МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

Факультет прикладної математики
Кафедра програмного забезпечення комп’ютерних систем

КУРСОВИЙ ПРОЕКТ
ТЕХНІЧНЕ ЗАВДАННЯ

з дисципліни “Бази даних”

спеціальність 121 – Програмна інженерія

на тему: Моніторингова система ресурсів пошуку оголошень про оренду нерухомості в курортних зонах Карпат

Студент
групи КП-81(82,83) **Дикий І. М.** _____
(підпис)

Викладач
к.т.н, доцент кафедри
СПіСКС **Петрашенко А.В.** _____
(підпис)

Київ – 2021

Анотація

Метою розробки даного курсового проекту є набуття практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з постреляційними базами даних, а також здобуття навичок оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті виконання курсового проекту було опановано навички розробляти програмне забезпечення для постреляційних баз даних, володіння основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних.

Темою даного курсового проекту є створення моніторингової системи ресурсів з пошуку оголошень про оренду апартаментів в курортних зонах карпат. У документі викладена актуальність та проблематика аналізу великого обсягу даних, аналіз використаного інструментарію (опис мови програмування, використаних бібліотек та СУБД), описана структура бази даних, опис розробленого програмного забезпечення (загальний, опис модулів та основних алгоритмів роботи), аналіз функціонування засобів масштабування, та опис результатів проведеного аналізу.

Результатами даного проекту стали діаграми та графіки, що зображають результати аналізу оголошень про оренду апартаментів в курортних зонах карпат. З ними можна ознайомитися в додатку Б.

Зміст

Вступ

В сучасному світі кількість даних зростає експоненціально. І для отримання інформації з цих даних вже не вистачає класичних інструментів аналізу. Через це набула значного розвитку розвитку Data Science.

Data Science – галузь інформатики, що вивчає проблеми аналізу, обробки і представлення даних у цифровій формі. Говорячи простіше, це наука про методи обробки великих масивів даних і вилучення з них цінної інформації, завдяки чому можна більш ефективно приймати рішення. Все це стало можливо завдяки появі хмарних сервісів для зберігання даних, зростання обчислювальних здібностей комп'ютерів, розвитку технологій машинного навчання і нейромереж.

Сучасний Data Science спеціаліст (дослідник даних) має оперувати великими обсягами даних та вміти виокремити з них приховані залежності, на основі яких зробити прогноз про те, як будуть надалі відбуватися ті чи інші явища. Дослідники даних використовують свої дані та аналітичні здібності для пошуку та інтерпретації великих джерел даних; керують великими обсягами даних безвідносно до апаратного та програмного забезпечення і обмежень пропускної здатності; об'єднують джерела даних; забезпечують цілісність наборів даних; створюють візуалізації для кращого розуміння даних; з використанням даних будують математичні моделі; надають тлумачення даних та висновки.

Метою даного курсового проекту був аналіз оголошень про оренду житла в курортних зонах карпат. Мета дослідження полягає в прогнозуванні ціни на квартири відповідно до площі, кількості кімнат та району, де розташовані квартири. Також метою даного курсового проекту є набуття навичок з масштабування високонавантажених системи, роботою з Big Data, науковими бібліотеками мови програмування Python3.

1. Аналіз інструментарію для виконання курсового проекту

1.1 Аналіз СУБД

Під час виконання курсового проекту виникла потреба зберігати велику кількість даних. Найкращий варіант для зберігання великої кількості даних-використання СУБД.

В якості СУБД були розглянуті варіанти: PostgreSQL, MongoDB. З порівняльною характеристикою цих СУБД можна ознайомитися в таблиці 1.

таблиця 1. Порівняльна характеристика СУБД

Критерій порівняння	Назва СУБД	
	MongoDB	PostgreSQL
Має відкритий вихідний код	так	так
Схема даних	динамічна	статична і динамічна
Підтримка ієрархічних даних	так	так
Реляційні дані	ні	так

Транзакції	ні	так
Атомарністі операцій	всередині документа	по всій БД
Мова запитів	JSON/JavaScript	SQL
Найлегший спосіб масштабування	горизонтальний	вертикальний
Підтримка шардингів	так	так (важка конфігурація)
Приклад використання	Великі дані (мільярди записів) з великою кількістю паралельних оновлень, де цілісність і узгодженість даних не потрібно.	Транзакційні і операційні програми, вигода яких в нормалізованому формі, об'єднаннях, обмеження даних і підтримки транзакцій.
Наявність бібліотек для мови програмування Python 3	так	так

Підтримка реплікації	так, автоматичне переобрання головного процесу	За принципом master-slave
Засіб збреження та відновлення даних	mongodump	pg_dump
Форма збереження даних	документи JSON	таблиця

За результатами порівняння цих СУБД було прийнято рішення зупинитися на NoSQL рішеннях. Оскільки вони чудово поєднують в собі переваги неструктурованих баз даних та простоту використання горизонтального масштабування. Крім цього, класичним прикладом використання NoSQL СУБД є системи збору та аналізу даних, до яких можна застосувати індексування за первинними та вторинними ключами.

NoSQL база даних є об’єктно орієнтованою та дозволяє зберігати великі масиви неструктурованих даних. На відміну від SQL баз даних ми можемо зберігати дані у “сирому” об’єктному вигляді, який використовується програмою та є більш близьким за структурою до моделі даних, яку буде використовувати ПЗ написане з використанням мови програмування Python. Це пришвидшить збір, збереження та отримання даних програмним забезпеченням. Оскільки MongoDB є представником NoSQL баз даних, вона не потребує жорсткої схеми даних, що дозволяє пришвидшити процес розробки та зробити його більш гнучким. Окрім цього дана СУБД підтримує горизонтальне масштабування за допомогою шардингу з метою зменшення навантаження на кожен окремий вузол шляхом розподілення навантаження між ними.

1.2 Обґрунтування вибору мови програмування

Мовою програмування для ПЗ було обрано Python 3.8. Ключовою особливістю Python є широкий інструментарій засобів розробки систем збору та аналізу даних. Фактично ця мова є стандартом у світі математичних розрахунків та обробки даних у реальному часі. Тож під час виконання курсового проекту було відносно легко знайти необхідну документацію та приклади роботи із цією мовою.

1.3 Обґрунтування вибору бібліотек і фреймворків

Використані бібліотеки та фреймворки:

- ***numpy*** - математична бібліотека мови Python, що додає підтримку великих багатовимірних масивів і матриць, разом з великою бібліотекою високорівневих математичних функцій для операцій з цими масивами.
- ***pandas*** - це бібліотека на Python для обробки і аналізу даних. Робота pandas з даними будується поверх бібліотеки NumPy.
- ***matplotlib*** - це бібліотека Python 2D, яка представляє числові дані у різноманітних форматах та інтерактивних середовищах на різних платформах
- ***folium*** - це бібліотека мови Python для візуалізації даних на інтерактивній мапі.
- ***scrappy*** - це швидкий веб-фреймворк з відкритим вихідним кодом, написаний на Python, який використовується для отримання даних з веб-сторінки за допомогою селекторів на основі XPath.
- ***pymongo*** - є дистрибутивом Python, що містить інструменти для роботи з MongoDB, і є рекомендованим способом роботи з MongoDB від Python.

2. Структура бази даних

База даних складається з однієї колекції, в якій зберігаються відомості про оголошення. Загальна структура документа в базі даних приведена у таблиці 2.

таблиця 2. Опис властивостей документа у базі даних

Назва властивості	**Тип**	**Опис**
_id	ObjectId	Ідентифікатор запису
resort	String	курортна зона, де знаходиться житло
price	Integer	вартість оренди житла
name	String	назва отелю, садиби
location	String	опис місцезнаходження готелю
latitude	Float	широта місцезнаходження
longitude	Float	довгота місцезнаходження
desc	String	опис готелю
distance	Float	відстань від готелю до гірнолижного комплексу

3. Опис програмного забезпечення

3.1 Загальна структура програмного забезпечення

Загалом проект складається з бази даних, клієнтського додатку, для отримання статистики та двох скраперів, котрі можна запускати як окремо так і в рамках додатку.

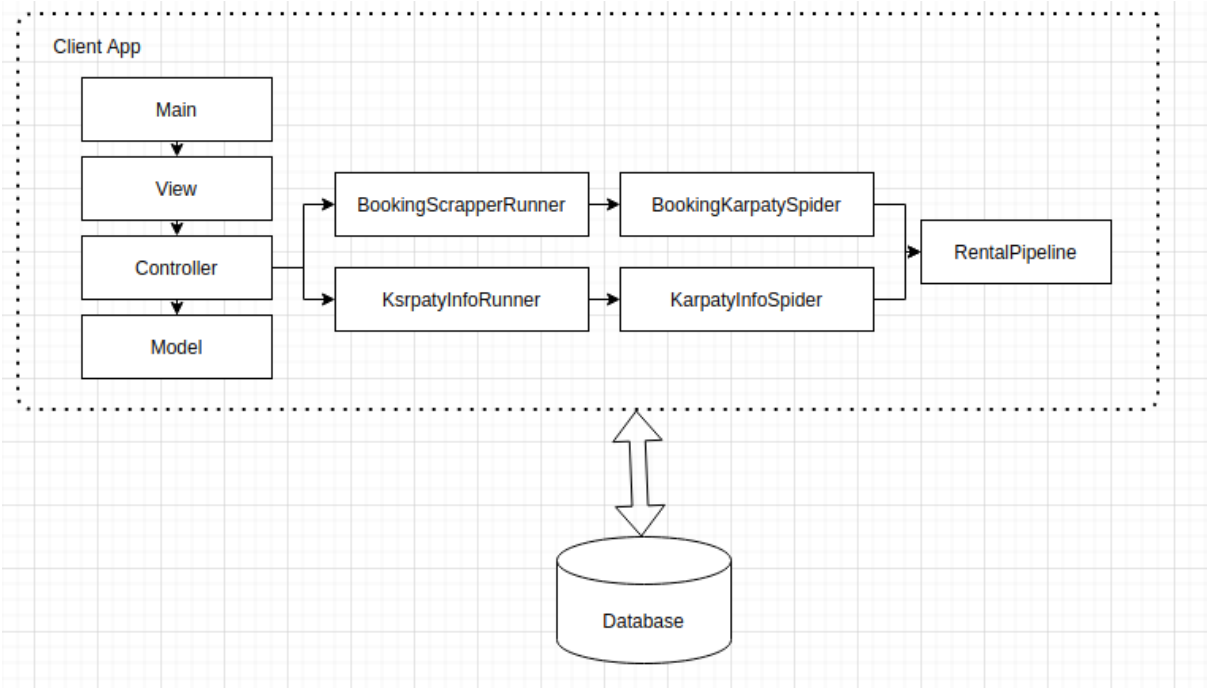

рис 1. Архітектура системи

3.2 Опис модулів програмного забезпечення

Додаток складається з наступних модулів:

Main - в даному модулі відбувається підключення до бази даних та запуск інтерфейсу користувача

View - модуль графічного інтерфейсу користувача

Controller - модуль, в якому відбувається весь аналіз даних, побудова графіків, мапи.

Model - модуль, в котрому відбувається доступ до бази даних

BookingScrapperRunner, KarpatyInfoRunner - модулі, які запускають скрапери в окремих потоках.

KarpatyInfoSpider, BookingKarpatySpider - модулі, котрі стягують інформацію про оренду житла з відповідних сайтів.

Pipeline - модуль в котрому відбувається фільтрація та валідація даних та збереження їх в базу даних.

4. Аналіз функціонування засобів масштабування

В якості засобів масштабування було обрано реплікацію та шардинг.

Реплікація - це процес синхронізації даних на декількох серверах. Даний механізм зменшує витрати ресурсів і збільшує доступність даних, копії яких зберігаються на різних серверах. Реплікація захищає базу даних від втрати єдиної сервера і дозволяє зберегти дані в разі технічної несправності на одному з серверів. У MongoDB реплікація досягається шляхом використання набору копій (replica set). Це група примірників mongod, який зберігають однакові набори даних. У копії один вузол - це ключовий вузол, який отримує всі операції запису. Всі інші вузли - вторинні, приймають операції з першого, таким чином, зберігаючи такі ж записи, як і первинний вузол. Набір копій може мати тільки один первинний вузол.

Шардінг - це процес зберігання документів на декількох серверах і це спосіб, яким MongoDB справляється з великими даними. З ростом кількості даних, один сервер не може зберігати всі даних, ні записувати їх, ні давати до них доступ. Шардінг вирішує проблему шляхом горизонтального масштабування. Завдяки даному механізму ми можемо підключати додаткові сервери для зберігання, записи і читання даних.

Для емуляції існування багатьох серверів із СКБД було запущено необхідну кількість процесів-демонів *mongod.*

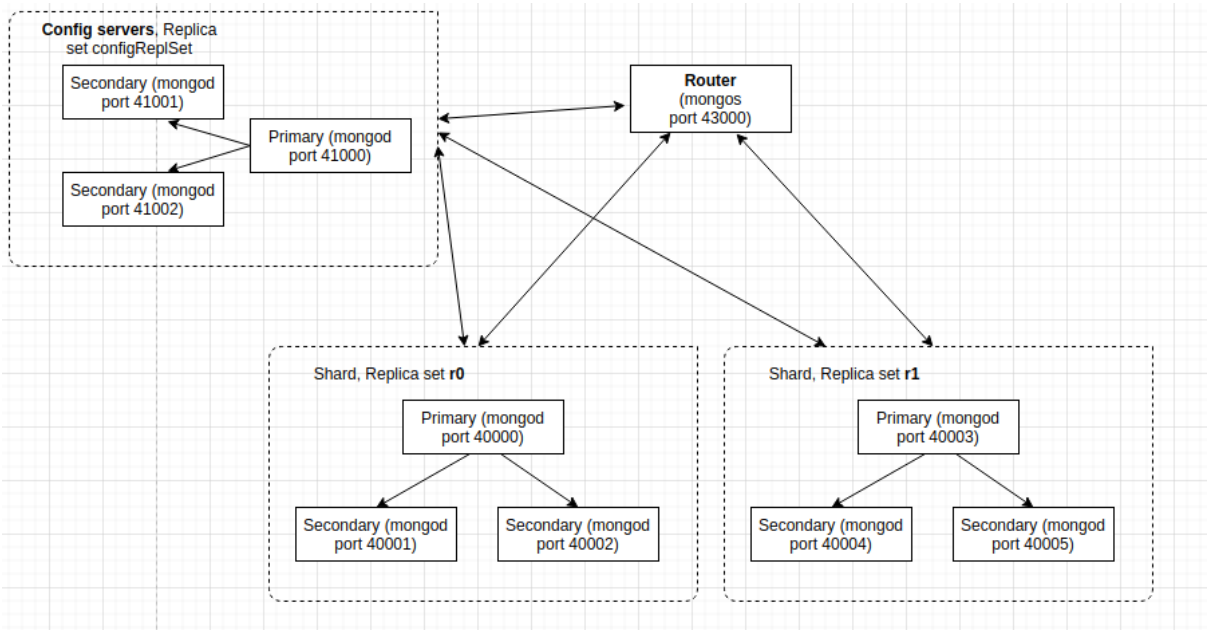

рис 2.Схема використаної моделі шардингу MongoDB у даному ПЗ

В якості ключа для розбивання даних на шарди було використано поле *resort.* Були створені окремі шард-зони для кожного курорту. До шарду **r0** увійшли Буковель і Славське, а до шарду **r1** - Пилипець, Яблуниця і Драгобрат.

рис 3. Розподіл даних між шардами

4.1 Тестування масштабування

В процесі перевірки здатності системи масштабуватися, було проведено наступні тести:

1. Тестування відмовостійкості шляхом зупинки декількох процесів із репліки сету одного із шардингу
2. Тестування відмовостійкості системи шляхом зупинки декількох процесів із репліки сети серверів конфігурації
3. Перевірка того, яку частину даних ми втратимо, в разі відмови однієї реплікації серверів шардингу

4.1.1 Тестування відмовостійкості шляхом зупинки декількох процесів із репліки сету одного із шардингу

Нами було зупинено процес mongod на порту 40003 в результаті чого, ми очікували, що система перенаправить усі запити, що йшли до нього до його реплікації на порту 40004 чи 40005, та обере його головним,

в разі якщо він не був таким. Вивід процесу mongos серверу маршрутизації наведено на рис. 3.

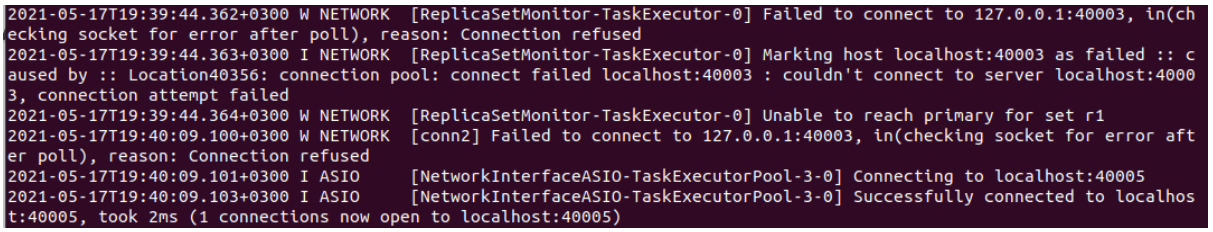

рис 3. Вивід процесу mongos після того, вимкнення mongod 40003

Як видно із виводу, після вимкнення mongod на порту 40003, який був первинним вузлом для шарду, сервер надіслав запит до процесу на порті 40005, який був вторинним але став первинним вузлом в шарді.

4.1.2 Тестування відмовостійкості системи шляхом зупинки декількох процесів із репліки сети серверів конфігурації

Сервери конфігурація, які знаходять у реплікації поводять себе так само, як і при тестуванні реплікації кластерів шардів. Після невдачі отримати відповідь на запит перевірки серцебиття (рис 5) від серверу конфігурації на порту 41000, кожен з процесів серверу конфігурації, що ще онлайн робить запит до вимкненого сервера кілька раз на секунду. Крім цього відбувається обрання нового лідера реплікації.

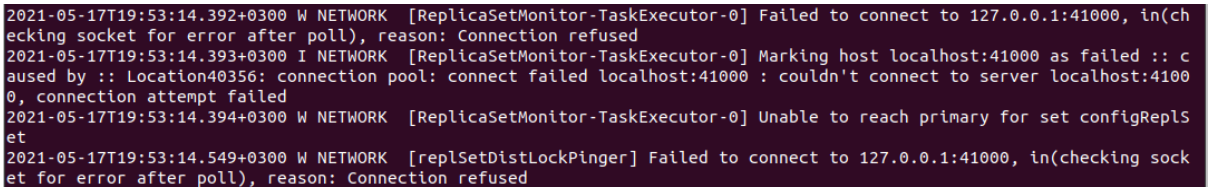

рис 5. Спроби отримати відповідь від вимкненого сервера конфігурації

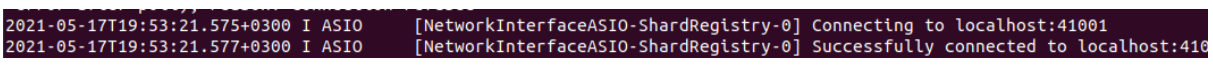

рис 6. Обрання нового лідера репліки

4.1.3 Перевірка того, яку частину даних ми втратимо, в разі відмови однієї реплікації серверів шардингу

Нами було зупинено всі процеси що стосуються шарду **r0**. В результаті чого частина даних, яка зберігалися на цих серверах, повинна бути втрачена, а після поновлення хоча б одного процесу що стосуються шарду r0 вони знову повинні бути доступними. Внаслідок розірвання зв'зку між серверами конфігурації та кластерами шардингу, ми не могли запустити запит на виконання до бд, що стосувався даних у колекції, яка була розбита на партиції. Після поновлення процесів r0 цілісність даних було відновлено і ми знову змогли виконати запит до колекції.

5. Опис результатів аналізу предметної галузі

В результаті виконання курсового проекту було проаналізовано дані про оголошення оренди житла в курортних зонах карпат. Було отримано наступні дані:

1. Проведено кореляційний аналіз між вартістю оренди та відстання до гірськолижного комплексу. Кореляція відсутня (-0.054803).
2. Знайдено середньо-арифметичну вартість оренди для кожного курорта окремо (Додаток Б,1)
3. Знайдено найбільш поширену вартість оренди(мода) для кожного курорта окремо (Додаток Б,2)
4. Пораховано відсоткове співвідношення кількості оголошень оренди житла в курортних зонах (Додаток Б,3)

Висновки

В процесі виконання даного курсового проекту було отримано практичні навички обробки великих масивів даних за допомогою мови програмування Python 3 та СУБД MongoDB.

Було проаналізовано сучасні методи та інструменти для роботи із великими даними, знайдено гарно працюючу комбінацію із мови програмування Python 3 та бібліотек до нього: Scrappy, Matplotlib, Numpy. Було складено порівняльну таблицю із двох баз даних, які найчастіше використовують. На основі цих даних було обрано в якості СУБД NoSQL рішення MongoDB.

Для забезпечення горизонтального масштабування було використано засоби MongoDB, такі як: реплікація та шардинг. Практичним шляхом було знайдено спосіб партиціювання даних часових рядів на основі діапазону значень з шард-зонами за назвою курорту. Це дозволило розподілити дані, які зберігаються в рамках однієї колекції між багатьма серверами кластерів шардингу. За допомогою реплікації було досягнуто відмовостійкість системи, в результаті чого, під час тестування ми впевнилися, що система продовжує функціонування після виходу із ладу кількох реплік.

На основі зібраних даних було проаналізовано ринок оренди житла в курортних зонах карпат. Знайдено кореляції та залежності між різними показниками (кількості кімнат в квартирі, площі та району, де знаходиться квартира). Дані аналізу приведені у Додатку Б.

В ході виконання даного курсового проекту було досягнуто поставленої мети: було набуто практичних навичок розробки сучасного програмного забезпечення, що взаємодіє з NoSQL базами даних, а також були здобуті навички оформлення відповідного текстового, програмного та ілюстративного матеріалу у формі проектної документації. У результаті виконання курсового проекту я навчилася писати програмне забезпечення для NoSQL баз даних, володіти основами використання СУБД, а також інструментальними засобами аналізу великих обсягів даних, а саме відкритими бібліотеками мови Python.

Література

1. MongoDB від теорії до практики. Керівництво по установці кластера mongoDB [Електронний ресурс].

   – Режим доступу до ресурсу: https://m.habr.com/ru/post/217393/;

2. Чому Python такий хороший в наукових обчисленнях [Електронний ресурс].

   – Режим доступу до ресурсу: https://habr.com/post/349482/;

3. Scrapy: збираємо дані і зберігаємо в базу даних [Електронний ресурс].

   – Режим доступу до ресурсу: https://habr.com/ru/post/308660/

4. Python [Електронний ресурс].

   – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python;

6. Matplotlib [Електронний ресурс].

   – Режим доступу до ресурсу: https://matplotlib.org/stable/tutorials/index.html;

7. NumPy [Електронний ресурс].

   – Режим доступу до ресурсу: https://numpy.org/doc/;

8. Pandas [Електронний ресурс].

   – Режим доступу до ресурсу: https://pandas.pydata.org/docs/;

Додатки

Додаток А

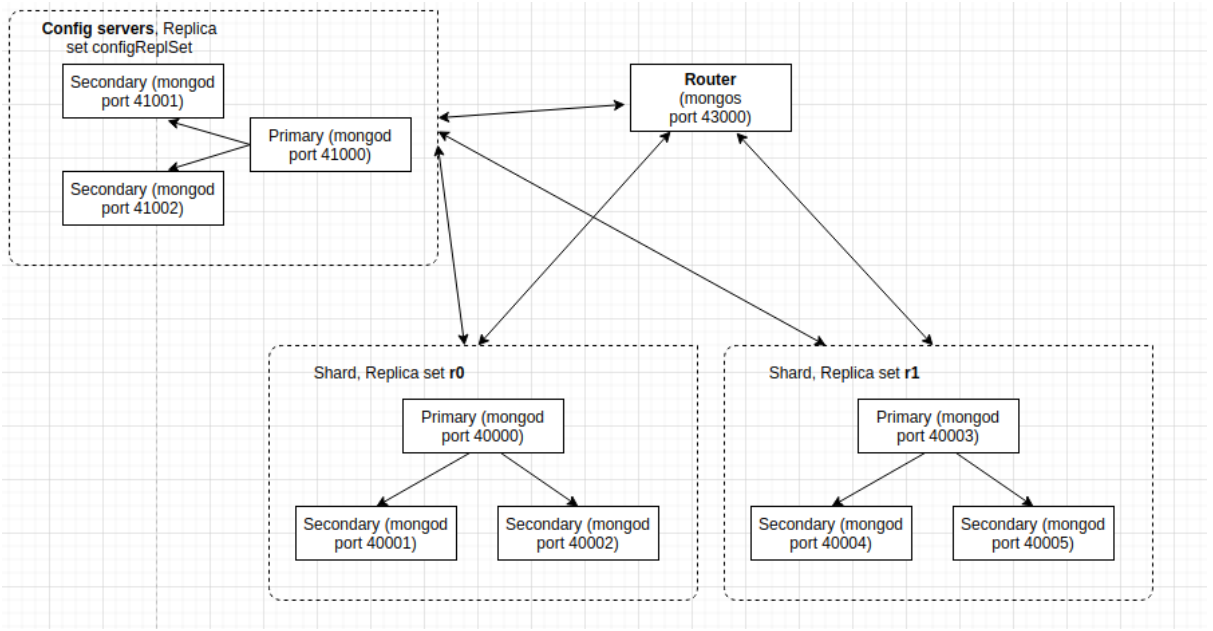

рис 1.Схема використаної моделі шардингу MongoDB у даному ПЗ

Додаток Б

1.

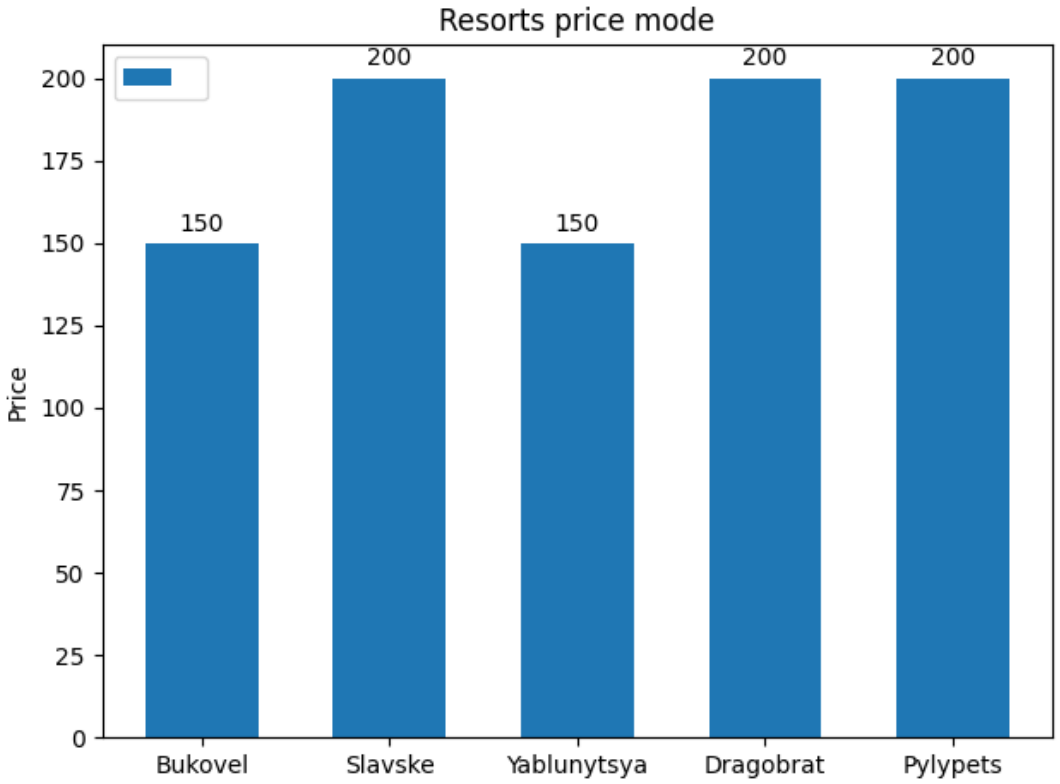

рис 8. Найчастіше зустрічаєма ціна оренди відповідно до курорту

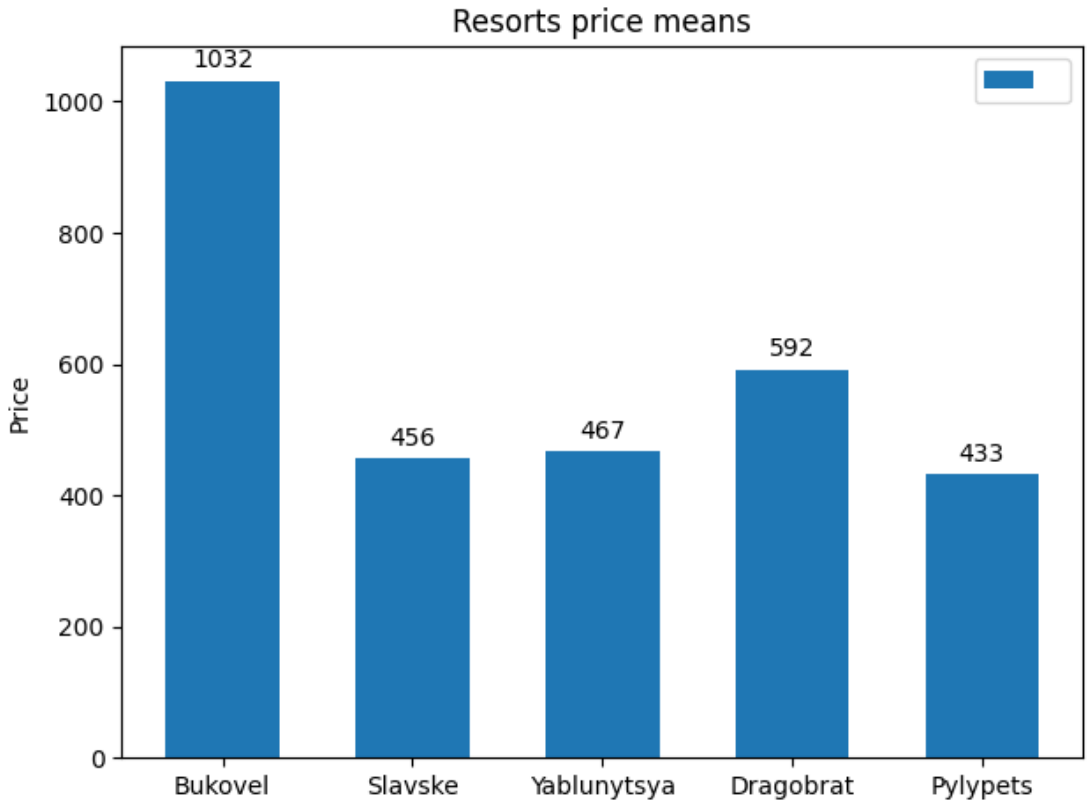

рис 9. Середньо-арефметична ціна оренди відповідно до курорту

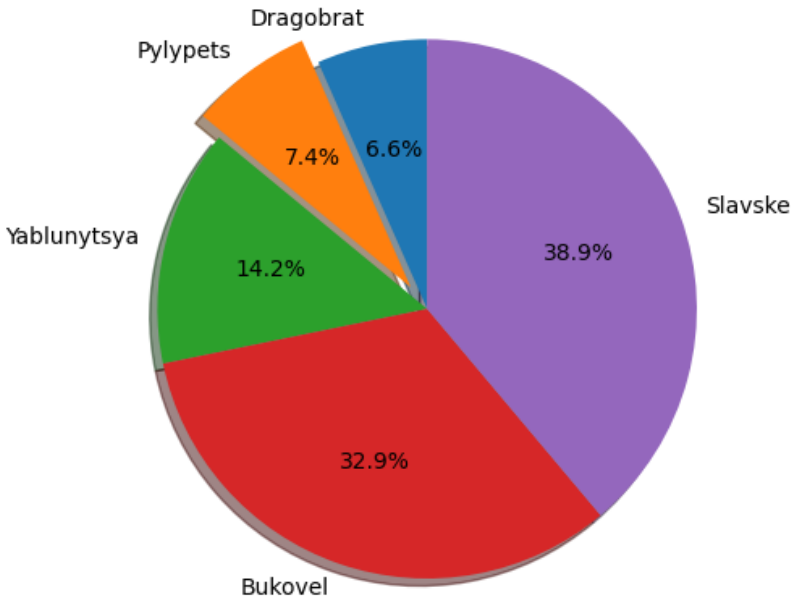

рис 10. Відсоткове співвідношення кількості оголошень про оренду в певній курортній зоні

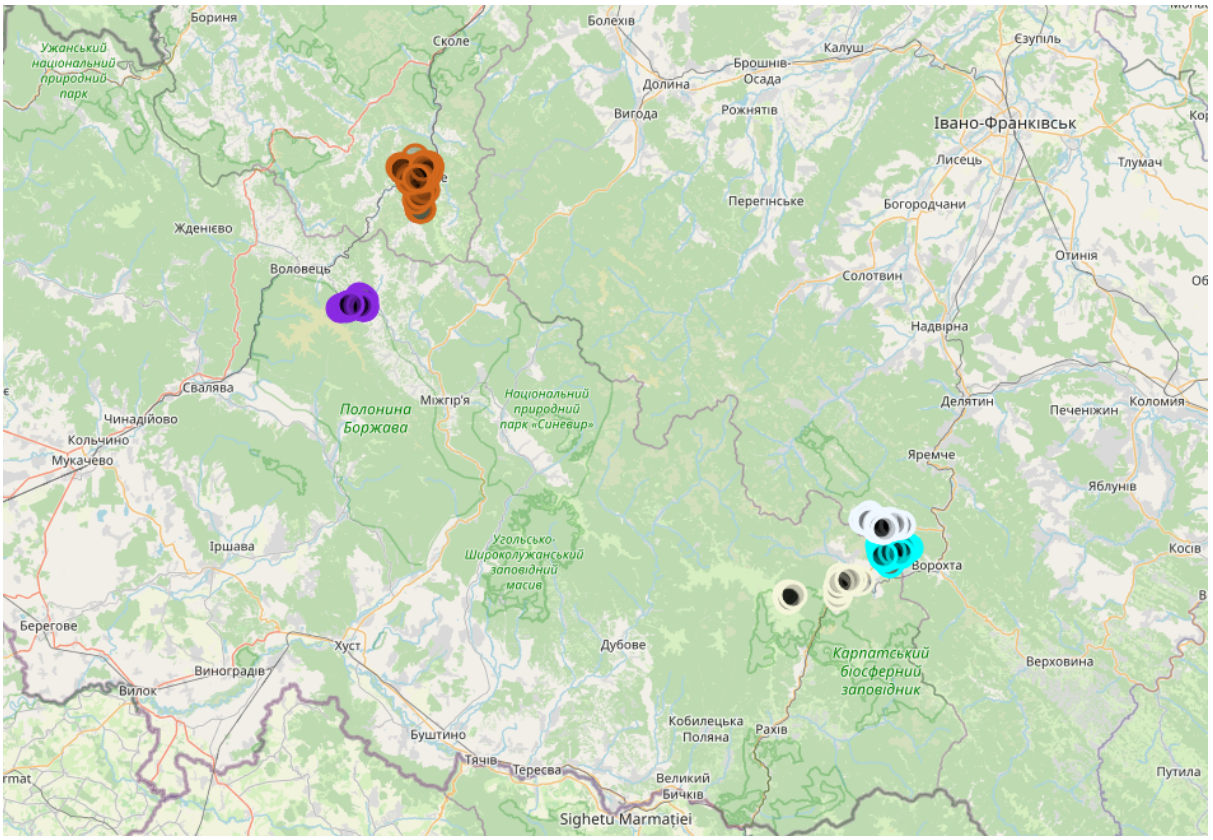

рис 11. Мапа розміщення орендного житла з оголошень

Додаток В

Фрагменти програмного коду

booking_karpaty_spider.py

```
class BookingKarpaty(scrapy.Spider):
    name = "booking_karpaty"
    allowed_domains = ['booking.karpaty.ua']
    start_urls = [
        # готелі Буковель

'https://booking.karpaty.ua/uk/cities/539876186e656f114e440000?dr=true&ga
=2#539876186e656f114e5d0000|539876186e656f114e440000',
        # гостинний двір

'https://booking.karpaty.ua/uk/cities/539876186e656f114e440000?dr=true&ga
=2#539876186e656f114e5b0000|539876186e656f114e440000',
        # приватна садиба

'https://booking.karpaty.ua/uk/cities/539876186e656f114e440000?dr=true&ga
=2#539876186e656f114e5a0000|539876186e656f114e440000',
        # готелі Славське
```

```
'https://booking.karpaty.ua/uk/cities/539876186e656f114e540000?dr=true&ga
=2#539876186e656f114e5d0000|539876186e656f114e540000',
        # гостинний двір

'https://booking.karpaty.ua/uk/cities/539876186e656f114e540000?dr=true&ga
=2#539876186e656f114e5b0000|539876186e656f114e540000',
        # приватна садиба

'https://booking.karpaty.ua/uk/cities/539876186e656f114e540000?dr=true&ga
=2#539876186e656f114e5a0000|539876186e656f114e540000',
        # Дргобрат

'https://booking.karpaty.ua/uk/cities/54d1e8064b617207dab82a00?dr=true&ga
=2#54d1e8064b617207dab82a00',
        # Ясіня

'https://booking.karpaty.ua/uk/cities/539876186e656f114e4f0000?dr=true&ga
=2#539876186e656f114e4f0000',
        # Пилипець

'https://booking.karpaty.ua/uk/cities/55d48d2ac0557c6378000faa?dr=true&ga
=2#539876186e656f114e5a0000|539876186e656f114e5d0000|55d48d2ac0557c637800
0faa',
        # Яблуниця

'https://booking.karpaty.ua/uk/cities/539876186e656f114e480000?dr=true&ga
=2#539876186e656f114e5d0000|539876186e656f114e5b0000|539876186e656f114e5a
0000|539876186e656f114e590000|539876186e656f114e480000',
    ]

    def parse(self, response):
        print("_____URL_____", response.url)
        for next_page in response.xpath('//ul[@class="list
clearfix"]/li/a/@href').getall():
            print("NEXT_PAGE", next_page)
            yield response.follow(next_page, self.parse_hotel)

    def parse_hotel(self, response):
        price_per =
response.xpath('//span[@class="ap-price__per"]/text()').get()
        if 'котедж' in price_per:
            print('Cottage')
            return
        price =
int(response.xpath('//span[@class="ap-price__amount"]/text()').get().repla
ce(" ", ""))
        if 'особа' in price_per:
            price *= 2
        name = response.xpath('//div[@class="ap-name"]/h1/text()').get()
        desc =
response.xpath('//div[@class="container--text"]/p/text()').get()
        location =
response.xpath('//div[@class="ap-address"]/p/text()').get()
        coords =
response.xpath('//div[@class="location-coords"]/text()').get()
        (latitude, longitude) = coords.split(', ')
        latitude = float(latitude)
        longitude = float(longitude)
        # print("_____DATA:", name, price, latitude, longitude, location,
```

```
desc)
        yield {
            'name': name,
            'price': price,
            'location': location,
            'latitude': latitude,
            'longitude': longitude,
            'desc': desc
        }
```

pipline.py

```
import math
import pymongo
from pymongo.errors import DuplicateKeyError


def calc_distance(ltd, lng, ltd_c, lng_c):
    return math.sqrt(pow(ltd_c - ltd, 2) + pow(lng_c - lng, 2))


def get_resort_and_distance(ltd, lng):
    print('LTD_____________LNG', ltd, lng)
    resort = ''
    distance = None
    if 48.34 < ltd < 48.37 and 24.4 <= lng < 24.5:
        resort = 'Bukovel'
        distance = calc_distance(ltd, lng, resort_centers['Bukovel'][0],
resort_centers['Bukovel'][1])
    elif 48.24 < ltd < 48.275 and 24.237 < lng < 24.4:
        resort = 'Dragobrat'
        distance = calc_distance(ltd, lng, resort_centers['Dragobrat'][0],
resort_centers['Dragobrat'][1])
    elif 48.29 < ltd < 48.32 and 24.44 < lng < 24.51:
        resort = 'Yablunytsya'
        distance = calc_distance(ltd, lng,
resort_centers['Yablunytsya'][0], resort_centers['Yablunytsya'][1])
    elif 48.65 < ltd < 48.75 and 23.27 < lng < 23.328:
        resort = 'Pylypets'
        distance = calc_distance(ltd, lng, resort_centers['Pylypets'][0],
resort_centers['Pylypets'][1])
    elif 48.79 < ltd < 48.93 and 23.40 < lng < 23.47:
        resort = 'Slavske'
        distance = calc_distance(ltd, lng, resort_centers['Slavske'][0],
resort_centers['Slavske'][1])
    return resort, distance


resort_centers = {
    'Bukovel': [48.35826, 24.40846],
    'Dragobrat': [48.24957, 24.24956],
    'Yablunytsya': [48.31447, 24.48858],
    'Slavske': [48.79964, 23.44442],
    'Pylypets': [48.64943, 23.2901],
```

```
}


class CarpathiansRentalPipeline:

    def __init__(self):
        self.my_client = pymongo.MongoClient("mongodb://localhost:43000/")
        self.mydb = self.my_client["karpatydb"]
        self.hotels = self.mydb["hotels"]
        self.count = 0

    def open_spider(self, spider):
        print("___PIPLiNE WORKS___")
        print(self.mydb.list_collection_names())

    def close_spider(self, spider):
        print("___COUNT PAGES_________", self.count)
        self.my_client.close()

    def process_item(self, item, spider):
        if not item:
            return
        (resort, distance) = get_resort_and_distance(item['latitude'],
item['longitude'])
        print("____RESORT_____", resort, distance)
        if 40000 > item['price'] > 0 and distance and item['name']:
            print("_____DATA_____:", item['name'], item['price'], distance,
resort)
            item["resort"] = resort
            item['distance'] = distance
            try:
                self.hotels.insert_one(item)
                self.count += 1
            except DuplicateKeyError:
                print("The same hotel is already exist!")

        return item
```

model.py

```
class Hotel:
    def __init__(self, state):
        self.state = state
        self.db = state["db"]
        self.collection = self.db["hotels"]

    def get_hotel(self, name):
        return self.collection.find_one({"name": name})

    def get_all_hotels(self):
        return self.collection.find({}, {"_id": 0, "desc": 0, "location":
0})
```