НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра інформатики та програмної інженерії
(повна назва кафедри)

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

спеціальності «121 Інженерія програмного забезпечення»

на тему: Сервіс агрегації та аналізу даних про аренду житла

Виконав студент IV курсу, групи ІТ-01
(шифр групи)

Гончаренко Андрій Андрійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доцент, к.т.н., доц., Фіногенов О.Д. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент доцент каф. НГІ та КГ, к.т.н., доц. Яблонський П.М. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ –2024

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 Інженерія програмного забезпечення

Освітньо-професійна програма – Інженерія програмного забезпечення інформаційних систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Едуард ЖАРІКОВ
(підпис) (ім'я прізвище)
"___"_____2024 р.

ЗАВДАННЯ
на дипломний проєкт студенту

Гончаренко Андрію Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема проєкту Сервіс агрегації та аналізу даних про аренду житла
керівник проєкту Фіногенов Олексій Дмитрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «27»квітня 2024 р. №2112-с

2. Термін подання студентом проєкту « 17 » червня 2024 року

3. Вихідні дані до проєкту: технічне завдання

4. Зміст пояснювальної записки
1) Загальні положення: основні визначення та терміни, опис предметного середовища, огляд ринку програмних продуктів, постановка задачі.
2) Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних.
3) Математичне забезпечення: змістовна та математична постановки задачі, обґрунтування та опис методу розв'язання.
4) Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів.
5) Технологічний розділ: керівництво користувача, методика випробувань програмного продукту.

5. Перелік графічного матеріалу

1) Схема структурна варіантів використань

2) Схема структурна бізнес процесів

3) Схема бази даних

4) Схема структурна компонентів програмного забезпечення

6. Консультанти розділів проєкту

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

7. Дата видачі завдання «11» березня 2024 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Вивчення рекомендованої літератури	11.03.2024	
2	Аналіз існуючих методів розв’язання задачі	21.03.2024	
3	Постановка та формалізація задачі	26.03.2024	
4	Розробка інформаційного забезпечення	07.04.2024	
5	Алгоритмізація задачі	15.04.2024	
6	Обґрунтування вибору використаних технічних засобів	17.04.2024	
7	Розробка програмного забезпечення	01.05.2024	
8	Налагодження програми	10.05.2024	
9	Виконання графічних документів	12.05.2024	
10	Оформлення пояснювальної записки	01.06.2024	
11	Подання ДП на попередній захист	02.06.2024	
12	Подання ДП рецензенту	10.06.2024	
13	Подання ДП на основний захист	14.06.2024	

Студент _____ Андрій ГОНЧАРЕНКО

(підпис) (ініціали, прізвище)

Керівник _____ Олексій ФІНОГЕНОВ

(підпис) (ініціали, прізвище)

Анотація

Пояснювальна записка дипломного проєкту складається з п’яти розділів, містить 34 таблиць, 24 рисунків та 33 джерела – загалом 58 сторінок.

Дипломний проєкт присвячений сервісу для агрегації та аналізу оголошень про оренду.

Мета: полегшити та пришвидшити процес пошуку оголошень про оренду квартири.

Об'єкт дослідження: сервіс агрегації та аналізу даних про оренду житла.

Предмет дослідження: визначення вимог до програмного забезпечення та архітектури, розробка та тестування.

В першому розділі даної роботи було досліджено предметну область. Було проведено аналіз відомих технічних рішень та успішних аналогів.

В другому розділі було розроблено вимоги до програмного забезпечення.

В третьому розділі описано моделювання та конструювання додатку. Була розроблена архітектура застосунку, описані програмні продукти, які були використані.

В четвертому розділі було складено тестовий план, визначено об’єкти тестування, визначено підходи до тестування та описано основні задачі тестування.

В п’ятому розділі наведено опис впровадження додатку та представлено керівництво користувача

КЛЮЧОВІ СЛОВА: АГРЕГАТОР ОГОЛОШЕНЬ, ШТУЧНИЙ ІНТЕЛЕКТ, API, БАЗА ДАНИХ, ПАРСИНГ.

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 34 tables, 24 figures and 33 sources - a total of 58 pages.

The purpose of the diploma project is a service for the aggregation and analysis of rental ads.

Purpose: implementation and maintenance of the service of aggregation and analysis of ads for housing rent.

The object of the study: the service of aggregation and analysis of data on housing rent.

Research subject: definition of software and architecture requirements, development and testing.

The subject area was investigated in the first section of this work. It was an analysis of known technical solutions and successful analogues was carried out.

In the second section, the software requirements were developed.

The third chapter describes the modeling and construction of the application. The application architecture was developed, the software products that were used were described.

In the fourth chapter, a test plan was drawn up, test objects were defined, testing approaches were defined, and the main tasks of testing were described.

The fifth chapter describes the implementation of the application and user manual is presented

KEYWORDS: AD AGGREGATOR, ARTIFICIAL INTELLIGENCE, API, DATABASE, PARSING.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	КПІ.ІТ-0104.045440.01.91	Технічне завдання	12	
3	А4	КПІ.ІТ-0104.045440.02.81	Пояснювальна записка	58	
4	А4	КПІ.ІТ-0104.045440.03.12	Текст програми	34	
5	А4	КПІ.ІТ-0104.045440.04.51	Програма та методика тестування	6	
6	А4	КПІ.ІТ-0104.045440.05.34	Керівництво користувача	10	
7	А3	КПІ.ІТ-0104.045440.06.99	Графічний матеріал	4	

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

СЕРВІС АГРЕГАЦІЇ ТА АНАЛІЗУ ДАНИХ ПРО АРЕНДУ ЖИТЛА

Технічне завдання

КПІ. ІТ-0104.045440.01.91

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Андрій ГОНЧАРЕНКО

Київ – 2024

Зміст

1 НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Сервіс агрегації та аналізу даних про аренду житла.

Галузь застосування: наведене технічне завдання поширюється на розробку програмного забезпечення, котре використовується для агрегації, аналізу та оцінки нерухомості та призначена для підвищення ефективності процесу пошуку оголошень про оренду житла, забезпечуючи зручність і задоволення потреб користувачів у цій сфері.

2 ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки сервісу агрегації та аналізу даних про аренду житла є завдання на дипломне проєктування, затверджене кафедрою інформатики та програмної інженерії Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3 ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для пошуку та аналізу оголошень про оренду нерухомості.

Метою розробки є покращення ефективності пошуку оголошень про оренду нерухомості з різних джерел.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Програмне забезпечення повинно забезпечувати виконання наступних основних функції:

4.1.1 Користувацького інтерфейсу

– відображення усіх ендпоінтів сервісу;
– перелік можливих відповідей серверу;
– перелік параметрів запиту.

Приклад відображення ендпоінтів вказано на рисунку 4.1.

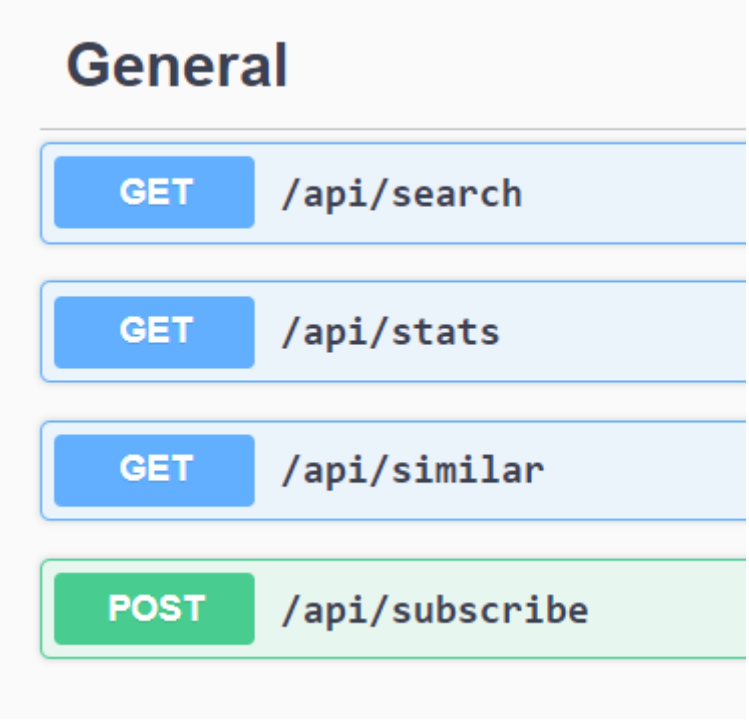


Рисунок 4.1 – Приклад переліку ендроінтів

Приклад відображення параметрів запиту вказано на рисунку 4.2.

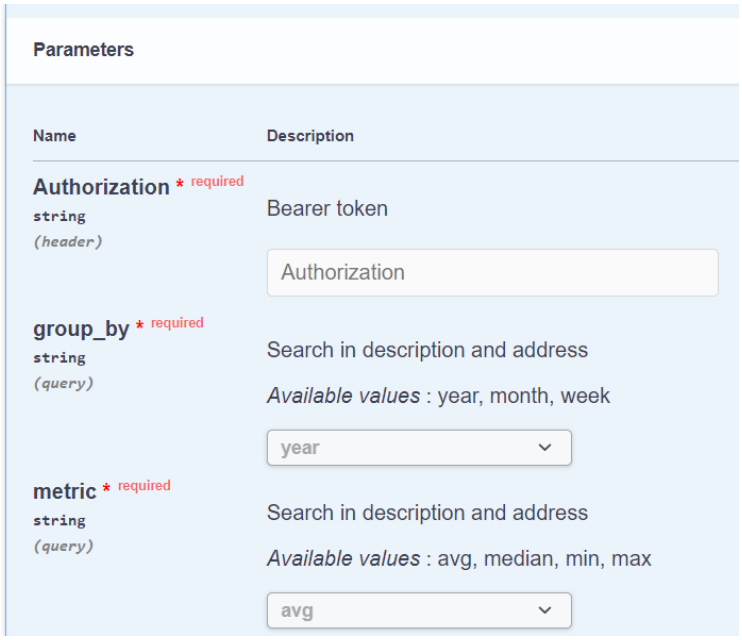


Рисунок 4.2 – Приклад переліку параметрів запиту

4.1.2 Для неавторизованого користувача:

– реєстрація;
– авторизація.

4.1.3 Для авторизованого користувача):

– переглядання списку оголошень за фільтрами;
– переглядання статистики за фільтрами;
– оцінка вручну введених даних про квартиру;
– підписка на щоденну розсилку нових оголошень за фільтрами;
– пошук оголошень, схожих на вже існуюче оголошення;
– пошук оголошень, схожих на вручну введене оголошення.

4.1.4 Додаткові вимоги:

– модель оцінки вартості оренди квартири.

4.2 Вимоги до надійності

Передбачити контроль введення інформації та захист від некоректних дій користувача. Забезпечити цілісність інформації в базі даних.

4.3 Умови експлуатації

Умови експлуатації згідно СанПін 2.2.2.542 – 96.

4.3.1 Вид обслуговування

Вимоги до виду обслуговування не висуваються.

4.3.2 Обслуговуючий персонал

Вимоги до обслуговуючого персоналу не висуваються.

4.4 Вимоги до складу і параметрів технічних засобів

Мінімальна конфігурація технічних засобів:

– тип процесору: Intel Core i3;
– об‘єм ОЗП: 4 Гб;
– підтримка контейнеризації;
– підключення до мережі Інтернет зі швидкістю від 20 мегабіт.

Рекомендована конфігурація технічних засобів:

– тип процесору: Intel Core i5;
– об‘єм ОЗП: 16 Гб;
– підтримка контейнеризації;
– підключення до мережі Інтернет зі швидкістю від 100 мегабіт.

4.5 Вимоги до інформаційної та програмної сумісності

Програмне забезпечення повинно працювати під управлінням операційних систем сімейства WIN32 (Windows'XP, Windows NT і т.д.) або Unix.

4.5.1 Вимоги до вхідних даних

Вхідні дані повинні бути представлені в наступному форматі: запити по протоколу HTTP, які можуть містити URL параметри та тіло запиту у форматі JSON.

4.5.2 Вимоги до вихідних даних

Результати повинні бути представлені в наступному форматі: відповіді на по HTTP протоколу у форматі JSON.

4.5.3 Вимоги до мови розробки

Розробку виконати на мові програмування Python.

4.5.4 Вимоги до середовища розробки

Розробку виконати на платформ PyCharm та Jupyter Notebook.

4.5.5 Вимоги до представленню вихідних кодів

Вихідний код програми має бути представлений у вигляді текстових файлів Python (.py) та Jupyter (.ipynb).

4.6 Вимоги до маркування та пакування

Вимоги до маркування та пакування не висуваються.

4.7 Вимоги до транспортування та зберігання

Вимоги до транспортування та зберігання не висуваються.

4.8 Спеціальні вимоги

Спеціальні вимоги не висуваються.

5 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

5.1 Попередній склад програмної документації

У склад супроводжувальної документації повинні входити наступні документи на аркушах формату А4:

– пояснювальна записка;
– технічне завдання;
– текст програми;
– програма та методика тестування;
– керівництво користувача.

Графічна частина повинна бути виконана на аркушах формату А3 та містити наступні документи:

– схема структурна варіантів використання;
– схема структурна бізнес процесів;
– схема структурна компонентів програмного забезпечення;
– схема бази даних.

5.2 Спеціальні вимоги до програмної документації

Програмні модулі, котрі розробляються, повинні бути задокументовані, тобто тексти програм повинні містити всі необхідні коментарі.

6 СТАДІЇ І ЕТАПИ РОЗРОБКИ

№	Назва етапу	Строк	Звітність
1.	Вивчення літератури за тематикою проекту	21.02	
2.	Розробка технічного завдання	03.03	Технічне завдання
3.	Аналіз вимог та уточнення специфікацій	19.04	Специфікації програмного забезпечення
4.	Проектування структури програмного забезпечення, проектування компонентів	30.04	Схема структурна програмного забезпечення та специфікація компонентів (діаграма класів, схема алгоритму)
5.	Програмна реалізація програмного забезпечення	05.05	Тексти програмного забезпечення
6.	Тестування програмного забезпечення	10.05	Тести, результати тестування
7.	Розробка матеріалів текстової частини проекту	14.05	Пояснювальна записка
8.	Розробка матеріалів графічної частини проекту	20.05	Графічний матеріал проекту
9.	Оформлення технічної документації проекту	29.05	Технічна документація

7 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Пояснювальна записка
до дипломного проєкту

на тему: **Сервіс агрегації та аналізу даних про аренду житла**

КПІ.ІТ-0104.045440.02.81

Київ – 2024

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД	–	База даних
ІТ	–	Інформаційні технології
ОС	–	Операційна система
СУБД	–	Система управління базою даних
ШІ	–	Штучний інтелект
API	–	Application programming interface, прикладний програмний Інтерфейс
ER	–	Entity-Relation diagram
HTML	–	HyperText Markup Language
IDE	–	Integrated Development Environment – інтегроване середовище розробки
REST	–	Representational State Transfer
SDK	–	Software development kit
Скрапер	–	програмний інструмент, який автоматизує процес збору даних із веб-сайтів

ВСТУП

У сучасному світі об'єм даних, що генерується і доступний для аналізу, зростає експоненційно. Особливо це стосується сегменту нерухомості, де кількість даних про оренду житла швидко збільшується завдяки популярності платформ для бронювання та оренди житла, таких як rieltor, olx, та інші.

У контексті цього зростаючого обсягу даних виникає суттєва потреба в ефективних інструментах агрегації та аналізу цих даних. Сервіси агрегації даних про оренду житла можуть відігравати ключову роль у розумінні динаміки ринку, споживчих уподобань та інших ключових факторів.

Цей проект має на меті розробити сервіс агрегації та аналізу даних про оренду житла, який використовує методи машинного навчання для рекомендації цін. Цей проект буде корисним для платформ оренди житла, агентств нерухомості та інших зацікавлених сторін, які шукають нові способи аналізувати великі обсяги даних для прийняття бізнес-рішень.

Дослідження включатиме розгляд існуючих методів агрегації та аналізу даних, а також розробку нових підходів, що дозволять ефективно і з точністю аналізувати та класифікувати дані про оренду житла. Даний проєкт покращує процес пошуку оголошень та аналізу ринку житла.

1 ПЕРЕДПРОЄКТНЕ ОБСТЕЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз предметної області

Сайт-агрегатор – веб-ресурс, де зібрані й розподілені пропозиції від різних компаній, що продають товари і пропонують послуги. Основний прибуток таких сайтів йде від реалізації товарів і послуг фірм, які представлені на інтернет-майданчику[1].

Процес агрегації оголошень не є новим, нескладно знайти багато вдалих прикладів даних застосунків. Задачею є покращити вже існуючі рішення за допомогою машинного навчання.

Машинне навчання (Machine learning, ML) – звід методів в області штучного інтелекту, набір алгоритмів, які застосовують, щоб створити машину, яка вчиться на власному досвіді. В якості навчання машина обробляє величезні масиви вхідних даних і знаходить у них закономірності[2].

NLP (Natural Language Processing, обробка природної мови) – це напрям у машинному навчанні, присвячений розпізнаванню, генерації та обробці усного та письмового людського мовлення[3].

Для збору оголошень найчастіше використовуються скрапери.

Вебскрапінг – перетворення у структуровані дані інформації з вебсторінок, які призначені для перегляду людиною за допомогою браузера[4].

Основними функціями агрегатора оголошень оренди житла є зручний перегляд останніх оголошень, аналітика по нинішньому стану ринку нерухомості, можливість гнучкого пошуку оголошень.

Також важливою складовою агрегаторів оголошень є збір даних з інтернет джерел.

Дане програмне забезпечення має унікальну функцію передбачення ціни на оренду житла та знаходження схожих квартир методами машинного навчання. Передбачення ціни надає можливість знайти квартири, що здаються

дешевше за їх ринкову вартість, а пошук схожих дозволяє знайти оголошення, схоже на вже існуюче, або введене користувачем.

1.2 Аналіз існуючих рішень

Проаналізуємо відоме на сьогодні алгоритмічне забезпечення у даній області та технічні рішення, що допоможуть у реалізації сервісу агрегації та аналізу оголошень про оренду житла. Далі будуть розглянуті допоміжні програмні засоби, засоби розробки та готові програмні рішення.

1.2.1 Аналіз відомих програмних продуктів

Порівняємо функціонал з двома агрегаторами оголошень, що є найпопулярнішими на ринку України: Bird AI[5] та ЛУН[6].

Для порівняння проєкту з аналогом можна скористатись таблицею 1.1.

Таблиця 1.1 – Порівняння з аналогом

Функціонал	Дипломний проєкт	Bird AI	ЛУН
Перегляд оголошень	Оголошення можна переглянути списком. Також можлива відправка нових оголошень на пошту	Оголошення можна переглянути на мапі міста	Оголошення можна переглянути на мапі міста або списком
Кількість міст що підтримується	Тільки Київ та його передмістя	Київ, Львів, Харків, Одеса, Дніпро	Більш ніж 20 міст, але не всі міста підтримують
Реєстрація	Для використання необхідна реєстрація	Реєстрація відсутня	Реєстрація не обов'язкова
Використання штучного інтелекту	ШІ використовується для передбачення ринкової вартості нерухомості. Також наявний пошук схожих квартир	ШІ використовується для пошуку квартир з ремонтом та персонального підбору оголошень	ШІ використовується для пошуку квартир з ремонтом
Статистика	Надає статистику згруповану по місяцям або тижням з гнучкими фільтрами по оголошенням	Не надає статистику	Надає статистику за останній місяць по районах для квартир з 1-3 кімнатами. Можна подивитися історичну динаміку по місту окремо для квартир з 1-2 кімнатами

Таким чином дане програмне забезпечення має перевагу в наданні статистики та по новому використовує штучний інтелект, в порівнянні з аналогами.

1.2.2 Аналіз відомих алгоритмічних та технічних рішень

Для скрипту збору оголошень можна використовувати парсинг – збір даних безпосередньо з HTML коду сторінки. Такий спосіб дає змогу зібрати дані з усіх загальнодоступних веб-сторінок, але кожен сайт повинен бути окремо оброблений. Іншим способом є використання API сайту оголошень. Даний метод надає можливість збирати дані використовуючи програмний інтерфейс, що може облегшити розробку та знизити ризики нестабільної роботи застосунку. Українські сайти з оголошеннями не надають можливість збирати дані про оголошення через API, через що є необхідність розробки парсеру сторінок.

Для вирішення задачі перетворення опису оголошення у вектор чисел використовують алгоритм Word2Vec. Word2Vec – це двошарова нейронна мережа, яка обробляє текст, приймаючи пакети необроблених текстових даних, обробляючи їх і створюючи векторний простір у кілька сотень вимірів. Кожному унікальному слову в даних присвоюється відповідний вектор у просторі. Розташування цих векторів у просторі визначається семантичним значенням слів і близькістю до інших слів. Word2Vec поділяється на два види. Модель CBOW передбачає цільове слово на основі контекстних слів, що його оточують. Іншими словами, він використовує навколишні слова, щоб передбачити слово в середині. Ця модель бере всі контекстні слова, агрегує їх і використовує результуючий вектор для прогнозування цільового слова. Модель Skip-Gram передбачає навколишні контекстні слова з цільового слова. Іншими словами, він використовує одне слово, щоб передбачити навколишній контекст. Наприклад, якщо ми знову візьмемо речення «The cat sit on the mat»,

модель Skip-Gram візьме «cat» як вхідні дані та передбачить «The», «sat», «on», «the», «mat». Для нашого випадку модель CBOW підходить краще, так як вона не враховує порядок слів при навчанні. Порядок слів у оголошенні не є доволі важливим, більше сенсу мають саме ключові слова[7].

На етапі аналізу існуючих технологій необхідно обрати алгоритми регресії, які будуть протестовані та оцінені на етапі розробки програмного забезпечення за допомогою метрики r2 score[8].

Для подальшої оцінки були відібрані алгоритми регресії. Одним з критеріїв вибору була невисока обчислювальна складність алгоритмів через відсутність високопродуктивної обчислювальної техніки для їхнього навчання.

Таким чином було обрано три алгоритми, що описані нижче.

CatBoost або Categorical Boosting – це бібліотека з відкритим кодом. Він розроблений для використання в таких задачах, як регресія та класифікація, що мають дуже велику кількість незалежних ознак.

Catboost – це варіант градієнтного бустінгу, який може обробляти як категориальні, так і числові значення. Він не потребує жодних методів кодування ознак, таких як One-Hot Encoder або Label Encoder для перетворення категоріальних ознак у числові. Він також використовує алгоритм, званий симетричним зваженим квантильним ескізом (SWQS), який автоматично обробляє відсутні значення в наборі даних, щоб зменшити перенавчання та покращити загальну продуктивність набору даних[9].

Алгоритм Random Forest – хоча дерева рішень є звичайними алгоритмами навчання з вчителем, вони можуть бути схильні до проблем, таких як упередженість і перенавчання. Однак, коли декілька дерев рішень утворюють ансамбль, вони передбачають більш точні результати, особливо коли окремі дерева не корельовані одне з одним.

Даний алгоритм складається з ансамблю дерев рішень, і кожне дерево в ансамблі складається із вибірки даних, отриманої з навчального набору із заміною, яка називається початковою вибіркою. Залежно від типу проблеми визначення прогнозу буде різним. Для завдання регресії буде усереднено окремі дерева рішень, а для завдання класифікації – більшість голосів, тобто. найчастіша категоріальна змінна – дасть прогнозований клас[10].

Змодельовано нейромережу, що складається з чотирьох прихованих Dense шарів з функцією активації relu та одного вихідного лінійного Dense шару[11].

1.3 Опис бізнес-процесів

Для опису бізнес процесу використовується BPMN. Модель, що описує послідовність користування сервісом незареєстрованого користувача наведена у графічному матеріалі, креслення 1.

Опис послідовності роботи неавторизованого користувача з сервісом:

– користувач надсилає запит на реєстрацію;
– користувач надсилає запит на отримання токену доступу;
– користувач надсилає запит на отримання списку актуальних оголошень.

Модель, що описує послідовність користування сервісом зареєстрованого користувача наведена у графічному матеріалі, креслення 2.

Опис послідовності роботи авторизованого користувача з сервісом:

– користувач надсилає запит на отримання токену доступу;
– користувач оцінює вручну введену квартиру;
– користувач шукає оголошення, схожі на вже існуюче або введене вручну.

1.4 Постановка задачі

Задачею є розробка застосунку, що надає можливість користувачу переглядати актуальні оголошення оренди нерухомості з оцінкою ціни, отримати гнучку історичну статистику. Користувач повинен мати можливість оцінити введену вручну дані про квартиру, знайти схожі оголошення та отримувати список нових оголошень по пошуку на електронну пошту.

Висновки до розділу

Проаналізовано предметну область та описано основні компоненти та специфіки обраної сфери. Також проаналізовано відомі програмні продукти, що довело актуальність та цінність даного програмного забезпечення

Аналіз відомих алгоритмічних та технічних рішень виявив найефективніші методи та підходи, які можуть бути використані або адаптовані для розробки даного програмного забезпечення. Обрано алгоритми для векторизації тексту та навчання моделі оцінки вартості.

У ході опису бізнес-процесів були описані основні послідовності використання застосунку за допомогою BPMN моделей.

2 РОЗРОБЛЕННЯ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Варіанти використання програмного забезпечення

Головною функцією програмного забезпечення є агрегація та розрахунок приблизної ринкової ціни квартири, більше функцій можна побачити на рисунку 2.1.

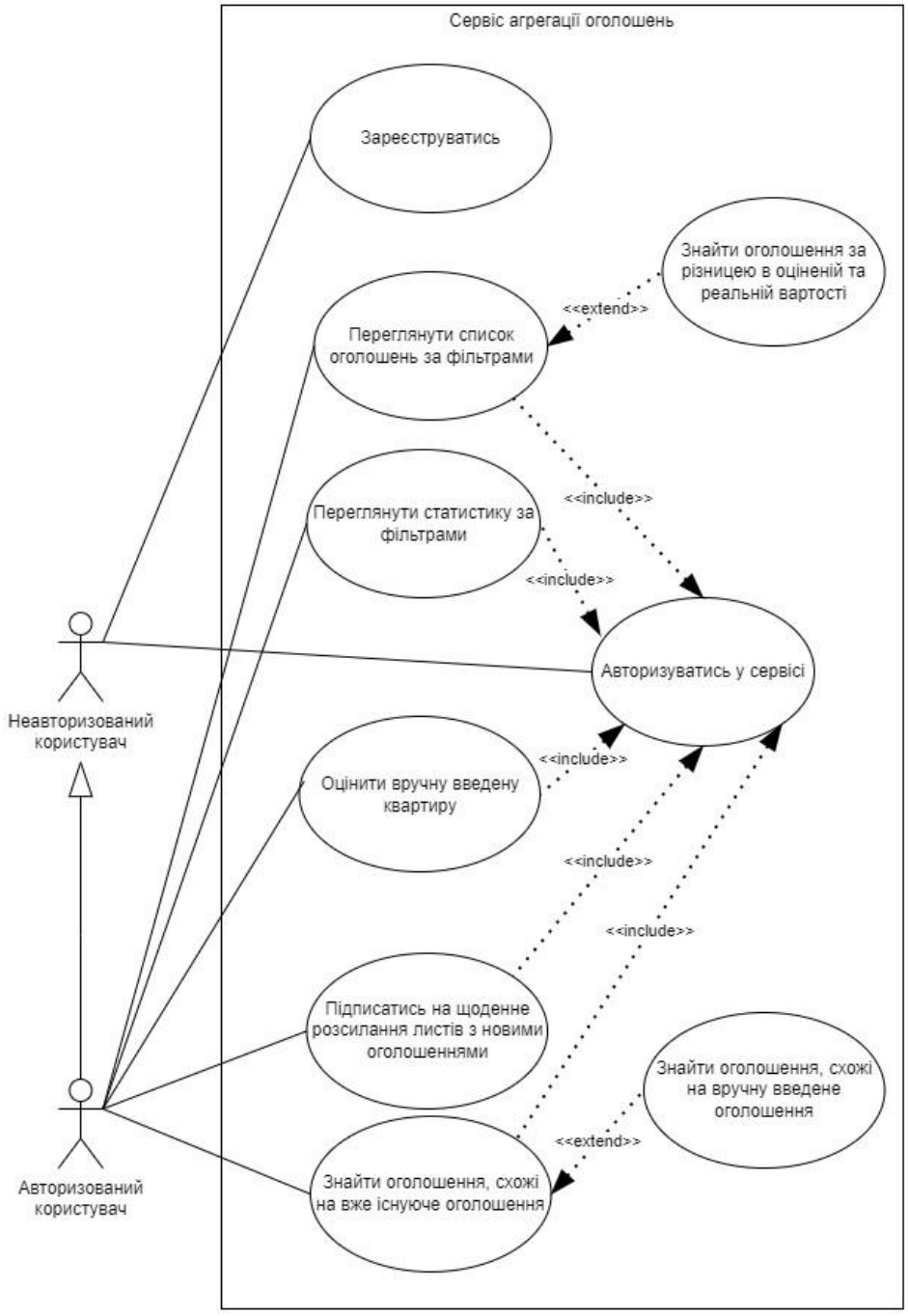


Рисунок 2.1 – Діаграма варіантів використання

В таблицях 2.1 - 2.9 наведені варіанти використання програмного забезпечення.

Таблиця 2.1 - Варіант використання UC-1

Use case name	Зареєструватись
Use case ID	UC-01
Goals	Створити нового користувача в системі
Actors	Користувач
Trigger	Користувач бажає зареєструватись
Pre-conditions	-
Flow of Events	Користувач робить запит з логіном та паролем. Якщо такого користувача ще не існувало то його аккаунт створюється, відправляється повідомлення про успішну реєстрацію. В іншому випадку відправляється повідомлення про помилку
Extension	
Post-Condition	Користувач зареєстрований у системі

Таблиця 2.2 - Варіант використання UC-2

Use case name	Авторизуватись у сервісі
Use case ID	UC-02
Goals	Отримати токен авторизації
Actors	Користувач
Trigger	Користувачу необхідно почати роботу з сервісом
Pre-conditions	Користувач зареєстрований
Flow of Events	Користувач надсилає запит з логіном та паролем. Якщо дані правильні, то користувачу відправляється тимчасовий токен доступу до сервісу. В іншому випадку користувачу відправляється повідомлення про помилку
Extension	
Post-Condition	Тимчасовий токен доступу

Таблиця 2.3 - Варіант використання UC-3

Use case name	Переглянути список оголошень за фільтрами
Use case ID	UC-03
Goals	Отримати список оголошень
Actors	Користувач
Trigger	Користувач бажає отримати список оголошень
Pre-conditions	Користувач отримав тимчасовий токен доступу
Flow of Events	Користувач надсилає запит з бажаними фільтрами. Користувачу повертається повідомлення зі списком актуальних оголошень. Якщо фільтри введені неправильно користувачу повертається повідомлення з помилкою
Extension	
Post-Condition	Повідомлення зі списком оголошень

Таблиця 2.4 - Варіант використання UC-4

Use case name	Переглянути статистику за фільтрами
Use case ID	UC-04
Goals	Отримати історичні дані про оголошення аренди житла
Actors	Користувач
Trigger	Користувач бажає отримати статистику про ринок житла
Pre-conditions	Користувач отримав тимчасовий токен доступу
Flow of Events	Користувач надсилає запит з бажаними фільтрами, групуванням та метрикою. Користувачу відправляється повідомлення з агрегованою статистикою. Якщо вхідні дані введені неправильно користувачу повертається повідомлення з помилкою
Extension	
Post-Condition	Повідомлення зі статистикою про ринок оренди

Таблиця 2.5 - Варіант використання UC-5

Use case name	Оцінити вручну введену квартиру
Use case ID	UC-05
Goals	Отримати оцінку вартості оренди квартири користувача
Actors	Користувач
Trigger	Користувач бажає оцінити квартиру, якої немає в оголошеннях сервісу
Pre-conditions	Користувач отримав тимчасовий токен доступу
Flow of Events	Користувач надсилає запит з даними про квартиру. Користувачу повертається повідомлення з оцінкою квартири. Якщо дані введені некоректно повертається повідомлення про помилку
Extension	
Post-Condition	Оцінка вартості оренди квартири

Таблиця 2.6 - Варіант використання UC-6

Use case name	Підписатись на щоденне розсилання електронних листів з новими оголошеннями
Use case ID	UC-06
Goals	Щоденно отримувати дані про нові оголошення на пошту
Actors	Користувач
Trigger	Користувач бажає отримувати повідомлення про нові оголошення на пошту
Pre-conditions	Користувач отримав тимчасовий токен доступу
Flow of Events	Користувач надсилає запит з електронною адресою та фільтрами пошуку. Пошта додається до розсилки. Якщо дані введені некоректно повертається повідомлення про помилку
Extension	
Post-Condition	Користувач отримує щоденні електронні листи з новими оголошеннями по пошуку

Таблиця 2.7 - Варіант використання UC-7

Use case name	Знайти оголошення, схожі на вже існуюче оголошення
Use case ID	UC-07
Goals	Знайти схожі квартири
Actors	Користувач
Trigger	Користувач бажає знайти квартири, що схожі на вже існуюче оголошення у сервісі
Pre-conditions	Користувач отримав тимчасовий токен доступу
Flow of Events	Користувач надсилає запит з посиланням на квартиру. Користувачу повертається список з 5 схожих оголошень. Якщо дані введені некоректно повертається повідомлення про помилку
Extension	
Post-Condition	Список схожих оголошень

Таблиця 2.8 - Варіант використання UC-8

Use case name	Знайти оголошення, схожі на вручну введене оголошення
Use case ID	UC-08
Goals	Знайти схожі квартири
Actors	Користувач
Trigger	Користувач бажає знайти квартири, що схожі на оголошення, якого немає у сервісі
Pre-conditions	Користувач отримав тимчасовий токен доступу
Flow of Events	Користувач надсилає запит з даними про квартиру. Користувачу повертається список з 5 схожих оголошень. Якщо дані введені некоректно повертається повідомлення про помилку
Extension	
Post-Condition	Список схожих оголошень

Таблиця 2.9 - Варіант використання UC-9

Use case name	Знайти оголошення за різницею в оціненій та реальній вартості
Use case ID	UC-09
Goals	Знайти оголошення, ціна в яких менше за ринкову
Actors	Користувач
Trigger	Корисутвач бажає знайти квартиру, ціна якої менша за ринкову
Pre-conditions	Користувач отримав тимчасовий токен доступу
Flow of Events	Користувач надсилає запит з бажаними фільтрами та сортування по різниці ринкової та оголошеної ціни. Користувачу повертається повідомлення зі списком актуальних оголошень. Якщо фільтри введені неправильно користувачу повертається повідомлення з помилкою
Extension	
Post-Condition	Список оголошень

Варіанти використання надають опис основних сценаріїв роботи користувачів з застосунком.

2.2 Розроблення функціональних вимог

Програмне забезпечення розділене на модулі. Кожен модуль має свій певний набір функцій. В таблиці 2.10 наведено загальну модель вимог.

Таблиця 2.10 – Загальна модель вимог

Номер	Опис	Код	Пріоритет	Ризик
1	Періодичний збір оголошень	FR-1	1	Високий
2	Перегляд актуальних оголошень	FR-2	1	Низький
3	Авторизація користувача	FR-3	2	Низький
4	Реєстрація користувача	FR-4	2	Низький
5	Перегляд історичної статистики	FR-5	2	Середній
6	Перегляд схожих оголошень	FR-6	3	Середній
7	Підписка на щоденну розсилку	FR-7	3	Низький
8	Оцінка вартості оренди введеної квартири	FR-8	1	Високий

Періодичний збір оголошень та перегляд актуальних оголошень мають найвищий пріоритет так як це є невід’ємною частиною будь-якого агрегатора. Перегляд схожих оголошень та щоденна підписка мають низький пріоритет через очікувану низьку кількість використань даних функцій. Оцінка вартості оренди введеної квартири має високий пріоритет через те що це є основною особливістю даного сервісу.

Періодичний збір оголошень має високий ризик через нестабільність збору даних зі сторінки оголошень: якщо код сторінки буде змінено, то скрипт збору може перестати працювати. Оцінка вартості оренди введеної квартири має високий ризик через необхідність точних обчислень. Перегляд схожих оголошень має середній ризик через очікуваний великий час обробки запиту. Перегляд історичної статистики має середній ризик через постійне збільшення об’єму даних, що може призвести до сповільнення системи.

В таблиці 2.11 наведений опис функціональних вимог до програмного забезпечення. Матрицю трасування вимог можна побачити в таблиці 2.12.

Таблиця 2.11 – Перелік функціональних вимог

ID вимоги	Назва та опис
FR-1	Періодичний збір оголошень Дані про оголошення повинні оновлюватися періодично з декількох джерел.
FR-2	Перегляд актуальних оголошень Користувач отримує список актуальних оголошень за фільтрами
FR-3	Авторизація користувача Користувач отримує токен тимчасового доступу для користування сервісом
FR-4	Реєстрація користувача Користувач створює свій аккаунт у сервісі
FR-5	Перегляд історичної статистики Користувач отримує історичну статистику за бажаними фільтрами, групуванням та метрикою
FR-6	Перегляд схожих оголошень Користувач отримує список оголошень, що схожі на вже існуюче оголошення або введене вручну
FR-7	Підписка на щоденну розсилку Користувач щоденно отримує актуальний список оголошень на пошту
FR-8	Оцінка вартості оренди введеної квартири Користувач отримує оцінку вартості оренди введеної квартири

Таблиця 2.12 – Матриця трасування вимог

	FR1	FR2	FR3	FR4	FR5	FR6	FR7	FR8
UC1				+				
UC2			+					
UC3	+	+						
UC4	+				+			
UC5								+
UC6							+	
UC7	+					+		
UC8	+					+		
UC9	+	+						

Матриця трасування вимог описує які функціональні вимоги необхідні для забезпечення відповідного варіанту використання.

2.3 Розроблення нефункціональних вимог

Розроблене програмне забезпечення повинно відповідати наступним нефункціональним вимогам:

- мати зрозумілий програмний інтерфейс;
- мінімізувати час відповіді сервера на запит користувача;
- бути простою у розгортанні;
- розпізнавати лише дані, які введені українською мовою.

Висновки до розділу

Розроблено діаграму варіантів використання. Детально описані варіанти використання програмного забезпечення.

Створено загальну модель вимог та обґрунтовано вибір пріоритету та ризику функціональних вимог. Надано детальний опис функціональних вимог.

Побудовано модель трасування вимог, що описує функціональні вимоги, що необхідні для забезпечення варіантів використання.

Розроблено нефункціональні вимоги, яким повинно відповідати програмне забезпечення.

3 КОНСТРУЮВАННЯ ТА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектура програмного забезпечення

Для забезпечення виконання функціональних вимог найкращим варіантом архітектури є клієнт-серверна. Вся логіка застосунку та управління даними зосереджена на сервері. Для розробки архітектури API було використано принципи REST. REST – це архітектура програмного забезпечення, яка визначає умови роботи API. Дана архітектура передбачає дотримання таких принципів: єдиний інтерфейс, відсутність збереження стану, кешування. Така архітектура має такі переваги як гнучкість та можливість масштабування[12]. Архітектура програмного забезпечення у вигляді діаграми компонентів наведена у графічному матеріалі, креслення 3.

Компонент скраперу складається з двох частин, кожна з яких відповідає за окреме джерело. Вони збирають «сирі дані» про оголошення та трансформують у структуровані таблиці. Дані таблиці передаються до бази даних та до моделі, що розраховує приблизну ринкову ціну. Окремий компонент відповідає за щоденну відправку електронних листів.

Основний API використовує Auth0 для авторизації користувачів. Для зберігання даних було обрано реляційні бази даних через необхідність зберігання структурованих та пов’язаних між собою даних.

Для розміщення даних використано систему управління базами даних PostgreSQL[13].

3.2 Обґрунтування вибору засобів розробки

При виборі основної мови програмування головними критеріями є наявність фреймворків для розробки API, можливість зручної обробки даних та присутність бібліотек для машинного навчання. Базуючись на даних вимогах було вирішено обрати мову програмування Python.

Для розробки API є велика кількість Python фреймворків, наприклад: Django[14], CherryPy[15], Flask[16] тощо. Дане програмне забезпечення передбачає простий API. Для даного випадку обрано Flaks за його простоту у розробці та налаштуванні. Недоліком фреймворку є не така велика кількість можливостей, як, наприклад, у Django.

Для скрипту збору оголошень буде використано парсинг. Наприклад бібліотека requests[17] у поєднанні з beautiful soup[18] дають можливість знайти інформацію на будь-якому сайті з оголошеннями.

Для роботи з даними у Python обрано бібліотеку pandas[19]. Дана бібліотека написана на базі numpy[20], що прискорює роботу з великими об’ємами даних. Також перевагою є велика кількість вбудованих методів для зчитування, трансформації та аналізу даних.

Для регресії, тобто розрахунку рекомендованої ціни, вирішено використовувати бібліотеки sklearn[21], catboost[22] та keras[23] через широкий вибір алгоритмів машинного навчання, а саме для NLP моделей[24-25], що будуть використані в даному програмному забезпеченні.

Для перетворення категоріальних значень в вектор чисел вирішено використати Embedding шари з бібліотеки keras [26].

Для перетворення опису оголошення в вектор чисел обрано алгоритм Word2Vec з бібліотеки gensim[27].

Середовище розробки обрано PyCharm[28] через його мультифункціоналність та полегшення розробки мовою Python, що була обрана як основна. Python IDLE[29] не була обрана основним середовищем розробки через відсутність допоміжних функцій для розробки великих проектів. Visual Studio Code[30] поступається PyCharm через необхідність встановлення додаткових пакетів для початку розробки обраною мовою.

Допоміжною середою розробки обрано Jupyter Notebook[31] для розробки моделей машинного навчання. Дана середа дає можливість виконувати код блоками та зберігати результат роботи після виконання.

Google Colab[32] також дозволяє працювати з ноутбуками, але має менше можливостей для встановлення доповнень для зручної розробки.

Для адміністрування бази даних використано застосунок TablePlus[33] через простоту у налаштуванні та гнучкість підключення до різних БД.

3.3 Конструювання програмного забезпечення

Для оцінки вартості квартири вирішено створювати свою регресійну модель на основі Embedding layers, Word2Vec та одного з алгоритмів регресії. Для навчання зібрано близько 10 тисяч оголошень про оренду квартир в Києві.

Для створення векторів що описують райони створено нейромережу, що складається з одного Embedding шару, одного прихованого шару з функцією активації relu та вихідного шару. Дана модель була навчена на тренувальних даних. Як результат було створено словник, де для кожного району Києва міститься вектор чисел. Для візуалізації вектори були зменшені до розмірності 2 за допомогою алгоритму PCA. Результат векторизації зображений на рисунку 3.2.

Рисунок 3.1 – Результат векторизації районів

Для векторизації опису оголошення використано використано алгоритм Word2Vec, а саме Continuous Bag of Words. Щоб отримати оптимальний розмір

вікна, що використовується для навчання, на результуючих векторах навчено модель градієнтного бустінгу з однаковими параметрами та оцінено r2 score. Результати оцінки відображено на рисунку 3.3.

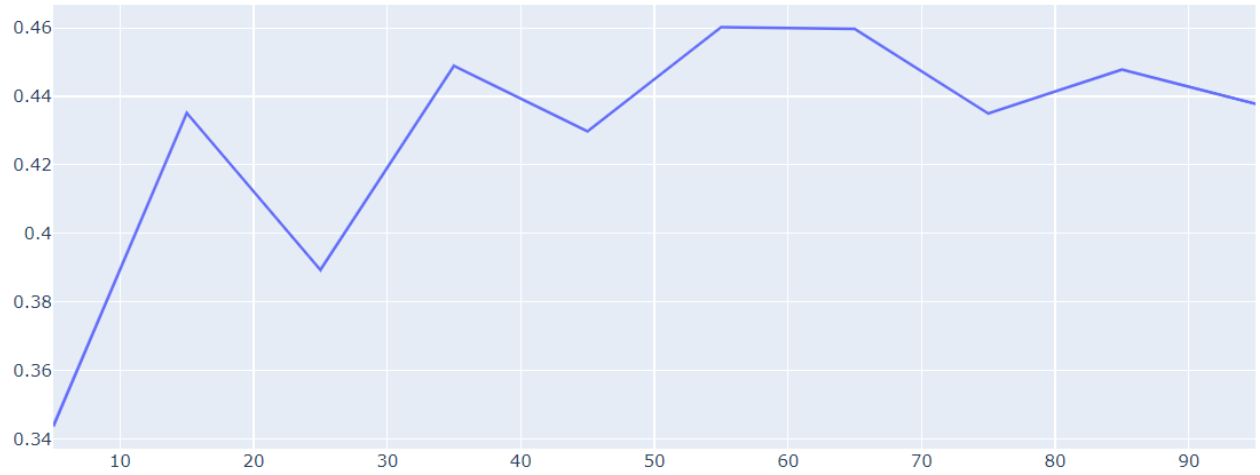


Рисунок 3.2 – Залежність r2 score від розміру вікна

Зі збільшенням розміру вікна оцінка моделі росте до 0.46 при розмірі 55, а потім значно не змінюється. Таким чином розмір вікна 55 обрано як оптимальний. Аналогічним чином розраховано r2 score для різних розмірів вектору. Результат оцінки відображено на рисунку 3.4.

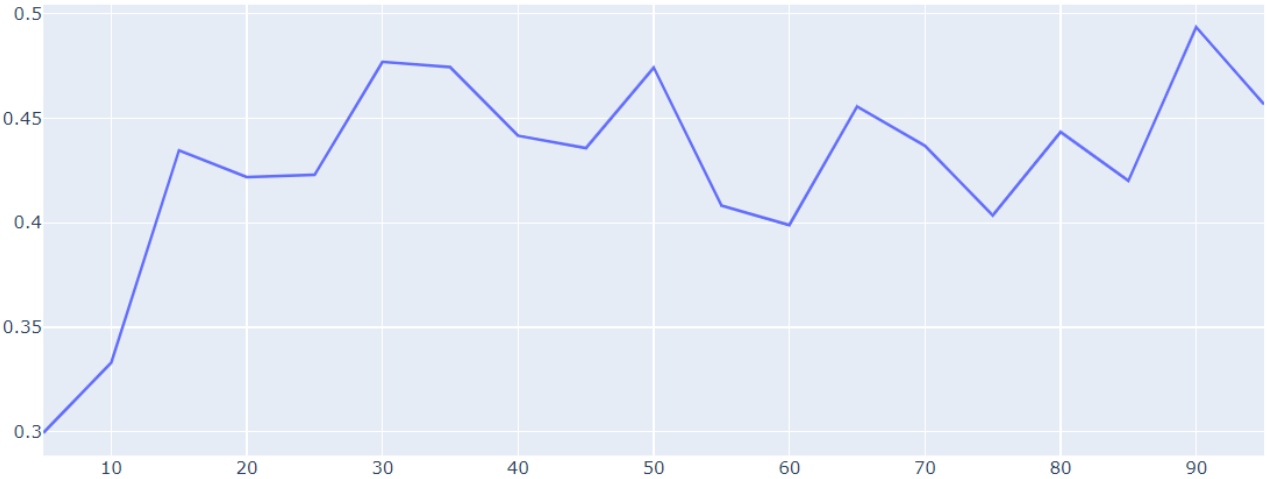


Рисунок 3.3 – Залежність r2 score від розміру вектору

При розмірі вектору 30 оцінка моделі досягає максимуму, та значно не збільшується при збільшенні розміру. Таким чином оптимальним розміром вектору є 30. Створені вектори опису та району були використані для навчання регресійної моделі.

Регресійна модель навчалася трьома алгоритмами. Результати навчання наведені в таблиці 3.1.

Таблиця 3.1 – Результати навчання регресійної моделі

Алгоритм	R2 score	Середнє відхилення від реальної вартості (грн)
CatBoost	0.74	-965
Random Forest	0.68	-2033
Нейромережа	0.66	1100

Судячи з оцінок моделей, доцільно обрати модель, що була навчена алгоритмом CatBoost, так як вона має найкращу оцінку R2 score та найменше середнє відхилення від реальної ціни. Важливо зазначити, що підтримка додатку передбачає подальше покращення моделі через збір нових даних.

База даних призначена для зберігання оголошень, векторів оголошень, даних про користувачів. Модель бази даних наведена на рисунку 3.5.

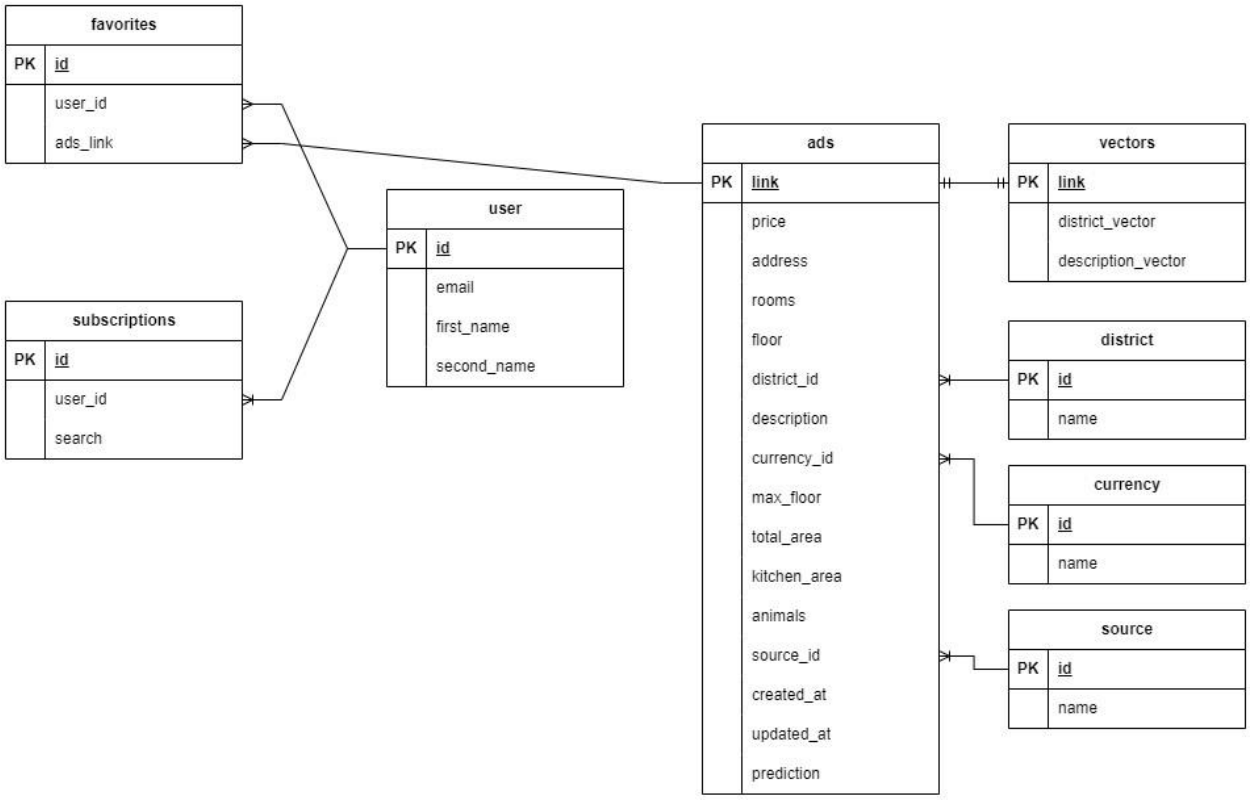


Рисунок 3.4 – Модель бази даних

Опис таблиць бази даних наведено у таблицях 3.2 - 3.10.

Таблиця 3.2 – Опис таблиці user

Назва поля	Тип даних	Опис
email	text	Email користувача
first_name	text	Ім’я користувача
last_name	text	Прізвище користувача

Таблиця 3.3 – Опис таблиці ads

Назва поля	Тип даних	Опис
link	text	Посилання на оголошення
price	int	Ціна в оголошенні
address	text	Адреса нерухомості
rooms	int	Кількість кімнат
floor	int	Поверх
district_id	int	ID району
description	text	Опис
currency_id	int	ID валюти в якій зазначена ціна
max_floor	int	Кількість поверхів у будинку
total_area	float	Загальна площа квартири
kitchen_area	float	Площа кухні
animals	bool	Чи можна з тваринами
source_id	int	ID джерела оголошення
created_at	timestamp	Час створення оголошення у сервісі
updated_at	timestamp	Час останнього оновлення
prediction	float	Ціна передбачена моделлю

Таблиця 3.4 – Опис таблиці favorites

Назва поля	Тип даних	Опис
id	int	Унікальний ідентифікатор
user_id	text	Ім’я користувача
ads_link	text	Посилання на оголошення

Таблиця 3.5 – Опис таблиці district

Назва поля	Тип даних	Опис
id	int	Унікальний ідентифікатор
name	text	Назва району

Таблиця 3.6 – Опис таблиці vectors

Назва поля	Тип даних	Опис
link	text	Посилання на оголошення
district_vector	float[]	Вектор району
description_vector	float[]	Вектор опису

Таблиця 3.7 – Опис таблиці subscriptions

Назва поля	Тип даних	Опис
id	int	Унікальний ідентифікатор
user_id	text	Електронна пошта користувача
search	text	Збережений пошук

Таблиця 3.8 – Опис таблиці currency

Назва поля	Тип даних	Опис
id	int	Унікальний ідентифікатор
name	text	Назва валюти

Таблиця 3.9 – Опис таблиці currency

Назва поля	Тип даних	Опис
id	int	Унікальний ідентифікатор
name	text	Назва валюти

Таблиця 3.10 – Опис таблиці source

Назва поля	Тип даних	Опис
id	int	Унікальний ідентифікатор
name	text	Назва джерела даних

Опис утиліт, бібліотек та іншого стороннього програмного забезпечення в таблиці 3.11.

Таблиця 3.11 – Опис утиліт

№ п/п	Назва утиліти	Опис застосування
1	PyCharm	Головне середовище розробки програмного забезпечення.
2	Jupyter Notebook	Середовище для роботи з даними, навчання моделей
3	TablePlus	Програмне забезпечення яке надає легкий графічний інтерфейс для доступу до бази даних.

Утиліти були використані при розробці програмного забезпечення.

3.4 Аналіз безпеки даних

Для аутентифікації використано Auth0. Персональні дані користувача передаються виключно у хедері. При використанні протоколу HTTPS даний підхід є безпечним.

Програмне забезпечення відповідає Загальному регламенту із захисту даних. Дані про звичайних користувачів, не збираються.

Висновки до розділу

У ході моделювання та конструювання програмного забезпечення було розроблено архітектуру програмного забезпечення. Визначено основні

програмні модулі, їх функціональні можливості та інтерфейси для взаємодії між собою.

Проведено аналіз інструментів та технологій для розробки програмного забезпечення. Обґрунтовано вибір фреймворків, бібліотек та середовищ розробки, виходячи з вимог до продуктивності, безпеки та зручності використання.

Реалізовано основні функціональні можливості системи, включаючи обробку даних, управління користувачами. Також були протестовані алгоритми, які використовувались для навчання моделі, обрано оптимальний алгоритм та навчена модель оцінки оголошення.

Обрано систему управління базою даних та розроблено ER діаграму, що описує базу даних. Описано основні сутності бази даних.

Прийнято міри для захисту даних у програмному забезпеченні.

4 АНАЛІЗ ЯКОСТІ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Аналіз якості ПЗ

Аналіз якості програмного забезпечення – це процес оцінки, який визначає, наскільки програмне забезпечення відповідає встановленим стандартам якості, вимогам і очікуванням користувачів. Цей процес охоплює оцінку різних характеристик програмного забезпечення, таких як функціональність, продуктивність, надійність, безпека, зручність використання та інші аспекти.

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- перевірка збереження даних;
- перевірка коректності обчислень;
- знаходження проблем, помилок і недоліків з метою їх усунення;
- знаходження вразливостей, що можуть вплинути на безпеку;
- перевірка зручності програмного інтерфейсу.

Метриками для оцінки якості ПЗ обрано наступні:

- читабельність;
- портативність;
- надійність;
- безпека;
- зручність використання.

Для розрахунку даних метрик обрано сервіс Embold, що оцінює необхідні метрики, базуючись на коді програмного забезпечення. Результати аналізу представлені на рисунках 4.1 – 4.2.

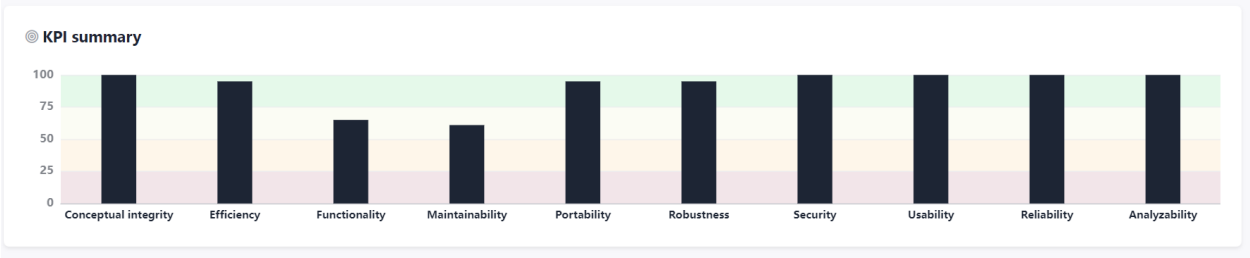


Рисунок 4.1 – Оцінка метрик якості програмного забезпечення

Рисунок 4.2 – Загальна оцінка коду

Знайдено 18 проблем, з яких 8 мають середній рівень загрози, 10 низький рівень загрози. Помилки не впливають на стабільність та безпеку застосунку.

4.2 Опис процесів тестування

Тестування виконується згідно документу «Програма та методика тестування».

Проведено мануальне тестування програмного забезпечення. Опис відповідних тестів наведено у таблицях 4.1 – 4.10.

Таблиця 4.1 – Тест 1.1 Реєстрація користувача

Початковий стан системи	Користувач не зареєстрований у системі.
Вхідні дані	Електронна пошта, пароль.
Опис проведення тесту	Надсилається запит з такими параметрами: коректна електронна пошта, яка до цього не була зареєстрована в системі, пароль від 8 символів, який містить хоча б з одну велику та маленьку літери і одне число.
Очікуваний результат	Користувач отримує повідомлення про успішну реєстрацію. Користувача додано до системи.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.2 – Тест 1.2 Некоректна реєстрація користувача

Початковий стан системи	Користувач не зареєстрований у системі.
Вхідні дані	Електронна пошта, пароль.
Опис проведення тесту	Надсилається запит з поштою, що введена в некоректному форматі (test.gmail.com). Пароль пароль від 8 символів, який містить хоча б з одну велику та маленьку літери і одне число.
Очікуваний результат	Користувач отримує повідомлення про некоректну пошту. Новий обліковий запис у системі не створюється.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.3 – Тест 1.3 Авторизація користувача

Початковий стан системи	Користувач зареєстрований.
Вхідні дані	Електронна пошта, пароль.
Опис проведення тесту	Надсилається запит з поштою та паролем, що відповідають зареєстрованому у системі користувачу.
Очікуваний результат	Користувач отримує повідомлення з токеном тимчасового доступу.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.4 – Тест 1.4 Пошук актуальних оголошень

Початковий стан системи	Користувач зареєстрований.
Вхідні дані	Токен доступу, перелік фільтрів для оголошень.
Опис проведення тесту	Надсилається запит з токеном доступу у хедері та таким списком фільтрів: query=Борщагівська, animal=true, district=Солом’янський. Фільтри передаються як параметри запиту.
Очікуваний результат	Користувач отримує повідомлення зі списком оголошень. Повідомлення включає в себе посилання на оголошення, перелік усіх атрибутів оголошення (ціна, опис, район, площа тощо), ціну, що була передбачена сервісом. Повідомлення включає тільки ті оголошення, що підходять під фільтри. Усі оголошення є актуальними.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.5 – Тест 1.5 Статистика оголошень

Початковий стан системи	Користувач зареєстрований.
Вхідні дані	Токен доступу, перелік фільтрів для оголошень, параметр групування та метрика.
Опис проведення тесту	Надсилається запит з токеном доступу у хедері та таким списком фільтрів: animal=true, district=Солом'янський. Параметр групування week, метрика median.
Очікуваний результат	Користувач отримує повідомлення зі списком тижнів. Для кожного тижня виводиться медіана ціни для тих оголошень, що підпадають під фільтри.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.6 – Тест 1.6 Пошук схожих оголошень

Початковий стан системи	Користувач зареєстрований.
Вхідні дані	Токен доступу, посилання на оголошення.
Опис проведення тесту	Надсилається запит з токеном доступу у хедері та посиланням на оголошення. Посилання є валідним. Оголошення існує в сервісі.
Очікуваний результат	Користувач отримує повідомлення зі списком з 10 схожих актуальних оголошень.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.7 – Тест 1.7 Підписка на щоденну розсилку

Початковий стан системи	Користувач зареєстрований.
Вхідні дані	Токен доступу, електронна пошта, фільтри пошуку.
Опис проведення тесту	Надсилається запит з токеном доступу у хедері, поштою та таким списком фільтрів: max_price=15000, district=Печерський.
Очікуваний результат	Користувач отримує повідомлення про успішну підписку. Щодня о 9:00 користувач отримує список нових актуальних квартир, що підходять під введені фільтри.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.8 – Тест 1.8 Оцінка введеної квартири

Початковий стан системи	Користувач зареєстрований.
Вхідні дані	Токен доступу, дані про квартиру.
Опис проведення тесту	Надсилається запит з токеном доступу у хедері та такими параметрами: кількість кімнат, поверх, площа, площа кухні, район та опис. Дані взято з реального оголошення.
Очікуваний результат	Користувач отримує повідомлення з оцінкою введеної квартири. Оцінка є близькою до реальної ціни в оголошенні.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.9 – Тест 1.9 Пошук оголошень, схожих на вручну введену квартиру

Початковий стан системи	Користувач зареєстрований.
Вхідні дані	Токен доступу, дані про квартиру.
Опис проведення тесту	Надсилається запит з токеном доступу у хедері та такими параметрами: кількість кімнат, поверх, площа, площа кухні, район та опис.
Очікуваний результат	Користувач отримує повідомлення з 10 актуальними оголошеннями, що схожі на введену квартиру.
Фактичний результат	Збігається з очікуваним.

Таблиця 4.10 – Тест 1.10 Некоректна авторизація користувача

Початковий стан системи	-
Вхідні дані	Електронна пошта, пароль.
Опис проведення тесту	Надсилається запит з поштою, що є в сервісі, та неправильним паролем.
Очікуваний результат	Користувач отримує повідомлення з помилкою, де написано що введені дані є некоректними.
Фактичний результат	Збігається з очікуваним.

Результат усіх тестів збігся з очікуваним, що підтверджує покриття потрібної функціональності розробленим ПЗ.

4.3 Опис контрольного прикладу

Для реєстрації користувача надсилається запит з електронною поштою, паролем, ім’ям та прізвищем (рис. 4.3 - 4.4).

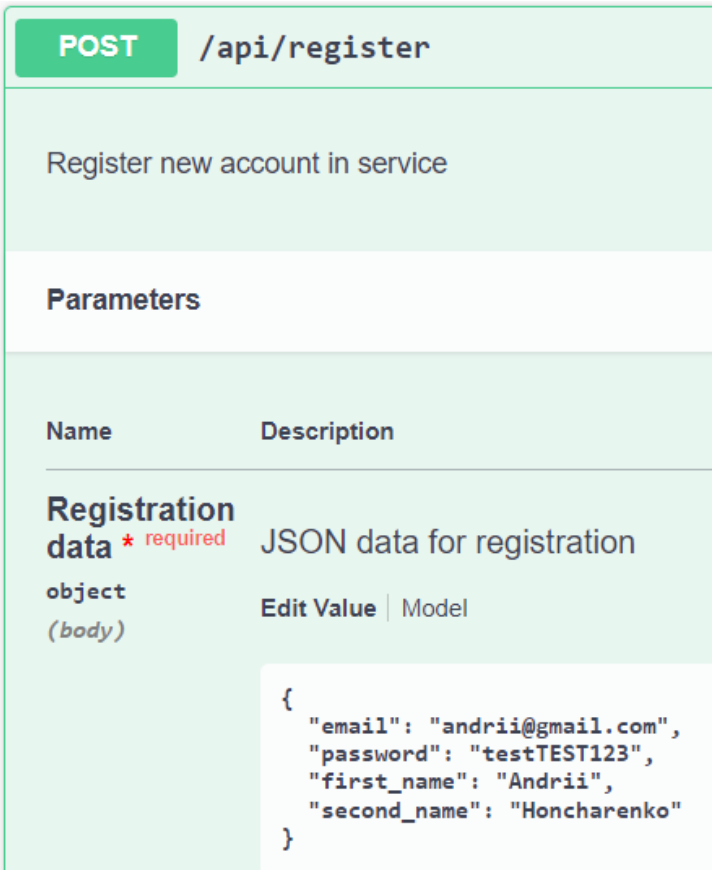


Рисунок 4.3 – Запит на реєстрацію

```
{
  "created_at": "2024-05-29T19:28:53.434Z",
  "email": "andrii@gmail.com",
  "email_verified": false,
  "identities": [
    {
      "connection": "Username-Password-Authentication",
      "user_id": "665781f5b1fb2d3be86d382e",
      "provider": "auth0",
      "isSocial": false
    }
  ],
  "name": "andrii@gmail.com",
  "nickname": "andrii",
  "picture": "https://s.gravatar.com/avatar
    /8d7f8f96834d152aa4f1ab35a7e8ebd4?s=480&r=pg&d
    =https%3A%2F%2Fcdn.auth0.com%2Favatars%2Faa.png",
  "updated_at": "2024-05-29T19:28:53.434Z",
  "user_id": "auth0|665781f5b1fb2d3be86d382e"
}
```


Рисунок 4.4 – Результат запиту на реєстрацію

Авторизація виконується за допомогою логіну та паролю (рис. 4.5 - 4.6).

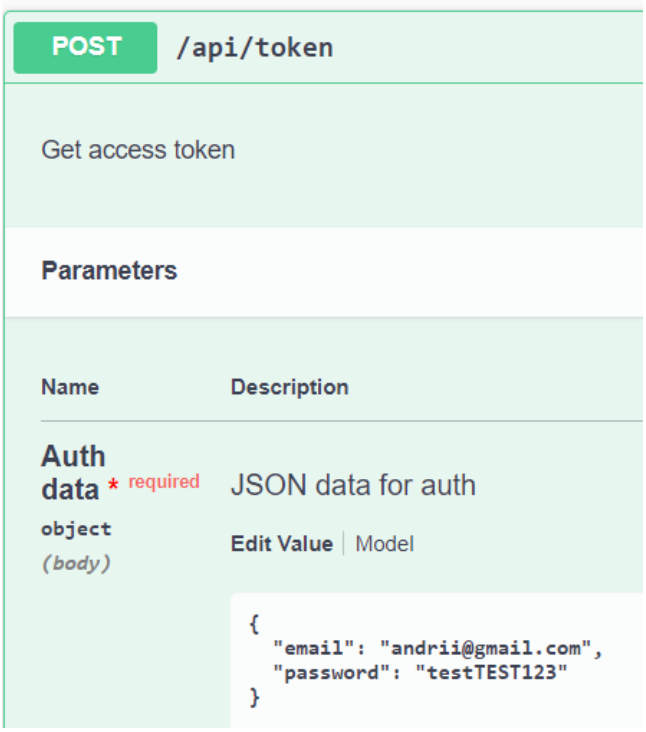


Рисунок 4.5 – Запит на отримання токену доступу

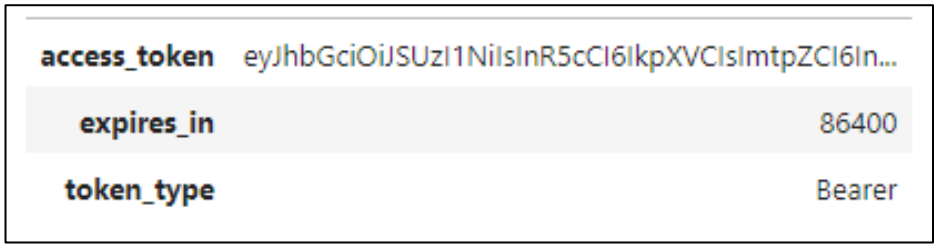


Рисунок 4.6 – Результат запиту на отримання токену доступу

Користувач робить запит на пошук актуальних оголошень (рис. 4.7 - 4.8).

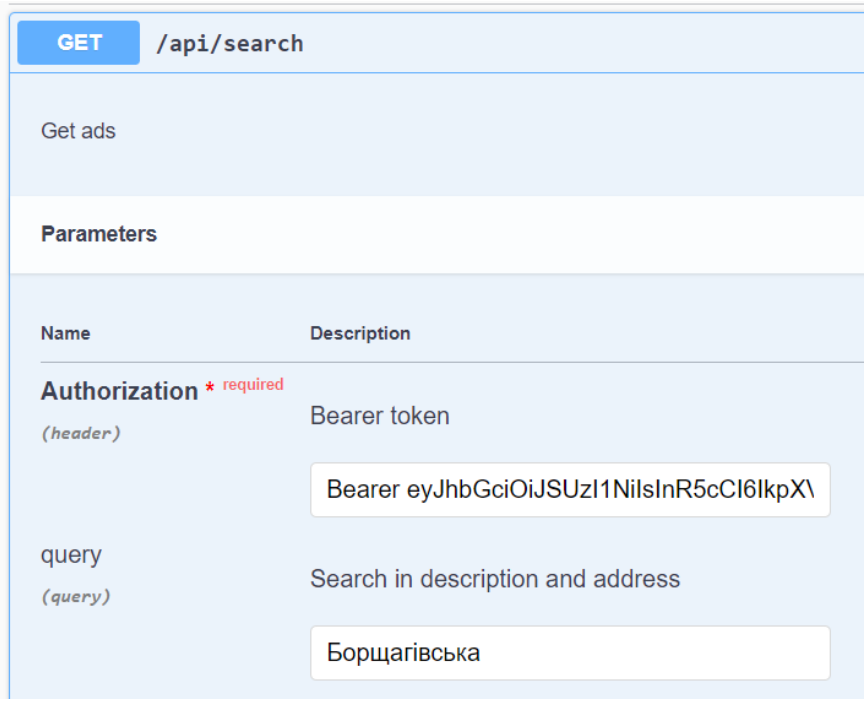


Рисунок 4.7 – Запит на пошук оголошень за фільтрами

price	rooms	floor	district	description	building_floors	total_area	kitchen_area	animals	source	prediction	price_diff
8000	1	3	Шевченківський	Сдам 1к.кв. Метро Политехнический Институт ул....	9	35	7.0	0.0	rieltor	11250.026081	-28.889054
14000	1	4	Солом'янський	Індустріальний міст Київ, Борщагівська вул., 145	16	40	9.0	0.0	rieltor	14462.316231	-3.196696
40000	1	15	Шевченківський	Оренда 1-к квартири,вул.Борщагівська,ЖК Manhat...	36	53	15.0	NaN	olx	31096.813307	28.630544
42000	2	10	Солом'янський	Оренда 2к квартири, вул. Борщагівська, ЖК Manh...	12	95	20.0	NaN	olx	30598.884469	37.259906
10000	1	5	Солом'янський	Здам 1-кіімнатну квартиру, Солом янський , Бор...	5	31	4.5	NaN	olx	6865.166348	45.662894

Рисунок 4.8 – Результат запиту на пошук оголошень

Користувач робить запит на отримання щотижневої статистики (рис. 4.9 - 4.10).

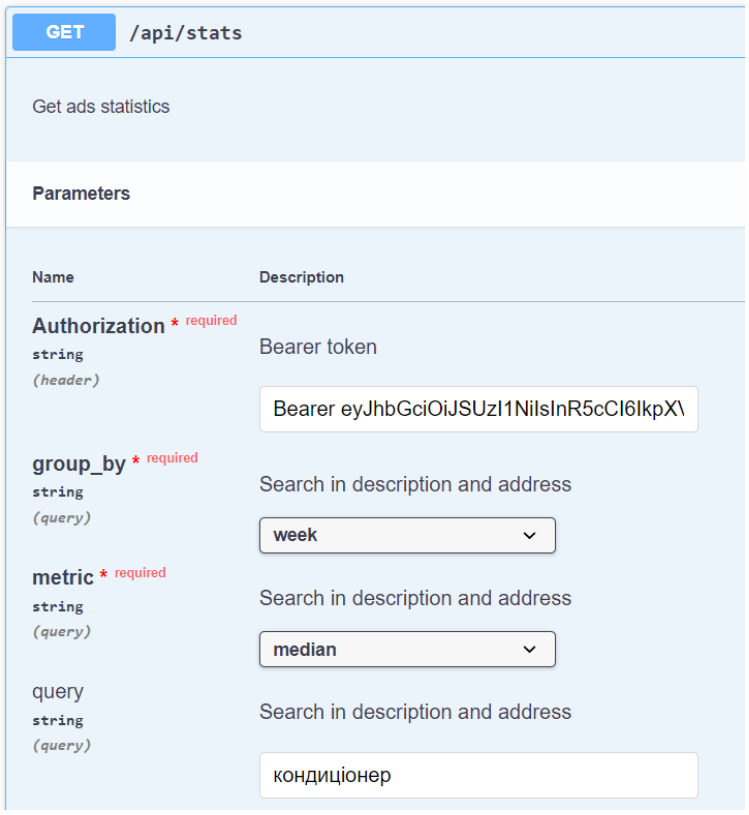


Рисунок 4.9 – Запит на отримання статистики

year	week	avg
2024	19	17860.283063
2024	20	20686.454392
2024	21	20255.754438
2024	22	26226.912935

Рисунок 4.10 – Результат запиту на отримання статистики

Користувач надсилає запит на пошук схожих квартир (рис. 4.11 - 4.12).

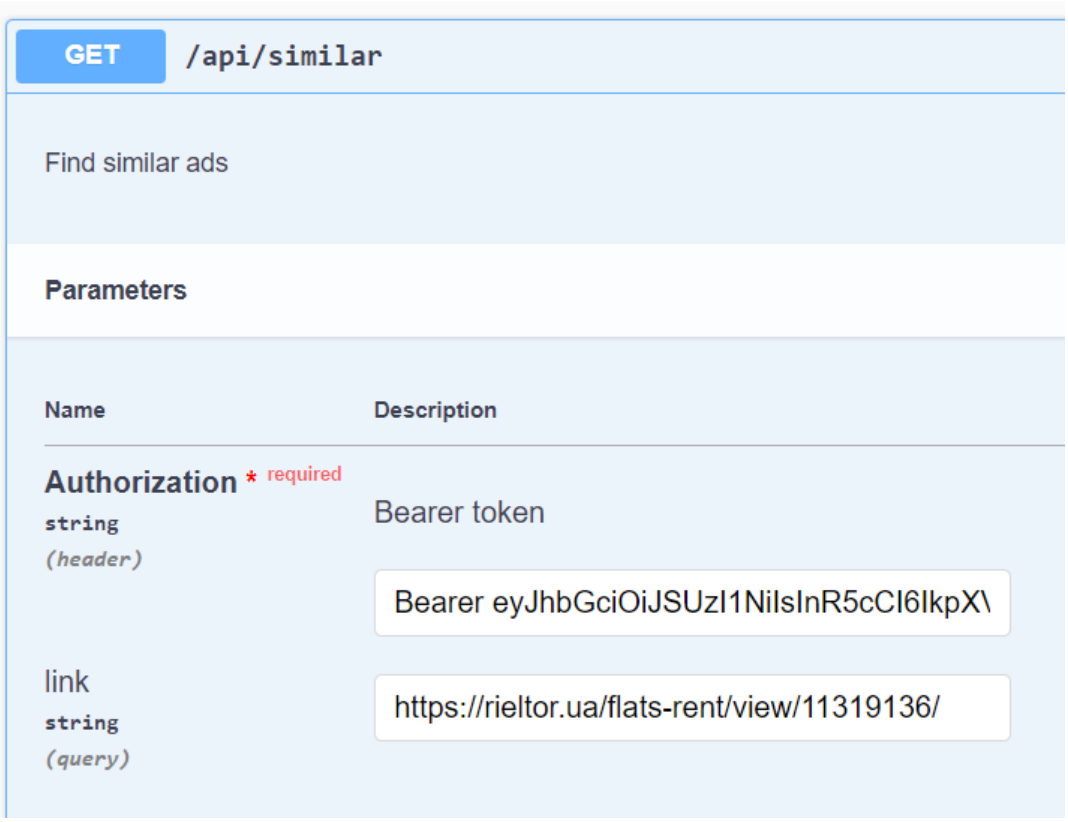


Рисунок 4.11 – Запит на отримання схожих оголошень

price	rooms	floor	district	description	building_floors	total_area	kitchen_area	animals	source	prediction	price_diff
23000	3	9	Дарницький	Актуальна и доступна у заезду\nУл. Днепровская...	23	120	20.0	0.0	rieltor	48981.497088	-53.043493
80000	3	2	Печерський	Оренда трикімнатної квартири в клубному будинк...	6	145	62.0	0.0	rieltor	71430.225167	11.997407
15000	2	15	Дарницький	Вільна! Є відео.\n\nпроспект Миколи Бажана 7\n...	17	60	11.0	1.0	rieltor	12892.963104	16.342534
72000	3	7	Печерський	Оренда квартири 115м2, ЖК Новопечерські Липки ...	22	115	40.0	NaN	olx	133776.690277	-46.178964
26000	1	18	Дарницький	Актуально! Можливі перегляди!\n\nПропонуємо в ...	25	51	10.0	0.0	rieltor	27605.958432	-5.817434
40000	2	16	Шевченківський	Перша здача!!!\n3атишна та стильна квартира у ...	36	54	22.0	0.0	rieltor	40523.100265	-1.290869
30000	1	19	Дарницький	Оренда однокімнатної квартири в сучасному житл...	24	47	13.7	1.0	rieltor	24054.395839	24.717329
72000	3	5	Печерський	Пропонується в довгострокову оренду квартира н...	13	90	40.0	0.0	rieltor	76044.642611	-5.318774
17000	3	14	Дніпровський	3х кімнатаквартира по адресу Пантелеймона Кулі...	16	69	23.0	1.0	rieltor	18125.399788	-6.208965
60000	2	17	Голосіївський	Предлагаю в аренду 2 комнатную квартиру по адр...	23	90	25.0	0.0	rieltor	76021.270183	-21.074720

Рисунок 4.12 – Результат запиту на отримання схожих оголошень

Користувач робить запит на підписку на щоденну розсилку (рис. 4.13 - 4.14).

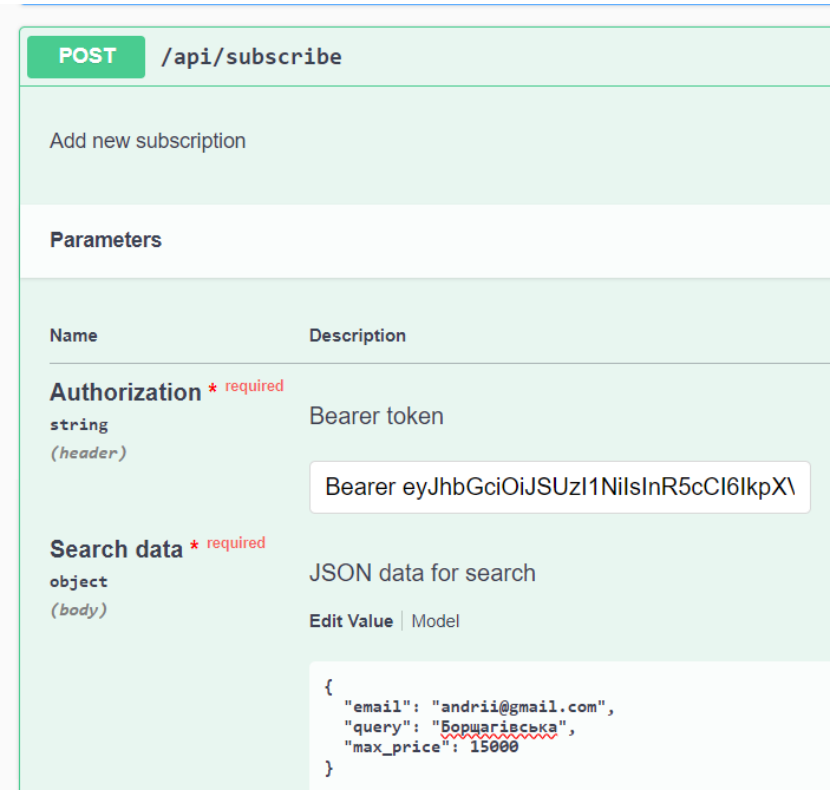


Рисунок 4.13 – Запит на підписку на щоденну розсилку

```
{
  "status": "success"
}
```


Рисунок 4.14 – Результат запиту на підписку

Користувач вводить дані про квартиру для отримання її оцінки від моделі (рис. 4.15 - 4.16).

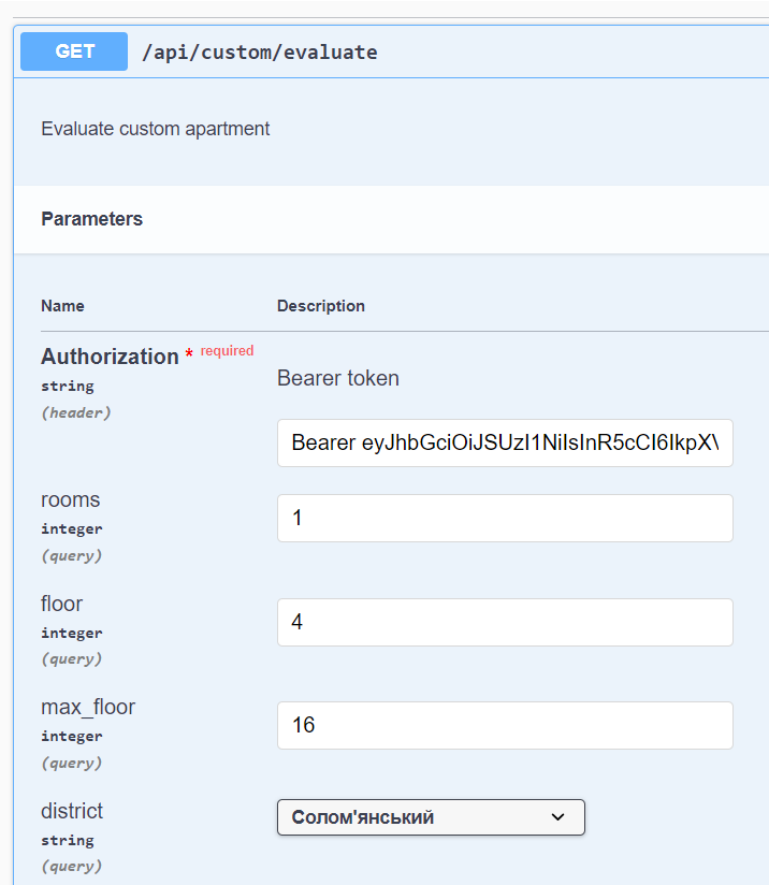


Рисунок 4.15 – Запит на оцінку квартири


```
{
  "prediction": 12629.75
}
```


Рисунок 4.16 – Результат запиту на оцінку квартири

Корисутвач робить запит на отримання оголошень, схожих на квартиру, що введена вручну (рис. 4.17 - 4.18).

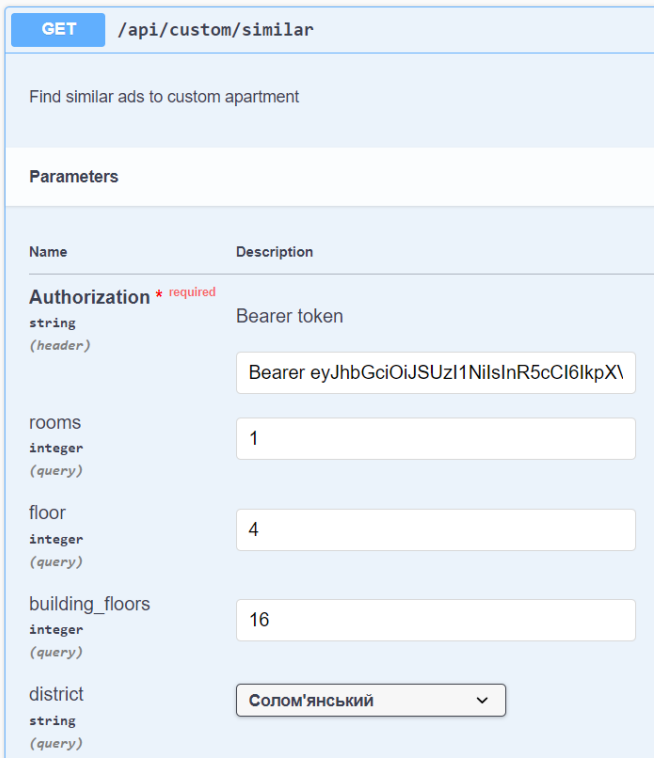


Рисунок 4.17 – Запит на отримання схожих оголошень

price	rooms	floor	district	description	building_floors	total_area	kitchen_area	animals	source	prediction	price_diff
30000	2	11	Печерський	Дипломат-Холл, Жилянська 59, м. Університет, В...	13	75	25.0	NaN	olx	50412.944641	-40.491475
6000	1	4	Деснянський	Оренда просторої 40 кв.м 1-кімн. квартири на п...	18	40	9.0	NaN	olx	7146.940468	-16.047992
52000	2	5	Печерський	Актуально! Можливі перегляди!\n\nПропонуємо в ...	7	70	27.0	0.0	rieltor	53452.643241	-2.717627
48000	2	7	Печерський	Оренда 2к 48м2, ЖК Новопечерські Липки, ул. Др...	18	48	9.0	NaN	olx	45094.312577	6.443578
120000	3	5	Печерський	Оренда 3-к квартири в ЖК Бульвар Фонтанів\nПро...	14	100	20.0	NaN	olx	98377.938072	21.978568
25000	1	4	Голосіївський	До вашої уваги пропонується :\nОренда Кухня-ст...	9	42	6.0	0.0	rieltor	19745.941055	26.608299
48000	2	5	Печерський	Здається в довгострокову оренду 2к квартира в ...	20	50	20.0	1.0	rieltor	36868.484567	30.192495
44000	2	7	Печерський	Оренда 2к квартири 90м2, ЖК Новопечерські Липк...	22	90	15.0	NaN	olx	70972.956378	-38.004555
60000	3	5	Шевченківський	Шевченківський р-н, вул. Богдана Хмельницького...	6	130	13.4	0.0	rieltor	59982.286297	0.029532
12000	1	16	Дарницький	Сдам 1 ком кв новострой ул. Ващенко, м. Осокор...	20	45	10.0	NaN	olx	9680.869871	23.955803

Рисунок 4.18 – Результат запиту на отримання схожих оголошень

Висновки до розділу

В ході аналізу якості та тестування програмного забезпечення зроблено загальну оцінку якості веб застосунку.

Для оцінки якості програмного забезпечення застосовано сервіс Embold. Цей інструмент аналізу виявляє проблеми в коді, включаючи антипатерни та повторення. Було представлено результати аналізу.

Проведено мануальне тестування методом послідовного виконання кроків і порівняння реальних результатів з очікуваними.

У результаті тестування встановлено, що додаток відповідає основним вимогам і демонструє заявлену функціональність та надійність.

Наведено контрольний приклад, що показує звичайний спосіб використання застосунку клієнтом з усіма розгалуженнями.

5 РОЗГОРТАННЯ ТА СУПРОВІД ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Розгортання програмного забезпечення

Для розгортання обрано Docker так як він ізолює кожен «контейнер», через що застосунок не може нашкодити основній (хостовій) системі, дає змогу контролювати версії білду. Також контейнер є готовою середою для запуску програми, що дозволяє завантажити готовий контейнер на будь-який сервер, що підтримує контейнеризацію, і бути впевненим, що програма запуститься нормально, що додає гнучкості. Також докер контейнер займає менше пам’яті ніж повноцінна віртуальна машина.

Для розгортання веб застосунку, скраперу, бази даних та сервісу відправки електронних листів був написаний docker-compose файл, що додає усі потрібні файли, встановлює бібліотеки та запускає усі компоненти програмного забезпечення.

Контейнери працюють у одній мережі для простоти зв’язку між компонентами.

Розгорнуті контейнери можна побачити на рисунку 5.1.

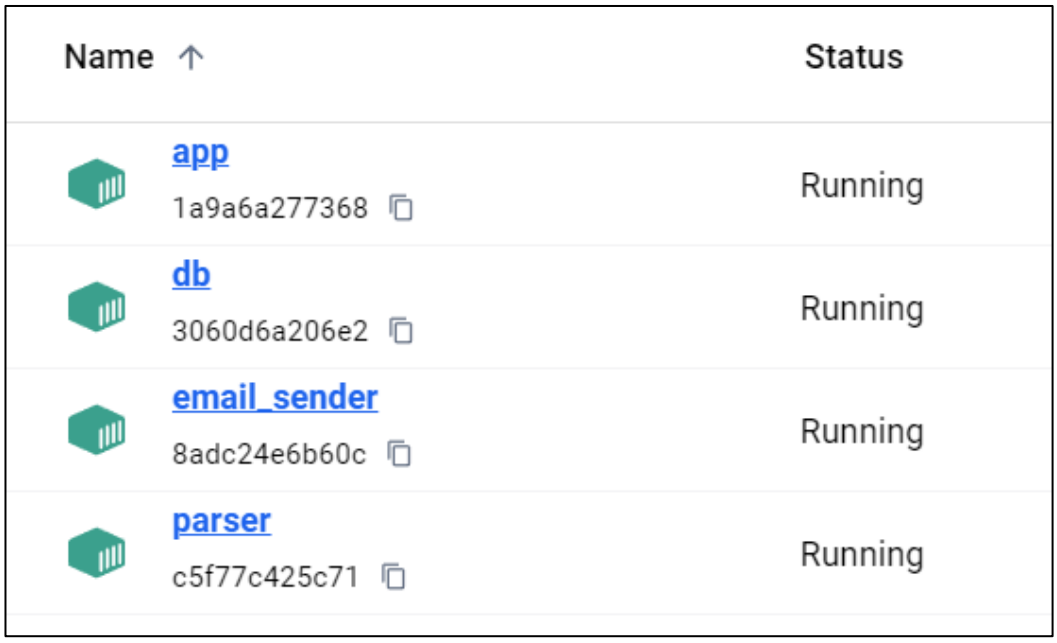


Рисунок 5.1 – Розгорнуті контейнери

5.2 Супровід програмного забезпечення

Інструкція користувача наведена в окремому документі.

Підтримку даного застосунку можна розділити на три частини: підтримка веб застосунку, підтримка скраперу, розробка нових версій моделі.

Для зміни веб застосунку необхідно знову білдити представлення програми та на його основі запускати контейнер.

Для підтримки скраперу можна оновлювати статуси джерел через адмін панель або, у випадку необхідності розробки інших методів збору інформації, використати метод аналогічний методу для веб застосунку.

Оновлення моделей машинного навчання відбувається за допомогою розробки нових моделей та зберігання їх дампів у відповідній дерикторії.

Адміністрування бази даних може виконуватись за допомогою сторонніх програм, наприклад TablePlus.

Висновки до розділу

В ході розробки впровадження та супроводу програмного забезпечення виконано розгортання застосунку, скраперу та бази даних. Розроблено docker-compose файл для швидкого розгортання застосунку.

Було описано процес підтримки програмного забезпечення, створення нових версій API, скраперу, моделі та адміністрування бази даних.

6 ВИСНОВКИ

У ході роботи над аналізом предметної області проаналізовано існуючі програмні продукти, порівняно їх з дипломним проєктом. Проведено аналіз відомих алгоритмічних рішень, обрано алгоритми навчання моделей штучного інтелекту. Описано бізнес процеси, сформовано задачі.

У ході аналізу вимог до програмного забезпечення спроектовано вимоги до функцій додатку. Створено діаграму варіантів використання. Детально описані варіанти використання сервісу користувачем. Розроблено функціональні та нефункціональні вимоги та скомпоновано у модель вимог. Розроблена матриця трасування вимог, яка показує використання функціональних вимог у варіантах використання.

У ході конструювання та розроблення програмного забезпечення розроблено архітектуру програмного забезпечення. Описано алгоритми, які використовувались для навчання моделі. Обрано СУБД, спроєктовано ER діаграму бази даних та описано основні сутності. Обґрунтовано вибір утиліт, бібліотек, фреймворків, що використовувалися при розробці.

В ході аналізу якості та тестування програмного забезпечення зроблено загальну оцінку якості застосунку. Проведене мануальне тестування. Усі тести були пройдені успішно, що підтверджує заявлену функціональність даного застосунку. Описаний контрольний приклад, що показує звичайний спосіб використання застосунку клієнтом з усіма розгалуженнями.

В ході розробки впровадження та супроводу програмного забезпечення виконано розгортання програмного забезпечення. Розроблено docker-compose для швидкої розгортки застосунку. Описано процес підтримки програмного забезпечення.

В результаті спроєктовано програмне забезпечення для агрегації та аналізу оголошень про оренду квартир. Робота містить API, скрапер, сервіс відправки електронних листів та базу даних.

В якості середовища розробки веб застосунку обрано PyCharm. Для роботи з даними та розробки моделей використано Jupyter Notebook. У якості СУБД використано PostgreSQL.

Створене програмне забезпечення повністю відповідає усім поставленим вимогам, пройшов усі необхідні тестування та показав високу якість реалізації. Впровадження даного застосунку може покращити процес пошуку оголошень про оренду квартир.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Навіщо потрібні сайти-агрегатори [Електронний ресурс]. – Режим доступу до ресурсу: https://jam.in.ua/blog/navishcho-potribni-sajty-ahrehatory-chomu-google-ikh-tak-liubyt/#:~:text=%D0%A1%D0%B0%D0%B9%D1%82%2D%D0%B0%D0%B3%D1%80%D0%B5%D0%B3%D0%B0%D1%82%D0%BE%D1%80%20%E2%80%94%20%D0%B2%D0%B5%D0%B1%2D%D1%80%D0%B5%D1%81%D1%83%D1%80%D1%81,%D1%8F%D0%BA%D1%96%20%D0%BF%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D1%96%20%D0%BD%D0%B0%20%D1%96%D0%BD%D1%82%D0%B5%D1%80%D0%BD%D0%B5%D1%82%2D%D0%BC%D0%B0%D0%B9%D0%B4%D0%B0%D0%BD%D1%87%D0%B8%D0%BA%D1%83 (дата звернення: 02.06.2024).
2) Штучний інтелект, машинне навчання та нейронні мережі: в чому різниця і для чого їх використовують [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://evergreens.com.ua/ua/articles/machine-learning-overview.html (дата звернення: 02.06.2024).
3) NLP [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://www.ibm.com/topics/natural-language-processing (дата звернення: 02.06.2024).
4) Mitchell R. Web Scraping with Python: Collecting Data from the Modern Web / Ryan Mitchell., 2015.
5) Bird AI [Електронний ресурс]. – Режим доступу до ресурсу: https://birdrent.com/ (дата звернення: 02.06.2024).
6) ЛУН [Електронний ресурс]. – Режим доступу до ресурсу: https://lun.ua/
7) What Is Word2Vec and How Does It Work? [Електронний ресурс]. – Режим доступу до ресурсу: https://swimm.io/learn/large-language-models/what-is-word2vec-and-how-does-it-work#:~:text=Technically%2C%20Word2Vec%20is%20a%20two,corresponding%20vector%20in%20the%20space (дата звернення: 02.06.2024).

8) Coefficient of Determination: How to Calculate It and Interpret the Result [Електронний ресурс]. – Режим доступу до ресурсу: https://www.investopedia.com/terms/c/coefficient-of-determination.asp (дата звернення: 02.06.2024).
9) CatBoost in Machine Learning [Електронний ресурс]. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/catboost-ml/ (дата звернення: 02.06.2024).
10) What is random forest? [Електронний ресурс]. – Режим доступу до ресурсу: https://www.ibm.com/topics/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems (дата звернення: 02.06.2024).
11) Basic regression: Predict fuel efficiency [Електронний ресурс]. – Режим доступу до ресурсу: https://www.tensorflow.org/tutorials/keras/regression (дата звернення: 02.06.2024).
12) What is a RESTful API? [Електронний ресурс]. – Режим доступу до ресурсу:https://aws.amazon.com/what-is/restful-api/ (дата звернення: 02.06.2024).
13) PostgreSQL [Електронний ресурс]. – Режим доступу до ресурсу: https://www.postgresql.org/ (дата звернення: 02.06.2024).
14) Vincent W. Django for APIs: Build web APIs with Python and Django / William S. Vincent., 2018.
15) CherryPy Framework documentation [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.cherrypy.dev/ (дата звернення: 02.06.2024).
16) Grinberg M. Flask Web Development: Developing Web Applications with Python / Miguel Grinberg., 2018.
17) Requests documentation [Електронний ресурс]. – Режим доступу до ресурсу: https://pypi.org/project/requests/ (дата звернення: 02.06.2024).

18) Beautiful soup documentation [Електронний ресурс]. – Режим доступу до ресурсу: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата звернення: 02.06.2024).

19) Pandas documentation [Електронний ресурс]. – Режим доступу до ресурсу: https://pandas.pydata.org/ (дата звернення: 02.06.2024).

20) NumPy [Електронний ресурс]. – Режим доступу до ресурсу: https://numpy.org/ (дата звернення: 02.06.2024).

21) Scikit-Learn [Електронний ресурс]. – Режим доступу до ресурсу: https://scikit-learn.org/ (дата звернення: 02.06.2024).

22) CatBoost - open-source gradient boosting library [Електронний ресурс]. – Режим доступу до ресурсу: https://catboost.ai/ (дата звернення: 02.06.2024).

23) Keras: Deep Learning for humans [Електронний ресурс]. – Режим доступу до ресурсу: https://keras.io/ (дата звернення: 02.06.2024).

24) Manning C. Foundations of Statistical Natural Language Processing / C. Manning, H. Schütze., 1999.

25) Goldberg Y. Neural Network Methods for Natural Language Processing / Yoav Goldberg., 2017.

26) Embedding layer [Електронний ресурс]. – Режим доступу до ресурсу: https://keras.io/api/layers/core_layers/embedding/ (дата звернення: 02.06.2024).

27) Word2vec embeddings [Електронний ресурс]. – Режим доступу до ресурсу: https://radimrehurek.com/gensim/models/word2vec.html (дата звернення: 02.06.2024).

28) PyCharm [Електронний ресурс]. – Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/pycharm/ (дата звернення: 02.06.2024).

29) Python IDLE [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.python.org/3/library/idle.html (дата звернення: 02.06.2024).

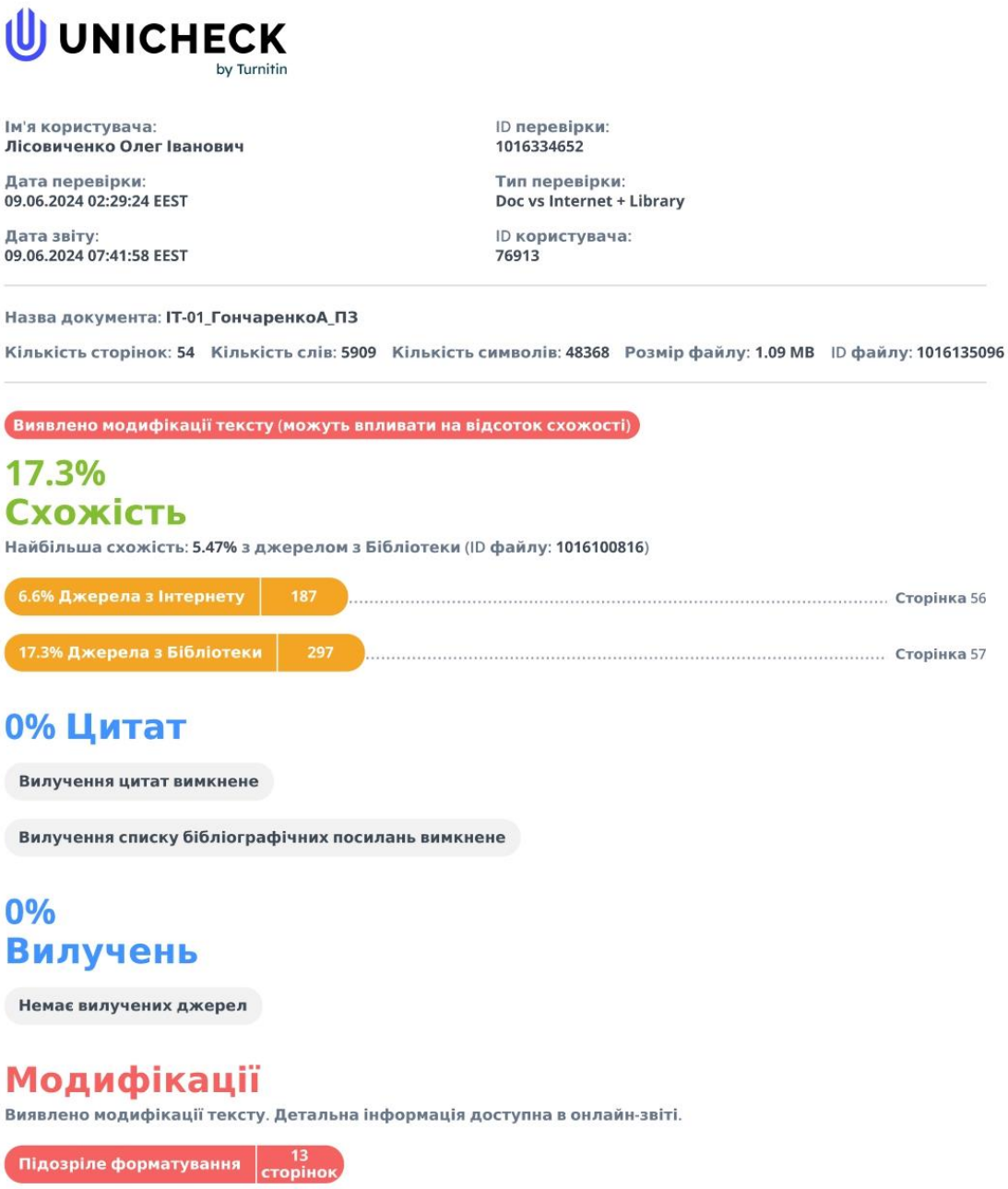
30) Visual Studio Code [Електронний ресурс]. – Режим доступу до ресурсу: https://code.visualstudio.com/ (дата звернення: 02.06.2024).

31) Jupyter Notebook [Електронний ресурс]. – Режим доступу до ресурсу: https://jupyter.org/ (дата звернення: 02.06.2024).

32) Google Colab [Електронний ресурс]. – Режим доступу до ресурсу: https://colab.research.google.com/ (дата звернення: 02.06.2024).

33) Table Plus [Електронний ресурс]. – Режим доступу до ресурсу: https://tableplus.com/ (дата звернення: 02.06.2024).

ДОДАТОК А ЗВІТ ПОДІБНОСТІ



UNICHECK
by Turnitin

Ім'я користувача:
Лісовиченко Олег Іванович

ID перевірки:
1016334652

Дата перевірки:
09.06.2024 02:29:24 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
09.06.2024 07:41:58 EEST

ID користувача:
76913

Назва документа: ІТ-01_ГончаренкоА_ПЗ

Кількість сторінок: 54 Кількість слів: 5909 Кількість символів: 48368 Розмір файлу: 1.09 MB ID файлу: 1016135096

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

17.3%
Схожість

Найбільша схожість: 5.47% з джерелом з Бібліотеки (ID файлу: 1016100816)

6.6% Джерела з Інтернету 187 ..... Сторінка 56

17.3% Джерела з Бібліотеки 297 ..... Сторінка 57

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 13 сторінок

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

Сервіс агрегації та аналізу даних про аренду житла

Текст програми

КПІ.ІТ-0104.045440.03.12

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Андрій ГОНЧАРЕНКО

Київ – 2024

Посилання на репозиторій з повним текстом програмного коду
https://github.com/chikichunchik/diplom

Файл rieltor.py
Реалізація функціональної вимоги періодичного збору оголошень.

```
import datetime
import sys
import traceback

import pandas
import asyncio
import aiohttp
import pandas as pd
import requests
import sqlalchemy
from bs4 import BeautifulSoup as BS
from fake_useragent import UserAgent
from sqlalchemy.dialects.postgresql import insert
from sqlalchemy import create_engine
from logs import Logger
from predictor import Predictor

predictor = Predictor()
logger = Logger('logs.txt')
try:
    DATABASE_URL = f"postgresql://postgres:admin@172.18.0.2:5432/postgres"
    engine = create_engine(DATABASE_URL)
except:
    logger.add_log(
        message='Error on connecting to db',
        level='ERROR',
        data={'exception': traceback.format_exc()}
    )


async def rieltor():
    data = []
    staturl = "https://rieltor.ua/kiev/flats-rent/?page=1"
    agent = {'User-Agent': UserAgent().random}
    response = requests.get(staturl)
    bs = BS(response.text, 'lxml')
    maxp = bs.find('li', {'class': 'last'}).find('a', {'class': 'pager-
btn'}).text
    first_page = 1
    last_page = int(maxp)

    for page in range(first_page, last_page + 1):
        try:
            print(page)
            async with aiohttp.ClientSession() as session:
                url = f"https://rieltor.ua/kiev/flats-rent/?page={page}"
                async with session.get(url,
                                       headers=agent) as page_response:
                    r = await
aiohttp.StreamReader.read(page_response.content)
                    page_bs = BS(r, 'lxml')
                    ads = page_bs.findAll('div', {'class': 'catalog-card'})

                    for ad in ads:
```

```
                        row = {}
                        row['link'] = ad.find('a', {'class': 'catalog-card-
media'}).get('href')
                        row['price'] = ad.find('strong', {'class': 'catalog-
card-price-title'}).text
                        row['address'] = ad.find('div', {'class': 'catalog-
card-address'}).text
                        row['rooms'] = ad.find('div', {'class': 'catalog-
card-details'}).findAll('span', {'class': ''})[
                            0].text
                        row['sizes'] = ad.find('div', {'class': 'catalog-
card-details'}).findAll('span', {'class': ''})[
                            1].text
                        row['floor'] = ad.find('div', {'class': 'catalog-
card-details'}).findAll('span', {'class': ''})[
                            2].text
                        location = ad.findAll('a', {'data-analytics-event':
'card-click-region'})
                        row['district'] = location[1].text.strip() if
len(location) > 1 else None
                        row['tags'] = [i.text.strip() for i in
                                       ad.find('div', {'class': 'catalog-
card-chips'}).findAll('a')]
                        description = ad.find('div', {'class': 'catalog-card-
description'})
                        row['description'] =
description.find('span').text.strip() if description is not None else None
                        data.append(row)
        except Exception as e:
            logger.add_log(
                message='Exception in extracting data',
                level='ERROR',
                data={'exception': traceback.format_exc()}
            )

    try:
        df = pandas.DataFrame(data)
        df['currency'] = df['price'].str.split(' ').str.get(-
1).str.split('/').str.get(0).str.strip()
        df['price'] = df['price'].str.replace(r'\D', '',
regex=True).astype('int')
        df['max_floor'] = df['floor'].str.split(' ').str.get(3).astype('int')
        df['floor'] = df['floor'].str.split(' ').str.get(1).astype('int')
        df['total_area'] = df['sizes'].str.split(' /
').str.get(0).astype('float')
        df['kitchen_area'] = df['sizes'].str.split(' /
').str.get(2).str.replace(r'[^\d\.]', '',

regex=True).astype('float')
        df['rooms'] = pd.to_numeric(df['rooms'].str.split(' ').str.get(0),
errors='coerce').fillna(1)
        df['district'] = df['district'].str.split(' ').str.get(0)
        df['animals'] = df['tags'].map(lambda x: 'Можна з тваринами' in x)
        df['source'] = 'rieltor'
        df = df.drop(columns=['sizes', 'tags'])
        print(len(df))
        load_to_db(df)
    except Exception as e:
        logger.add_log(
            message='Exception in transforming and loading data',
            level='ERROR',
            data={'exception': traceback.format_exc()}
        )
```

```
def load_to_db(df):
    df['prediction'], vectors = predictor.predict(df.copy())
    df.drop_duplicates(subset=['link']).set_index('link').to_sql('ads',
engine, if_exists='append',

method=postgres_upsert)

vectors.drop_duplicates(subset=['link']).set_index('link').to_sql('vectors',
engine, if_exists='append',

method=postgres_upsert)


def postgres_upsert(table, conn, keys, data_iter):
    data = [dict(zip(keys, row)) for row in data_iter]

    insert_statement = insert(table.table).values(data)
    upsert_statement = insert_statement.on_conflict_do_update(
        constraint=f"{table.table.name}_pkey",
        set_={c.key: c for c in insert_statement.excluded},
    )
    conn.execute(upsert_statement)


if __name__ == '__main__':
    start = datetime.datetime.now()
    loop = asyncio.get_event_loop()
    loop.run_until_complete(domria())
    print(datetime.datetime.now() - start)
```

Файл olx.py

Реалізація функціональної вимоги періодичного збору оголошень.

```
import datetime
import traceback
import requests
import pandas as pd
from bs4 import BeautifulSoup
import re
from sqlalchemy.dialects.postgresql import insert
from sqlalchemy import create_engine
from logs import Logger
from predictor import Predictor

predictor = Predictor()
logger = Logger('logs.txt')
try:
    DATABASE_URL = f"postgresql://postgres:admin@172.18.0.2:5432/postgres"
    engine = create_engine(DATABASE_URL)
except:
    logger.add_log(
        message='Error on connecting to db',
        level='ERROR',
        data={'exception': traceback.format_exc()}
    )

URL = "https://olx.ua/uk/nedvizhimost/kvartiry/dolgosrochnaya-arenda-
kvartir/kiev/"
```

```
def get_max_page():
    r = requests.get(URL)
    soup = BeautifulSoup(r.content, "html.parser")
    return int(soup.findAll('a', class_='css-1mi714g')[-1].text)


def get_links(page):
    r = requests.get(URL + f'?page={page}')
    print(r.url)
    soup = BeautifulSoup(r.content, "html.parser")
    links = []
    news_elements = soup.find_all("a", class_="css-z3gu2d")
    for element in news_elements:
        link = element["href"]
        links.append(link)
    return set(links)


def get_page_content(link):
    try:
        if link[3:5] != 'uk':
            link = link[:3] + 'uk/' + link[3:]
        full_url = "https://www.olx.ua" + link
        page = requests.get(full_url)
        soup = BeautifulSoup(page.content, "html.parser")

        name = soup.find("h4", class_="css-1juynto").text if soup.find("h4",
class_="css-1juynto") is not None else ''
        cost = soup.find("h3", class_="css-12vqlj3").text

        description = soup.find("div", class_="css-1t507yq er34gjf0").text if
soup.find("div",

class_="css-1t507yq er34gjf0") is not None else None
        district = soup.find("ol", class_='css-xv75xi').text
        tags = [i.text for i in soup.find('ul', class_='css-
sfcl1s').findAll('li')]
        return {'link': full_url, 'price': cost, 'description': name + '\n' +
description, 'district': district,
                'tags': tags}
    except:
        logger.add_log(
            message='Error on loading from olx',
            level='ERROR',
            data={'exception': traceback.format_exc()}
        )


def parse_tags(x):
    fields = {'floor': None,
              'max_floor': None,
              'total_area': None,
              'kitchen_area': None,
              'animals': None,
              'rooms': None
              }

    for value in x:
        if 'Поверх:' in value or 'Этаж:' in value:
            fields['floor'] = int(re.sub('\D', '', value))
        if 'Поверховість:' in value or 'Этажность:' in value:
            fields['max_floor'] = int(re.sub('\D', '', value))
        if 'Загальна площа:' in value or 'Общая площадь:' in value:
```

```
            fields['total_area'] = float(re.sub('[^\d\.]', '', value))
        if 'Площа кухні:' in value or 'Площадь кухни:' in value:
            fields['kitchen_area'] = float(re.sub('[^\d\.]', '', value))
        if 'Домашні улюбленці:' in value or 'Домашние питомцы:' in value:
            fields['animals'] = 'Так' in value or 'Да' in value
        if 'Кількість кімнат:' in value or 'Количество комнат:' in value:
            fields['rooms'] = int(re.sub('\D', '', value))
    return fields


def load_to_db(df):
    df['prediction'], vectors = predictor.predict(df.copy())
    df.drop_duplicates(subset=['link']).set_index('link').to_sql('ads',
engine, if_exists='append',

method=postgres_upsert)

vectors.drop_duplicates(subset=['link']).set_index('link').to_sql('vectors',
engine, if_exists='append',

method=postgres_upsert)


def postgres_upsert(table, conn, keys, data_iter):
    data = [dict(zip(keys, row)) for row in data_iter]

    insert_statement = insert(table.table).values(data)
    upsert_statement = insert_statement.on_conflict_do_update(
        constraint=f"{table.table.name}_pkey",
        set_={c.key: c for c in insert_statement.excluded},
    )
    conn.execute(upsert_statement)


def main():
    max_page = get_max_page()
    total = pd.DataFrame()
    print(max_page)
    for page in range(1, max_page + 1):
        print(page)
        links = get_links(page)
        result = []
        for link in list(links):
            content = get_page_content(link)
            if content is not None:
                result.append(content)
        try:
            result = pd.DataFrame(result)
            result['tags'] = result['tags'].map(parse_tags)
            result = pd.concat([result, pd.json_normalize(result['tags'])],
axis=1)
            result = result.drop(columns=['tags'])
            result['currency'] = result['price'].str.split(' ').str.get(-
1).str.strip().str.replace('.', '')
            result['price'] = result['price'].str.replace(r'\D', '',
regex=True).astype('int')
            result['district'] = result['district'].str.split('-').str.get(-
1).str.strip()
            result['address'] = None
            result['source'] = 'olx'
            total = pd.concat([total, result])
        except:
            logger.add_log(
                message='Error on preprocessing',
```

```
                level='ERROR',
                data={'exception': traceback.format_exc()}
            )
    try:
        load_to_db(total)
    except:
        logger.add_log(
            message='Error on loading to db',
            level='ERROR',
            data={'exception': traceback.format_exc()}
        )


if __name__ == "__main__":
    start = datetime.datetime.now()
    main()
    print(datetime.datetime.now() - start)
```

Файл predictor.py

Реалізація функціональної вимоги обчислення ринкової ціни.

```
import traceback

import pandas as pd
from nltk.corpus import stopwords
import pickle
from nltk.tokenize import RegexpTokenizer
import numpy as np
from sqlalchemy import create_engine
import nltk
from sqlalchemy.dialects.postgresql import insert
import requests
nltk.download('stopwords')

class Predictor:
    def __init__(self):
        url = 'https://raw.githubusercontent.com/olegdubetcky/Ukrainian-
Stopwords/main/ukrainian'
        r = requests.get(url)
        with open('/root/nltk_data/corpora/stopwords/ukrainian', 'wb') as f:
            f.write(r.content)
        with open('/app/catboost_model.pickle', 'rb') as f:
            self.model = pickle.load(f)
        with open('/app/districts_vectors.pickle', 'rb') as f:
            self.districts_vectors = pickle.load(f)
        with open('/app/word2vec_model.pickle', 'rb') as f:
            self.word2vec_model = pickle.load(f)

    @staticmethod
    def preprocess_description(doc_set):
        # initialize regex tokenizer
        tokenizer = RegexpTokenizer(r'\w+')
        stop =
set(stopwords.words('ukrainian')).union(set(stopwords.words('russian'))).unio
n({'та'})
        # list for tokenized documents in loop
        texts = []
        # loop through document list
        for i in doc_set:
            # clean and tokenize document string
            raw = i.lower()
            tokens = tokenizer.tokenize(raw)
```

```
            tokens = [i for i in tokens if not i in stop]
            texts.append(tokens)
        return texts

    def __transform(self, df):
        df['district'] = df['district'].fillna('None')
        df['description'] = df['description'].str.replace('.',
'').str.replace(',', '').str.replace('\n', ' ').fillna(
            'None')
        df['animals'] = df['animals'].fillna(False)

        df['description'] =
self.preprocess_description(df['description'].values)
        df_description_vector = df['description'].apply(lambda text:
                                                                np.mean(

[self.word2vec_model.wv[word] for word in text
                                                                    if
word in self.word2vec_model.wv],

axis=0))
        df_description_vector.loc[df_description_vector.isna()] =
df_description_vector.loc[df_description_vector.isna()].map(
            lambda x: self.word2vec_model.wv['квартира'])
        df_description_vector = pd.DataFrame(df_description_vector.to_list(),
columns=[f'description_{i}' for i in range(13)])
        df = pd.concat([df.reset_index(drop=True), df_description_vector],
axis=1)

        df_district_vector =
pd.DataFrame(df['district'].map(self.districts_vectors).to_list())
        df_district_vector.columns = [f'district_{i}' for i in range(4)]
        df = pd.concat([df.reset_index(drop=True), df_district_vector],
axis=1)

        return df

    def predict(self, df):
        df = self.__transform(df)
        predictors = ['rooms',
                      'floor',
                      'max_floor',
                      'total_area',
                      'kitchen_area',
                      'animals'
                  ] + [i for i in df if 'description_' in i or 'district_'
in i]

        return self.model.predict(df[predictors]), df[['link'] + [i for i in
df.columns if 'district_' in i or 'description_' in i]]


if __name__ == '__main__':
    def postgres_upsert(table, conn, keys, data_iter):
        data = [dict(zip(keys, row)) for row in data_iter]

        insert_statement = insert(table.table).values(data)
        upsert_statement = insert_statement.on_conflict_do_update(
            constraint=f"{table.table.name}_pkey",
            set_={c.key: c for c in insert_statement.excluded},
        )
        conn.execute(upsert_statement)
    pred = Predictor()
    DATABASE_URL = f"postgresql://postgres:admin@localhost:5433/postgres"
```

```
    engine = create_engine(DATABASE_URL)
    df = pd.read_sql("""SELECT * FROM ads""", engine)
    print(len(df))
    df['prediction'], vectors = pred.predict(df.copy())
    # df.set_index('link').to_sql('ads', engine, if_exists='append',
method=postgres_upsert)
    try:
        vectors.set_index('link').to_sql('vectors', engine,
if_exists='append', method=postgres_upsert)
    except Exception as e:
        with open('test.txt', 'w') as f:
            f.write(traceback.format_exc())
        raise Exception from e
```

Файл modeling.ipynb

Реалізація функціональної вимоги обчислення ринкової ціни.

```
#!/usr/bin/env python
# coding: utf-8


# In[1]:



import pandas as pd
from sklearn.metrics import r2_score
from sklearn.metrics import mean_absolute_error,
mean_absolute_percentage_error, median_absolute_error
from sqlalchemy import create_engine
from sklearn.model_selection import train_test_split
import catboost
from nltk.corpus import stopwords
import pickle
from nltk.tokenize import RegexpTokenizer
import plotly.express as px
import numpy as np
import shap
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import RandomForestRegressor
from keras.layers import Dense, Dropout, Embedding, Input, Reshape,
Concatenate
from keras.models import Model
from sklearn.preprocessing import StandardScaler
from tensorflow.keras import Sequential
from sklearn.svm import LinearSVR
```

```
# # Data preparation

# In[48]:


DATABASE_URL = f"postgresql://postgres:admin@localhost:5433/postgres"
engine = create_engine(DATABASE_URL)
df = pd.read_sql("""SELECT * FROM ads WHERE created_at <= '2024-05-05'""",
engine)


# In[49]:


df['district'] = df['district'].fillna('None')
df.loc[df['currency'] != 'PiCЂPS', 'price'] = df.loc[df['currency'] !=
'PiCЂPS', 'price'] * 40
df['description'] = df['description'].str.replace('.', '').str.replace(',',
'').str.replace('\n', ' ').fillna('None')
df['animals'] = df['animals'].fillna(False)


# In[50]:


X_train, X_test, y_train, y_test =
train_test_split(df.drop(columns=['price']), df['price'], test_size=0.33,
random_state=42,
                                                    stratify=df['district'])


# In[51]:


def preprocess_data(doc_set):
    # initialize regex tokenizer
    tokenizer = RegexpTokenizer(r'\w+')
    stop =
set(stopwords.words('ukrainian')).union(set(stopwords.words('russian'))).unio
n(set(['C,P°']))
    # list for tokenized documents in loop
    texts = []
```

```
    # loop through document list
    for i in doc_set:
        # clean and tokenize document string
        raw = i.lower()
        tokens = tokenizer.tokenize(raw)
        tokens = [i for i in tokens if not i in stop]
        texts.append(tokens)
    return texts


# In[52]:


X_train['description'] = preprocess_data(X_train['description'].values)
X_test['description'] = preprocess_data(X_test['description'].values)


# # Description vectors

# In[53]:


with open('word2vec_model.pickle', 'rb') as f:
    word2vec_model = pickle.load(f)


# In[54]:


X_train_vector = pd.DataFrame(X_train['description'].apply(lambda text:
np.mean([word2vec_model.wv[word] for word in text if word in
word2vec_model.wv], axis=0))\
                              .to_list(), columns=[f'description_{i}' for i
in range(30)])
X_test_vector = pd.DataFrame(X_test['description'].apply(lambda text:
np.mean([word2vec_model.wv[word] for word in text if word in
word2vec_model.wv], axis=0))\
                              .to_list(), columns=[f'description_{i}' for i
in range(30)])


# In[55]:
```

```
X_train = pd.concat([X_train.reset_index(drop=True), X_train_vector], axis=1)
X_test = pd.concat([X_test.reset_index(drop=True), X_test_vector], axis=1)


# # District vectors

# In[56]:


with open('districts_vectors.pickle', 'rb') as f:
    weights_dict = pickle.load(f)


# In[57]:


X_train_vector =
pd.DataFrame(X_train['district'].map(weights_dict).to_list())
X_test_vector = pd.DataFrame(X_test['district'].map(weights_dict).to_list())


# In[58]:


X_train_vector.columns = [f'district_{i}' for i in range(4)]
X_test_vector.columns = [f'district_{i}' for i in range(4)]


# In[59]:


X_train = pd.concat([X_train.reset_index(drop=True), X_train_vector], axis=1)
X_test = pd.concat([X_test.reset_index(drop=True), X_test_vector], axis=1)


# In[60]:


predictors = [
    'rooms',
    'floor',
    'max_floor',
```

```
    'total_area',
    'kitchen_area',
    'animals'
] + [i for i in X_train if 'description_' in i or 'district_' in i]


# # CatBoost

# ## Train model

# In[66]:


parameters = {'depth': 7, 'learning_rate': 0.07}
model = catboost.CatBoostRegressor(**parameters).fit(X_train[predictors],
y_train)


# ## Test model

# In[67]:


r2_score(y_test, model.predict(X_test[predictors]))


# In[65]:


(y_test - model.predict(X_test[predictors])).describe()


# In[42]:


px.bar(x=model.get_feature_importance(), y=predictors, orientation = "h")


# In[144]:


with open('catboost_model.pickle', 'wb') as f:
    pickle.dump(model, f)
```

```
# # CatBoost (district as categorical feature)

# In[45]:


predictors = [
    'rooms',
    'floor',
    'max_floor',
    'total_area',
    'kitchen_area',
    'animals',
    'district'
] + [i for i in X_train if 'description_' in i]


# ## Train model

# In[159]:


parameters = {'depth' : [4,5],
              'learning_rate' : [0.09, 0.1]
              }
model = catboost.CatBoostRegressor()
grid = GridSearchCV(estimator=model, param_grid = parameters, cv = 3,
n_jobs=2, verbose=10)
grid.fit(X_train[predictors], y_train, cat_features=['district'])
model = grid.best_estimator_


# In[160]:


grid.best_params_


# In[46]:


parameters = {'depth': 5, 'learning_rate': 0.09}
```

```
model = catboost.CatBoostRegressor(**parameters).fit(X_train[predictors],
y_train, cat_features=['district'])


# ## Test model

# In[47]:


r2_score(y_test, model.predict(X_test[predictors]))


# In[77]:


(y_test - model.predict(X_test[predictors])).describe()


# In[164]:


px.bar(x=model.get_feature_importance(), y=predictors, orientation = "h")


# In[144]:


with open('catboost_model.pickle', 'wb') as f:
    pickle.dump(model, f)


# In[ ]:




# # Gradient Boosting

# ## Train model

# In[104]:
```

```
parameters = {'max_depth' : [6],
              'learning_rate' : [  0.2]
              }
model = GradientBoostingRegressor()
grid = GridSearchCV(estimator=model, param_grid = parameters, cv = 3,
n_jobs=2, verbose=10)
grid.fit(X_train[predictors], y_train)
model = grid.best_estimator_


# In[105]:


grid.best_params_


# ## Test model

# In[106]:


r2_score(y_test, model.predict(X_test[predictors]))


# In[107]:


(y_test - model.predict(X_test[predictors])).describe()


# In[108]:


px.bar(x=model.feature_importances_, y=predictors, orientation = "h")


# # Random Forest

# ## Train model

# In[99]:
```

```
parameters = {'max_depth' : [20]
              }
model = RandomForestRegressor()
grid = GridSearchCV(estimator=model, param_grid = parameters, cv = 3,
n_jobs=2, verbose=10)
grid.fit(X_train[predictors], y_train)
model = grid.best_estimator_


# In[100]:


grid.best_params_


# ## Test model

# In[101]:


r2_score(y_test, model.predict(X_test[predictors]))


# In[102]:


(y_test - model.predict(X_test[predictors])).describe()


# In[103]:


px.bar(x=model.feature_importances_, y=predictors, orientation = "h")


# # Keras

# ## Train model

# In[68]:
```

```
len(predictors)


# In[69]:


for col in predictors:
    X_train[col] = X_train[col].astype('float')


# In[70]:


del model


# In[71]:


inputs = []
model = Sequential()
model.add(Dense(100, activation= 'relu'))
model.add(Dense(100, activation= 'relu'))
model.add(Dense(100, activation= 'relu'))
model.add(Dense(100, activation= 'relu'))
model.add(Dense(1))
model.compile(loss= "mean_squared_error",
                    optimizer="adam")


# In[72]:


model.fit(X_train[predictors], y_train.values, epochs=80)


# ## Test model

# In[73]:


for col in predictors:
    X_test[col] = X_test[col].astype('float')
```

```
# In[74]:


r2_score(y_test, model.predict(X_test[predictors]))


# In[76]:


(y_test - model.predict(X_test[predictors]).reshape(1, 1359)[0]).describe()
```

Файл email_sender.py

Реалізація функціональної вимоги щоденної відправки електронних листів.

```
import datetime
import smtplib
from email.message import EmailMessage
from sqlalchemy import create_engine
import pandas as pd



class EmailAlerts():
    def __init__(self):
        DATABASE_URL = f"postgresql://postgres:admin@localhost:5433/postgres"
        self.host = "imap.gmail.com"
        self.username = "andry010203@gmail.com"
        self.password = 'bstahvshemichpjb'
        self.connection = smtplib.SMTP_SSL(self.host)
        self.connection.login(self.username, self.password)
        self.engine = create_engine(DATABASE_URL)

    def send_message(self, text, file_path, email):
        msg = EmailMessage()
        msg['Subject'] = 'Daily rent alerts'
        msg['From'] = 'Andrii Honcharenko'
        msg['To'] = email
        msg.set_content(text)

        with open(file_path, 'rb') as f:
```

```
            file_data = f.read()
        msg.add_attachment(file_data, maintype="application", subtype="xlsx",
filename=file_path)
        self.connection.send_message(msg)

    def send_alerts(self):
        df = pd.read_sql("""SELECT * FROM subscriptions""", self.engine)
        current_date = datetime.date.today()
        for index, row in df.iterrows():
            temp_df = pd.read_sql(row['search'], self.engine)
            temp_df.to_excel(f"rent_data_{current_date}.xlsx")
            self.send_message(f"Alert on {current_date}:",
f"rent_data_{current_date}.xlsx",
                              row['email'])

        self.connection.quit()


if __name__ == '__main__':
    test = EmailAlerts()
    test.send_alerts()
```

Файл server.py

Реалізація API з усіма основними функціями.

```
from functools import wraps
import json
from os import environ as env
from typing import Dict
from flask_swagger_ui import get_swaggerui_blueprint
import numpy as np
import requests
from six.moves.urllib.request import urlopen

from dotenv import load_dotenv, find_dotenv
from flask import Flask, request, jsonify, _request_ctx_stack, Response,
make_response
from flask_cors import cross_origin
from jose import jwt
from sqlalchemy import create_engine
import pandas as pd
import pickle
from predictor import Predictor
```

```
AUTH0_DOMAIN = 'dev-lzz2ukzxdjuncnzl.us.auth0.com'
API_IDENTIFIER = 'https://dev-lzz2ukzxdjuncnzl.us.auth0.com/api/v2/'
CLIENT_ID = 'qpY3dU4mWc2DlBvBM8GiJkiStRTaIVDo'
CLIENT_SECRET =
'LzKTx5lfG75OVIA2JLbry8Ig7YzfitxLwuMrqfQNBxGiYXNnOYCWbSwFZiPZw62G'
ALGORITHMS = ["RS256"]
DATABASE_URL = f"postgresql://postgres:admin@localhost:5433/postgres"
PREDICTOR = Predictor()
with open('districts_vectors.pickle', 'rb') as f:
    DISTRICT_VECTORS = pickle.load(f)
with open('word2vec_model.pickle', 'rb') as f:
    WORD2VEC_MODEL = pickle.load(f)
APP = Flask(__name__)


# Format error response and append status code.
class AuthError(Exception):
    """
    An AuthError is raised whenever the authentication failed.
    """

    def __init__(self, error: Dict[str, str], status_code: int):
        super().__init__()
        self.error = error
        self.status_code = status_code


@APP.errorhandler(AuthError)
def handle_auth_error(ex: AuthError) -> Response:
    """
    serializes the given AuthError as json and sets the response status code
accordingly.
    :param ex: an auth error
    :return: json serialized ex response
    """
    response = jsonify(ex.error)
    response.status_code = ex.status_code
    return response


def get_app_access_token():
    # get access token
```

```
    data = {
        'audience': API_IDENTIFIER,
        'grant_type': 'client_credentials',
        'client_id': CLIENT_ID,
        'client_secret': CLIENT_SECRET,
    }
    headers = {'content-type': 'application/x-www-form-urlencoded'}
    response = requests.post(f'https://{AUTH0_DOMAIN}/oauth/token',
data=data, headers=headers)
    token_type = response.json()['token_type']
    access_token = response.json()['access_token']
    return token_type + ' ' + access_token


def get_search_params(inp_request):
    params = {}
    for param in ['query', 'max_price', 'min_price', 'min_rooms',
'max_rooms', 'min_floor', 'max_floor', 'district',
                  'min_total_area', 'max_total_area', 'min_kitchen_area',
'max_kitchen_area', 'animals', 'source',
                  'min_price_diff', 'max_price_diff']:
        if param in inp_request:
            params[param] = inp_request.get(param)
    query = []
    for param in params:
        if 'min_' in param:
            query.append(f"{param.split('_', 1)[1]} >= {params[param]}")
            continue
        if 'max_' in param:
            query.append(f"{param.split('_', 1)[1]} <= {params[param]}")
            continue
        if param == 'query':
            query.append(f"LOWER(description) LIKE
'%%{params[param].lower()}%%'")
            continue
        if param in ['district', 'source']:
            query.append(f"""{param} = '{params[param].replace("'",
"''")}'""")
            continue
        if param == 'animals':
            query.append(f"{param} = {params[param]}")
            continue
        query.append(f"{param.split('_', 1)[1]} = {params[param]}")
```

```
    return query


def get_token_auth_header() -> str:
    """Obtains the access token from the Authorization Header
    """
    auth = request.headers.get("Authorization", None)
    if not auth:
        raise AuthError({"code": "authorization_header_missing",
                         "description":
                             "Authorization header is expected"}, 401)

    parts = auth.split()

    if parts[0].lower() != "bearer":
        raise AuthError({"code": "invalid_header",
                         "description":
                             "Authorization header must start with"
                             " Bearer"}, 401)
    if len(parts) == 1:
        raise AuthError({"code": "invalid_header",
                         "description": "Token not found"}, 401)
    if len(parts) > 2:
        raise AuthError({"code": "invalid_header",
                         "description":
                             "Authorization header must be"
                             " Bearer token"}, 401)

    token = parts[1]
    return token


def requires_scope(required_scope: str) -> bool:
    """Determines if the required scope is present in the access token
    Args:
        required_scope (str): The scope required to access the resource
    """
    token = get_token_auth_header()
    unverified_claims = jwt.get_unverified_claims(token)
    if unverified_claims.get("scope"):
        token_scopes = unverified_claims["scope"].split()
        for token_scope in token_scopes:
            if token_scope == required_scope:
```

```
                return True
    return False


def requires_auth(func):
    """Determines if the access token is valid
    """

    @wraps(func)
    def decorated(*args, **kwargs):
        token = get_token_auth_header()
        jsonurl = urlopen("https://" + AUTH0_DOMAIN + "/.well-
known/jwks.json")
        jwks = json.loads(jsonurl.read())
        try:
            unverified_header = jwt.get_unverified_header(token)
        except jwt.JWTError as jwt_error:
            raise AuthError({"code": "invalid_header",
                             "description":
                                 "Invalid header. "
                                 "Use an RS256 signed JWT Access Token"},
401) from jwt_error
        if unverified_header["alg"] == "HS256":
            raise AuthError({"code": "invalid_header",
                             "description":
                                 "Invalid header. "
                                 "Use an RS256 signed JWT Access Token"},
401)
        rsa_key = {}
        for key in jwks["keys"]:
            if key["kid"] == unverified_header["kid"]:
                rsa_key = {
                    "kty": key["kty"],
                    "kid": key["kid"],
                    "use": key["use"],
                    "n": key["n"],
                    "e": key["e"]
                }
        if rsa_key:
            try:
                payload = jwt.decode(
                    token,
                    rsa_key,
```

```
                    algorithms=ALGORITHMS,
                    audience=API_IDENTIFIER,
                    issuer="https://" + AUTH0_DOMAIN + "/"
                )
            except jwt.ExpiredSignatureError as expired_sign_error:
                raise AuthError({"code": "token_expired",
                                 "description": "token is expired"}, 401)
from expired_sign_error
            except jwt.JWTClaimsError as jwt_claims_error:
                raise AuthError({"code": "invalid_claims",
                                 "description":
                                     "incorrect claims,"
                                     " please check the audience and
issuer"}, 401) from jwt_claims_error
            except Exception as exc:
                raise AuthError({"code": "invalid_header",
                                 "description":
                                     "Unable to parse authentication"
                                     " token."}, 401) from exc

            _request_ctx_stack.top.current_user = payload
            return func(*args, **kwargs)
        raise AuthError({"code": "invalid_header",
                         "description": "Unable to find appropriate key"},
401)

    return decorated


@APP.route('/api/register', methods=['POST'])
@cross_origin(headers=["Content-Type", "Authorization"])
def register():
    try:
        # get input values
        request_data = json.loads(request.data.decode(), strict=False)
        email = request_data.get('email')
        password = request_data.get('password')
        if email is None or password is None:
            return make_response("{'error': 'Invalid email or password'}",
400)

        # register new user
        data = json.dumps({
```

```
            "email": email,
            "connection": "Username-Password-Authentication",
            "password": password
        })
        headers = {
            'Content-Type': 'application/json',
            'Accept': 'application/json',
            'Authorization': get_app_access_token()
        }
        response = requests.post(f'https://{AUTH0_DOMAIN}/api/v2/users',
headers=headers, data=data)

        return make_response(json.dumps(response.json()),
response.status_code)
    except Exception as e:
        return make_response(json.dumps({'error': str(e)}), 500)


@APP.route('/api/token', methods=['POST'])
@cross_origin(headers=["Content-Type", "Authorization"])
def login():
    try:
        # get input values
        request_data = json.loads(request.data.decode(), strict=False)
        email = request_data.get('email')
        password = request_data.get('password')
        if email is None or password is None:
            return make_response("{'error': 'Invalid email or password'}",
400)

        # get access token
        headers = {
            'content-type': 'application/x-www-form-urlencoded'}

        data = {
            'grant_type': 'password',
            'username': email,
            'password': password,
            'audience': API_IDENTIFIER,
            'client_id': CLIENT_ID,
            'client_secret': CLIENT_SECRET,
            'scope': 'offline_access'
        }
```

```
        response = requests.post(f'https://{AUTH0_DOMAIN}/oauth/token',
headers=headers, data=data)

        return make_response(json.dumps(response.json(), indent=2),
response.status_code)
    except Exception as e:
        return make_response(json.dumps({'error': str(e)}), 500)


@APP.route('/api/search', methods=['GET'])
@cross_origin(headers=["Content-Type", "Authorization"])
@cross_origin(headers=["Access-Control-Allow-Origin",
"http://localhost:3000"])
@requires_auth
def search():
    try:
        query = get_search_params(request.args)
        select = """SELECT * FROM actual_ads"""
        if query:
            select += ' WHERE ' + ' AND '.join(query)
        try:
            df = pd.read_sql(select, create_engine(DATABASE_URL))
        except Exception as e:
            return make_response(json.dumps({'error': 'Invalid params ' +
str(e)}), 400)

        return make_response(df.set_index('link').to_json(orient="index",
force_ascii=False), 200)

    except Exception as e:
        return make_response(json.dumps({'error': str(e)}), 500)


@APP.route('/api/stats', methods=['GET'])
@cross_origin(headers=["Content-Type", "Authorization"])
@cross_origin(headers=["Access-Control-Allow-Origin",
"http://localhost:3000"])
@requires_auth
def get_stats():
    try:
        query = get_search_params(request.args)
        group_by = request.args.get('group_by')
        group_by_dict = {'year': 'EXTRACT(year from updated_at)',
```

```
                        'month': 'EXTRACT(year from updated_at),
EXTRACT(month from updated_at)',
                        'week': 'EXTRACT(year from updated_at), EXTRACT(week
from updated_at)'}
        if group_by not in group_by_dict:
            return make_response("{'error': 'Invalid groupby param'}", 400)

        metric = request.args.get('metric')
        if metric not in ['avg', 'median', 'min', 'max']:
            return make_response("{'error': 'Invalid metric param'}", 400)

        select = f"""SELECT {group_by_dict[group_by]}, {metric}(CASE WHEN
currency = 'грн' THEN price ELSE price * 40 END) FROM ads"""
        group_by_limit = {'year': 'EXTRACT(year from NOW()-created_at) <= 2',
                          'month': 'EXTRACT(month from NOW()-created_at) <=
5',
                          'week': 'EXTRACT(day from NOW()-created_at) <= 60'}
        select += ' WHERE ' + group_by_limit[group_by]
        if query:
            select += ' AND ' + ' AND '.join(query)
        select += f' GROUP BY {group_by_dict[group_by]}'
        try:
            df = pd.read_sql(select, create_engine(DATABASE_URL))
        except Exception as e:
            return make_response(json.dumps({'error': 'Invalid params ' +
str(e)}), 400)

        group_by_columns = {'year': ['year'],
                            'month': ['year', 'month'],
                            'week': ['year', 'week']}

        df.columns = group_by_columns[group_by] + [metric]
        return
make_response(df.set_index(group_by_columns[group_by]).to_json(orient="index"
, force_ascii=False), 200)

    except Exception as e:
        return make_response(json.dumps({'error': str(e)}), 500)


@APP.route('/api/custom/evaluate', methods=['GET'])
@cross_origin(headers=["Content-Type", "Authorization"])
```

```
@cross_origin(headers=["Access-Control-Allow-Origin",
"http://localhost:3000"])
@requires_auth
def evaluate_custom():
    try:
        predictors = ['rooms', 'floor', 'max_floor', 'total_area',
'kitchen_area', 'animals', 'district', 'description']
        df = {}
        try:
            for predictor in predictors:
                df[predictor] = [request.args.get(predictor)]
                if predictor == 'animals':
                    df[predictor] = [bool(df[predictor][0])]
                elif predictor not in ['district', 'description']:
                    df[predictor] = [float(df[predictor][0])]
        except Exception as e:
            return make_response(json.dumps({'error': 'Invalid param ' +
str(e)}), 400)
        df = pd.DataFrame(df)
        try:
            prediction, _ = PREDICTOR.predict(df)
        except Exception as e:
            return make_response(json.dumps({'error': 'Prediction error ' +
str(e)}), 400)
        return make_response(json.dumps({'prediction': round(prediction[0],
2)}), 200)
    except Exception as e:
        return make_response(json.dumps({'error': str(e)}), 500)


@APP.route('/api/subscribe', methods=['POST'])
@cross_origin(headers=["Content-Type", "Authorization"])
@cross_origin(headers=["Access-Control-Allow-Origin",
"http://localhost:3000"])
@requires_auth
def subscribe():
    try:
        request_data = json.loads(request.data.decode(), strict=False)
        email = request_data.get('email')
        if email is None:
            return make_response(f"{'error': 'Invalid email param'}", 400)
        query = get_search_params(request_data)
```

```
        select = """SELECT * FROM actual_ads WHERE EXTRACT(day from NOW()-
created_at) = 0 """
        if query:
            select += ' AND ' + ' AND '.join(query)
        try:
            pd.read_sql(select, create_engine(DATABASE_URL))
        except Exception as e:
            return make_response(json.dumps({'error': 'Invalid params ' +
str(e)}), 400)

        df = pd.DataFrame({'email': [email], 'search': [select]})
        df.set_index('email').to_sql('subscriptions',
create_engine(DATABASE_URL), if_exists='append')

        return make_response(json.dumps({'status': 'success'}), 200)

    except Exception as e:
        return make_response(json.dumps({'error': str(e)}), 500)


@APP.route('/api/similar', methods=['GET'])
@cross_origin(headers=["Content-Type", "Authorization"])
@cross_origin(headers=["Access-Control-Allow-Origin",
"http://localhost:3000"])
@requires_auth
def similar():
    try:
        link = request.args.get('link')
        select = f"""SELECT COUNT(*) FROM actual_ads WHERE link = '{link}'"""
        try:
            df = pd.read_sql(select, create_engine(DATABASE_URL))
            if df.iat[0, 0] == 0:
                return make_response(f"{'error': 'Link is not in actual
data'}", 400)
        except Exception as e:
            return make_response(json.dumps({'error': 'Invalid params ' +
str(e)}), 400)

        predictors = ['rooms',
                      'floor',
                      'building_floors',
                      'total_area',
                      'kitchen_area',
```

```
                    'price'
                    ]
        predictors = [f"(((al.{i} - (SELECT avg({i}) FROM
actual_ads))/(SELECT stddev({i}) FROM actual_ads)) " \
                    f"- ((target.{i} - (SELECT avg({i}) FROM
actual_ads))/(SELECT stddev({i}) FROM actual_ads))) * 2"
                    for i in predictors]
        pred_vectors = ['description_' + str(i) for i in range(13)] +
['district_' + str(i) for i in range(4)]
        predictors += [f"(((al.{i} - (SELECT avg({i}) FROM vectors))/(SELECT
stddev({i}) FROM vectors)) " \
                     f"- ((target.{i} - (SELECT avg({i}) FROM
vectors))/(SELECT stddev({i}) FROM vectors)))"
                    for i in pred_vectors]

        select = f"""SELECT al.link FROM ((SELECT * FROM actual_ads) a INNER
JOIN (SELECT * FROM vectors) v USING(link)) as al
cross join((SELECT * FROM actual_ads WHERE link = '{link}') a_1 INNER JOIN
(SELECT * FROM vectors WHERE link = '{link}') v_1 USING(link)) as target
ORDER BY abs(""" \
                + ' + '.join(predictors) + ')  LIMIT 11'
        select = f"SELECT * FROM actual_ads WHERE link IN ({select}) AND link
<> '{link}'"
        try:
            df = pd.read_sql(select, create_engine(DATABASE_URL))
        except Exception as e:
            return make_response(json.dumps({'error': 'Invalid params ' +
str(e)}), 400)

        return make_response(df.set_index('link').to_json(orient="index",
force_ascii=False), 200)

    except Exception as e:
        return make_response(json.dumps({'error': str(e)}), 500)


@APP.route('/api/custom/similar', methods=['GET'])
@cross_origin(headers=["Content-Type", "Authorization"])
@cross_origin(headers=["Access-Control-Allow-Origin",
"http://localhost:3000"])
@requires_auth
def custom_similar():
    try:
```

```
        predictors = ['rooms',
                      'floor',
                      'building_floors',
                      'total_area',
                      'kitchen_area',
                      'price'
                      ]
        predictors = [f"(((al.{i} - (SELECT avg({i}) FROM
actual_ads))/(SELECT stddev({i}) FROM actual_ads)) " \
                      f"- (({request.args.get(i)} - (SELECT avg({i}) FROM
actual_ads))/(SELECT stddev({i}) FROM actual_ads))) * 2"
                      for i in predictors if i in request.args]

        description = request.args.get('description')
        if description is not None:
            try:
                description = description.replace('.', '').replace(',',
'').replace('\n', ' ')
                description =
PREDICTOR.preprocess_description([description])[0]
                description_vector = np.mean(
                    [WORD2VEC_MODEL.wv[word] for word in description if word
in WORD2VEC_MODEL.wv], axis=0)
                predictors += [
                    f"(((al.{'description_' + str(i)} - (SELECT
avg({'description_' + str(i)}) FROM vectors))/(SELECT stddev({'description_'
+ str(i)}) FROM vectors)) " \
                    f"- (({description_vector[i]} - (SELECT
avg({'description_' + str(i)}) FROM vectors))/(SELECT stddev({'description_'
+ str(i)}) FROM vectors)))"
                    for i in range(13)]
            except:
                return make_response(json.dumps({'error': 'Invalid
description param'}), 400)

        district = request.args.get('district')
        if district is not None:
            try:
                district_vector = DISTRICT_VECTORS[district]
            except:
                return make_response(json.dumps({'error': 'Invalid district
param'}), 400)
            predictors += [
```

```
                f"(((al.{'district_' + str(i)} - (SELECT avg({'district_' +
str(i)}) FROM vectors))/(SELECT stddev({'district_' + str(i)}) FROM vectors))
" \
                f"- (({district_vector[i]} - (SELECT avg({'district_' +
str(i)}) FROM vectors))/(SELECT stddev({'district_' + str(i)}) FROM
vectors)))"
                for i in range(4)]

        if len(predictors) == 0:
            return make_response(json.dumps({'error': 'Invalid params'}),
400)

        select = f"""SELECT al.link FROM ((SELECT * FROM actual_ads) a INNER
JOIN (SELECT * FROM vectors) v USING(link)) as al ORDER BY
        abs(""" + ' + '.join(predictors) + ')  LIMIT 10'
        select = f"SELECT * FROM actual_ads WHERE link IN ({select})"
        try:
            df = pd.read_sql(select, create_engine(DATABASE_URL))
        except Exception as e:
            return make_response(json.dumps({'error': 'Invalid params ' +
str(e)}), 400)

        return make_response(df.set_index('link').to_json(orient="index",
force_ascii=False), 200)

    except Exception as e:
        return make_response(json.dumps({'error': str(e)}), 500)



if __name__ == "__main__":
    SWAGGER_URL = "/swagger"
    API_URL = "/static/swagger.json"

    swagger_ui_blueprint = get_swaggerui_blueprint(
        SWAGGER_URL,
        API_URL,
        config={
            'app_name': 'Rent API'
        }
    )
    APP.config['JSONIFY_PRETTYPRINT_REGULAR'] = True
    APP.register_blueprint(swagger_ui_blueprint, url_prefix=SWAGGER_URL)
```

```
APP.run(host="0.0.0.0", port=env.get("PORT", 3010))
```

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

СЕРВІС АГРЕГАЦІЇ ТА АНАЛІЗУ ДАНИХ ПРО АРЕНДУ ЖИТЛА

Програма та методика тестування

КПІ.ІТ-0104.045440.04.51

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Андрій ГОНЧАРЕНКО

Київ – 2024

ЗМІСТ

1 ОБ’ЄКТ ВИПРОБУВАНЬ

Об’єктом випробування є додаток для сервісу агрегації та аналізу даних про оренду житла. Програмне забезпечення створено з використанням мови програмування Python з використанням фреймворку Flask.

2 МЕТА ТЕСТУВАННЯ

Метою тестування є наступне:

- перевірка правильності роботи програмного забезпечення відповідно до функціональних вимог;
- знаходження проблем, помилок і недоліків з метою їх усунення;
- перевірка зручності графічного інтерфейсу.

3 МЕТОДИ ТЕСТУВАННЯ

Для тестування програмного забезпечення використовуються такі методи:

- статичне тестування – перевіряється програма разом з усією документацією, яка аналізується на предмет дотримання стандартів програмування;
- динамічне тестування – застосовується в процесі виконання програми. Коректність програмного засобу перевіряється на певній кількості тестів. При прогоні кожного з них збираються та аналізуються дані про проблеми та помилки в роботі програми;
- функціональне тестування – полягає у перевірці відповідності реальної поведінки програмного забезпечення очікуваній;
- мануальне тестування – тестування без використання автоматизації, тест-кейси пише особа, що тестує програмне забезпечення.

4 ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується мануально, з метою знаходження помилок та недоліків як у функціональній частині програмного забезпечення так і в зручності в користуванні. Для того, щоб перевірити працездатність та відмовостійкість застосунку, необхідно провести наступні тестування:

- динамічне тестування на відповідність функціональним вимогам;
- тестування на виведення повідомлень про помилку, коли це необхідно;
- тестування інтерфейсу користувача;
- тестування зручності використання.

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

СЕРВІС АГРЕГАЦІЇ ТА АНАЛІЗУ ДАНИХ ПРО АРЕНДУ ЖИТЛА

Керівництво користувача

КПІ.ІТ-0104.045440.05.34

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

Виконавець:
_____ Андрій ГОНЧАРЕНКО

Київ – 2024

ЗМІСТ

1 Призначення програми

Сервіс агрегації та аналізу оголошень про оренду житла - це застосунок, що періодично збирає оголошення про оренду з декількох джерел, оцінює їх ринкову вартість та надає можливість переглядати статистику ринку оренди житла.

2 ПІДГОТОВКА ДО РОБОТИ

2.1 Системні вимоги для коректної роботи

Для успішної роботи даного застосунку необхідне виконання наступних вимог:

- наявність пристрою з оперативною пам’яттю хочаб 512Мб;
- наявність доступу до Інтернету;
- браузер Google Chrome, Mozila Firefox, Opera або інший.

2.2 Завантаження застосунку

Для запуску застосунку необхідно перейти по посиланню на головну сторінку.

2.3 Перевірка коректної роботи

При коректному запуску сторінки браузер відобразить перелік ендпоінтів.

3 Виконання програми

Для реєстрації користувача надсилається запит з електронною поштою, паролем, ім’ям та прізвищем (рис. 3.1).

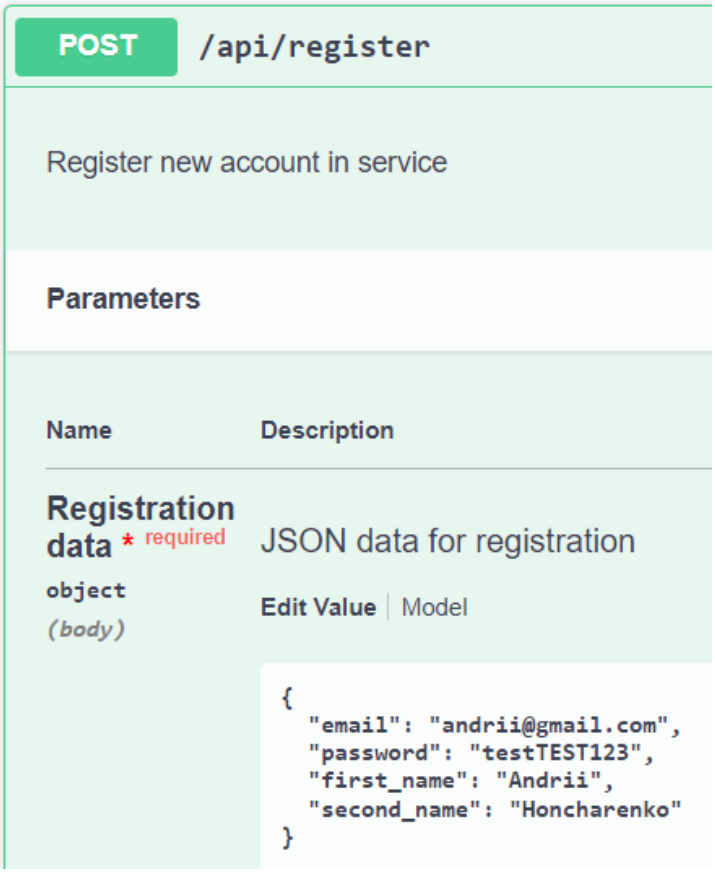


Рисунок 3.1 – Запит на реєстрацію

Авторизація виконується за допомогою логіну та паролю (рис. 3.2).

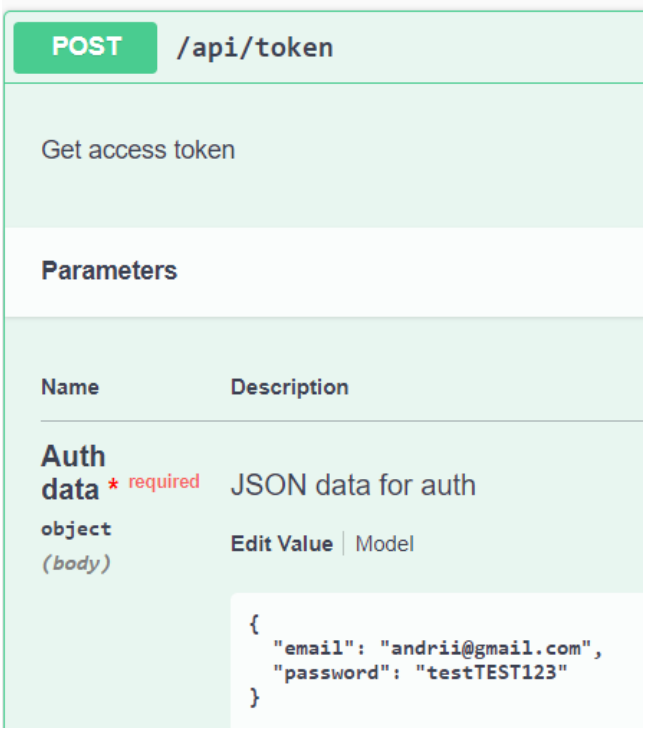


Рисунок 3.2 – Запит на отримання токену доступу

Користувач робить запит на пошук актуальних оголошень (рис. 4.7).

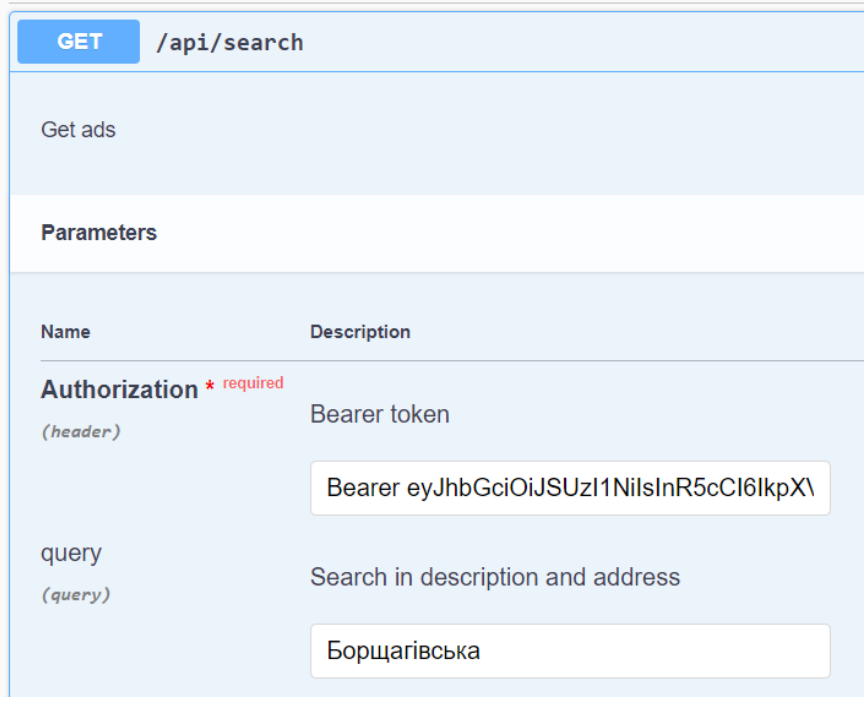


Рисунок 3.3 – Запит на пошук оголошень за фільтрами

Користувач робить запит на отримання щотижневої статистики (рис. 3.4).

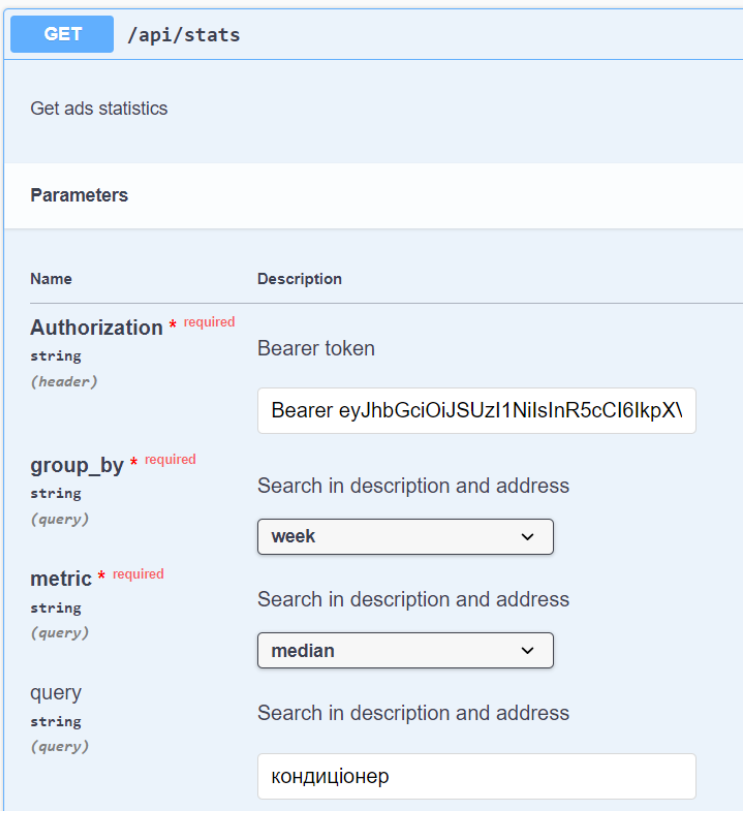


Рисунок 3.4 – Запит на отримання статистики

Користувач надсилає запит на пошук схожих квартир (рис. 3.5).

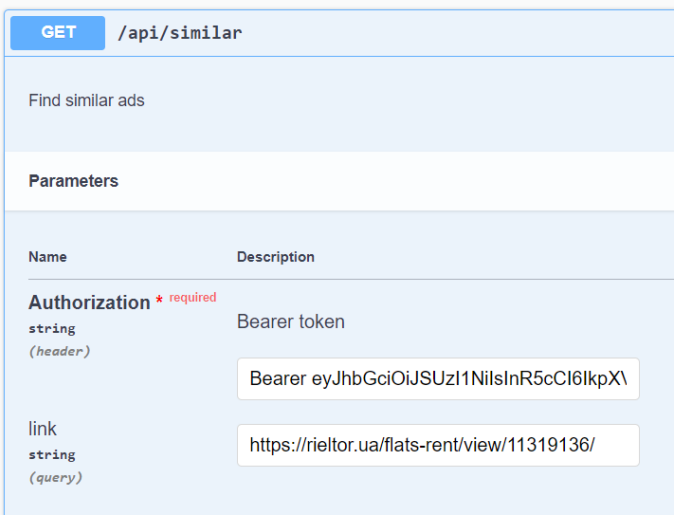


Рисунок 3.5 – Запит на отримання схожих оголошень

Користувач робить запит на підписку на щоденну розсилку (рис. 3.6)

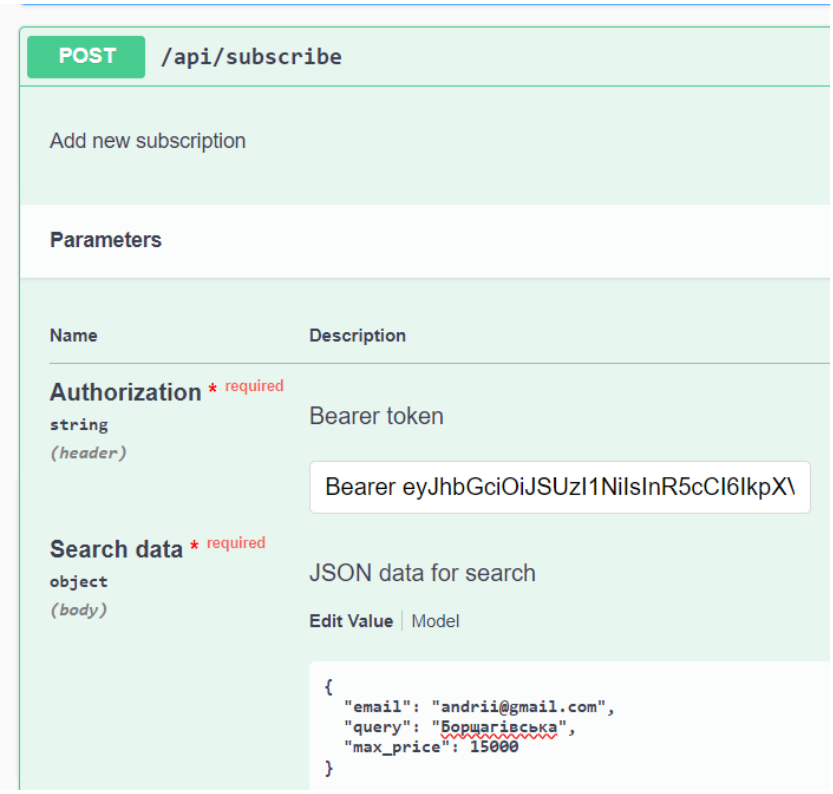


Рисунок 3.6 – Запит на підписку на щоденну розсилку

Користувач вводить дані про квартиру для отримання її оцінки від моделі (рис. 3.7).

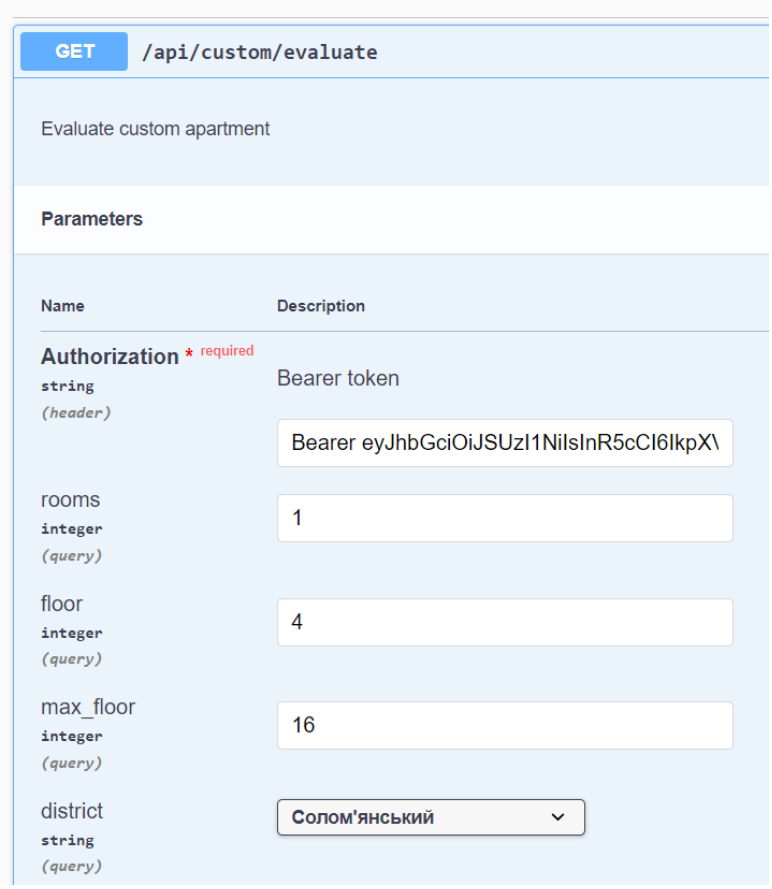


Рисунок 3.7 – Запит на оцінку квартири

Корисутвач робить запит на отримання оголошень, схожих на квартиру, що введена вручну (рис. 3.8).

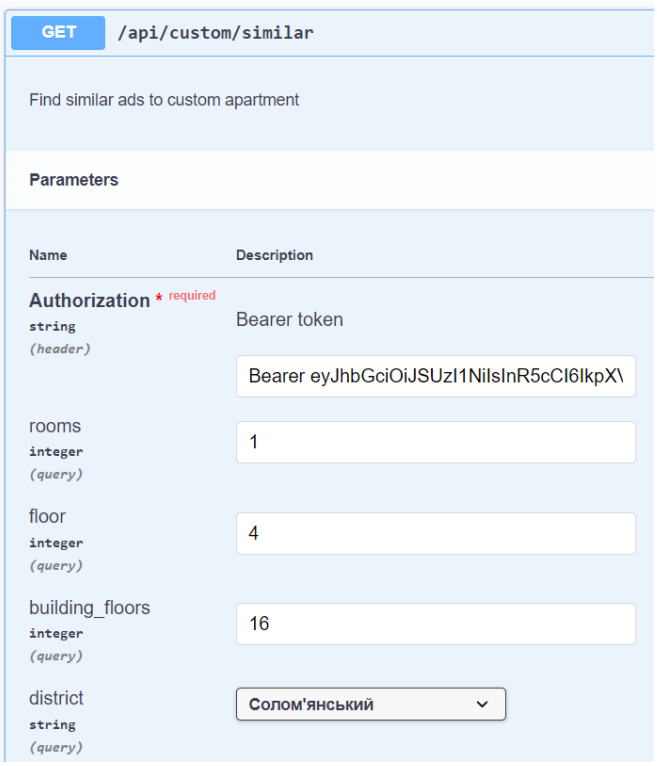


Рисунок 3.8 – Запит на отримання схожих оголошень

Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

“ЗАТВЕРДЖЕНО”
Завідувач кафедри
_____ Едуард ЖАРІКОВ
“___” _____ 2024 р.

СЕРВІС АГРЕГАЦІЇ ТА АНАЛІЗУ ДАНИХ ПРО АРЕНДУ ЖИТЛА

Графічний матеріал

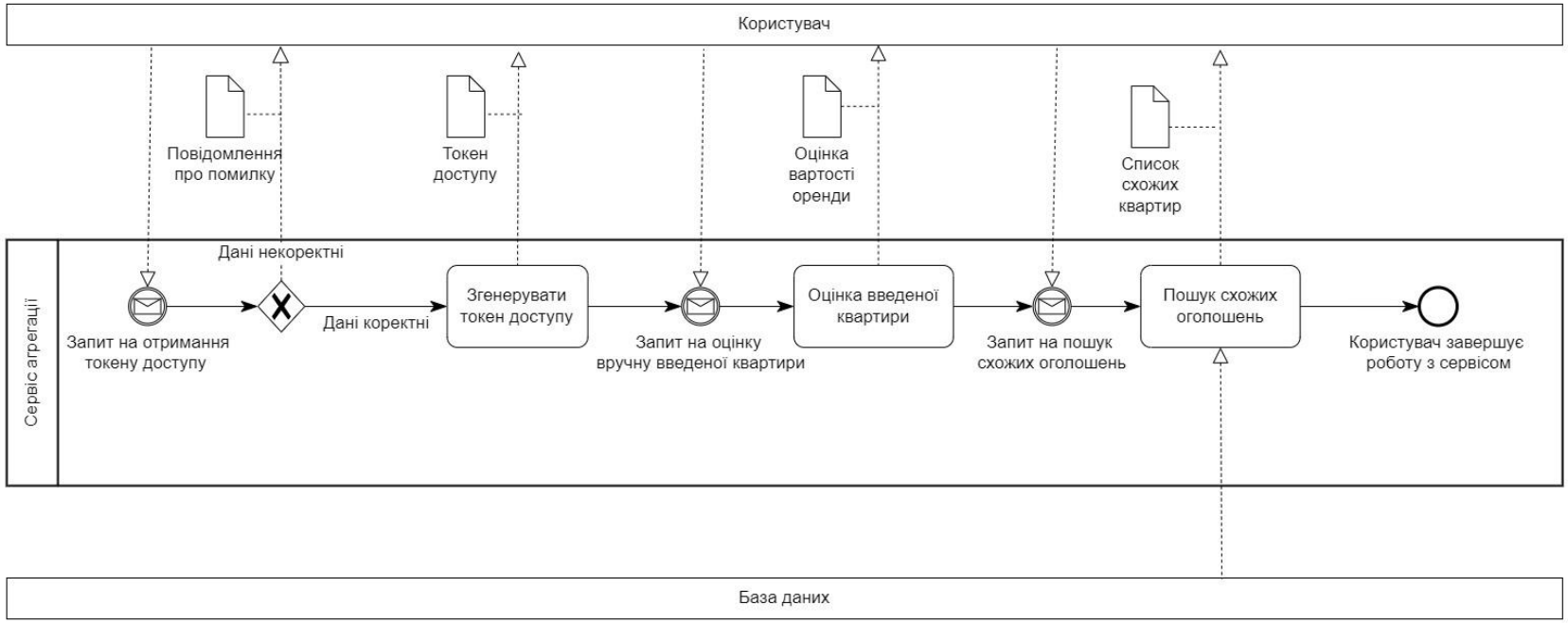
КПІ. ІТ-0104.045440.06.99

“ПОГОДЖЕНО”
Керівник проєкту:
_____ Олексій ФІНОГЕНОВ

Нормоконтроль:
_____ Ірина ВІТКОВСЬКА

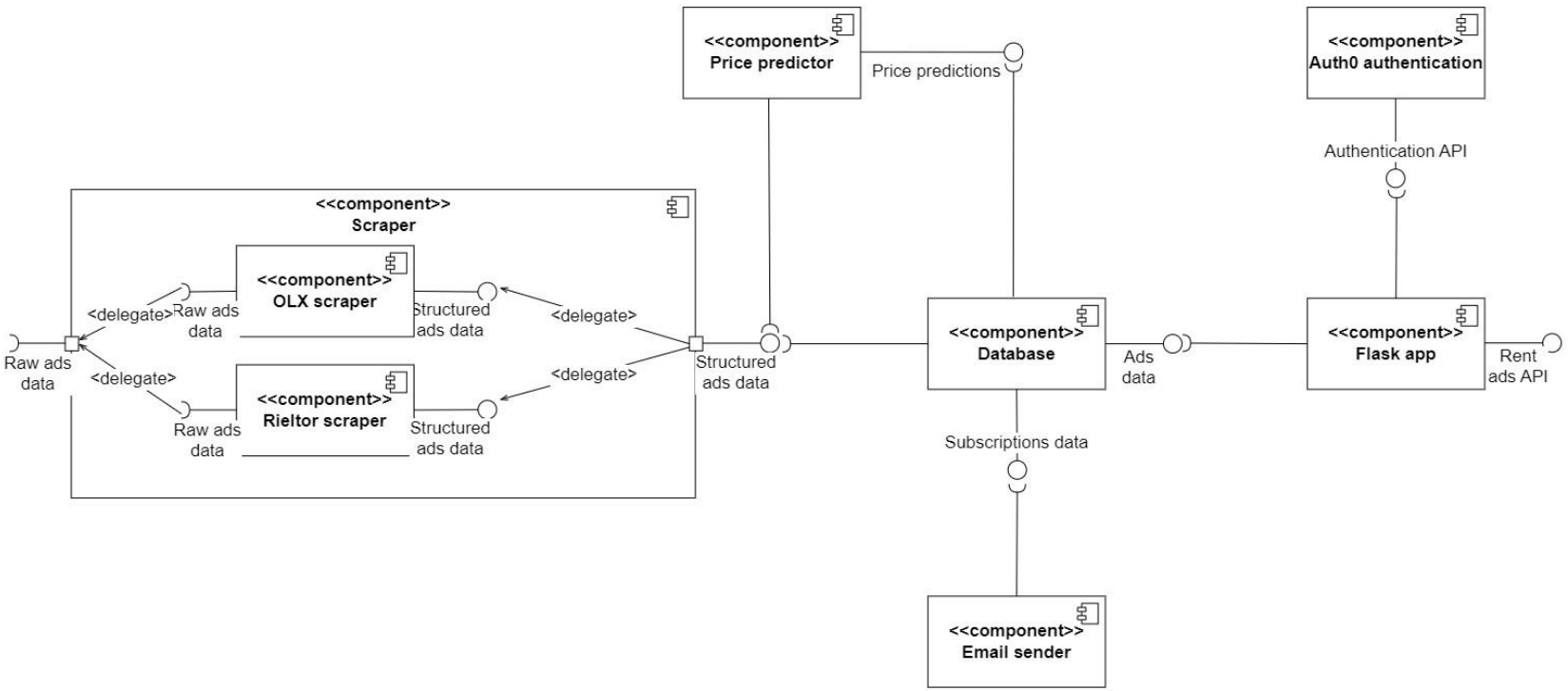
Виконавець:
_____ Андрій ГОНЧАРЕНКО

Київ – 2024



					ІТ-0104.045440.СБП1	
						Літера / Маса / Масштаб
Зм.	Арк.	№ документа	Підпис	Дата	Схема бізнес-процесів	
Розробив		Гончаренко А.А.				Аркуш / Аркушів
Перевірив		Фіногенов О.Д.				
Т. контр.						
Н. контр.		Вітковська І.І.			Сервіс агрегації та аналізу даних про аренду житла	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01
Затвердив		Жаріков Е.В.				

					ІТ-0104.045440.СБП2	
Зм.	Арк.	№ документа	Підпис	Дата	Схема бізнес-процесів	Літера / Маса / Масштаб
Розробив		Гончаренко А.А.				
Перевірив		Фіногенов О.Д.				Аркуш / Аркушів
Т. контр.						
Н. контр.		Вітковська І.І.			Сервіс агрегації та аналізу даних про аренду житла	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01
Затвердив		Жаріков Е.В.				



					ІТ-0104.045440.ДК	
Зм.	Арк.	№ документа	Підпис	Дата	Діаграма компонентів	Літера / Маса / Масштаб
Розробив		Гончаренко А.А.				
Перевірив		Фіногенов О.Д.				Аркуш / Аркушів
Т. контр.						
Н. контр.		Вітковська І.І.			Сервіс агрегації та аналізу даних про аренду житла	КПІ ім.Ігоря Сікорського Кафедра ІПІ гр. ІТ-01
Затвердив		Жаріков Е.В.				