, а деякі поля заповняться автоматично

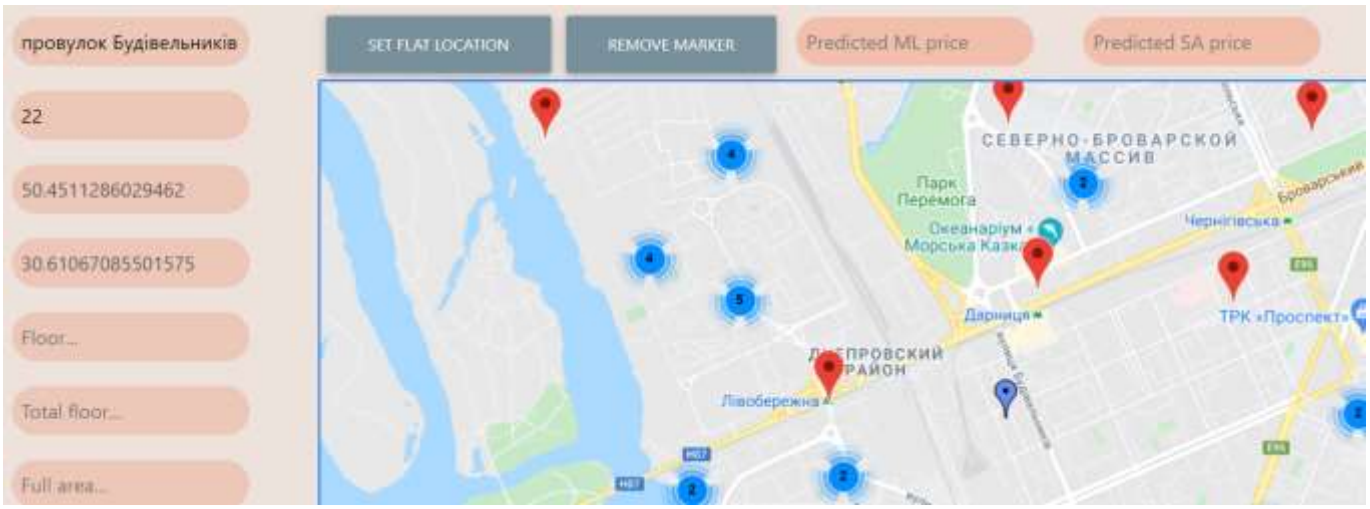

Рис. 3.6. Координати квартири

Під час того, як користувач вводить необхідні поля та натискає на кнопку «Predict price by System Analyze», формується запит з моделлю квартири та передається на сервер. Там вже створюється новий запит до бази даних, фільтрами якого є:

- загальна площа квартири становить +- 10 м² від моделі користувача
- квартира знаходиться в радіусі 1.5 км від моделі користувача

Після чого отримується перелік квартир, що є подібними до моделі.
Наступним кроком є виділення квартилей серед отриманих квартир за їх ціною.
Наприкад, серед отриманих 20 подібних квартир, що відфільтровані за ціною, будуть грати роль лише квартири з порядковими номерами від 5 до 15. Далі уже береться середньє значення цін даних квартир та зображується користувачеві.

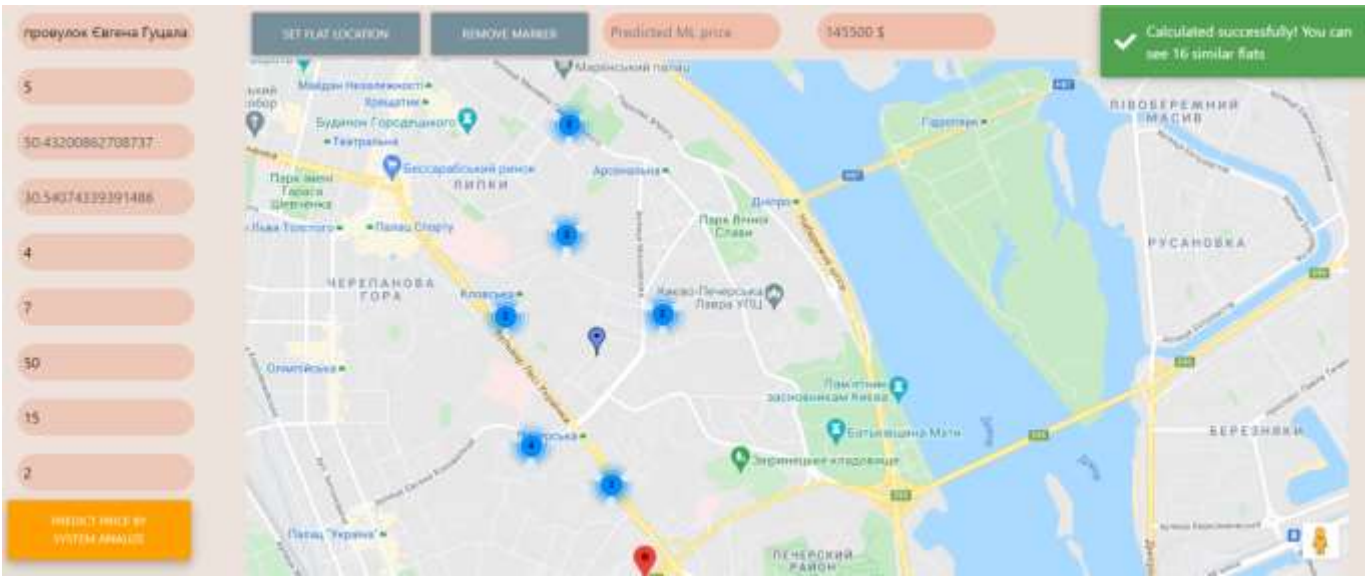

Рис.3.7. Отримання результату оцінки системного аналізу квартири

Також, у разі якщо користувачем була введена модель квартири, подібних до якої немає у БД – він отримає відповідне повідомлення:

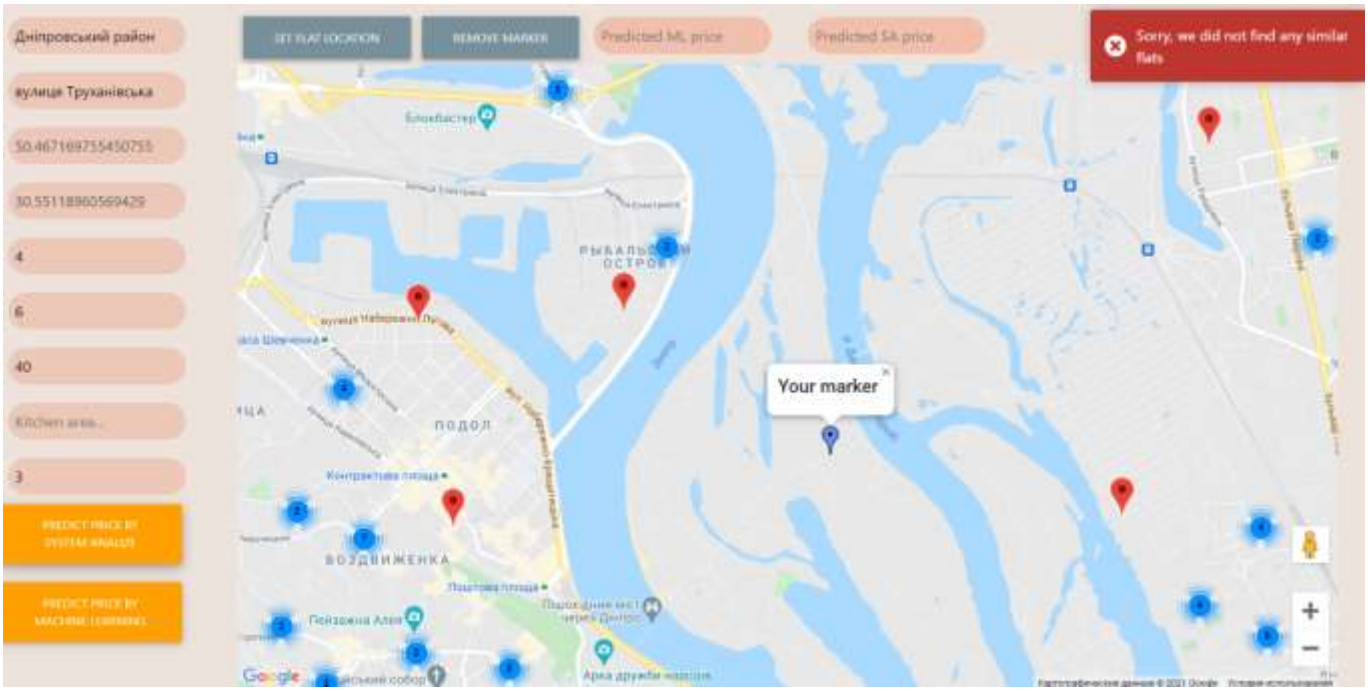

Рис.3.8. Отримання повідомлення про відсутність подібних квартир

3.3 Висновки до розділу

У даному розділі було показано реалізацію серверної та клієнтської частини системи. Використовувались такі технології: серверна частина – Python, Flask, Pandas, клієнтська частина – Angular, Google maps.

На серверній частині було розроблено API, що дозволяло клієнтській частині робити запити та отримувати на них відповіді. В API було розроблено 2 методи: allBuildings – повернення усіх записи з бази даних об’єктів нерухомості, відфільтруваних по поточним координатам інтерактивної карти, /calculateAnalitics – повернення найбільш подібних до моделі квартири користувача записів з бази даних об’єктів нерухомості. Відповідно клієнтська частина полягала у відображенні інтерактивної карти, маркерів на ній, та бокової частини, що стосувалася моделі квартири користувача.

ВИСНОВКИ

Загалом вся робота поділялась на 2 частини:

- формування бази даних
- реалізація веб-додатку

Формування бази даних виконувалось методом веб-скрапінгу, використовуючи технологію Selenium Webdriver. Кінцева версія БД налічувала понад 10тис. записів, збережених у форматі csv.

Реалізація додатку полягала у створенні серверної та клієнтської частини системи. Використовувались такі технології: серверна частина – Python, Flask, Pandas, клієнтська частина – Angular, Google maps.

На серверній частині було розроблено API, що дозволяло клієнтській частині робити запити та отримувати на них відповіді. В API було розроблено 2 методи: /allBuildings – повернення усіх записи з бази даних об’єктів нерухомості, відфільтруваних по поточним координатам інтерактивної карти, /calculateAnalitics – повернення найбільш подібних до моделі квартири користувача записів з бази даних об’єктів нерухомості. Відповідно клієнтська частина полягала у відображенні інтерактивної карти, маркерів на ній, та бокової частини, що стосувалася моделі квартири користувача.

Загалом додаток працює швидко та безперебійно.

Система готова до впровадження у роботу організації. Наразі проходить тестове впровадження у підприємстві «ВЕРІТЕКС».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. https://developer.mozilla.org Офіційна документація JavaScript [Електронний ресурс] – Режим доступу.
2. https://uk.wikipedia.org/wiki/Python Офіційна документація Python [Електронний ресурс] – Режим доступу.
3. https://angular.io/ Офіційна документація Angular [Електронний ресурс] Режим доступу.
4. https://redux.js.org Офіційна документація Redux [Електронний ресурс] Режим доступу.
5. https://habr.com/ Спільнота IT-спеціалістів [Електронний ресурс] – Режим доступу.
6. https://metanit.com/ Сайт про програмування [Електронний ресурс] – Режим доступу.
7. https://stackoverflow.com/ Спільнота IT-спеціалістів [Електронний ресурс] Режим доступу.
8. https://developers.google.com/maps/ Офіційна документація Google-Map

ДОДАТКИ

Додаток А Приклад копіювання інформації у локальну БД

```
def ReadDataOfApartments ():
    csvFile = open("fullDataset.csv", "w")
    csvAttributes = "href,city,buildingType,district,microDistrict,street,buildNumber,rooms," \
                    "metro,fullSquare,livingSquare,kitchenSquare,floor,floorCount,price"
    csvFile.write(csvAttributes + '\n')
    hrefsFile = open("hrefs.txt", "r")

    hrefs = hrefsFile.read().split('\n')
    for href in hrefs:
        try:
            start_time = datetime.now()

            driver.get(href)
            html = driver.page_source
            soup = BeautifulSoup(html.replace('  ', '').replace('\n', ''), features="lxml")
            attributeList = soup.find('ul', {'class': 'mh-estate__list__inner'})
            city = attributeList.find('li', {'id': 'mh-estate_attribute--3'}).contents[1]
            buildingType = attributeList.find('li', {'id': 'mh-estate_attribute--14'})
             .contents[1]
                    district = attributeList.find('li', {'id': 'mh-estate_attribute--15'}).contents[1]
                    microDistrict_attr = attributeList.find('li', {'id': 'mh-estate_attribute--36'})
             .contents[1]
                    street = attributeList.find('li', {'id': 'mh-estate_attribute--18'}).contents[1]
                    buildNumber = attributeList.find('li', {'id': 'mh-estate_attribute--12'})
             .contents[1]
                    rooms = attributeList.find('li', {'id': 'mh-estate_attribute--20'})
             .contents[1]
                    metro_attr = attributeList.find('li', {'id': 'mh-estate_attribute--16'})
             .contents[1]
                    fullSquare = attributeList.find('li', {'id': 'mh-estate_attribute--
             7'}).contents[1].replace('м²', '')
                    livingSquare = attributeList.find('li', {'id': 'mh-estate_attribute--
             21'}).contents[1].replace('м²', '')
                    kitchenSquare = attributeList.find('li', {'id': 'mh-estate_attribute--
             22'}).contents[1].replace('м²', '')
                    floor = attributeList.find('li', {'id': 'mh-estate_attribute--23'}).contents[1]
                    floorCount = attributeList.find('li', {'id': 'mh-estate_attribute--24'})
             .contents[1]
                    price = soup.find('div', {'class': 'mh-
             estate__details__price__single'}).contents[0].replace('грн.', '') \
                        .replace(' ', '')
```

```
            list_attr = [href, city, buildingType, district, microDistrict, street, buildNumber, rooms,
        metro,
                    fullSquare,
                    livingSquare, kitchenSquare, floor, floorCount, price]
            csvFile.write("\n" + ','.join(list_attr))

            print("Time spent while reading " + href + ": " + str(datetime.now() - start_time))
        except:
            print("Something went wrong while reading: " + href)

    csvFile.close()
    hrefsFile.close()
```

Приклад взяття інформації зі структури одного сайту.

Додаток Б Приклад видалення дублікатів по ключовим колонкам

```
import pandas as pd

df = pd.read_csv("DataKievAnalizeInclude.csv", encoding='windows-1251')
df.drop_duplicates('street,buildNumber,rooms,fullSquare,floor', keep='1')
df.to_csv("fullDataset.csv")
```

Приклад видалення дублікатів у базі даних

Додаток В Приклад реалізації серверного API

```
import pickle
from flask import Flask, request
from pandas.core.frame import DataFrame
from flask_cors import CORS, cross_origin
import pandas as pd

app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'

dataKiev = pd.read_csv("../DataKievAnalizeInclude.csv", nrows=2000)


@app.route('/allBuildings', methods = ['POST'])
@cross_origin()
def get_list():
```

```
  frame = request.get_json()['frame']
  datas = dataKiev[(frame['left'] < dataKiev['lon']) & (dataKiev['lon'] < frame['right']
)  & (frame['down'] < dataKiev['lat']) & (dataKiev['lat'] < frame['up'])]

  return datas.to_json(orient="records")

@app.route('/calculateAnalitics', methods = ['POST'])
@cross_origin()
def calculateAnalitics():
  model = request.get_json()['model']
  data = dataKiev[(abs(float(model["lat"]) - dataKiev.lat) + abs(float(model["lon"]) - d
ataKiev.lon) < 0.02) &
    (abs(float(model["fullArea"]) - dataKiev.area) < 20)]
  count = data.count()
  data = data[(data.price < data.price.quantile(.85)) & (data.price > data.price.quantil
e(.15))]
  response = {
    "collection": data.to_json(orient="records"),
    "fullCount": count
  }
  return data.to_json(orient="records")


if __name__ == '__main__':
    app.run(debug=True)
```

Додаток Г Приклад реалізації клієнтського додатку

app.module.ts

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { MapComponent } from './map/map.component';
import { InfoBarComponent } from './info-bar/info-bar.component';
import { AgmCoreModule } from '@agm/core';
import {HttpClientModule} from '@angular/common/http';
import { MDBBootstrapModule } from 'angular-bootstrap-md';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { BackService } from './shared/back.service';
import { AgmJsMarkerClustererModule } from '@agm/js-marker-clusterer';
import { FormsModule } from '@angular/forms';
import { ToastrModule } from 'ngx-toastr';
```

```
@NgModule({
  declarations: [
    AppComponent,
    MapComponent,
    InfoBarComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    AgmCoreModule.forRoot({
      apiKey: ‘’
    }),
    HttpClientModule,
    MDBBootstrapModule.forRoot(),
    BrowserAnimationsModule,
    AgmJsMarkerClustererModule,
    FormsModule,
    BrowserAnimationsModule,
    ToastrModule.forRoot()

  ],
  providers: [BackService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.less']
})
export class AppComponent {
  title = 'Front';
}
```

app.component.html

```
<section class = "parent">
    <tr>
        <td class="childBar">
            <app-info-bar></app-info-bar>
        </td>
        <td class="childMap">
```

```
            <app-map></app-map>
        </td>
    </tr>
</section>
```

app.component.less

```
* {
  background-color: #eee2dc;
}
.childBar {
  width: 280px;
}
.childMap {
  width: 1200px;
  height: min-content;
}
```

back.service.ts

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { BuildModel } from '../models/BuildModel';
import { MapFrame } from '../models/MapFrame';
import { Marker } from '../models/Marker';

@Injectable({
  providedIn: 'root'
})
export class BackService {

  readonly serverUrl: string = "http://127.0.0.1:5000/";
  constructor(private http: HttpClient) {

  }
  populateMarkers(frame: MapFrame) {
    return this.http.post<Marker[]>(this.serverUrl + "/allBuildings", {
      frame: frame
    });
  }
  calculateAnalitics(model: BuildModel) {
    return this.http.post<any>(this.serverUrl + "/calculateAnalitics", {
      model: model
    });
  }
  calculateML(model: BuildModel) {
    return this.http.post<any>(this.serverUrl + "/calculateML", {
```

```
      model: model
    });
  }
```

dataGateway.service.ts

```
import { Injectable } from '@angular/core';
import { BuildModel } from '../models/BuildModel';
import { Marker } from '../models/Marker';
@Injectable({
  providedIn: 'root'
})
export class DataGatewayService {
  buildModel: BuildModel = {};
  markers: Marker[] = [];
  modelPredicted: boolean = false;
constructor() { }

}
```

Marker.ts

```
export interface Marker {
    url?:string,
    lat: number,
    lon: number
    price? : number,
    markerAdded?: boolean,
    iconUrl: string,
    predictablePrice?: string;
}
```

MapFrame.ts

```
export interface MapFrame {
    up: number,
    down: number,
    left: number,
    right: number
}
```

BuildModel.ts

```
export interface BuildModel {
    lat?: string,
    lon?: string
    price? : number,
```

```
    streetName?: string,
    buildNumber?: string
    total_floor?: string,
    floor?: string,
    area?: string,
    rooms?: string,
    kitchen_area?:string,
    predictedML?: string,
    predictedSA?: string
}
```

Map.component.html

```
<button
    id = "btnSetLocation"
    mdbBtn type="button"
    color="blue-grey"
    (click)="onSetLocation()" mdbWavesEffect>Set flat location</button>
<button mdbBtn type="button" color="blue-
grey" (click)="onRemoveAddedMarker()" mdbWavesEffect>Remove marker</button>
<input type="text" id = "lat" disabled = "true" placeholder="Predicted ML price" autocom
plete="off" value="{{this.dataGateway.buildModel.predictedML}}"/>
<input type="text" id = "lat" disabled = "true" placeholder="Predicted SA price" autocom
plete="off" value="{{this.dataGateway.buildModel.predictedSA}}"/>

<agm-map id = "map"
    (boundsChange)="onBoundsChange($event)"
    (zoomChange)="onZoomChange($event)"
    (mapClick) = "addMarker($event)"
    [latitude]="lat"
    [longitude]="lng"
    [zoom]="zoom">
        <agm-marker-cluster imagePath="https://raw.githubusercontent.com/googlemaps/v3-
utility-library/master/markerclustererplus/images/m"> -->
            <agm-marker
            *ngFor="let m of this.dataGateway.markers; let i = index"
            [latitude]="m.lat"
            [longitude]="m.lon"
            [iconUrl]="m.iconUrl"
            (mouseOver)="onMouseOver(infoWindow, $event)">

            <agm-info-window [disableAutoPan]="false" #infoWindow>

                <h5 *ngIf="m.markerAdded ; else unset">Your marker</h5>
                <ng-template #unset>
                    <h5> Price {{m.price}} $</h5>
                    <a href={{m.url}}>{{m.url}}</a>
                </ng-template>
```

```
            </agm-info-window>
        </agm-marker>
        </agm-marker-cluster> -->
</agm-map>
```

map.component.ts

```
import { Component, OnInit } from '@angular/core';
import { LatLngBounds } from '@agm/core';
import { MapFrame } from '../models/MapFrame';
import { BackService } from '../shared/back.service';
import { Marker } from '../models/Marker';
import { DataGatewayService } from '../shared/dataGateway.service';

@Component({
  selector: 'app-map',
  templateUrl: './map.component.html',
  styleUrls: ['./map.component.less']
})
export class MapComponent implements OnInit {
  lat: number = 50.45;
  lng: number = 30.55;
  zoom: number = 12;
  isZoomed: boolean = false;
  enabledSettingLocation: boolean = false;
  infoWindow: any;
  prevInfoWindow: any;


  constructor(private backService: BackService, public dataGateway: DataGatewayService)
{}

  ngOnInit() {
  }

  onRemoveAddedMarker() {
     this.dataGateway.markers = this.dataGateway.markers.filter(elem => {
          return !elem.markerAdded;
       });
      this.dataGateway.buildModel = {};
      this.dataGateway.predictedBySA = false;
  }
  onSetLocation() {
    this.enabledSettingLocation = true;
  }
```

```
  addMarker(event: any) {
    if (this.enabledSettingLocation) {
      this.onRemoveAddedMarker();
      this.dataGateway.buildModel.lat = event.coords.lat.toString();
      this.dataGateway.buildModel.lon = event.coords.lng.toString();
      var geocoder = new google.maps.Geocoder();

      var latlng = {
        lat: parseFloat(event.coords.lat),
        lng: parseFloat(event.coords.lng),
      };

      geocoder.geocode({ location: latlng }, (results: any, status: any) => {
        if (status === "OK") {
          if (results[0]) {
            this.dataGateway.buildModel.streetName = results[0].address_components[1]?.l
ong_name;
            this.dataGateway.buildModel.buildNumber = results[0].address_components[0]?.
long_name
          }
        }
      });

      this.dataGateway.markers.push({
        lat: event.coords.lat,
        lon: event.coords.lng,
        markerAdded: true,
        iconUrl: "http://maps.google.com/mapfiles/ms/icons/blue-dot.png"
      });
      this.enabledSettingLocation = false;
    }
  }
  onMouseOver(infoWindow: any, gm: any) {
    if (this.prevInfoWindow) {
      this.prevInfoWindow.close();
    }
    infoWindow.open();
    this.prevInfoWindow = infoWindow;
  }
  onZoomChange(zoom: number) {
    this.isZoomed = zoom >= 12;
  }
  onBoundsChange(bounds: LatLngBounds) {
    if (this.isZoomed && !this.dataGateway.predictedBySA)
    {
      let frame: MapFrame =
        {
          up: bounds.getNorthEast().lat(),
```

```
          down: bounds.getSouthWest().lat(),
          left: bounds.getSouthWest().lng(),
          right: bounds.getNorthEast().lng()
        };
      this.dataGateway.markers = this.dataGateway.markers.filter(elem => {
          return elem.lat > frame.down
            && elem.lat < frame.up
            && elem.lon > frame.left
            && elem.lon < frame.right;
        });
      this.populateMarkers(frame);
    }
   }

  populateMarkers(frame: MapFrame) {
    this.backService.populateMarkers(frame).subscribe(
      (res: Marker[]) => {
        res.forEach(element => {
            if (!this.dataGateway.markers.find(marker => { return marker.lat == element.
lat && marker.lon == element.lon })) {
              this.dataGateway.markers.push(element);
            }
        });
      },
      err => {
        console.log(err);
      }
    );
  }
}
```

map.component.less

```
agm-map {
    height: 680px;
    margin: 1px;
}
#btnSetLocation:focus {
    color: burlywood;
}
input {
    background-color: #f1bfac;
    margin: 10px 30px 0 10px;
    border-radius: 50px;
    padding: 0.5rem;
    outline:none;
    border:0;
}
```

info-bar.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Marker } from '../models/Marker';
import { BackService } from '../shared/back.service';
import { DataGatewayService } from '../shared/dataGateway.service';
import { ToastrService } from 'ngx-toastr';

@Component({
  selector: 'app-info-bar',
  templateUrl: './info-bar.component.html',
  styleUrls: ['./info-bar.component.less']
})
export class InfoBarComponent implements OnInit {

  constructor(private backService: BackService, public dataGateway: DataGatewayService,
    private toastr: ToastrService)
    {}

  ngOnInit(): void {

  }

  handleClickAnalitic(res: any[]) {
    this.toastr.success("Calculated successfully! You can see " + res.length + " similar
 flats");
    this.calculateMeanPredictableSA(res);
    var addedMarker = this.dataGateway.markers.find(item => item.markerAdded);
    this.dataGateway.markers = res;
    this.dataGateway.markers.push(<Marker>addedMarker);
    this.dataGateway.predictedBySA = true;

  }
  calculateMeanPredictableSA(res: any[]) {
    let sum: number = 0;
    res.forEach(item=> sum+= item.price);
    this.dataGateway.buildModel.predictedSA = (Math.round(sum/res.length)).toString() +
" $";
  }
  onClickAnalitic() {
    this.backService.calculateAnalitics(this.dataGateway.buildModel).subscribe(
      (res: any) => {
        this.handleClickAnalitic(res);
      },
      err => {
        console.log(err);
```

```
      }
    );
  }
  onCalculateML() {
    this.backService.calculateML(this.dataGateway.buildModel).subscribe(
      (res: any) => {
        this.handleClickML(res);
      },
      err => {
        console.log(err);
      }
    );
  }
}
```

info-Bar.component.html

```
<button mdbBtn type="button" (click)="onClickAnalitic()" color="amber" mdbWavesEffect>Pr
edict price by system analize </button>
<button mdbBtn type="button" (click)="onCalculateML()"color="amber" mdbWavesEffect>Predi
ct price by machine learning </button>
<input type="text" id = "streetName" [(ngModel)]="dataGateway.buildModel.streetName"  pl
aceholder="Street name..." autocomplete="off" />
<input type="text" id = "buildNumber"[(ngModel)]="dataGateway.buildModel.buildNumber" pl
aceholder="Build number..." autocomplete="off" />
<input type="text" id = "lat" [(ngModel)]= "dataGateway.buildModel.lat" disabled = "true
" placeholder="Latitude..." autocomplete="off" />
<input type="text" id = "lon" [(ngModel)]= "dataGateway.buildModel.lon" disabled = "true
" placeholder="Longitude..." autocomplete="off" />
<input type="text" id = "floor" [(ngModel)]= "dataGateway.buildModel.floor" placeholder=
"Floor..." autocomplete="off"/>
<input type="text" id = "totalFloor" [(ngModel)]= "dataGateway.buildModel.total_floor" p
laceholder = "Total floor..." autocomplete="off"/>
<input type="text" id = "fullArea" [(ngModel)]= "dataGateway.buildModel.area" placeholde
r="Full area..." autocomplete="off"/>
<input type="text" id = "kitchenArea" [(ngModel)]= "dataGateway.buildModel.kitchen_area"
 placeholder="Kitchen area..." autocomplete="off"/>
<input type="text" id = "rooms" [(ngModel)]= "dataGateway.buildModel.rooms" placeholder=
"Rooms..." autocomplete="off"/>
```

info-Bar.component.less

```
input {
    background-color: #edc7b7;
    margin: 20px 20px 0 30px;
    border-radius: 50px;
    padding: 0.5rem;
    outline:none;
```

```
    border:0;
}
input:focus {
    background-color: #bab2b5;
}
input:hover {
    background-color: #bab2b5;
}
input[type="number"]::-webkit-inner-spin-button {
    -webkit-appearance: none;
}
button {
    margin: 10px;
    margin-left: 20px;
    width: 200px;
}
```