МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
Ігор ШЕЛЕХОВ
(підпис)
2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 Комп'ютерні науки,

освітньо-професійної програми «Інформатика»

на тему: «Інформаційна система для автоматизованого пошуку об'єктів за геолокацією з використанням телеграм бота»

здобувача групи ІН - 06 - 2 Кальченко Микити Володимировича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.
Микита КАЛЬЧЕНКО
(підпис)

Керівник, Старший викладач (старш. викл. каф. КН) Олег БЕРЕСТ

(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавра

зі спеціальності 122 Комп'ютерні науки, освітньо-професійної програми «Інформатика»
здобувача групи ІН-06-2 Кальченко Микити Володимировича

1. Тема роботи: «Інформаційна система для автоматизованого пошуку об'єктів за геолокацією з використанням телеграм бота»
затверджую наказом по СумДУ від *«22» квітня 2024 р. №0414-VI*
2. Термін здачі здобувачем кваліфікаційної роботи *до 13 червня 2024 року*
3. Вхідні дані до кваліфікаційної роботи _____
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити) *1) Аналіз проблеми предметної області, постановка й формування завдань дослідження 2) Огляд технологій, що використовуються для пошуку об'єктів за геолокацією 3) Розробка системи автоматизованого пошуку об'єктів за геолокацією .4) Аналіз результатів.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «06» травня 2024 р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються для пошуку об'єктів за геолокацією*		
3	*Розробка системи автоматизованого пошуку об'єктів за геолокацією*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ (підпис) Керівник _____ (підпис)

АНОТАЦІЯ

Записка: 59 стр., 30 рис., 5 додаток, 14 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною, оскільки присвячена розв’язанню важливої практичної задачі автоматизованого пошуку об'єктів за геолокацією. Це включає розробку відповідних методів, моделей та інформаційної технології, що забезпечують швидкий доступ до інформації, зокрема пошук бомбосховищ.

Об’єкт дослідження — процес автоматизованого пошуку об’єктів за геолокацією.

Мета роботи — створення інформаційної системи, яка базується на використанні телеграм бота для автоматизованого пошуку об'єктів за геолокацією.

Методи дослідження — інструменти пошуку геоданих на основі поточного місцезнаходження, інструменти зберігання даних, алгоритми обчислення відстані між 2 точками координат на земній поверхні

Результати — Розроблено інформаційну систему для автоматизованого пошуку об'єктів за геолокацією через телеграм бота. Система дозволяє швидко знаходити об'єкти за координатами, зберігає історію пошуків та включає функцію пошуку бомбосховищ. Проведено тестування на реальних даних.

ІНФОРМАЦІЙНА СИСТЕМА, АВТОМАТИЗОВАНИЙ ПОШУК ОБ'ЄКТІВ ЗА ГЕОЛОКАЦІЄЮ, PYTHON, SQLITE3, TELEBOT, HAVERSINE, REQUESTS.

ЗМІСТ

ВСТУП

У сучасному світі розвиток інформаційних технологій неухильно змінює наше оточення та спосіб взаємодії з ним. Швидкі темпи технологічного прогресу породжують нові можливості для автоматизації різних аспектів життя. Однією з таких можливостей є використання інформаційних систем для автоматизованого пошуку об'єктів за геолокацією.

В контексті даного дослідження пропонується розглянути створення інформаційної системи, спрямованої на автоматизований пошук об'єктів за їх географічним розташуванням з використанням телеграм бота. Це стає актуальним завдяки зростаючій популярності телеграму та його можливостям у сфері автоматизації та зручного взаємодії з користувачами.

Одним із ключових аспектів використання телеграму є забезпечення простоти та зручності у взаємодії з користувачем, що робить його ідеальним інструментом для реалізації різноманітних інформаційних систем. Зокрема, використання телеграм ботів для автоматизованого пошуку об'єктів за геолокацією дає можливість користувачам швидко та ефективно отримувати необхідну інформацію без зайвих зусиль.

Метою даного дослідження є розробка та реалізація інформаційної системи, яка дозволить користувачам здійснювати пошук об'єктів за географічними координатами з використанням телеграм бота. Предметом дослідження є аналіз існуючих підходів до реалізації подібних систем, розробка архітектури системи та її програмна реалізація.

Основними завданнями дослідження є:

- Аналіз існуючих методів та технологій для реалізації інформаційних систем з автоматизованим пошуком об'єктів за геолокацією.
- Розробка архітектури інформаційної системи та вибір необхідних технологій.

- Програмна реалізація інформаційної системи з використанням телеграм бота.

Тестування та оцінка ефективності розробленої системи.

У результаті дослідження планується отримати працездатну інформаційну систему, яка забезпечить користувачам зручний та швидкий доступ до інформації про об'єкти за їх географічним розташуванням через телеграм бота.

Ця робота має важливе значення в контексті розвитку сучасних інформаційних технологій та може бути корисною для широкого кола зацікавлених осіб, включаючи дослідників, розробників програмного забезпечення та підприємців.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Розробка телеграм бота

Телеграм-боти є важливими інструментами для автоматизації спілкування з користувачами та надання різноманітних послуг через популярну месенджерову платформу Telegram. Розробка таких ботів може бути цікавою та корисною задачею для програмістів і бізнес-власників. Ключовими аспектами розробки телеграм-бота є [1]:

- Визначення мети бота: Першим кроком є визначення цілей та завдань, які бот повинен виконувати. Це може бути надання інформації, обробка замовлень, автоматизація діяльності тощо.
- Вибір мови програмування: Для розробки телеграм-бота ви можете використовувати різні мови програмування, такі як Python, Node.js, Ruby тощо. Вибір мови залежить від вашого досвіду та конкретних потреб проекту.
- Використання Telegram Bot API: Telegram надає Bot API, яке дозволяє взаємодіяти з ботом. Вам потрібно буде зареєструвати бота і отримати токен доступу для використання API.
- Розробка бота: Розробка включає в себе створення логіки бота, обробку повідомлень користувачів, відправку відповідей і виконання запитів.
- Збереження та аналіз даних: Якщо ваш бот збирає дані користувачів, важливо забезпечити безпеку та конфіденційність цих даних, а також можливість їх аналізу для вдосконалення роботи бота.
- Інтеграція зі сторонніми сервісами: Бот може бути інтегрований зі сторонніми сервісами, такими як бази даних, зовнішні API, електронні платіжні системи тощо.
- Тестування і вдосконалення: Перед запуском бота важливо провести тестування на помилки і забезпечити високу якість роботи.
- Розгортання і публікація: Після успішного тестування можна розгорнути бота і забезпечити його доступність для користувачів.

- Підтримка і оновлення: Робота над ботом не закінчується після запуску. Важливо надавати підтримку і вносити оновлення для поліпшення функціональності.

Розробка телеграм-бота – це процес, який може допомогти автоматизувати багато рутинних завдань і полегшити спілкування з користувачами. З правильним підходом і ресурсами можливо створити дуже корисний та ефективний бот.

1.2 Огляд технологій та систем що використовують геолокацію

В сучасному світі існує багато інформаційних систем, спрямованих на автоматизований пошук об'єктів за їх геолокацією. Деякі з них використовують технології геолокації для визначення місця розташування об'єктів, інші — мобільні додатки або веб-сервіси, що дозволяють користувачам швидко знаходити потрібні об'єкти в конкретному регіоні.

Google Maps є одним із найпопулярніших сервісів, який надає можливість швидко знаходити об'єкти за їх географічним розташуванням. Він використовує технології геолокації та карту, щоб допомогти користувачам знаходити різноманітні об'єкти, такі як ресторани, магазини, готелі тощо (рисунок 1.1).

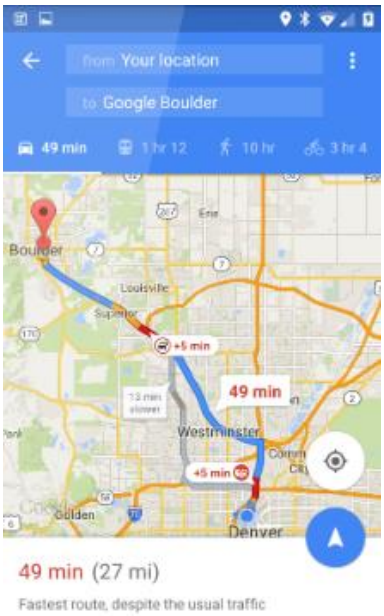

Рисунок 1.1 – Пошук маршруту в Google Maps

Foursquare — ще одна популярна платформа, яка дозволяє користувачам знаходити різноманітні заклади за їх географічним розташуванням. Вона також надає можливість додавати відгуки, оцінки та рецензії про об'єкти, що дозволяє отримувати більш детальну інформацію про них.

У контексті використання телеграм бота для автоматизованого пошуку об'єктів за геолокацією, слід звернути увагу на існуючі боти, які надають схожі послуги. Наприклад, українська платформа для пошуку роботи Joberz запустила Telegram-бот для пошуку роботи поруч із домом. Він показує усі актуальні вакансії за обраною геолокацією. Знайти їх можна як в Україні, так і за кордоном.

Технологічна складова інформаційних систем для автоматизованого пошуку об'єктів за геолокацією також включає в себе різноманітні API та сервіси для роботи з геоданими. Наприклад, Google Maps API та OpenStreetMap надають можливості для отримання геоданих та інтеграції з ними у власні додатки.

1.3 Зберігання постійних та тимчасових файлів

В рамках розробки інформаційної системи для автоматизованого пошуку об'єктів за геолокацією з використанням телеграм бота, виникає необхідність ефективного зберігання як постійних, так і тимчасових файлів у базах даних. Постійні файли, такі як дані про об'єкти, повинні бути збережені на постійній основі для подальшого використання, тимчасові файли можуть зберігатися протягом обмеженого часу під час роботи з додатком.

Прикладом бази даних, яка може бути використана для зберігання постійних та тимчасових файлів у цьому проекті, є SQLite. SQLite — це легка, вбудовувана база даних, яка зберігає базу даних у одному файлі на диску.

У розробці інформаційної системи для автоматизованого пошуку об'єктів за геолокацією з використанням телеграм бота можуть бути застосовані різні технології, такі як мови програмування Python для розробки бота, бази даних для зберігання інформації про об'єкти, а також інструменти для роботи з геоданими та їх візуалізації.

1.4 Постановка задачі

Метою даного проекту є створення інформаційної системи, яка базується на використанні телеграм бота для автоматизованого пошуку об'єктів за геолокацією. Розроблена система має наступні основні завдання:

- Збір та збереження інформації про об'єкти за географічними координатами, включаючи дані про їхню назву, тип, адресу та інші характеристики.
- Реалізація функціоналу пошуку об'єктів за геолокацією через телеграм бота з можливістю вказання типу об'єкта та радіусу пошуку.
- Відображення результатів пошуку на карті для користувача з можливістю детального перегляду інформації про об'єкти.
- Імплементація функціоналу зберігання історії пошуків та можливості зберігання обраних об'єктів у "вибраному".

Для вирішення вказаних завдань потрібно виконати наступні кроки:

- Оцінити можливості та аналізувати аналогічні системи для пошуку об'єктів за геолокацією та їхній функціонал.
- Вибрати відповідні технології для розробки та інтеграції телеграм бота та картографічного сервісу.
- Розробити та реалізувати архітектуру та дизайн інформаційної системи, включаючи модулі для зберігання даних, обробки запитів користувачів та взаємодії з телеграм API.
- Провести тестування та вдосконалення системи з урахуванням отриманих результатів.
- Реалізувати функціонал надсилання сповіщень та забезпечити масштабованість системи для ефективної роботи з великою кількістю користувачів.

Предметом дослідження є методологія обліку та аналізу даних про об'єкти за геолокацією з використанням телеграм бота.

Наукова новизна полягає в можливості ефективного збору, зберігання та аналізу даних про об'єкти за геолокацією з використанням телеграм бота.

Практичне значення полягає в підвищенні зручності та ефективності пошуку об'єктів за геолокацією через використання автоматизованої інформаційної системи на основі телеграм бота.

2 ВИБІР МЕТОДІВ РІШЕННЯ ЗАДАЧІ

2.1 Розробка телеграм бота на мові програмування Python

Python — це універсальна та потужна мова програмування, яка відома своєю простотою вивчення та ефективністю. Її елегантний синтаксис і динамічна типізація сприяють швидкому створенню програмних рішень у багатьох сферах.

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об’єктно-орієнтованого програмування, що робить його ідеальним для широкого спектру завдань. Від веб-розробки до аналізу даних, від штучного інтелекту до наукових обчислень — Python має засоби для будь-яких потреб.

Однією з великих переваг Python є наявність великої кількості бібліотек та фреймворків у відкритому доступі. Ці інструменти спрощують вирішення різноманітних завдань і роблять Python популярним вибором серед розробників[2, 12].

Для створення телеграм боту були використані такі бібліотеки:

- telebot (pyTelegramBotAPI)[3] - це бібліотека для мови програмування Python, яка спрощує створення та розробку ботів для Telegram. Вона надає простий і зручний інтерфейс для взаємодії з Telegram Bot API і дозволяє створювати різноманітні функції та можливості для вашого бота.
- Requests[4] – це бібліотека, що широко використовується для виконання HTTP-запитів на мові Python. Вона створена для спрощення взаємодії з API та веб-сервісами, отримання даних із веб-сайтів та виконання інших завдань на основі HTTP.
- sqlite3[5, 14] - це вбудована бібліотека мови програмування Python, яка надає інтерфейс для взаємодії з базами даних SQLite. SQLite - це легковагова, сервер-менше, вбудована система управління базами даних, яка зберігає дані у локальних файлових базах.

- Haversine[6-7] - це бібліотека Python, яка надає зручний і простий спосіб обчислення відстаней між точками на поверхні Землі за допомогою формули Гаверсайна. Ця бібліотека особливо корисна для проектів, пов'язаних із геолокацією, таких як системи навігації, аналіз маршрутів, обчислення відстаней між географічними точками.

Імпорт всіх бібліотек до проекту в компіляторі (рисунок 2.1):

```
import telebot
import requests
import bd
import conf
from telebot import types
from haversine import haversine
import sqlite3
```

Рисунок 2.1 - Імпорт всіх бібліотек

Отже, Python — це не лише мова програмування, але й потужний інструмент для творення інноваційних програмних рішень у різних областях. Його простота та ефективність дозволяють розробляти як малі сценарії, так і великі програмні проєкти з високою продуктивністю.

Тож після встановлення всіх програмних засобів для роботи з телеграм ботом потрібно створити самого бота.

2.2 Реєстрація телеграм бота за допомогою BotFather

BotFather в Telegram - це спеціальний бот, який дозволяє створювати та керувати іншими ботами на платформі Telegram. Він надає широкі можливості для налаштування та керування вашими ботами[8]. Ось деякі основні можливості BotFather:

- Створення нового бота: Ви можете створити нового телеграм-бота, визначивши його ім'я та отримавши унікальний API-токен для доступу до Telegram API.

- Керування налаштуваннями бота: BotFather дозволяє налаштовувати різні параметри для вашого бота, такі як опис, аватар, команди бота, повідомлення про вітаємо та багато інших.
- Отримання API-токену: Після створення бота BotFather надасть вам API-токен, який необхідно використовувати при розробці бота для доступу до Telegram API.
- Керування командами бота: Ви можете налаштовувати команди, які бот розуміє, і пов'язувати їх з конкретними діями.
- Налаштування аватара та імені бота: Ви можете встановити зображення профілю та ім'я для вашого бота.
- Керування правами доступу: BotFather дозволяє включати або вимикати різні можливості для бота, такі як можливість відправляти повідомлення в групи чи канали.
- Генерація URL-запрошення: Ви можете згенерувати URL-запрошення для вашого бота, щоб інші користувачі могли легко підписатися на нього.
- Відправлення повідомлень користувачам: BotFather дозволяє відправляти повідомлення іншим користувачам бота, щоб повідомити їх про оновлення або важливу інформацію.
- BotFather є потужним інструментом для створення та керування телеграм-ботами на платформі Telegram і надає багато можливостей для налаштування та розвитку вашого бота.

Тож нам потрібно зареєструвати бота. Для цього пишемо боту @BotFather команду /newbot, після цього даємо боту ім'я та тег. Після цих дій робота надішле нам токен, який нікому давати не можна (рисунок 2.2):

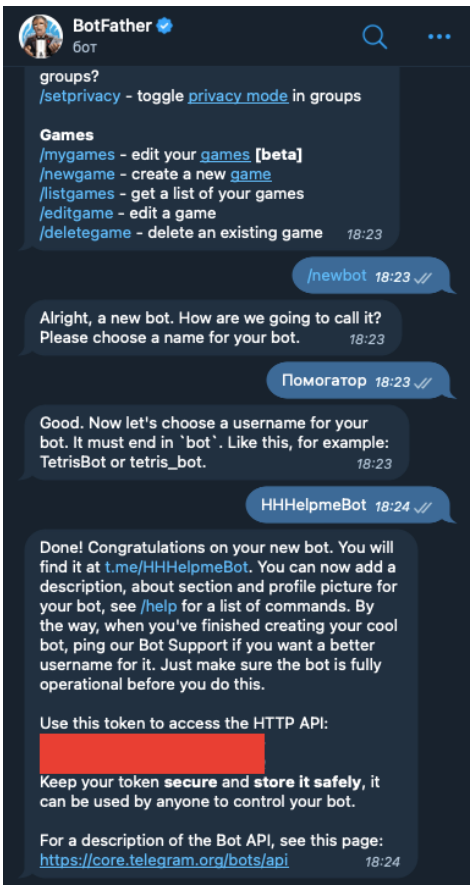

Рисунок 2.2 – Реєстрація бота

2.3 Використання Google Maps API та його сервісів

У рамках нашого проекту "Інформаційна система для автоматизованого пошуку об'єктів за геолокацією з використанням телеграм бота" велике значення приділяється використанню Google Maps API та його сервісу Places API. Ці інструменти забезпечують широкі можливості для реалізації функціоналу системи та забезпечення високої якості обслуговування користувачів.

Google Maps API[13] виступає основним інструментом для взаємодії з геоданими та картографічною інформацією. З його допомогою ми можемо отримувати дані про місцеположення об'єктів, визначати відстані між ними, а також відображати ці об'єкти на мапі. Цей API дозволяє розробникам інтегрувати картографічні функції Google Maps у свої власні веб-сайти та додатки, створюючи різноманітні та інтерактивні картографічні застосунки. Google Maps

API надає дуже зручний та документований інтерфейс програмування, який дозволяє легко і ефективно інтегрувати його в нашу інформаційну систему.

Google Maps API надає доступ до різних сервісів (рис. 2.3), які можна використовувати для взаємодії з картами та отримання різних типів інформації. Ось короткий огляд основних сервісів Google Maps API:

- Картографічні сервіси: Ці сервіси дозволяють вставляти інтерактивні карти Google Maps у ваші веб-сайти та додатки. Вони надають функції для зміни виду карти, додавання маркерів, полігонів, ліній, а також для взаємодії користувача з картами.
- Геокодування: Сервіс геокодування дозволяє перетворювати адреси у географічні координати (широту та довготу) та навпаки. Він допомагає знаходити місцезнаходження об'єктів за їхніми адресами або, навпаки, отримувати адреси за вказаними координатами.
- Маршрутизація та навігація: Ці сервіси дозволяють створювати маршрути між різними точками та отримувати навігаційні інструкції для подорожей автомобілем, пішки, велосипедом або громадським транспортом.
- Пошук місць: Пошукові сервіси дозволяють виконувати пошук різних об'єктів за різними критеріями, такими як назва місця, категорія, географічні координати тощо. Вони допомагають знаходити ресторани, готелі, магазини та інші об'єкти.
- Служби геозон: Ці сервіси дозволяють створювати та управляти геозонами - областями на мапі з певними географічними властивостями. Вони допомагають визначати, чи знаходиться користувач в певній геозоні, та виконувати певні дії в залежності від цього.
- Аналітика та візуалізація даних: Ці сервіси дозволяють візуалізувати дані на мапі та отримувати аналітичні звіти щодо використання вашого додатку. Вони допомагають відслідковувати рух користувачів, відображати теплові карти, аналізувати поведінку користувачів та інше.

Рисунок 2.3 – Google maps api service

Одним із ключових сервісів Google Maps API, який буде використовуватися в нашому проекті, є Places API[9](рисунок 2.4). Цей сервіс дозволяє отримувати детальну інформацію про різноманітні місця та об'єкти в певній географічній області. Використання Places API дозволить нам забезпечити користувачів нашого додатку актуальною та корисною інформацією про об'єкти, такі як ресторани, готелі, магазини тощо, що знаходяться поруч з їх поточним місцеположенням.

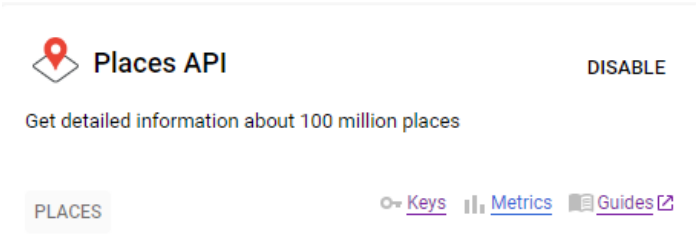

Рисунок 2.4 – Places API

Основні можливості Places API включають:

- Пошук об'єктів за ключовими словами або категоріями: Користувачі можуть шукати об'єкти за ключовими словами, такими як "піцерія" або "бензозаправка", що дозволяє їм знаходити потрібні послуги швидко та зручно.

- Детальна інформація про об'єкти: Places API надає доступ до різноманітної інформації про кожен об'єкт, таку як адреса, контактні дані, години роботи, відгуки користувачів та інше.
- Відстань та маршрутизація: З Places API ми можемо обчислювати відстані між об'єктами та створювати оптимальні маршрути для користувачів на основі їх поточного місцеположення.

Використання Google Maps API та сервісу Places API в нашому проекті дозволить нам створити потужний та зручний інструмент для автоматизованого пошуку об'єктів за геолокацією через телеграм бота, що відповідає сучасним стандартам і вимогам користувачів.

Документація до Places API від Google надає зрозумілі та докладні інструкції щодо використання цього сервісу у вашому проекті. Вона включає в себе розширені пояснення функцій та можливостей API, приклади коду для різних використань, а також інформацію про параметри запитів та відповіді сервера (рисунок 2.5).

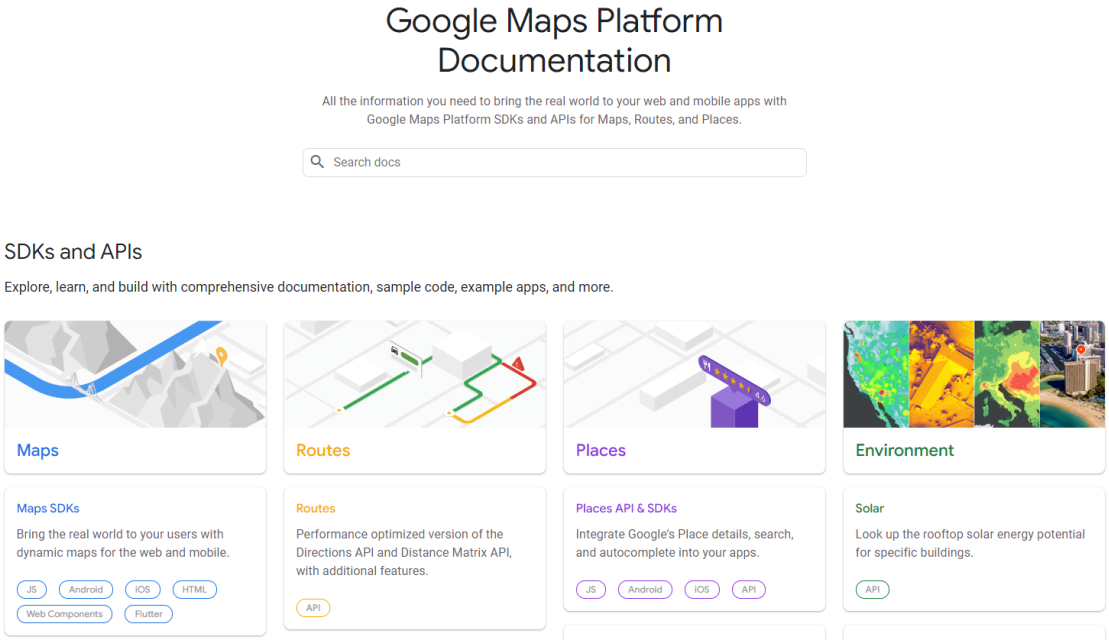

Рисунок 2.5 – Документація до Places API

Іншим сервісом Google Maps API, який буде використовуватися в нашому проекті, є Geocoding API[10].

Geocoding API - це інтерфейс програмування додатків, який дозволяє перетворювати адреси в географічні координати (широту та довготу), які можуть бути використані для розміщення маркерів на карті, або визначення місця на Землі. Також можливий зворотний процес - перетворення географічних координат у людські адреси.

Навіщо використовувати Geocoding API

Використовуйте Geocoding API для веб-сайту або мобільного додатку, коли ви хочете використовувати геокодування даних на картах, наданих одним із API Google Maps Platform. За допомогою Geocoding API ви використовуєте адреси для розміщення маркерів на карті, або Перетворіть маркер на карті на адресу. Цей сервіс призначений для геокодування заздалегідь визначених, статичних адрес для розміщення додатків вміст на карті.

Що можна робити за допомогою Geocoding API

Ви можете використовувати Geocoding API для отримання даних геокодування для однієї або декількох адрес або місць, у тому числі таких:

- Географічні координати адрес.
- Адреси для наборів координат широти та довготи.
- Адреси для ідентифікаторів місць.

Ви можете контролювати, де відображаються результати, і обмежувати результати певним регіоном, округу, або поштовий індекс.

Як працює Geocoding API

Geocoding API виконує як геокодування, так і зворотне геокодування:

- Геокодування: перетворює адреси, такі як "1600 Amphitheatre Parkway, Mountain View, CA" у координати широти та довготи або місця Ідентифікатори. Ці координати можна використовувати для розміщення маркерів на карті, а також для центрування чи переміщення карти у кадрі перегляду.

- Зворотне геокодування: перетворює координати широти/довготи або ідентифікатор місця в доступну для читання адресу. Ви можете використовувати адреси для різних сценаріїв, зокрема для доставки або самовивозу.

У наведеній нижче демонстрації (рисунок 2.6) використовується служба геокодування щоб продемонструвати, як працює Geocoding API.

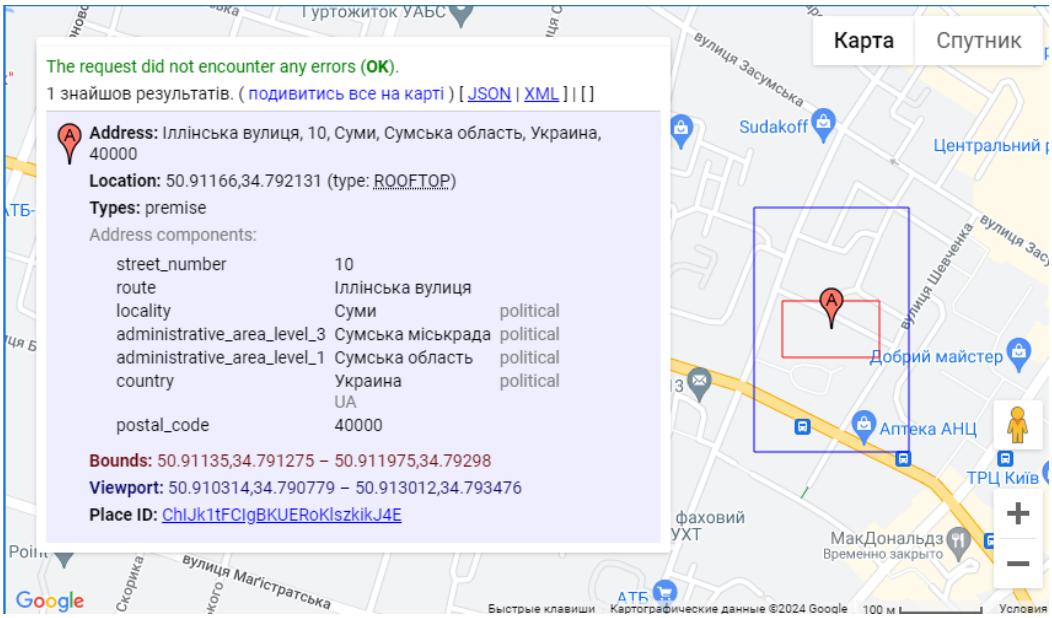

Рисунок 2.6 – Демонстрація використання Geocoding API

2.4 Зберігання даних за допомогою Sqlite

У рамках розробки інформаційної системи для автоматизованого пошуку об'єктів за геолокацією з використанням телеграм бота, було прийняте рішення використовувати базу даних SQLite для зберігання тимчасових та постійних файлів[11]. SQLite було обрано через його легкість використання, портативність та можливість працювати без необхідності окремого сервера баз даних.

Зберігання тимчасових даних: Для зберігання тимчасових даних, таких як історія пошуку користувачів та їх сесійні дані, було створено відповідні таблиці у базі даних SQLite. Ці дані зберігаються локально на пристрої та використовуються для підтримки функціоналу та персоналізації взаємодії з користувачами.

Зберігання постійних файлів: Постійні дані, такі як інформація про об'єкти (ресторани, готелі, лікарні, заправки), доступні для пошуку користувачів, також зберігаються у базі даних SQLite. Для цього було створено відповідні таблиці, які відображають різні типи об'єктів та їх властивості.

Використання транзакцій: З метою забезпечення цілісності даних та виконання групових операцій з даними, використовуються транзакції у базі даних SQLite. Це дозволяє здійснювати складні операції з даними без ризику втрати чи пошкодження інформації.

Ось деякі переваги використання SQLite у даному проекті:

- Простота використання: SQLite не потребує окремого сервера, що робить його дуже простим у використанні. Ви можете використовувати його безпосередньо з ваших програм, включаючи додатки на Python, що є зручним для розробки телеграм бота.
- Легка вбудовуваність: SQLite може бути вбудованою у вашу програму, що спрощує розповсюдження та встановлення.
- Ефективність: Для невеликих обсягів даних та простих операцій SQLite може бути досить ефективним, що важливо для швидкої реакції телеграм бота на запити користувачів.
- Підтримка транзакцій: SQLite підтримує транзакції, що дозволяє забезпечити консистентність даних при виконанні групи операцій.
- Кросплатформенність: SQLite підтримується на багатьох платформах, включаючи Windows, macOS та різні дистрибутиви Linux.

В мові програмування Python навіть існує вбудована бібліотека sqlite3 (рисунок 2.7), яка дозволяє легко взаємодіяти з базами даних SQLite. Ця бібліотека надає зручний інтерфейс для створення, читання, запису та видалення даних у базі даних SQLite без необхідності встановлення додаткових зовнішніх пакетів.

За допомогою бібліотеки sqlite3 можна створити нову базу даних SQLite та визначити таблиці і структуру даних безпосередньо з коду Python. Навіть якщо

база даних SQLite ще не існує, бібліотека sqlite3 автоматично створить її під час виконання першого з'єднання з базою даних.

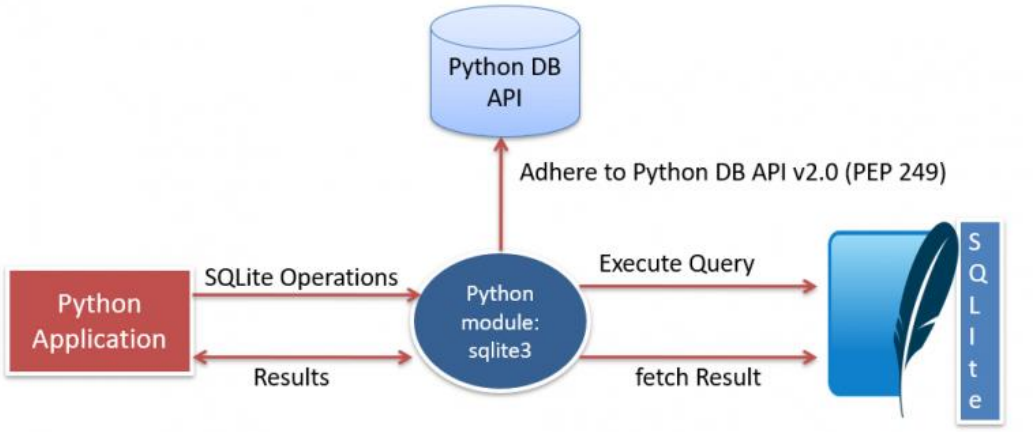

Рисунок 2.7 – Бібліотека sqlite

3 ІНФОРМАЦІЙНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Розробка дизайну інформаційної системи

У цьому розділі буде представлено розробку дизайну інформаційної системи для автоматизованого пошуку об'єктів за геолокацією з використанням телеграм бота. Основним завданням є створення зручного та ефективного інтерфейсу для взаємодії користувачів з системою, а також розробка алгоритмів, які забезпечать точний та швидкий пошук необхідних об'єктів.

Опис функціональних можливостей системи

Система дозволяє користувачам виконувати такі основні дії:

- Відправлення місцезнаходження: Користувач може поділитися своїм поточним місцезнаходженням з ботом. Ця функція реалізується через інтерфейс телеграму, де користувач натискає кнопку "Відправити місцезнаходження". Бот використовує цю інформацію для подальшого пошуку об'єктів у зазначеній зоні.
- Вибір типу об'єкта для пошуку: Користувач обирає тип об'єкта, який він хоче знайти, серед доступних варіантів: ресторан, готель, заправка, лікарня або бомбосховище. Такий підхід дозволяє адаптувати пошук відповідно до потреб користувача та забезпечити релевантність результатів.
- Пошук найближчих об'єктів: Після отримання типу об'єкта бот здійснює пошук об'єктів обраного типу в радіусі 20 кілометрів від поточного місцезнаходження користувача. Використовуючи Google Maps API, бот отримує точну та актуальну інформацію про об'єкти в зазначеній зоні.
- Пошук найближчих бомбосховищ: Однією з основних функцій системи є можливість пошуку найближчих бомбосховищ. Ця функція є надзвичайно актуальною в наш час, коли безпекові питання стають все більш важливими. Користувачі можуть швидко знайти найближчі бомбосховища, що розташовані в радіусі 20 кілометрів від їхнього поточного місцезнаходження.
- Перегляд історії пошуку: Користувач може переглядати історію своїх попередніх пошуків. Ця функція дозволяє зберігати та легко відновлювати

інформацію про раніше знайдені об'єкти, що може бути корисним для повторного доступу до потрібних даних.

- Додавання об'єктів до обраного: Користувач може додавати знайдені об'єкти до списку обраного для швидкого доступу в майбутньому. Це забезпечує можливість зберігати найважливіші та найчастіше використовувані об'єкти в персональному списку, що спрощує їх подальший пошук.
- Перехід до Google Maps: Користувач може отримати маршрут до обраного об'єкта через Google Maps. При натисканні на кнопку "Вибрати це місце", бот надає посилання, яке відкриває Google Maps з побудованим маршрутом до обраного об'єкта, що значно полегшує навігацію.

Розробка use case diagram.

Для ілюстрації взаємодії користувача з системою було створено діаграму використання (Use Case Diagram)(рисунок 3.1). На діаграмі представлені основні сценарії використання системи та взаємодії між користувачем та телеграм ботом.

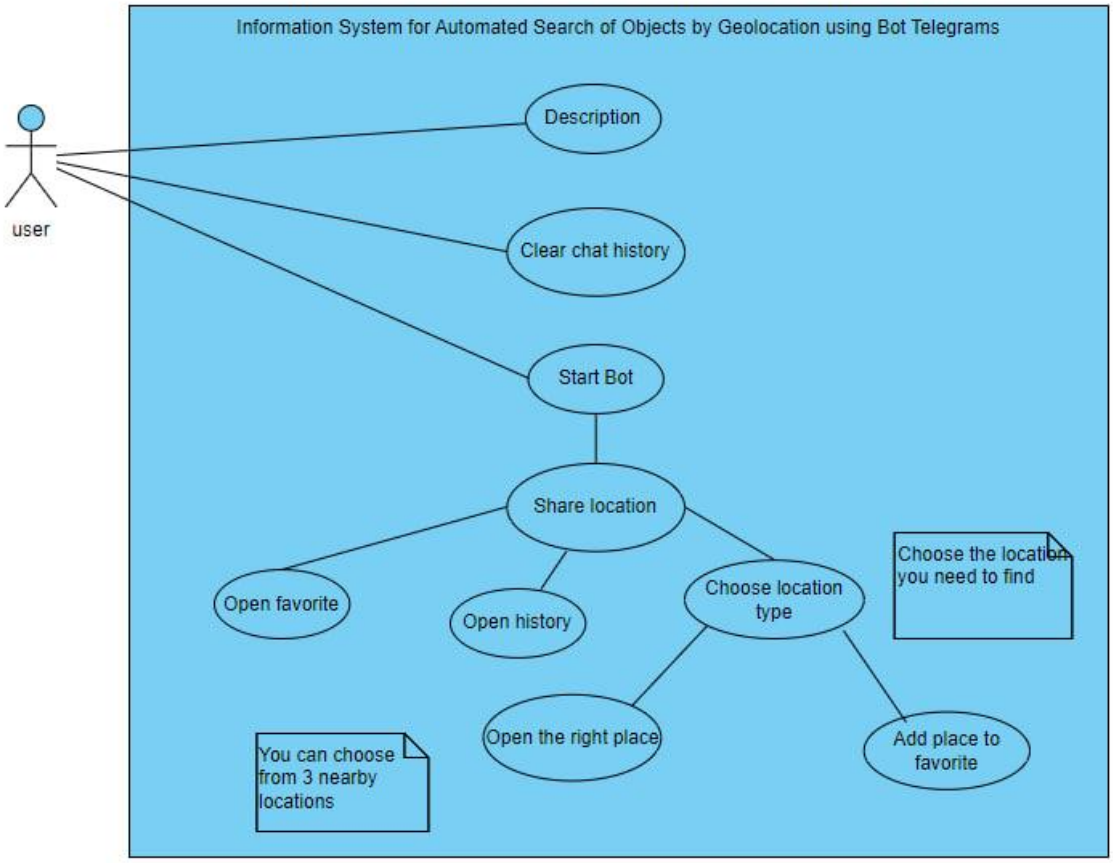

Рисунок 3.1 – Bot use case diagram

Опис діаграми використання

На діаграмі використання представлені наступні взаємодії:

- Користувач має можливість взаємодіяти з ботом через інтерфейс телеграму.
- Бот обробляє запити користувача та виконує наступні функції:
- Отримання та збереження місцезнаходження користувача.
- Надання можливості вибору типу об'єкта для пошуку (включаючи ресторани, готелі, заправки, лікарні та бомбосховища).
- Виконання пошуку найближчих об'єктів за допомогою Google Maps API.
- Збереження та відображення історії пошуків.
- Додавання об'єктів до списку обраного.
- Надання посилання для переходу до Google Maps з побудованим маршрутом до обраного об'єкта.

Користувач, взаємодіючи з ботом, може швидко знайти необхідні об'єкти, отримати детальну інформацію про них, зберегти цікаві місця для подальшого використання та побудувати маршрут до обраного місця. Це робить систему зручною та ефективною для щоденного використання, забезпечуючи користувачам доступ до важливої інформації у будь-який момент.

Розробка ERD до бази даних.

Етап розробки ERD (Entity-Relationship Diagram)(рисунок 3.2) є важливим кроком у створенні бази даних для телеграм бота, що здійснює автоматизований пошук об'єктів за геолокацією. ERD дозволяє візуалізувати структуру бази даних, визначити сутності, атрибути та взаємозв'язки між ними, що сприяє ефективному проєктуванню та подальшій реалізації системи.

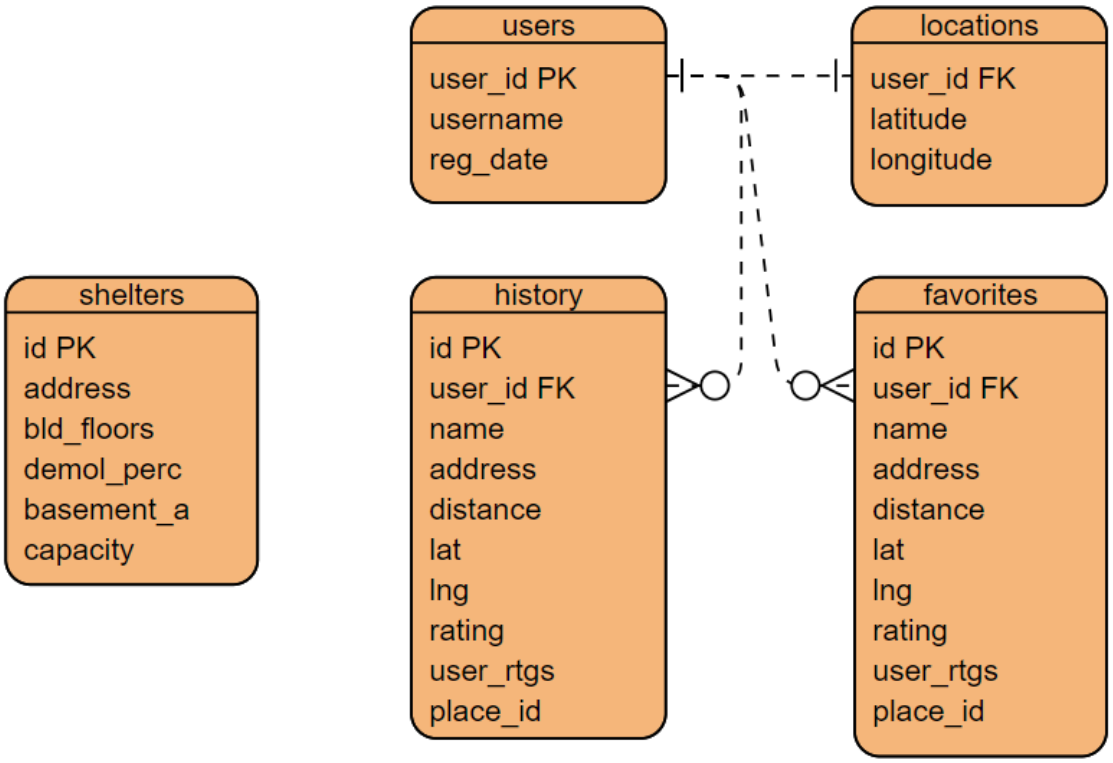

Рисунок 3.2 – Entity relationship diagram

Взаємозв'язки між сутностями

Користувачі (users) мають один-до-багатьох зв'язок з тимчасовим місцезнаходженням (locations), Історією пошуків (history), та Обраним (favorites). Кожен користувач може зробити багато запитів, мати багато записів в історії пошуків та обраних об'єктів, але кожен з цих записів належить лише одному користувачу.

3.2 Розроблення та імплементація

Проєкт "Інформаційна система для автоматизованого пошуку об'єктів за геолокацією з використанням телеграм бота" має на меті надати користувачам зручний інструмент для пошуку об'єктів поблизу їх поточного місцезнаходження. Основними компонентами системи є телеграм бот, модуль

для роботи з базою даних, а також модулі для обробки команд та «callback» запитів.

Основні компоненти системи:

Телеграм бот є основним інтерфейсом взаємодії користувача з системою. Він відповідає за прийом та обробку повідомлень від користувачів, виконання команд та відправку результатів пошуку. Використовуючи бібліотеку «telebot», ми створюємо бота, який взаємодіє з користувачем через API Telegram. Основні функції бота включають обробку команд «/start», отримання геолокації користувача та обробку «callback» запитів.

У файлі «main.py»(рисунок 3.3) відбувається ініціалізація бота та обробників команд та callback запитів. Використовуючи бібліотеку «telebot», створюється об'єкт бота з використанням API токена. Цей файл відповідає за запуск бота та забезпечення його безперервної роботи за допомогою методу «polling()».

```python
# Створюємо бота
bot = telebot.TeleBot(conf.BOT_API_TOKEN)
# Створюємо базу данних і таблиці
bd.create_db('locations.db')

# Обробник команди /start
@bot.message_handler(commands=['start'])
def start(message):
    CommandHandler.start(bot, message)

# Обробник місцезнаходження
@bot.message_handler(content_types=['location'])
def location(message):
    CommandHandler.location(bot, message)


# Обробник натискання на кнопку з вибором пошуку
@bot.callback_query_handler(func=lambda call: call.data.startswith('search'))
def callback_search(call):
    CallbackHandler.callback_search(bot, call)

# Обробник натискання на кнопку з вибором історії
@bot.callback_query_handler(func=lambda call: call.data.startswith('Menu'))
def callback_menu(call):
    CallbackHandler.callback_menu(bot, call)

bot.polling()
```

Рисунок 3.3 – Файл main.py

У цьому фрагменті коду визначено обробники для команди «/start», повідомлень з геолокацією та «callback» запитів. Ці обробники спрямовують

запити до відповідних функцій у модулях «command_handler» та «callback_handler».

Модуль конфігурації «conf.py»(рисунок 3.4) містить всі необхідні налаштування для роботи бота, включаючи API токен, Google API ключ, радіус пошуку, типи об'єктів для пошуку та дані про сховища. Цей модуль дозволяє легко змінювати налаштування системи без необхідності змінювати код основної програми. Наприклад, зміна радіусу пошуку або додавання нових типів об'єктів може бути виконана лише шляхом редагування конфігураційного файлу.

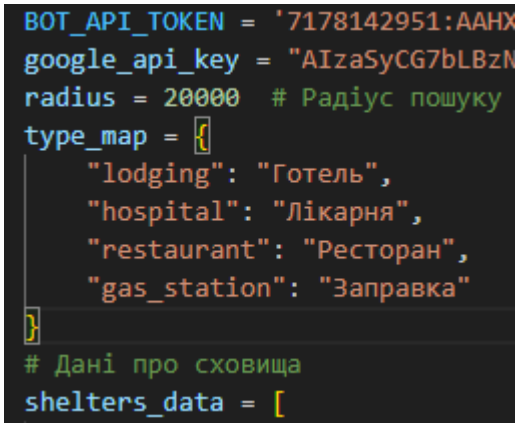

Рисунок 3.4 – Файл conf.py

Конфігураційний файл містить API токен для Telegram бота, ключ для Google API, радіус пошуку об'єктів та інші важливі налаштування. Наприклад, «type_map» використовується для пошуку типів об'єктів з англійської на українську мову, що дозволяє користувачам отримувати результати на їхній рідній мові.

Обробник команд «command_handler.py»(рисунок 3.5) відповідає за обробку основних команд, таких як «/start» та повідомлення з геолокацією. Він взаємодіє з базою даних для збереження та отримання необхідних даних. Основні функції цього модуля включають реєстрацію користувачів, отримання та збереження геолокації, а також ініціювання процесу пошуку об'єктів на основі отриманої геолокації.

```python
def start(bot, message):···


def location(bot, message):
    bot.delete_message(message.chat.id, message.message_id)
    bot.delete_message(message.chat.id, message.message_id - 1)
    latitude = message.location.latitude
    longitude = message.location.longitude
    msg_id = message.message_id

    conn = sqlite3.connect('locations.db')
    bd.add_location(conn, message.from_user.username, latitude, longitude)
    conn.close()
```

Рисунок 3.5 – Файл command_handler.py

Функція «start» реєструє нового користувача в базі даних та вітає його, пропонуючи надіслати геолокацію. Функція «location» отримує геолокацію користувача та зберігає її в базі даних, після чого ініціює процес пошуку об'єктів.

Обробник запитів «callback_handler.py»(рисунок 3.6) відповідає за обробку callback запитів від користувачів. Він дозволяє користувачам виконувати додаткові дії, такі як пошук об'єктів, перегляд історії запитів та управління улюбленими місцями. Callback запити використовуються для забезпечення інтерактивності ботів, дозволяючи користувачам вибирати різні опції через інтерфейс Telegram.

```python
def callback_search(bot, call):
    msg_del = int(call.data.split(':')[1])
    chat_id = call.message.chat.id
    bot.delete_message(chat_id, call.message.message_id)
    for i in range(msg_del + 1, call.message.message_id):
        bot.delete_message(chat_id, i)

    msg_id = call.message.message_id
    markup = telebot.types.InlineKeyboardMarkup()
    btn1 = telebot.types.InlineKeyboardButton('Готель', callback_data=f'locate-lodging-{msg_id}')
    btn2 = telebot.types.InlineKeyboardButton('Лікарня ', callback_data=f'locate-hospital-{msg_id}')
    btn3 = telebot.types.InlineKeyboardButton('Ресторан ', callback_data=f'locate-restaurant-{msg_id}')
    btn4 = telebot.types.InlineKeyboardButton('Заправка ', callback_data=f'locate-gas_station-{msg_id}')
    btn5 = telebot.types.InlineKeyboardButton('Сховища🚀 ', callback_data=f'safezone:{msg_id}')
    btn6 = telebot.types.InlineKeyboardButton(text="🔙", callback_data=f"Menu:{msg_id}")
    markup.row(btn1, btn2)
    markup.row(btn3, btn4)
    markup.row(btn5, btn6)

    bot.send_message(call.message.chat.id, "Тепер вибери місце яке хочеш знайти:", reply_markup=markup)

def callback_menu(bot, call):···

def callback_history(bot, call):···

def callback_favor(bot, call):···
```

Рисунок 3.6 – Функція «callback_search»

Функція «callback_search»(рисунок 3.6) отримує збережене місцезнаходження користувача з бази даних та виконує пошук об'єктів. Якщо місцезнаходження відсутнє, користувачеві надсилається повідомлення з проханням надіслати геолокацію.

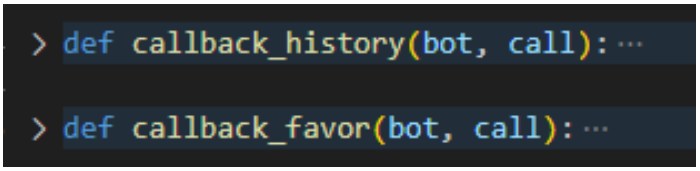

Рисунок 3.7 - history та favor

Функції «callback_history» та «callback_favor»(рисунок 3.7) повертають список з історією запитів та запити в обраному користувачеві у вигляді повідомлення з кнопками та посиланням на маршрут в Google Maps. Якщо місцезнаходження історія чи обране відсутнє бот відправить відповідне повідомлення .

```
def callback_locate(bot, call):
conn = sqlite3.connect('locations.db')
location = bd.get_location(conn, call.from_user.username)
conn.close()

if location:
    latitude = location[0]
    longitude = location[1]

locate_info = gma.find_nearest_locate(latitude, longitude, type)

if locate_info:
    # Відправляємо повідомлення з інформацією про всі місця
    for idx, locate in enumerate(locate_info, start=1):
        message_text = f"{idx}. {loc_type}\n" \
                       f"     Назва: {locate['name']}\n" \
                       f"     Адреса: {locate['address']}\n" \
                       f"     Відстань: {locate['distance']} км\n" \
                       f"     Рейтинг: {locate['rating']}⭐({locate['user_ratings_total']})\n" \
                       "     Натисніть⬇️, щоб перейти до google maps:"
        # Створюємо Inline Keyboard
        markup = types.InlineKeyboardMarkup()
        # Створюємо кнопку для вибору місця
        button = types.InlineKeyboardButton(text=f"Вибрати це місце", callback_data=f"locate_{idx}", url=f"ht
        button2 = types.InlineKeyboardButton(text="🌟", callback_data=f"Ffavorite||{locate['place_id']}")

        markup.add(button, button2)
        bot.send_message(call.message.chat.id, message_text, reply_markup=markup)
```

Рисунок 3.8 – Функція «callback_locate»

Функція «callback_locate»(рисунок 3.8) на основі отриманого місцезнаходження користувача та вибраного типу локації знаходить потрібні місця в радіусі 20 км та надсилає повідомлення з посиланням на маршрут в Google Maps.

```
def callback_safezone(bot, call):
    conn = sqlite3.connect('locations.db')
    location = bd.get_location(conn, call.from_user.username)
    shelters = bd.get_shelters(conn)

    if location:
        latitude = location[0]
        longitude = location[1]

    closest_idist = safe.closest_coordinate(latitude, longitude, shelters)
    closest_shelter = bd.get_closesd_shelter(conn, closest_idist[0])
    message_text = (
                f"\nАдреса: {closest_shelter[1].split(',')[0]}\n"
                f"Поверхів: {closest_shelter[2]}\n"
                f"Відсоток зносу: {closest_shelter[3]}%\n"
                f"Площа підвалу: {closest_shelter[4]}\n"
                f"Місткість(осіб): {closest_shelter[5]}\n"
                f"Відстань: { round(closest_idist[1], 1)} км\n"
                f".
            )
```

Рисунок 3.9 – Функція «callback_safezone»

Функція «callback_safezone»(рисунок 3.9) розраховує відстань між користувачем та сховищами що знаходять в базі даних та знаходить найближче сховище в місті та надсилає користувачеві всі данні про це місце разом з посиланням на маршрут в Google Maps.

Модуль бази даних «bd.py»(рисунок 3.10) забезпечує створення та керування базою даних, збереження даних користувачів, історії запитів та улюблених місць. Використання бази даних дозволяє зберігати інформацію про користувачів та їхні запити, забезпечуючи надійність зберігання даних. Модуль бази даних також включає функції для створення таблиць, додавання нових записів та отримання інформації з бази даних.

```python
def create_db(name):
    conn = sqlite3.connect(name)
    cursor = conn.cursor()

    cursor.execute('''...
    cursor.execute('''...
    cursor.execute('''...
    cursor.execute('''...
    cursor.execute('''...
    conn.close()

def clear_user_data(conn, user_id):...

def add_location(conn, user_id, latitude, longitude):
    cursor = conn.cursor()
    cursor.execute('INSERT INTO locations VALUES (?, ?, ?)', (user_id, latitude, longitude))
    conn.commit()
    cursor.close()

def get_location(conn, user_id):...

def get_closesd_shelter(conn, shalter_id):...
```

Рисунок 3.10 – Файл «bd.py»

3.3 Аналіз результатів(Тестування)

Після переходу за посиланням на телеграм бота ви отримаєте доступ до анотації щодо роботи, яка містить корисні поради та рекомендації до використання бота (рисунок 3.11).

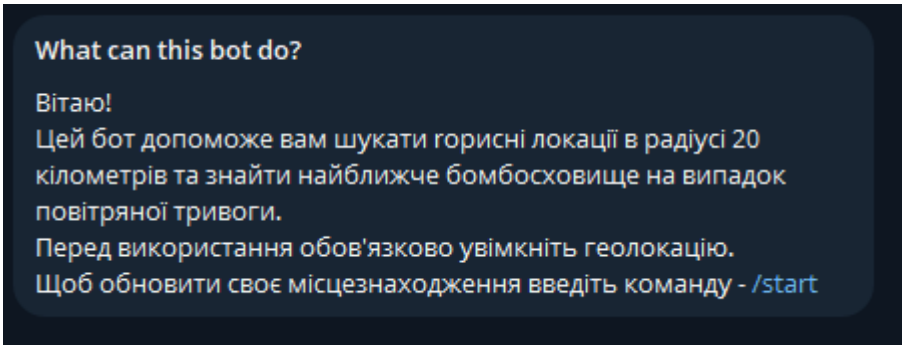

Рисунок 3.11 - Анотація до бота

Після запуску боту та введення команди /start (рисунок 3.12) бот відправляє повідомлення з проханням натиснути на кнопку «Відправити місцезнаходження» щоб поділитися своїм місцезнаходженням.

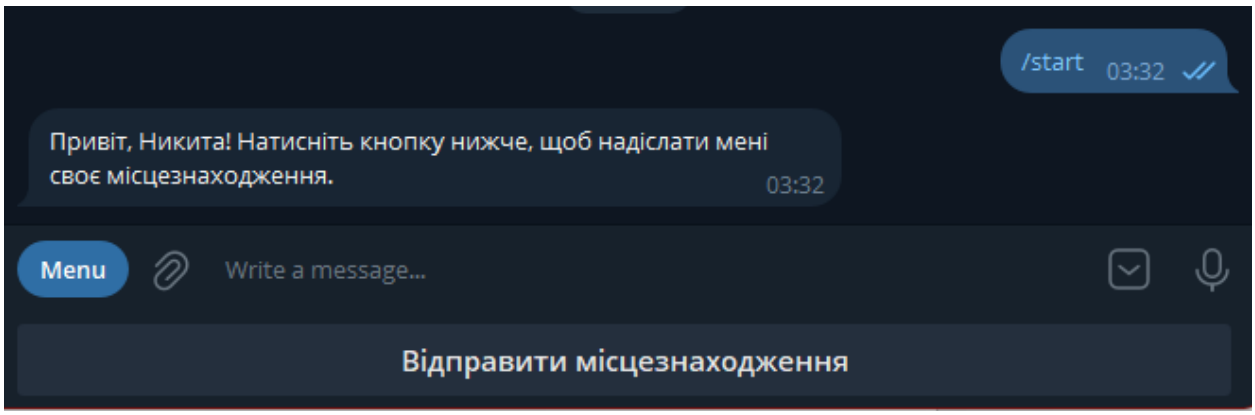

Рисунок 3.12 – Початок роботи з ботом

Після натискання на кнопку «Відправити місцезнаходження» ти поділишся своїм місцезнаходженням та бот відправить повідомлення з проханням вибрати розділ який тобі потрібен. (рисунок 3.13).

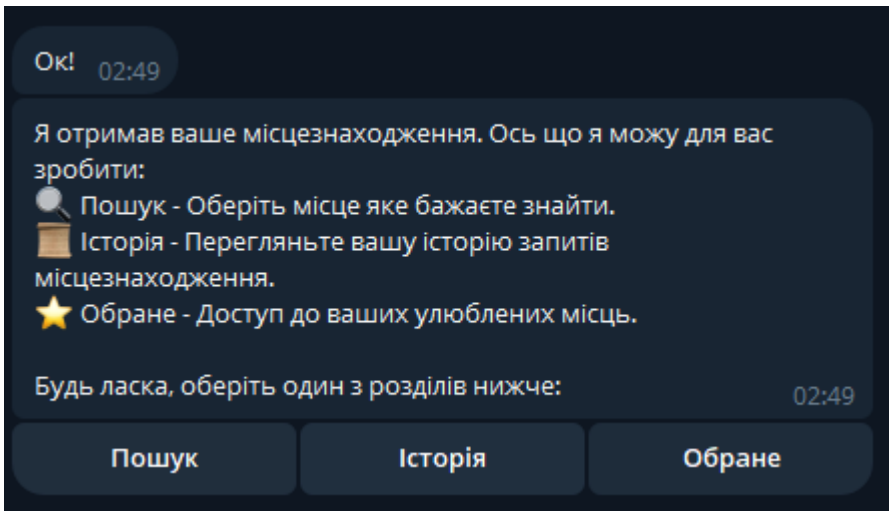

Рисунок 3.13 – Вибір розділу

Після натискання на кнопку «Пошук» бот відправить повідомлення з проханням вибрати місце яке ти хочеш знайти, до вибору представлені такі варіанти: «Готель», «Лікарня», «Ресторан», «Заправка», «Сховища» та кнопкою «Назад» (рисунок 3.14).

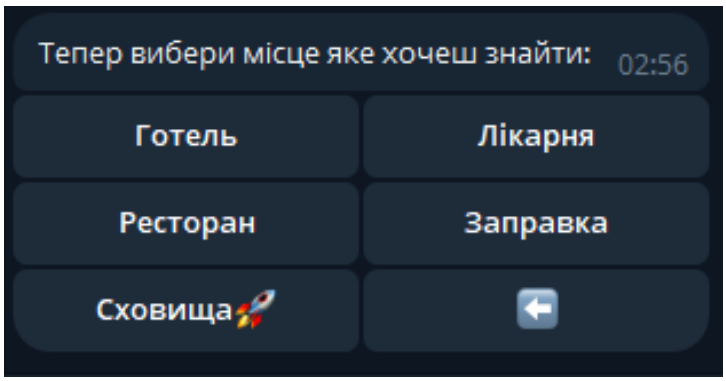

Рисунок 3.14 – Вибір місця пошуку

При натисканні на 1 з 4 варіантів: «Готель», «Лікарня», «Ресторан», «Заправка» бот надішле 3 найближчі місцезнаходження обраного типу в радіусі 20 кілометрів з можливістю перейти на Google Maps з маршрутом до цього місця, можливістю додати це місце до обраного та кнопкою назад (рисунок 3.15). А при натисканні на кнопку «Сховища» у відповідь бот відправить місцезнаходження найближчого бомбосховища в Сумах та можливістю перейти до маршруту на Google Maps з цим сховищем (рисунок 3.16).

Такий підхід дозволяє забезпечити користувачів інформацією про найближчі об'єкти без необхідності вручну вводити дані чи шукати їх самостійно.

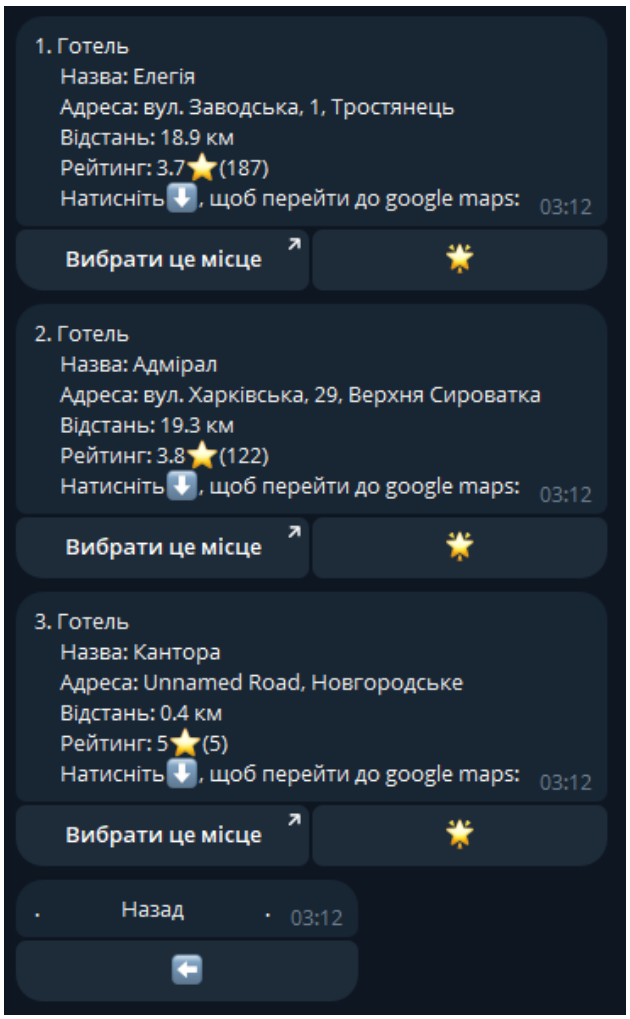

Рисунок 3.15 – Список найближчих місць

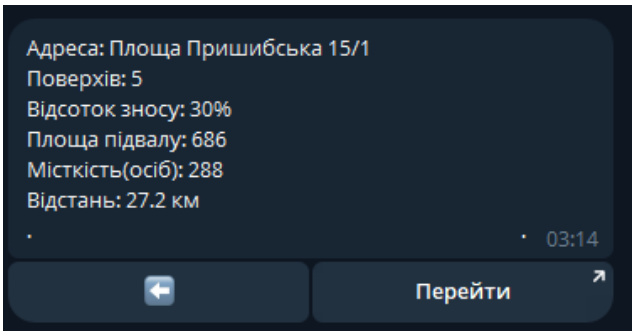

Рисунок 3.16 – Наближче бомбосховище

Після натискання на кнопку «Історія» бот відправить історію пошуку місць по 3 на одній сторінці з можливість переходити між сторінками вперед і назад, обирати їх в обране та кнопкою «назад» (рисунок 3.17).

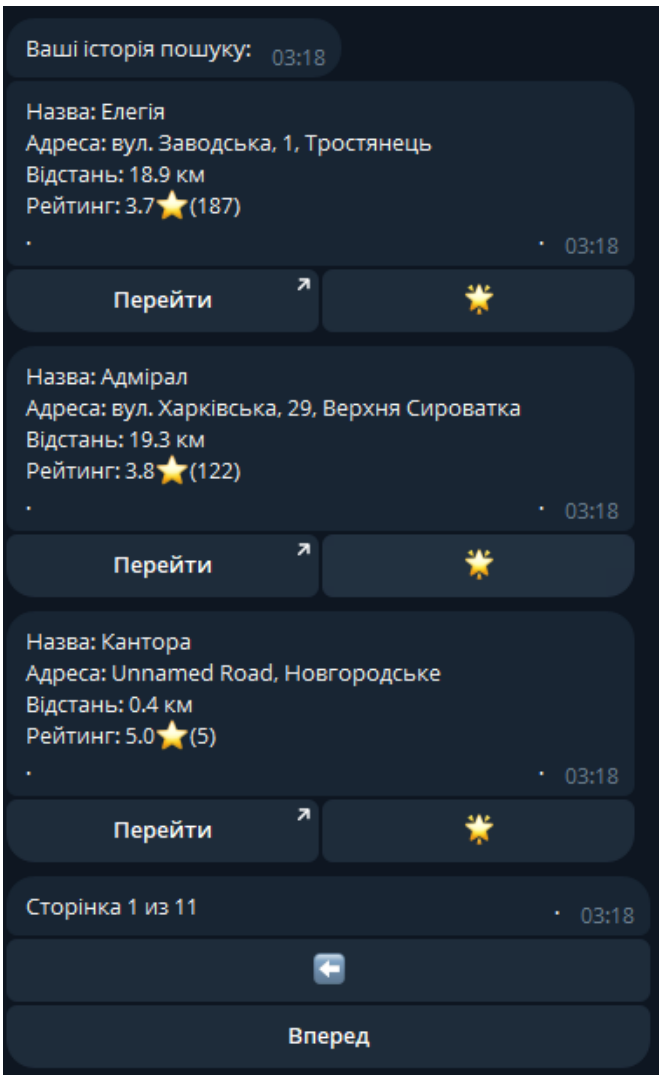

Рисунок 3.17 – Історія пошуку

Після натискання на кнопку «Обране» бот відправить обрані місця в тому ж форматі що і історія пошуку але замість можливості додати місце в обране буде видалення цього місцезнаходження з обраного (рисунок 3.18).

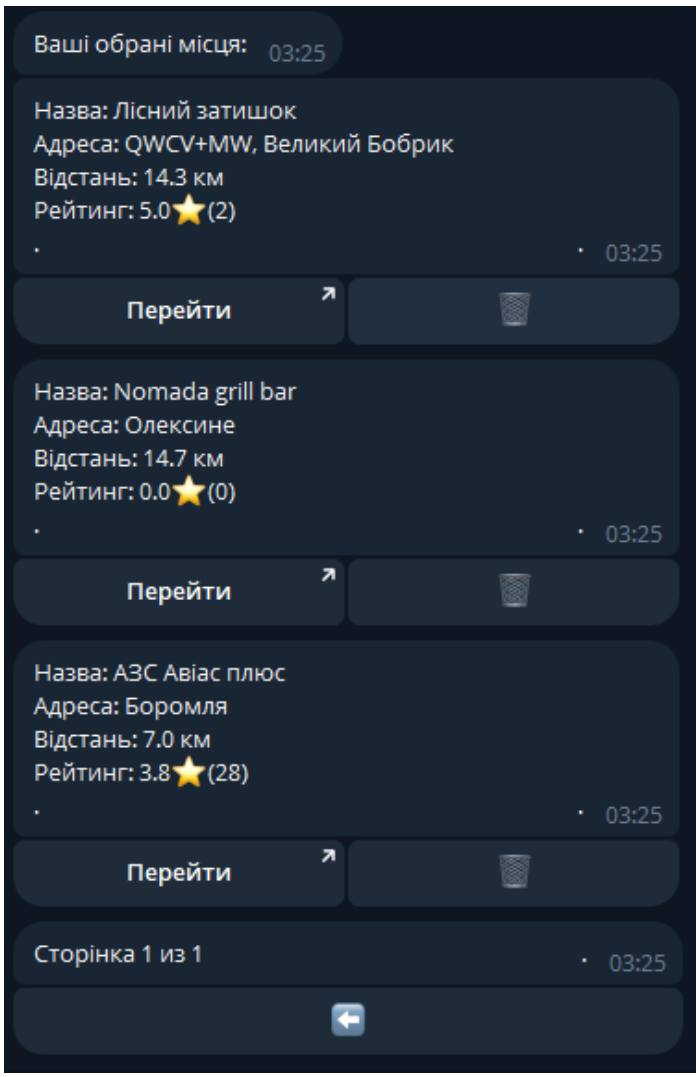

Рисунок 3.18 – Обране

Після натисканні на кнопку «Перейти» в любому з розділів з’явиться вікно з підтвердженням відкриття Google Maps з маршрутом до обраного місцезнаходження (рисунок 3.19).

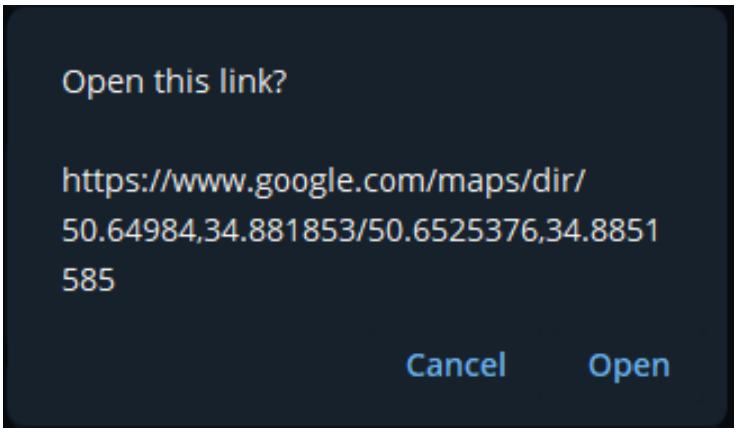

Рисунок 3.19 - Перехід до маршруту

Після підтвердження відкриється карта з маршрутом до вибраного нами раніше місцезнаходження (рисунок. 3.20).

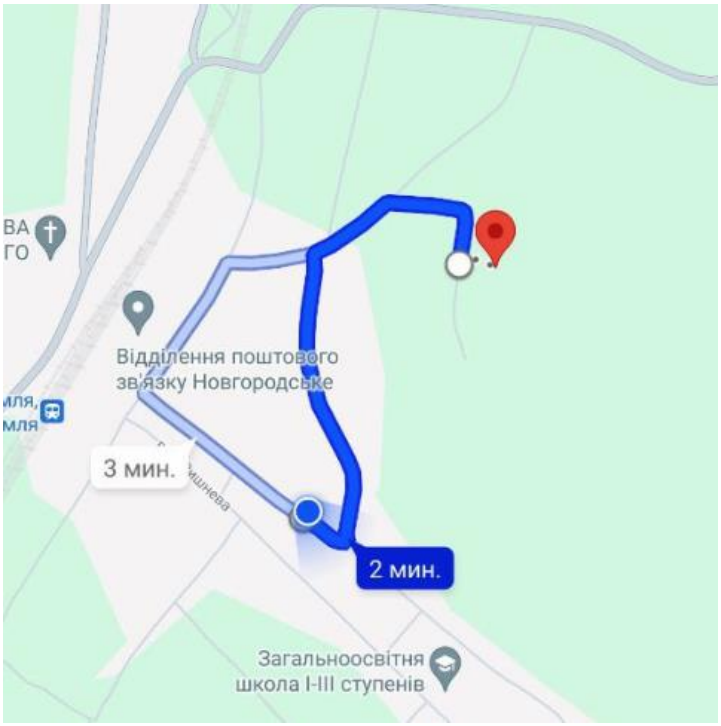

Рисунок 3.20 - Маршрут на Google Maps

Після натискання на кнопку щоб додати місце в обране з'явиться повідомлення про успішне додавання місцезнаходження до обраного, чи зауваження що місце вже в обраному (рисунок 3.21).

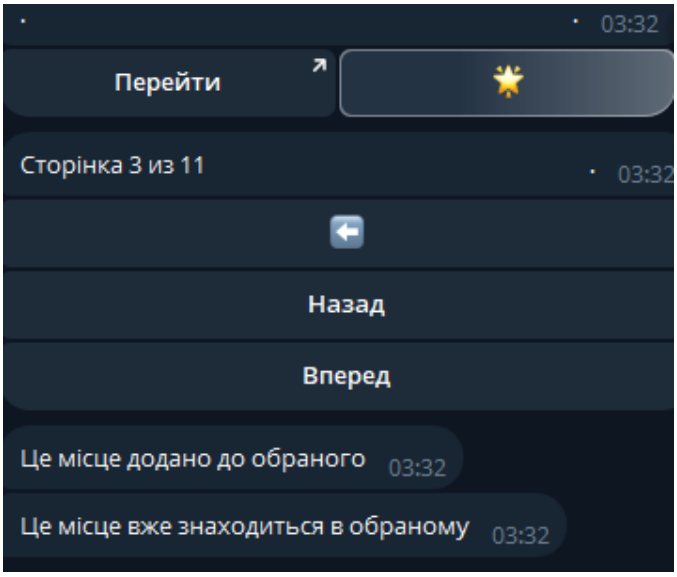

Рисунок 3.21 – Додавання до обраного

При натисканні на кнопку видалення місця з обраного бот відправить повідомлення про успішне видалення місця та кнопкою «Оновити» список обраного (рисунок 3.22).

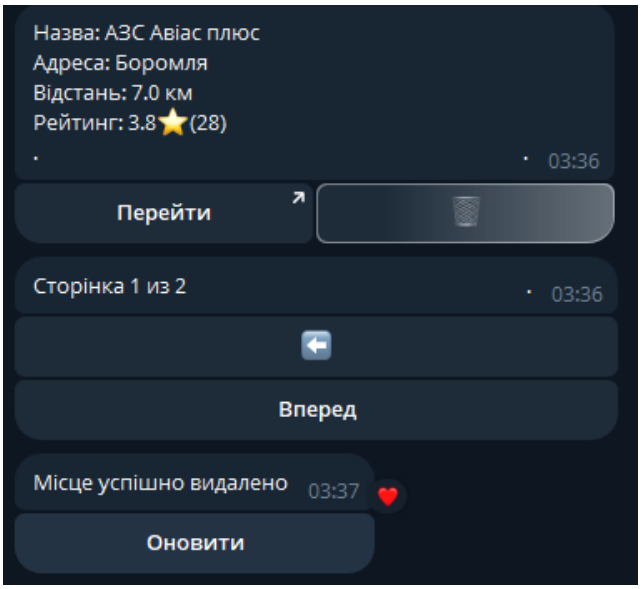

Рисунок 3.22 – Видалення з обраного

ВИСНОВКИ

У ході цієї роботи було розроблено інформаційну систему для автоматизованого пошуку об'єктів за геолокацією з використанням телеграм бота. Для реалізації цієї системи було використано технології Python, Google Maps API та базу даних SQLite для зберігання інформації.

Спершу було проведено аналіз сучасних методів та технологій, пов'язаних з автоматизованим пошуком об'єктів за геолокацією. З урахуванням зібраної інформації було обрано відповідні технології та засоби для реалізації системи.

На основі Google Maps API було розроблено телеграм бота, який надає користувачам можливість швидкого та зручного пошуку об'єктів за географічними координатами. Користувачі можуть здійснювати запити через інтерфейс телеграму та отримувати необхідну інформацію про об'єкти в обраному районі. Особливо корисною є функція пошуку бомбосховищ, яка забезпечує користувачам швидкий доступ до інформації про найближчі укриття в разі надзвичайної ситуації.

Для зберігання даних про об'єкти та історію пошуків було використано базу даних, яка забезпечує надійність та ефективність роботи системи.

Отже, результатом цієї роботи стало створення функціональної та ефективної інформаційної системи, яка дозволяє користувачам швидко та зручно знаходити потрібні об'єкти за географічними координатами з використанням телеграм бота.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Самий повний стартовий гайд по ботам Telegram (python) [Електронний ресурс], режим доступу – вільний, URL: https://habr.com/ru/articles/697052/
2. Python [Електронний ресурс], режим доступу – вільний, URL: https://www.python.org/
3. pyTelegramBotAPI [Електронний ресурс], режим доступу – вільний, URL: https://pypi.org/project/pyTelegramBotAPI/
4. Requests [Електронний ресурс], режим доступу – вільний, URL: https://pypi.org/project/requests/
5. sqlite3 - DB-API 2.0 interface for SQLite databases [Електронний ресурс], режим доступу – вільний, URL: https://docs.python.org/3/library/sqlite3.html
6. GeeksforGeeks. "Haversine Formula to find distance between two points on a sphere" [Електронний ресурс], режим доступу – вільний, URL: https://www.geeksforgeeks.org/haversine-formula-to-find-distance-between-two-points-on-a-sphere/.
7. Haversine, calculate the distance [Електронний ресурс], режим доступу – вільний, URL: https://pypi.org/project/haversine/
8. Botostore. "BotFather" [Електронний ресурс], режим доступу – вільний, URL: https://botostore.com/c/botfather/
9. Google Developers. "Places API Overview". Документація Google Maps. [Електронний ресурс], Режим доступу – вільний, URL: https://developers.google.com/maps/documentation#places
10. Google Developers. "Geocoding API Overview". Документація Google Maps. [Електронний ресурс], Режим доступу – вільний, URL: Geocoding API overview | Google for Developers
11. SQLite. Документація [Електронний ресурс], режим доступу – вільний, URL: https://www.sqlite.org/docs.html

12. Лутц, М. Programming Python. 4 видання. Себастопол, Каліфорнія: O'Reilly Media, 2010. – 1632 с.

13. Моррісон, М., Міткел, Е. Google Maps API. 3 видання. Індіанаполіс, Індіана: Sams Publishing, 2011. – 448 с.

14. Оу, М. The Definitive Guide to SQLite. 2 видання. Нью-Йорк: Apress, 2010. – 528 с.

Додаток А. Головний модуль

```
import telebot
import bd
import conf
import command_handler as CommandHandler
import callback_handler as CallbackHandler

bot = telebot.TeleBot(conf.BOT_API_TOKEN)
bd.create_db('locations.db')

@bot.message_handler(commands=['start'])
def start(message):
    CommandHandler.start(bot, message)

@bot.message_handler(content_types=['location'])
def location(message):
    CommandHandler.location(bot, message)


@bot.callback_query_handler(func=lambda call: call.data.startswith('search'))
def callback_search(call):
    CallbackHandler.callback_search(bot, call)

@bot.callback_query_handler(func=lambda call: call.data.startswith('Menu'))
def callback_menu(call):
    CallbackHandler.callback_menu(bot, call)

@bot.callback_query_handler(func=lambda call: call.data.startswith('history'))
```

```
def callback_history(call):
    CallbackHandler.callback_history(bot, call)

@bot.callback_query_handler(func=lambda call: call.data.startswith('favor'))
def callback_favor(call):
    CallbackHandler.callback_favor(bot, call)

@bot.callback_query_handler(func=lambda call: call.data.startswith('delete'))
def callback_delete(call):
    CallbackHandler.callback_delete(bot, call)

@bot.callback_query_handler(func=lambda call: call.data.startswith('locate-'))
def callback_locate(call):
    CallbackHandler.callback_locate(bot, call)

@bot.callback_query_handler(func=lambda call: call.data.startswith('Ffavorite||'))
def callback_favorite(call):
    CallbackHandler.callback_favorite(bot, call)

@bot.callback_query_handler(func=lambda call: call.data.startswith('safezone'))
def callback_safezone(call):
    CallbackHandler.callback_safezone(bot, call)

bot.polling()
```

Додаток Б. Модуль обробки команд

```
import telebot
import bd
import sqlite3

def start(bot, message):
    conn = sqlite3.connect('locations.db')
    try:
        bd.register_user(conn, message.from_user.username, user.first_name)
    except sqlite3.IntegrityError:
        pass
    finally:
        conn.close()
    bd.clear_user_data(conn, message.from_user.username)
    conn.close()
    user = message.from_user
    markup = telebot.types.ReplyKeyboardMarkup(resize_keyboard=True, one_time_keyboard=True)
    markup.add(telebot.types.KeyboardButton('Відправити місцезнаходження', request_location=True))

    bot.send_message(message.chat.id, f"Привіт, {user.first_name}! Натисніть кнопку нижче, щоб надіслати мені своє місцезнаходження.", reply_markup=markup)


def location(bot, message):
    bot.delete_message(message.chat.id, message.message_id)
```

```
    bot.delete_message(message.chat.id, message.message_id - 1)
    latitude = message.location.latitude
    longitude = message.location.longitude
    msg_id = message.message_id

    conn = sqlite3.connect('locations.db')
    bd.add_location(conn, message.from_user.username, latitude, longitude)
    conn.close()

    markup = telebot.types.InlineKeyboardMarkup()
    btn1 = telebot.types.InlineKeyboardButton('Пошук', callback_data=f'search:{msg_id}')
    btn2 = telebot.types.InlineKeyboardButton('Історія', callback_data=f"history:1:{msg_id}")
    btn3 = telebot.types.InlineKeyboardButton('Обране', callback_data=f"favor:1:{msg_id}")
    markup.row(btn1, btn2, btn3)
    bot.send_message(message.chat.id, "Ок!", reply_markup=telebot.types.ReplyKeyboardRemove())
    bot.send_message(message.chat.id,
                     "Я отримав ваше місцезнаходження. Ось що я можу для вас зробити:\n"
                     "🔍 Пошук - Оберіть місце яке бажаєте знайти.\n"
                     "📜 Історія - Перегляньте вашу історію запитів місцезнаходження.\n"
                     "⭐ Обране - Доступ до ваших улюблених місць.\n\n"
                     "Будь ласка, оберіть один з розділів нижче:",
                     reply_markup=markup)
```

Додаток В. Модуль обробки запитів

```
import telebot
import bd
import conf
from telebot import types
from haversine import haversine
import GoogleMapsApi as gma
import sqlite3
import safezone as safe

def callback_search(bot, call):
    msg_del = int(call.data.split(':')[1])
    chat_id = call.message.chat.id
    bot.delete_message(chat_id, call.message.message_id)
    for i in range(msg_del + 1, call.message.message_id):
        bot.delete_message(chat_id, i)

    msg_id = call.message.message_id
    markup = telebot.types.InlineKeyboardMarkup()
    btn1 = telebot.types.InlineKeyboardButton('Готель', callback_data=f'locate-lodging-{msg_id}')
    btn2 = telebot.types.InlineKeyboardButton('Лікарня ', callback_data=f'locate-hospital-{msg_id}')
    btn3 = telebot.types.InlineKeyboardButton('Ресторан ', callback_data=f'locate-restaurant-{msg_id}')
    btn4 = telebot.types.InlineKeyboardButton('Заправка ', callback_data=f'locate-gas_station-{msg_id}')
```

```
    btn5 = telebot.types.InlineKeyboardButton('Сховища 🚀 ',
callback_data=f'safezone:{msg_id}')
    btn6 = telebot.types.InlineKeyboardButton(text="⬅️",
callback_data=f"Menu:{msg_id}")
    markup.row(btn1, btn2)
    markup.row(btn3, btn4)
    markup.row(btn5, btn6)

    bot.send_message(call.message.chat.id, "Тепер вибери місце яке хочеш знайти:",
reply_markup=markup)

def callback_menu(bot, call):

def callback_history(bot, call):
    current_page = int(call.data.split(':')[1])
    del_msg = int(call.data.split(':')[2])
    chat_id = call.message.chat.id
    bot.delete_message(chat_id, call.message.message_id)

    for i in range(del_msg + 1, call.message.message_id):
        bot.delete_message(chat_id, i)

    msg_id = call.message.message_id

    conn = sqlite3.connect('locations.db')
    history = bd.get_history(conn, call.from_user.username)
    location = bd.get_location(conn, call.from_user.username)
    conn.close()
```

```
if location:
    latitude = location[0]
    longitude = location[1]
items_per_page = 3
start_idx = (current_page - 1) * items_per_page
end_idx = start_idx + items_per_page
current_history = history[start_idx:end_idx]
total_pages = (len(history) + items_per_page - 1) // items_per_page

if current_history:
    bot.send_message(call.message.chat.id, "Ваші історія пошуку:")
    for place in current_history:
        point1 = (latitude, longitude)
        point2 = (place[5], place[6])
        # Обчислюємо відстань
        distance = round(haversine(point1, point2), 1)
        message_text = (
            f"\nНазва: {place[2]}\n"
            f"Адреса: {place[3]}\n"
            f"Відстань: {distance} км\n"
            f"Рейтинг: {place[7]} ⭐ ({place[8]})\n"
            f".                                                                                    ."
        )
        # Створюємо Inline Keyboard
        markup = types.InlineKeyboardMarkup()
        # Створюємо кнопку для вибору місця
        button = types.InlineKeyboardButton(text=f"Перейти",
```

```
url=f"https://www.google.com/maps/dir/{latitude},{longitude}/{place[5]},{place[6]}
")
        button2 = types.InlineKeyboardButton(text="🌟", callback_data=f"Ffavorite||{place[9]}")
        markup.add(button, button2)
        bot.send_message(call.message.chat.id, message_text, reply_markup=markup, parse_mode="Markdown")

    markup = telebot.types.InlineKeyboardMarkup()
    markup.add(telebot.types.InlineKeyboardButton(text="⬅️", callback_data=f"Menu:{msg_id}"))
    if current_page > 1:
        markup.add(telebot.types.InlineKeyboardButton(text="Назад", callback_data=f"history:{current_page - 1}:{msg_id}"))
    if end_idx < len(history):
        markup.add(telebot.types.InlineKeyboardButton(text="Вперед", callback_data=f"history:{current_page + 1}:{msg_id}"))

    bot.send_message(call.message.chat.id, f"Сторінка {current_page} из {total_pages} .", reply_markup=markup)
  else:
    markup = telebot.types.InlineKeyboardMarkup()
    markup.add(telebot.types.InlineKeyboardButton(text="⬅️", callback_data=f"Menu:{msg_id}"))
    bot.send_message(call.message.chat.id, "У вас немає історії пошуку.", reply_markup=markup)

def callback_favor(bot, call):
```

```
def callback_delete(bot, call):

def callback_locate(bot, call):

ef callback_favorite(bot, call):

def callback_safezone(bot, call):
```

Додаток Г. Модуль гугл запитів

```
import requests
from haversine import haversine
import conf

def find_nearest_locate(latitude, longitude, type, num_locate=3):
    # URL для запиту до Google Places API
    url = f"https://maps.googleapis.com/maps/api/place/nearbysearch/json?location={latitude},{longitude}&radius={conf.radius}&type={type}&key={conf.google_api_key}&language=uk"
    locate_info = []
    # Виконуємо запит
    response = requests.get(url)
    data = response.json()

    # Перевірка результату пошуку
    if "results" in data and data["results"]:

        # Проходимося по першим num_restaurants результатам
        for result in data["results"][:num_locate]:
            name = result.get("name", "Unnamed")
            address = result.get("vicinity", "Unknown")
            # Отримуємо координати
            lat = result.get("geometry", {}).get("location", {}).get("lat", 0)
            lng = result.get("geometry", {}).get("location", {}).get("lng", 0)
            point1 = (latitude, longitude)
            point2 = (lat, lng)
```

```
            # Обчислюємо відстань
            distance = round(haversine(point1, point2), 1)
            # Отримуємо рейтинг і кількість оцінок
            rating = result.get("rating", 0)
            user_ratings_total = result.get("user_ratings_total", 0)
            place_id = result.get("place_id", None)
            # Додавання інформації про ресторан у список
            locate_info.append({"name": name, "address": address, "distance": distance,
"coordinate1": point1, "coordinate2": point2,
                                "rating": rating, "user_ratings_total": user_ratings_total,
"place_id": place_id})
        return locate_info
    else:
        return None

def get_shelters_coordinates(address, api_key):
    base_url = "https://maps.googleapis.com/maps/api/geocode/json"
    params = {
        "address": address,
        "key": api_key,
        "region": "UA"
    }
    response = requests.get(base_url, params=params)
    if response.status_code == 200:
        results = response.json().get("results")
        if results:
            location = results[0].get("geometry").get("location")
            return location["lat"], location["lng"]
    return None, None
```

Додаток Ґ. Модуль бази даних

```
import sqlite3


def create_db(name):
    conn = sqlite3.connect(name)
    cursor = conn.cursor()

    cursor.execute('''
        CREATE TABLE IF NOT EXISTS users (
            user_id TEXT PRIMARY KEY,
            username TEXT,
            registration_date TIMESTAMP DEFAULT CURRENT_TIMESTAMP
        )
    ''')
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS locations (
            user_id TEXT PRIMARY KEY,
            latitude REAL,
            longitude REAL,
            FOREIGN KEY (user_id) REFERENCES users(user_id)
        )
    ''')
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS history (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            user_id TEXT,
            name TEXT,
```

```
        address TEXT,
        distance REAL,
        lat REAL,
        lng REAL,
        rating REAL,
        user_ratings_total INTEGER,
        place_id TEXT,
        FOREIGN KEY (user_id) REFERENCES users(user_id)
    )
''')
cursor.execute('''
    CREATE TABLE IF NOT EXISTS favorites (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        user_id TEXT,
        name TEXT,
        address TEXT,
        distance REAL,
        lat REAL,
        lng REAL,
        rating REAL,
        user_ratings_total INTEGER,
        place_id TEXT UNIQUE,
        FOREIGN KEY (user_id) REFERENCES users(user_id)
    )
''')
cursor.execute('''
    CREATE TABLE IF NOT EXISTS shelters (
        id INTEGER PRIMARY KEY,
        address TEXT,
```

```
            building_floors INTEGER,
            demolition_percentage INTEGER,
            basement_area INTEGER,
            capacity INTEGER,
            lat REAL,
            lng REAL
        )
    ''')
    conn.close()

def clear_user_data(conn, user_id):
    cursor = conn.cursor()
    cursor.execute('DELETE FROM locations WHERE user_id = ?', (user_id,))
    conn.commit()
    cursor.close()

def add_location(conn, user_id, latitude, longitude):
    cursor = conn.cursor()
    cursor.execute('INSERT INTO locations VALUES (?, ?, ?)', (user_id, latitude,
longitude))
    conn.commit()
    cursor.close()

def get_location(conn, user_id):
    cursor = conn.cursor()
    cursor.execute('SELECT latitude, longitude FROM locations WHERE user_id = ?',
(user_id,))
    location = cursor.fetchone()
    cursor.close()
```

```
    return location

def get_closesd_shelter(conn, shalter_id):
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM shelters WHERE id = ?', (shalter_id, ))
    location = cursor.fetchone()
    cursor.close()
    return location

def get_shelters(conn):
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM shelters')
    shelters = cursor.fetchall()
    cursor.close()
    return shelters

def add_history(conn, user_id, name, address, distance, coordinate1, coordinate2, rating, user_ratings_total, place_id):
    cursor = conn.cursor()
    cursor.execute('''
        INSERT INTO history (user_id, name, address, distance, lat, lng, rating, user_ratings_total, place_id)
        VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
    ''', (user_id, name, address, distance, coordinate1, coordinate2, rating, user_ratings_total, place_id))
    conn.commit()
    cursor.close()

def copy_to_favorites(conn, user_id, place_id):
```

```
    cursor = conn.cursor()
    cursor.execute('''
        INSERT INTO favorites (user_id, name, address, distance, lat, lng, rating, user_ratings_total, place_id)
        SELECT user_id, name, address, distance, lat, lng, rating, user_ratings_total, place_id
        FROM history
        WHERE place_id = ? AND user_id = ?
    ''', (place_id, user_id))
    conn.commit()
    cursor.close()

def get_history(conn, user_id):
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM history WHERE user_id = ?', (user_id, ))
    history = cursor.fetchall()
    cursor.close()
    return history

def get_favor(conn, user_id):
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM favorites WHERE user_id = ?', (user_id, ))
    favor = cursor.fetchall()
    cursor.close()
    return favor


def get_favor_count(conn, user_id, place_id):
    cursor = conn.cursor()
```

```
    cursor.execute('SELECT count(*) FROM favorites WHERE user_id = ? and place_id = ?', (user_id, place_id))
    favor_count = cursor.fetchall()
    cursor.close()
    return favor_count

def del_favor(conn, user_id, place_id):
    cursor = conn.cursor()
    cursor.execute('DELETE FROM favorites WHERE user_id = ? AND place_id = ?', (user_id, place_id))
    conn.commit()
    cursor.close()

def register_user(conn, user_id, username):
    cursor = conn.cursor()
    cursor.execute('''
        INSERT INTO users (user_id, username) VALUES (?, ?)
        ''', (user_id, username))
    conn.commit()
    cursor.close()
```