

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ
СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТИРЕНКО

(підпис)

«___» _____ 2024 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: Телеграм-бот для пошуку найближчих укриттів у м. Київ під час
повітряної тривоги

Виконав : студент 4 курсу, групи ІО-04

(шифр групи)

Молочко Ігнатій Михайлович

(прізвище, ім'я, по батькові)

(підпис)

Керівник ст. викл. Васильєва М.Д.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Консультант (нормоконтроль) асистент Іваніщев Б.В.

(назва розділу)

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

(підпис)

Рецензент доцент каф.ІПІ, к.т.н. Вєчерковська А.С.

(посада, науковий ступінь, вчене звання, прізвище та ініціали)

(підпис)

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

(підпис)

Київ 2024

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма «Комп’ютерні системи та мережі»

спеціальності 123 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТИРЕНКО

(підпис)

«___» _____ 2024 р.

ЗАВДАННЯ

на бакалаврський дипломний проєкт студента

Молочка Ігнатія Михайловича

1. Тема проєкту Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної тривоги
керівник проєкту Васильєва Марія Давидівна, ст. викл.
затверджені наказом по університету від «27» травня 2024 р. №2112-с
2. Термін здачі студентом закінченого проєкту «07» червня 2024 р.
3. Вихідні дані до проєкту технічна документація, теоретичні дані.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)

Розділ 1. Аналіз предметної області та розгляд наявних рішень

Розділ 2. Огляд та вибір технологій для реалізації проєкту

Розділ 3. Огляд процесу розробку процесу телеграм-бота

Розділ 4. Огляд та тестування розробленого телеграм-бота

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень):

Структура системи (Структурна схема)

Функціональна схема (діаграма класів)

Алгоритм дій програми (Принципова схема)

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Іваніщев Б. В.		

7. Дата видачі завдання «16» листопада 2023 р.

Календарний план

№	Найменування етапів дипломного проєкту	Терміни виконання етапів проєкту	Примітки
1.	<i>Затвердження теми проєкту</i>	16.11.2023-13.12.2023	
2.	<i>Вивчення та аналіз завдання</i>	14.12.2023-19.01.2024	
3.	<i>Розробка архітектури та загальної структури системи</i>	20.01.2024-17.02.2024	
4.	<i>Розробка структур окремих підсистем</i>	18.02.2024-23.03.2024	
5.	<i>Програмна реалізація системи</i>	24.03.2024-21.05.2024	
6.	<i>Оформлення пояснювальної записки</i>	22.05.2024-30.05.2024	
7.	<i>Захист програмного продукту</i>	31.05.2024	
8.	<i>Передзахист</i>	07.06.2024	
9.	<i>Захист</i>	21.06.2024	

Студент-дипломник _____ Ігнатій МОЛОЧКО
(підпис)

Керівник проєкту _____ Марія ВАСИЛЬЄВА
(підпис)

АНОТАЦІЯ

Даний дипломний проєкт був присвячений розробці Telegram боту, який допоможе мешканцям Києва відстежувати стан повітряної тривоги та знаходити, при потребі, найближчі укриття до їх місцезнаходження.

Під час виконання роботи було проведено аналіз розробленого застосунку та архітектурних рішень в його реалізації. Описано функціонал задіяних інструментів розробки програмного забезпечення.

Мовою реалізації чат-боту була вибрана Python, системою управління базами даних — PostgreSQL, бібліотека для розробки бота — Aiogram, а IDE для написання коду — PyCharm.

Ключові слова: Python, Aiogram, PostgreSQL, Telegram бот.

ABSTRACT

This diploma project was dedicated to the development of a telegram bot that will help Kyiv residents monitor the state of the air alert and find, if necessary, the nearest shelters to their location.

During the execution of the work, an analysis of the developed application and architectural solutions in its implementation was carried out. The functionality of the software development tools involved is described.

The chatbot implementation language was Python, the database management system was PostgreSQL, the bot development library was Aiogram, and the IDE for writing code was PyCharm.

Keywords: Python, Aiogram, PostgreSQL, Telegram bot.

[illegible]

**ТЕХНІЧНЕ ЗАВДАННЯ
ДО ДИПЛОМНОГО ПРОЄКТУ**

на тему: «Телеграм-бот для пошуку найближчих укриттів у м. Київ під час
повітряної тривоги»

Київ – 2024

ЗМІСТ

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ	2
2 ПІДСТАВИ ДЛЯ РОЗРОБКИ	2
3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ.....	2
4 ДЖЕРЕЛА РОЗРОБКИ.....	2
5 ТЕХНІЧНІ ВИМОГИ.....	3
5.1. Вимоги до розробленого продукту.....	3
5.2. Вимоги до програмного забезпечення	3
5.3. Вимоги до апаратної частини	3
6 ЕТАПИ РОЗРОБКИ	3

					ІАЛЦ.467200.002 ТЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної тривоги Технічне завдання	Літ.	Аркуш	Аркушів
Розробив		Молочко І.М.						
Перевірив		Васильєва М.Д.					1	3
						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-04		
Н. Контр.		Іваніщев Б.В.						
Затвердив								

1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Це технічне завдання поширюється на розробку Telegram бота для пошуку найближчих укриттів у м. Київ під час повітряної тривоги.

Областю використання цього боту є жителі міста Київ, які зможуть отримувати повідомлення про початок та закінчення тривоги та знаходити найближчі укриття в разі їх оголошення.

2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання для виконання бакалаврського проєкту по освітньо-професійній програмі “Комп’ютерні системи та мережі” спеціальності 123 “Комп’ютерна інженерія”, затверджене кафедрою Обчислювальної техніки Національного технічного Університету України “Київський Політехнічний інститут імені Ігоря Сікорського”.

3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка Telegram бота, який допоможе мешканцям міста Київ отримувати повідомлення про зміну статусу повітряної тривоги в реальному часі та швидко знаходити найближчі укриття відносно відправленої мітки на карті.

4 ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки даного дипломного проєкту є офіційні документації, публікації та статті в мережі Інтернет на тему розробки ботів, науково-технічна література.

					ІАЛЦ.467200.002 ТЗ	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

5 ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до розробленого продукту

- Інтуїтивно-зрозумілий інтерфейс системи.
- Оперативно повідомляти користувачам про статус тривоги.
- Швидко допомогати користувачам знайти найближче укриття.
- Бути безкоштовним для користувачів та розробників.

5.2. Вимоги до програмного забезпечення

- Операційна система Windows, MacOS або будь-який дистрибутив Linux.
- IDE PyCharm з встановленим компілятором мови Python.
- Встановлена PostgreSQL база даних з PostGIS розширенням.
- Встановлений месенджер Telegram (або його веб-версія).

5.3. Вимоги до апаратної частини

Будь-який електронний девайс з можливістю зайти на Web сторінку або скачати програму та стабільний інтернет зв'язок.

6 ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	16.11.2023-13.12.2023
Вивчення та аналіз завдання	14.12.2023-19.01.2024
Розробка архітектури та загальної структури системи	20.01.2024-17.02.2024
Розробка структур окремих частин системи	18.02.2024-23.03.2024
Програмна реалізація системи	24.03.2024-09.05.2024
Виправлення помилок	10.05.2024-21.05.2024
Оформлення пояснювальної записки	22.05.2024-30.06.2024

					ІАЛЦ.467200.002 ТЗ	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

**ПОЯСНЮВАЛЬНА ЗАПИСКА
ДО ДИПЛОМНОГО ПРОЄКТУ**

на тему: «Телеграм-бот для пошуку найближчих укриттів у м. Київ під час
повітряної тривоги»

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....	4
ВСТУП.....	5
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА РОЗГЛЯД НАЯВНИХ РІШЕНЬ.....	7
1.1 Telegram.....	7
1.1.1 Основний функціонал Telegram	8
1.1.2 Додаткові можливості Telegram	9
1.1.3 Переваги та недоліки в порівнянні з іншими месенджерами.....	10
1.2 Telegram боти [10].....	12
1.2.1 Що таке Telegram бот і як він може допомогти.....	12
1.2.2 Переваги використання бота Telegram	12
1.2.3 Як створити бота Telegram для особистого використання.....	12
1.2.4 Які є різні типи ботів Telegram	13
1.3 Огляд існуючих рішень	14
1.3.1 Телеграм-бот «Укриття!» [11]	14
1.3.2 Телеграм-бот «Київщина Безпечна»	15
Висновок до розділу 1.....	16
РОЗДІЛ 2. ОГЛЯД ТА ВИБІР ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОЄКТУ	17
2.1 Огляд мов програмування для реалізації проєкту	17
2.1.1 Python.....	17
2.1.2 JavaScript (Node.js)	18
2.1.3 Go	18

					ІАЛЦ.467200.003 ПЗ			
Зм.	Арк.	№ докум.	Підпис	Дата	Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної тривоги Пояснювальна записка			
Розробив		Молочко І.М.						
Перевірив		Васильєва М.Д.						
Н. Контр.		Іваніщев Б.В.			Літ. Аркуш Аркушів КПІ ім. Ігоря Сікорського, ФІОТ, ІО-04			
Затвердив								

2.2 Python бібліотеки для розробки Telegram ботів.....	19
2.2.1 Aiogram [19].....	19
2.2.2 python-telegram-bot [21]	20
2.2.3 pyTelegramBotAPI [22].....	20
2.3 Бази даних	21
2.3.1 Відмінності SQL та NoSQL баз даних [25].....	22
2.3.2 Реляційні СКБД.....	23
2.4. Джерела даних	25
Висновок до розділу 2.....	26
РОЗДІЛ 3. ОГЛЯД ПРОЦЕСУ РОЗРОБКУ ПРОЦЕСУ ТЕЛЕГРАМ-БОТА	
.....	27
3.1 Реєстрація бота та зазначення деяких параметрів	27
3.2 Отримання токена для використання API тривоги.....	29
3.3 Розробка телеграм бота	30
3.3.1 Створення змін середовища з конфігураційним файлом.....	31
3.3.2 Створення, налаштування та робота з базою даних	32
3.3.3 Реалізація отримання та сповіщення про статус тривоги	37
3.3.4 Обробники відправлених телеграм боту повідомлень	38
Висновок до розділу 3.....	44
РОЗДІЛ 4. ОГЛЯД ТА ТЕСТУВАННЯ РОЗРОБЛЕНОГО ТЕЛЕГРАМ-	
БОТА.....	45
4.1 Тестування ініціалізації бота	45
4.2 Тестування базових команд /help та /start.....	46
4.3 Тестування команди /alert.....	48
4.4 Тестування команди /nearest	50
4.5 Тестування отримання сповіщення про зміну статусу тривоги	54
Висновок до розділу 4.....	56

ВИСНОВОК.....57

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....58

ДОДАТОК 1

ДОДАТОК 2

ДОДАТОК 3

ДОДАТОК 4

ПЕРЕЛІК СКОРОЧЕНЬ

API	(Application Programming Interface)	Прикладний програмний інтерфейс
IDE	(Integrated Development Environment)	Інтегроване середовище розробки
SQL	(Structured query language)	Мова структурованих запитів
RDB	(Relational database)	Реляційна база даних
СУБД	(Система управління базами даних)	
БД	(База даних)	

					ІАЛЦ.467200.003 ПЗ	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Сучасна ситуація в Україні, зокрема в місті Київ, вимагає швидких і ефективних рішень для забезпечення безпеки населення під час повітряних тривог. З розвитком інформаційних технологій з'явилась можливість створення цифрових інструментів, які можуть просто та оперативно надавати необхідну інформацію та допомагати в критичних ситуаціях мешканцям, навіть які не дуже розуміються на сучасних технологіях. Одним з таких інструментів є Telegram-бот, який може використовуватися для пошуку найближчих укриттів під час повітряної тривоги буквально в декілька натискань.

Забезпечення населення своєчасною та точною інформацією про початок тривоги та розташування найближчих укриттів є критично важливим для збереження життя та здоров'я громадян. Застосування Telegram-ботів для цих цілей дозволяє автоматизувати процес пошуку, зменшити навички та скоротити час, необхідний для отримання потрібної інформації. Враховуючи широке розповсюдження месенджера Telegram в Україні, такий бот стане доступним для широкого кола користувачів, забезпечуючи ефективну комунікацію в надзвичайних ситуаціях.

Тема є надзвичайно актуальною, оскільки забезпечення безпеки населення під час надзвичайних ситуацій є одним з пріоритетних завдань держави. Створення ефективних та доступних інструментів для пошуку укриттів підвищить рівень готовності громадян до таких ситуацій і допоможе знизити ризики для їхнього життя та здоров'я.

Для розробки Telegram-бота використовуються такі вихідні дані: геолокаційні дані про розташування укриттів у місті Київ, інформація про їх тип, власника, форму власності та наявність пристосування для доступу осіб з інвалідністю та інших маломобільних груп населення (пандусу), дані про мережу повітряних тривог.

					ІАЛЦ.467200.003 ПЗ	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

Основними методами дослідження є аналіз існуючих систем оповіщення, вивчення геолокаційних технологій та алгоритмів маршрутизації, а також програмування ботів для месенджера Telegram з використанням відповідних API.

Метою дипломної роботи є розробка функціонального Telegram-бота, який дозволить отримувати повідомлення про зміну статусу повітряної тривоги в реальному часі та швидко знаходити найближчі укриття відносно відправленої мітки на карті у місті Київ.

Об'єктом дослідження є система оповіщення та укриття населення під час надзвичайних ситуацій у місті Київ. Предметом дослідження є програмне забезпечення для автоматизації процесу пошуку найближчих укриттів за допомогою Telegram-бота. Практичне значення отриманих результатів полягає у створенні інструменту, який може бути використаний широким колом користувачів для забезпечення своєї безпеки під час повітряних тривог, а також може бути адаптований для використання в інших містах України та за її межами.

Таким чином, розробка Telegram-бота для пошуку найближчих укриттів у місті Київ є важливим та актуальним завданням, що сприятиме підвищенню рівня безпеки та обізнаності населення в умовах надзвичайних ситуацій.

					ІАЛЦ.467200.003 ПЗ	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА РОЗГЛЯД НАЯВНИХ РІШЕНЬ

1.1 Telegram

Telegram [1] — це один за найпопулярніших хмарних кросплатформних сервісів миттєвих повідомлень (ІМ) із зашифрованим кодуванням. Він дозволяє користувачам обмінюватися повідомленнями, ділитися медіа та файлами, проводити приватні та групові голосові чи відеодзвінки, а також організовувати публічні трансляції. Сервіс доступний на більшості сучасних платформ і підтримує наскрізне шифрування голосових і відеодзвінків, а також у спеціальних чатах. Також Telegram має функції соціальної мережі.

Сервіс надає відкритий API для створення спеціальних ботів, які можуть виконувати різні завдання, інтегрувати інші сервіси в чати Telegram або діяти як міні-додатки.

Одною з основних переваг Telegram є його конфіденційність, яка досягається шляхом використання дуже ефективного протоколу MTProto[2], який теж був розроблений командою розробників Telegram (під керівництвом одного з засновників — Миколою Дуровими).

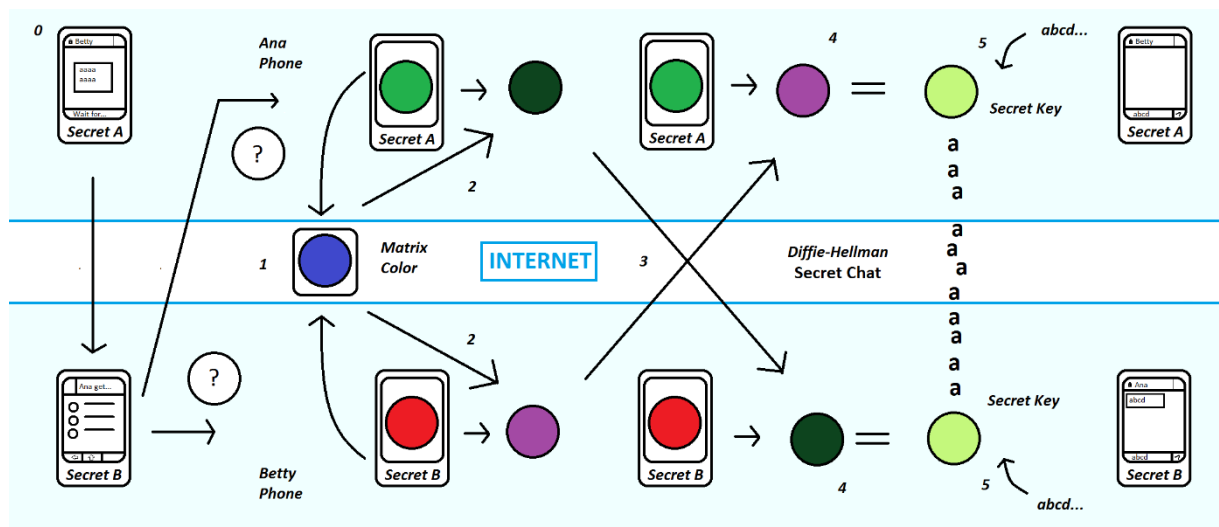


Рис. 1.1 – Спрощена ілюстрація схеми шифрування MTProto [1]

1.1.1 Основний функціонал Telegram

Telegram дуже розвинутий месенджер, він має багато різноманітних функцій, як і звичайних для майже будь-яких месенджерів, так і унікальних. Далі будуть представлені деякі з основних функцій та їх короткий опис:

1) Обмін повідомленнями:

Telegram надає можливість швидко і безпечно обмінюватись текстовими, аудіо та відео повідомленнями, а також надсилати фото, відео, аудіо, документи та інші файли. Ще треба звернути увагу, що будь яке повідомлення, щоб в нього не входило, буде надіслано одержувачу в той же час, як буде відправлено.

2) Групові чати та канали:

Також в Telegram є така важлива складова суспільства, як об'єднання в групи з друзями, колегами, родичами або взагалі незнайомими людьми. Telegram групи підтримують до 200 000 учасників [3], що дозволяє створювати великі спільноти для обговорень. Також в Telegram є канали, які можуть мати необмежену кількість підписників і використовуються для трансляції повідомлень на велику аудиторію.

3) Обмін файлами:

Telegram підтримує надсилання та отримування різноманітних файлів, включаючи фотографії, відео, аудіозаписи та документи, єдине обмеження полягає в максимальному обсязі в 1.5 ГБ [4], що є значно більшим лімітом, ніж у багатьох інших месенджерів.

4) Телефонні та відео дзвінки [5]:

Ще Telegram пропонує користувачам можливість здійснювати як телефонні, так і відеодзвінки. Ці функції, як і всі інші, розроблені для забезпечення високої якості зв'язку та конфіденційності.

					ІАЛЦ.467200.003 ПЗ	Арк.
						8
Зм.	Арк.	№ докум.	Підпис	Дата		

1.1.2 Додаткові можливості Telegram

Як вже згадувалось, окрім базових функцій месенджера, Telegram має унікальні можливості, які, здебільшого, більше ні в одному месенджері-конкуренті не зустрічаються. Далі я поверхнево згадаю і трохи розкажу про ці можливості.

1) Ігри [6]:

Telegram також є невеликою ігровою платформою. Він надає можливість створювати HTML5-ігри та грати в них всередині месенджера. Ігри можуть бути як одиночні так і в режимі змагання з іншими.

2) Підтримка локалізації:

Також мало того, що Telegram перекладений багатьма, так він ще й підтримує можливість перекладати іншомовні повідомлення з нещодавніх пір. Ця функція окремо включається в налаштуваннях та дуже корисна, якщо вам доводиться спілкуватись з багатомовними командами.

3) Боти [7]:

Одною з унікальних та головних можливостей Telegram є можливість створення ботів. Боти можуть виконувати різні завдання, від простих відповідей на запити до складних задач, таких як опитування, бронювання або інше. Ботів можна інтегрувати з майже всіми сервісами.

4) Секретні чати [8]:

Ще Telegram надає можливість створювати секретні чати. Повідомлення в таких чатах мають функцію самознищення (через проміжок часу від 1 секунди до 1 тижня). Повідомлення не можна пересилати, а зображення з короткими таймерами сильно розмиті, можуть бути переглянуті лише під час утримання пальця на фото після чого теж зникають.

					ІАЛЦ.467200.003 ПЗ	Арк.
						9
Зм.	Арк.	№ докум.	Підпис	Дата		

1.1.3 Переваги та недоліки в порівнянні з іншими месенджерами

Переваги Telegram [9]:

1. Розширені функції обміну повідомленнями

Telegram — має більше функцій в порівнянні з аналогами, такими функціями як голосові/відеодзвінки, групові чати, стікери, секретні чати та чат-боти. Можливість організації повідомлень у файли, групи до 200 000 учасників.

2. Аспект соціальних мереж

Також останнім часом в Telegram почали з'являться нові функції, через які його вже можна називати повноцінною соціальною мережею, наприклад не так давно з'явилась можливість публікувати свої історії.

3. Необмежене хмарне сховище

Ще одна унікальна перевага Telegram порівняно з іншими програмами для обміну миттєвими повідомленнями — це необмежене хмарне сховище для повідомлень та медіафайлів, що робить його комунікаційною платформою і платформою для обміну файлами. Доступ до файлів можна отримати з будь-якого пристрою, а вбудована система керування і пошукова система полегшують організацію та доступність файлів.

4. Безкоштовна, із шифруванням і відкритим кодом

Сервіс Telegram є абсолютно безкоштовним для всіх, але також пропонує платні підписки для розширення функціоналу, наприклад, для покращення групових чатів. Усі повідомлення і дзвінки зашифровані, забезпечуючи приватність, а елементи управління конфіденційністю дозволяють контролювати, якою інформацією ділиться користувач. Також важливо, що Telegram є застосунком з відкритим кодом, що означає доступність вихідного коду для перевірки та переконання у безпеці.

					ІАЛЦ.467200.003 ПЗ	Арк.
						10
Зм.	Арк.	№ докум.	Підпис	Дата		

Недоліки Telegram:

1. Усе ще має обмежену глобальну базу користувачів

В Telegram зареєстровано 700 мільйонів користувачів, але база ще менша порівняно з іншими послугами, такими як Facebook Messenger. Viber популярний в Південно-Східній Азії. Маленька база може обмежувати доступ до контактів та розробку сторонніх програм.

2. Недоступність деяких функцій на різних платформах

Так, додаток доступний на багатьох платформах, але функціональність може відрізнятись. Версії для Android та iOS мають найбільший функціонал, в той час як програма для macOS є спрощеною і не підтримує деякі функції, такі як розширені голосові виклики з відеодзвінками. Це також стосується версій для Windows і Linux. Існують затримки та розбіжності між версіями для комп'ютерів і мобільних пристроїв, які впливають на синхронізацію повідомлень та файлів.

3. Значні проблеми щодо конфіденційності даних Telegram

Telegram збирає особисті дані, включаючи номери телефонів, адреси електронної пошти та імена, що може бути розглянуто як недолік. Повідомлення зберігаються на серверах, а якість шифрування не перевірена. Влада окремих країн може вимагати доступ до даних користувачів і організація їх передасть.

4. Занепокоєння щодо доцільності бізнесу

Telegram - безкоштовна месенджерська служба без реклами. Головне джерело доходу - підписки. Однак, незріла бізнес-модель та великі витрати можуть підірвати стійкість платформи. Нещодавнє введення реклами може втратити користувачів, оскільки безрекламна пропозиція - основна перевага Telegram.

					ІАЛЦ.467200.003 ПЗ	Арк.
						11
Зм.	Арк.	№ докум.	Підпис	Дата		

1.2 Telegram боти [10]

1.2.1 Що таке Telegram бот і як він може допомогти

Telegram бот — це чат-бот для взаємодії з користувачами на платформі Telegram. Це програма, що імітує людське спілкування і надає автоматичні відповіді на запити. Telegram боти можуть автоматизувати завдання, надавати клієнтську підтримку, відправляти повідомлення та надавати інформацію.

Вони допомагають бізнесам, надаючи підтримку, відповідаючи на запитання, відправляючи підтвердження замовлень і оновлення продуктів, а також надаючи інформацію про продукти і ціни. Загалом, Telegram боти допомагають заощадити час і гроші, покращуючи обслуговування клієнтів.

1.2.2 Переваги використання бота Telegram

Використання Telegram ботів стає популярним завдяки його перевагам. Він може автоматизувати обслуговування клієнтів, швидко відповідаючи на запити, надаючи інформацію та обробляючи замовлення, що звільняє ресурси. Telegram боти також дозволяють створювати персоналізований досвід для користувачів, що підвищує їхню лояльність і задоволеність. Крім того, боти допомагають збирати відгуки для покращення. Таким чином, Telegram боти оптимізують діяльність підприємств і підвищують рівень задоволеності клієнтів.

1.2.3 Як створити бота Telegram для особистого використання

Боти Telegram — це автоматизовані програми, які можуть взаємодіяти з користувачами та надавати їм інформацію, послуги та інші функції. Їх можна використовувати, щоб відповідати на запити клієнтів, надавати інформацію про продукт і навіть автоматизувати певні завдання.

					ІАЛЦ.467200.003 ПЗ	Арк.
						12
Зм.	Арк.	№ докум.	Підпис	Дата		

Першим кроком у створенні бота Telegram є реєстрація облікового запису на платформі Telegram через BotFather. Це надасть вам ключ API та маркер, які використовуватимуться для автентифікації вашого бота. Після цього ви зможете створити нового бота з потрібними параметрами.

Коли ваш бот буде створено, ви можете почати його налаштовувати. Ви можете додавати команди, налаштовувати автоматичні відповіді та навіть створювати власні меню. Ви також можете додати до свого бота плагіни, щоб розширити його функціональність. Наприклад, ви можете додати плагін, який дозволяє користувачам шукати продукти чи послуги на вашому веб-сайті.

1.2.4 Які є різні типи ботів Telegram

Боти Telegram — це автоматизовані програми, які можуть виконувати різноманітні завдання, від надання інформації до допомоги користувачам керувати своїми обліковими записами. Доступно кілька типів ботів Telegram, кожен зі своїми унікальними функціями та можливостями.

1. Вбудовані боти: ці боти запускаються, коли користувач вводить команду у вікні чату. Вони можуть надавати інформацію, наприклад прогноз погоди, або виконувати завдання, наприклад установлювати нагадування.

2. Автономні боти: ці боти є незалежними програмами, які можна використовувати для виконання різноманітних завдань, таких як ігри, керування обліковими записами або надання інформації.

4. Магазины ботів: ці боти розроблені, щоб допомогти користувачам знаходити та купувати товари в онлайн-магазинах. Їх можна використовувати для пошуку товарів, порівняння цін і здійснення покупок.

5. API ботів: ці боти розроблені, щоб дозволити розробникам створювати власних ботів для власних програм. Їх можна використовувати для створення ботів, які можуть взаємодіяти з іншими службами.

					ІАЛЦ.467200.003 ПЗ	Арк.
						13
Зм.	Арк.	№ докум.	Підпис	Дата		

1.3 Огляд існуючих рішень

1.3.1 Телеграм-бот «Укриття!» [11]

Telegram бот «Укриття!» був розроблений ІТ компанією Stfalcon в середині 2023 року. Він являє собою хороше рішення, але, на мій розсуд, з деякими недоліками.

По-перше, цей Telegram бот надає тільки можливість знаходити найближчі укриття, ніяких сповіщень про тривоги ви в ньому не отримаєте. Вам доведеться про них дізнаватись в інших джерелах, що може виявитись вже складним для людей, які погано розуміються в техніці.

По-друге, Stfalcon являє собою дуже велику ІТ компанію з багатьма різноманітними проєктами, через що швидкість оновлення даних або виправлення якихось технічних помилок може бути з невеликим затриманням. Тим паче їх розробка є закритим продуктом, через що приватні розробники або районі адміністрації не можуть в особистому порядку зробити своїх ботів і особисто їх підтримувати.

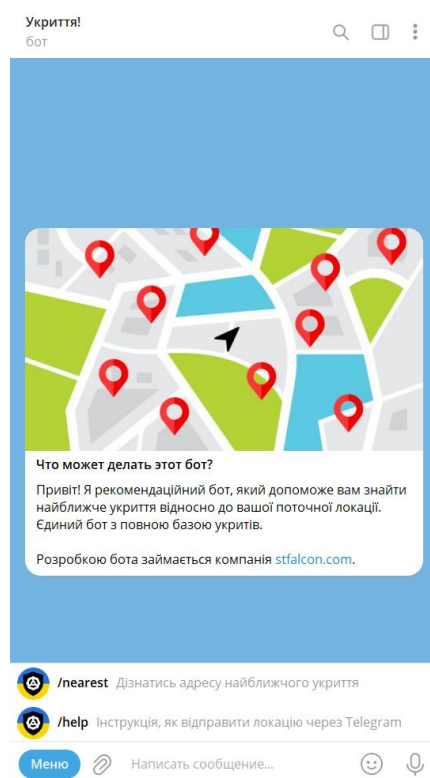


Рис. 1.2 – Інтерфейс Telegram боту «Укриття!»

					ІАЛЦ.467200.003 ПЗ	Арк.
						14
Зм.	Арк.	№ докум.	Підпис	Дата		

1.3.2 Телеграм-бот «Київщина Безпечна»

Ще одним аналогом моєї розробки є Telegram бот «Київщина Безпечна». Його розробник невідомий, але це не робить бота поганим, на відмінність від деяких моментів.

По-перше, цей бот теж не має відкритого коду, через що вільні розробники не мають можливості його вдосконалення під себе, а користувачі не мають впевненості, що їх дані будуть знаходитись в безпеці, що, як на мене, дуже великий мінус.

По-друге, цей бот знаходить укриття, але не обов'язково найближчі до вас, через що він може порадити вам добиратись до укриття в іншому кінці міста, або навіть їхати в іншу точку країни за багато кілометрів. Це ніби і не є великою проблемою, бо якщо в базі бота є сховище біля вас, то воно буде вам рекомендовано, але якщо такого не виявиться, недосвідчений користувач може в небезпечний час почати добиратись декілька кілометрів до укриття, що може стати дуже небезпечним.



Рис. 1.3 – Інтерфейс Telegram боту «Київщина Безпечна»

					ІАЛЦ.467200.003 ПЗ	Арк.
						15
Зм.	Арк.	№ докум.	Підпис	Дата		

Висновок до розділу 1

У цьому розділі було детально розглянуто месенджер Telegram. Весь його основний та унікальний функціонал, також було розглянуто переваги та недоліки в порівнянні з іншими аналогічними месенджерами. Також було коротко розібрано можливості Telegram чат-ботів, та вони такі корисні в сучасному житті. Після чого було розглянуто декілька аналогічних до мого Telegram ботів та розібрано, в чому полягають їх недоліки, через що варто займатись розробкою мого проєкту.

					ІАЛЦ.467200.003 ПЗ	Арк.
						16
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ОГЛЯД ТА ВИБІР ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ПРОЄКТУ

2.1 Огляд мов програмування для реалізації проєкту

Теоретично розробляти Telegram ботів можна використовуючи будь-яку мову програмування, бо Telegram надає відкрите API, яке дозволяє взаємодіяти з його платформою через HTTP-запити, а більшість мов мають інструменти для роботи з HTTP.

Але треба зауважити, що написання ботів виключно через HTTP-запити витрачає доволі багато часу та сил, тому для більшості мов існують офіційні та неофіційні бібліотеки [12], які спрощують взаємодію з Telegram API.

Одними з найпопулярніших мов програмування для написання Telegram ботів є такі мови, як: Python, JavaScript (node.js), Java, C#, Go, PHP. Далі ми розберемо деякі з цих мов, їх можливості, і оберемо більш підходящу для нашої задачі.

2.1.1 Python

Python [13] — це мова програмування високого рівня загального призначення. Його філософія дизайну наголошує на зручності читання коду з використанням значних відступів.

Python динамічно типізується та збирає сміття [14], підтримує кілька парадигм програмування, такі як: структуроване, об'єктно-орієнтоване та функціональне. Його часто називають мовою «з батареjkами», завдяки повній стандартній бібліотеці.

Python незмінно вважається однією з найпопулярніших мов програмування та широко використовується в різних напрямках, включаючи розробку Telegram ботів.

					ІАЛЦ.467200.003 ПЗ	Арк.
						17
Зм.	Арк.	№ докум.	Підпис	Дата		

2.1.2 JavaScript (Node.js)

JavaScript [15], часто відомий як JS, є мовою програмування і ключовою технологією для веб-розробки поряд з HTML та CSS. 99% веб-сайтів використовують JavaScript на клієнтській стороні для реалізації поведінки веб-сторінок. Веб-браузери мають спеціальний механізм для виконання JavaScript-коду на клієнті, який також застосовується на деяких серверах і в різних програмах. Найпопулярнішою платформою для виконання JavaScript поза браузером є Node.js.

Node.js [17] — це кросплатформне середовище виконання JavaScript із відкритим кодом, яке може працювати в Windows, Linux, Unix, macOS тощо. Node.js працює на механізмі JavaScript V8 і виконує код JavaScript поза веб-браузером.

Node.js дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та сценаріїв на стороні сервера. Можливість запуску коду JavaScript на сервері часто використовується для створення динамічного вмісту веб-сторінки перед тим, як сторінка надсилається у веб-браузер користувача. Node.js представляє парадигму «JavaScript всюди», об'єднуючи розробку веб-додатків навколо однієї мови програмування, на відміну від використання різних мов для програмування на стороні сервера чи клієнта.

Отже, JavaScript на платформі Node.js, дуже добре підходить та часто використовується для розробки серверної сторони застосунків, включаючи розробку різноманітних Telegram ботів.

2.1.3 Go

Go [18] — це високорівнева мова програмування з статичною типізацією та компіляцією, створена в Google Робертом Гріземером, Робом Пайком і Кеном Томпсоном. Вона має синтаксис, подібний до C, але

					ІАЛЦ.467200.003 ПЗ	Арк.
						18
Зм.	Арк.	№ докум.	Підпис	Дата		

включає безпеку пам'яті, збір сміття, структурну типізацію та паралельність у стилі CSP. Часто її називають Golang через колишнє доменне ім'я `golang.org`, хоча справжня назва — Go.

Існують дві основні реалізації:

- Інструментальний ланцюжок компілятора «gc» від Google, призначений для різних операційних систем і WebAssembly.
- `gofrontend`, інтерфейс для інших компіляторів з бібліотекою `libgo`, який у комбінації з GCC називається `gccgo`, а з LLVM — `gollvm`.

Також є сторонній компілятор GopherJS, який компілює Go у JavaScript для веб-розробки.

Go відома своєю простотою, ефективністю та підтримкою паралельного програмування. Ці особливості роблять її відмінним і частим вибором для розробки Telegram ботів.

2.2 Python бібліотеки для розробки Telegram ботів

В Python доступно доволі багато різноманітних бібліотек, які спрощують створення Telegram ботів. Якісь з них трохи частіше використовують, а якісь рідше, якісь ефективніші в одному, а інші в іншому. Далі я розберу деякі з цих бібліотек з їх перевагами та недоліками.

2.2.1 Aiogram [19]

Aiogram - це бібліотека для Python, яка дозволяє розробникам створювати ботів для Telegram. Основні функції aiogram включають у себе роботу з повідомленнями, клавіатурами, командами, а також інші інструменти для взаємодії з API Telegram. Основним з плюсів цієї бібліотеки є її асинхронність. Як можна зрозуміти вже по її назві, вона була створена виключно для реалізації асинхронного коду, а якщо говорити точніше, то

					ІАЛЦ.467200.003 ПЗ	Арк.
						19
Зм.	Арк.	№ докум.	Підпис	Дата		

вона базується на асинхронній, вбудованій в Python, бібліотеці `asyncio` [20]. Асинхронність, в свою чергу, робить код більш ефективним під час обробки великої кількості запитів та подій. Це особливо корисно для створення ботів з високим трафіком або вимогливих до продуктивності. Але також треба зауважити, що для розробки асинхронних програм треба володіти відповідними навичками та досвідом. З мінусів цієї бібліотеки можна виділити те, що її документація не настільки гарна, як сам фреймворк, деякі функції, які використовуються рідше, можуть мати доволі скудний опис та зовсім не мати прикладів, але якщо ви готові витратити сили на оволодіння цією бібліотекою, то вона стане дуже потужним інструментом у ваших руках.

2.2.2 python-telegram-bot [21]

Python-telegram-bot – це найпопулярніша бібліотека для Telegram ботів, вона є дуже потужним інструментом для їх розробки на мові програмування Python. Ця бібліотека дозволяє легко створювати і налаштовувати ботів з різним функціоналом, від простих авто-відповідачів до складних інтерактивних додатків. Цей фреймворк відомий своєю простотою та тим, що в ньому повністю підтримуються всі функції Telegram Bot API, а також є дуже гарна документація. Також в цій бібліотеці віднедавна реалізована можливість писати асинхронний код, але, на відміну від Aiogram, ця функція не була імплементована спочатку, а додана згодом.

2.2.3 pyTelegramBotAPI [22]

pyTelegramBotAPI – це ще одна популярна Python бібліотека для створення Telegram ботів, вона надає зручний і простий спосіб їх створення. Ця бібліотека дозволяє вам створювати ботів, які можуть відповідати на повідомлення, обробляти команди, отримувати оновлення з чатів і каналів,

					ІАЛЦ.467200.003 ПЗ	Арк.
						20
Зм.	Арк.	№ докум.	Підпис	Дата		

надсилати повідомлення та медіафайли, працювати з клавіатурою і багато іншого. Цей інструмент відомий своєю простотою, але також в ньому реалізовані далеко не всі функції Telegram Bot API. Через це цей фреймворк дуже добре підходить для початки створення своїх перших Telegram ботів та дізнавання основ цього напрямку програмування. До речі, ознайомлення з Telegram Bot API та створення своїх перших ботів я починав саме з цієї бібліотеки.

2.3 Базы даних

В обчислювальній техніці база даних (БД) [23] — це організований набір даних або тип сховища даних на основі використання системи керування базами даних (СКБД), програмного забезпечення, яке взаємодіє з кінцевими користувачами, програмами та самою базою даних для збору та аналізу даних. СКБД додатково включає в себе основні засоби, надані для адміністрування бази даних. Загальна сума бази даних, СКБД і пов'язаних програм може називатися системою баз даних. Часто термін «база даних» також використовується вільно для позначення будь-якої СКБД, системи баз даних або програми, пов'язаної з базою даних.

Невеликі бази даних можна зберігати у файловій системі, тоді як великі бази даних розміщуються на комп'ютерних кластерах або хмарних сховищах. Дизайн баз даних охоплює формальні методи та практичні міркування, включаючи моделювання даних, ефективне представлення та зберігання даних, мови запитів, безпеку та конфіденційність конфіденційних даних, а також питання розподілених обчислень, включаючи підтримку одночасного доступу та стійкість до відмов.

Інформатики можуть класифікувати системи керування базами даних відповідно до моделей баз даних, які вони підтримують. Реляційні бази даних стали домінуючими у 1980-х роках. Ці дані моделюють у вигляді

					ІАЛЦ.467200.003 ПЗ	Арк.
						21
Зм.	Арк.	№ докум.	Підпис	Дата		

рядків і стовпців у серії таблиць, і переважна більшість використовує SQL [24] для запису та запиту даних. У 2000-х роках стали популярними нереляційні бази даних, які разом називаються NoSQL, оскільки вони використовують різні мови запитів.

2.3.1 Відмінності SQL та NoSQL баз даних [25]

Для порівняння SQL та NoSQL використаємо такі параметри, як: модель даних, мова запитів, масштабованість, транзакції та гнучкість схеми:

- SQL використовує структуровану модель даних, відому як реляційна модель. Дані зберігаються у вигляді таблиць, де кожен рядок представляє окремий запис, а кожний стовпець - певну властивість. NoSQL, в свою чергу, є більш гнучкою і може використовувати різні моделі даних, такі як документи, ключ-значення, стовпцеві та графові бази даних. Це дозволяє краще відповідати різним потребам додатків.
- SQL бази даних, як не важко здогадатись, використовують мову запитів SQL (Structured Query Language), яка стандартизована для взаємодії з реляційними базами даних, що є великим плюсом, бо при зміні одної реляційної СКБД на іншу, методи взаємодії з самою БД якщо і зміняться, то не сильно. В NoSQL же навпаки, кожен тип NoSQL бази даних може використовувати свою власну мову запитів або API для доступу до даних. Наприклад, MongoDB використовує мову запитів, що базується на JavaScript.
- SQL БД Традиційно мають обмежену масштабованість, особливо у режимі горизонтального масштабування, а NoSQL, зазвичай, добре, особливо у великих розподілених системах, так як багато NoSQL БД побудовані з урахуванням горизонтального масштабування.

					ІАЛЦ.467200.003 ПЗ	Арк.
						22
Зм.	Арк.	№ докум.	Підпис	Дата		

- SQL бази даних, на відміну від NoSQL, підтримують транзакції ACID (атомарність, консистентність, ізолюваність, довіреність). Деякі NoSQL бази даних не підтримують повністю ACID транзакції через підвищену відмовостійкість або швидкодію. Деякі з них пропонують транзакційні можливості, але не завжди повністю дотримуються всіх принципів ACID.
- Бази даних з SQL керуванням, мають жорстку схему, де структура даних повинна бути заздалегідь визначена та відображена у схемі бази даних. А NoSQL, зазвичай, дозволяє гнучкішу схему, де дані можуть мати різну структуру без необхідності заздалегідь визначати схему. Це особливо корисно в розробці, де вимоги до структури даних можуть змінюватися.

2.3.2 Реляційні СКБД

Реляційна база даних (RDB) — це база даних, заснована на реляційній моделі даних, запропонованій Е. Ф. Коддом у 1970 році. Система керування базами даних, яка використовується для підтримки реляційних баз даних, є системою керування реляційною базою даних (RDBMS). Багато систем реляційних баз даних оснащені опцією використання SQL (Structured Query Language) для запитів і оновлення бази даних. До реляційних СКБД належать такі, як MySQL, PostgreSQL, Microsoft SQL Server, Oracle Database та SQLite, далі ми розберемо деякі з них:

- MySQL [26] - це відкрита система управління реляційною базою даних, що використовує мову SQL для обробки даних. Вона організовує дані у таблиці та дозволяє їх взаємодію через SQL. MySQL безкоштовна за умов GNU GPL та має комерційні опції. Початково належала MySQL AB, а потім була придбана Oracle. MySQL широко використовується в програмах та веб-сайтах, що

					ІАЛЦ.467200.003 ПЗ	Арк.
						23
Зм.	Арк.	№ докум.	Підпис	Дата		

потребують роботи з базами даних, у тому числі у популярних веб-проектах як Facebook, Twitter, і YouTube.

- PostgreSQL, або Postgres [27], безкоштовна система керування реляційними базами даних з відкритим кодом, що підтримує SQL. Має широкі можливості, включаючи ACID, підтримується на різних ОС та використовується для різних завдань, від окремих машин до веб-сервісів. Група розвитку фокусується на ядрі PostgreSQL, але існує велика спільнота розробників, що надає додаткові функції, наприклад плагін PostGIS. Виник унаслідок бази даних Ingres і був перейменований у 1996 році, зберігши псевдонім Postgres.
- SQLite [28] — це не просто програма, а скоріше набір інструментів для управління базами даних, що написані мовою програмування C. Він не функціонує як окремий додаток, а вбудовується розробниками безпосередньо у свої програми. Таким чином, він відноситься до категорії вбудованих баз даних. SQLite став досить популярним механізмом для зберігання та управління даними через своє використання в найкращих веб-браузерах, операційних системах, мобільних телефонах та інших вбудованих системах. Існує багато прив'язок до SQLite для різних мов програмування. Хоча він має схожий синтаксис з PostgreSQL, але за замовчуванням не перевіряє типи даних. Це означає, що, наприклад, можна вставити рядок у стовпець, що оголошений як ціле число. З усіма цими особливостями, SQLite є чудовим інструментом для ознайомлення з реляційними базами даних.

					ІАЛЦ.467200.003 ПЗ	Арк.
						24
Зм.	Арк.	№ докум.	Підпис	Дата		

2.4. Джерела даних

Для мого проєкту мені необхідні два джерела інформації: одне для отримання актуального статусу повітряної тривоги, та інше для доступу до повного актуального переліку сховищ міста Київ. На жаль в виборі джерел даних для виконання завдання нема такої свободи, як при виборі технологій.

- Для отримання статусів повітряних тривог є декілька API, але з них лише одне офіційне, яке гарантує стабільну роботу. Це API [29] створено компанією Stfalcon спільно з розробником Артемом Лемешевим, дані в застосунок «Повітряна тривога» [30] йдуть саме через це API.
- База даних укриттів є на сайті Київської міської державної адміністрації [31] в відкритому доступі, але там застарілі дані які оновлюються рідко, тому мною було залишено запит на публічну інформацію на відповідній офіційній електронній поштовій скриньці, який згодом був оброблений, в результаті чого мені була надана xlsx таблиця від Департаменту муніципальної безпеки з актуальним переліком сховищ.

					ІАЛЦ.467200.003 ПЗ	Арк.
						25
Зм.	Арк.	№ докум.	Підпис	Дата		

Висновок до розділу 2

В цьому розділі було проведено огляд можливих технологій для виконання поставленої задачі. В якості підбиття підсумку можна сказати, що оптимальним стеком технологій буде наступний:

- Python, як мова програмування, враховуючи мій власний досвід, підійде для реалізації проєкту більше.
- Із Python бібліотек для спрощення та покращення досвіду взаємодії з Telegram Bot API краще підійде Aiogram, так як навантаження на бота може бути доволі великим і через це краще писати асинхронний код, а з усіх бібліотек від початку під асинхронність писалась лише Aiogram, а в усі інші ця можливість згодом імплементувалась.
- Що стосовно баз даних, то під мою задачу краще всього підійде PostgreSQL база даних з встановленим плагіном PostGIS, бо він дозволяє зручно, просто та швидко проводити операції з географічними об'єктами, а мені саме такі операції й доведеться проводити для знаходження найближчих укриттів.
- Дані будуть братись з офіційного API тривоги, розробленого компанією Stfalcon спільно з розробником Артемом Лемешевим, а база сховищ отримана від Департаменту муніципальної безпеки через відділ обробки запитів на публічну інформацію в Київській міській державній адміністрації.

					ІАЛЦ.467200.003 ПЗ	Арк.
						26
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ОГЛЯД ПРОЦЕСУ РОЗРОБКУ ПРОЦЕСУ ТЕЛЕГРАМ-БОТА

3.1 Реєстрація бота та зазначення деяких параметрів

Як вже згадувалось в цій роботі, для реєстрації бота та отримання токenu необхідно написати офіційному боту телеграм BotFather. Для цього було використано команду /newbot та вказано пару обов'язкових параметрів, а саме назва бота та ім'я користувача (рис. 3.1).

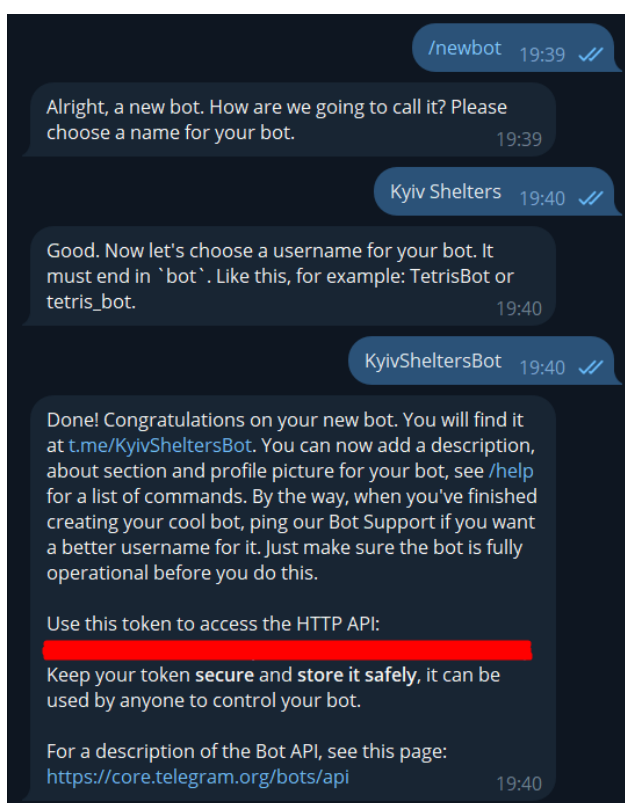


Рис. 3.1 – Отримання унікального Телеграм-бот токenu

Ще додатково для більшої якості й розуміння в функціональності бота було задано наступні параметри: «Інформація про бота» (рис. 3.2), «Опис бота» (рис. 3.3), «Зображення» (рис. 3.4), «Меню з командами» (рис. 3.5). Для цього ми використали зручне та інтуїтивно зрозуміле меню редагування бота в BotFather.

					ІАЛЦ.467200.003 ПЗ	Арк.
						27
Зм.	Арк.	№ докум.	Підпис	Дата		

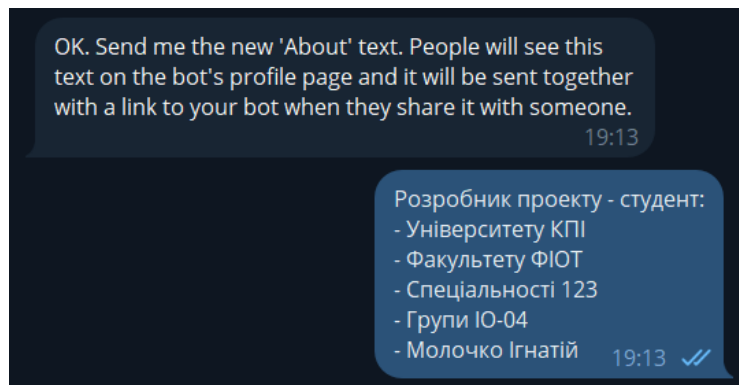


Рис. 3.2 – Зміна інформації про бота

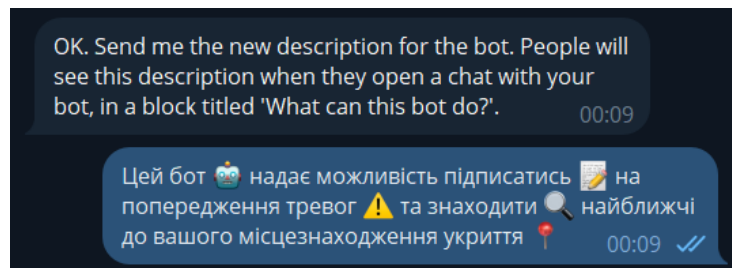


Рис. 3.3 – Зміна опису бота



Рис. 3.4 – Зміна зображення бота

					ІАЛЦ.467200.003 ПЗ	Арк.
						28
Зм.	Арк.	№ докум.	Підпис	Дата		

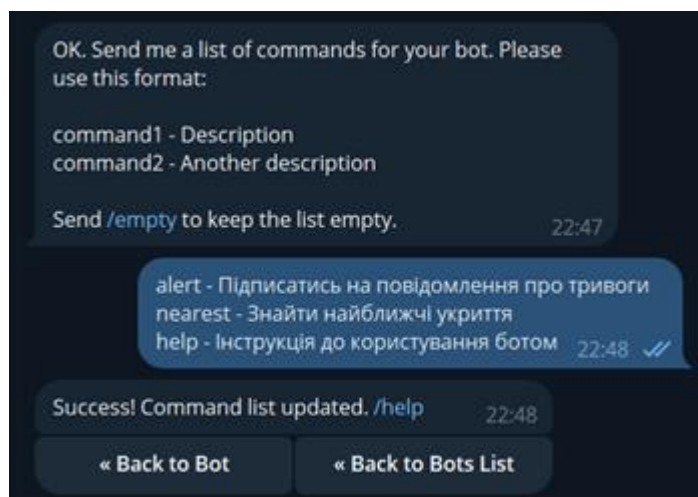


Рис. 3.5 – Створення меню бота з доступними командами

3.2 Отримання токена для використання API тривог

Для отримання токена до API від Stfalcon треба залишити запит на сайті <https://api.ukrainealarm.com/> вказавши трохи інформації про себе, вказавши призначення, для якого потрібно API, та погодившись з договором оферти (рис. 3.6).

Запит на доступ до API

Компанія **Stfalcon** спільно з розробником **Артемом Лемешевим** створили офіційний API для поширення інформації, яка надходить в застосунок Повітряна тривога.

Щоб отримати доступ до API, необхідно залишити заявку

Ім'я *

Компанія *

Номер телефону *

Email *

Посилання

Для чого потрібно API

☐
Приймаю та погоджуюсь з умовами [Договору оферти](#).

Відправити

Рис. 3.6 – Форма для відправки запиту на доступ до API

Після чого було отримано позитивну відповідь з токеном до API (рис. 3.7) та посиланням на документацію, реалізованою за допомогою Swagger [32].

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		29



Вітаємо

Вам видано доступ

Для користування API використовуйте наступний код доступу:



Документацію по API Ви зможете [отримати тут](#)

Увага, цей лист був надісланий автоматично. Не потрібно на нього відповідати

Рис. 3.7 – Лист від підтримки Stfalcon з токеном до API

3.3 Розробка телеграм бота

Для початку створимо основну структуру проєкту з усіма необхідними директоріями та файлами, також розмістимо надану таблицю зі сховищами за шляхом database/data, щоб зберегти чистоту проєкту та відразу розумітись на всьому необхідному в реалізації. (рис. 3.8)

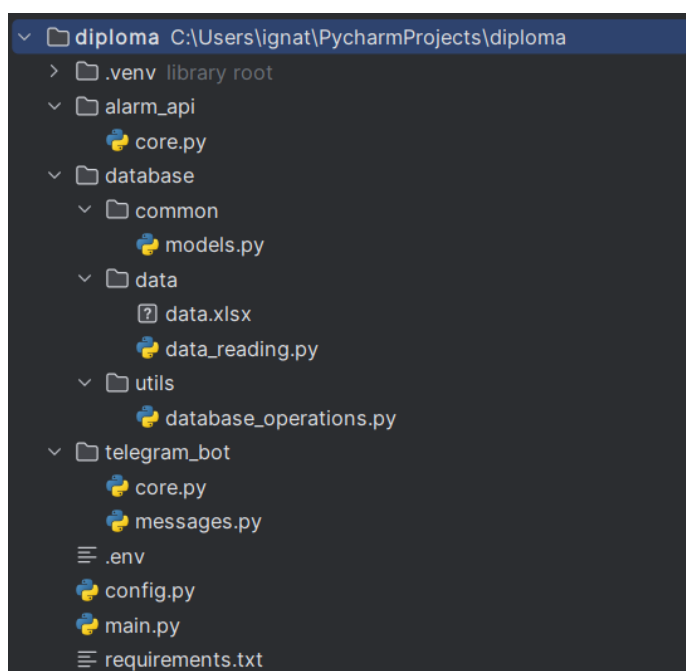


Рис. 3.8 – Структура проєкту

					ІАЛЦ.467200.003 ПЗ	Арк.
						30
Зм.	Арк.	№ докум.	Підпис	Дата		

Як можемо побачити в нас є три допоміжні директорії та основна. В основній в нас створено наступні файли: файл зі змінними середовища, конфігураційний файл з завантаженням цих змінних, main.py файл для запуску всього проєкту та файл з списком необхідних бібліотек для зручного та швидкого їх завантаження, а три допоміжні відповідають за роботу з API тривоги, базою даних та безпосередньо телеграм боту.

3.3.1 Створення змін середовища з конфігураційним файлом

Для початку пропишемо наші токени в файлі .env (рис. 3.9), які згодом будуть завантажені в середовище виконання Python за допомогою бібліотеки python-dotenv, це робиться, щоб цей окремий файл можна було виключити з системи контролю версій та випадково не оприлюднити секретну інформацію. Після цього в файлі config.py було безпосередньо завантажено ці змінні з файлу .env в середовище та створено екземпляр класу Settings з використанням типу SecretStr бібліотеки pydantic (рис. 3.10).

```

1 TELEBOT_TOKEN = "[REDACTED]"
2 ALERT_API_TOKEN = "[REDACTED]"

```

Рис. 3.9 – Файл .env зі вказаними змінами середовища

```

1 import os
2
3 from pydantic import BaseModel, SecretStr
4 from dotenv import load_dotenv
5
6 load_dotenv()
7
8
9 usage: Ignasha5451
10 class Settings(BaseModel):
11     telebot_token: SecretStr
12     alert_api_token: SecretStr
13
14 settings = Settings(
15     telebot_token=os.getenv("TELEBOT_TOKEN"),
16     alert_api_token=os.getenv("ALERT_API_TOKEN"),
17 )
18

```

Рис. 3.10 – Файл config.py

3.3.2 Створення, налаштування та робота з базою даних

Перед тим як створити базу даних та почати роботу з нею потрібно скачати PostgreSQL з розширенням PostGIS, для цього потрібно виконати декілька кроків, які описані на офіційних сайтах PostgreSQL [33] та PostGIS [34]. На будь-якому дистрибутиві Linux або MacOS достатньо буде виконати декілька команд в терміналі, а на Windows необхідно скачати відповідний установник з сайту PostgreSQL та в процесі установки вибрати розширення PostGIS. Після цього запустивши PostgreSQL потрібно створити нового користувача та, для зручності, надати йому права супер-користувача, для цього виконаємо наступні команди:

```
CREATE USER ignatiy WITH PASSWORD 'qwerty12345';  
ALTER USER ignatiy WITH SUPERUSER;
```

Після створення користувача, від якого будемо взаємодіяти з базою даних, можна створити базу даних, підключитись до неї та активувати в неї вже встановлене розширення PostGIS, для цього виконаємо ще декілька команд:

```
CREATE DATABASE mydatabase;  
\c mydatabase  
CREATE EXTENSION postgis;
```

Тепер можна повертатись до інтегрованої середовища розробки та продовжувати роботу з базою даних вже там. Безпосередньо взаємодіяти з БД ми будемо за допомогою ORM (Object-Relational Mapping) системи SQLAlchemy, що дозволить зробити код більш зрозумілим, за рахунок відсутності SQL запитів напряму в Python коді. Для початку потрібно буде створити асинхронний двигун для з'єднання з базою даних PostgreSQL та фабрику для створення асинхронних сесій (рис. 3.11).

					ІАЛЦ.467200.003 ПЗ	Арк.
						32
Зм.	Арк.	№ докум.	Підпис	Дата		

```

11 engine = create_async_engine(url="postgresql+asyncpg://ignatiy:qwerty12345@localhost:5432/mydatabase")
12
13 async_session = async_sessionmaker(engine)

```

Рис. 3.11 – Створення двигуна та фабрики для з'єднання з БД та створення асинхронних сесій

Далі створимо базовий клас, який буде успадковуватись від класів AsyncAttrs та DeclarativeBase для асинхронних атрибутів та декларативного стилю ORM відповідно. Від цього базового класу в майбутньому будуть успадковуватись всі моделі таблиць бази даних (рис. 3.12).

```

16 @ Ignasha5451 class Base(AsyncAttrs, DeclarativeBase):
17     pass

```

Рис. 3.12 – Створення базового класу

Тепер, використовуючи успадкування від базового класу, можемо створити дві моделі для двох таблиць, які будуть необхідні для зручної роботи телеграм бота.

Перша модель User таблиці users, в якій будуть зберігатись телеграм айди всіх зареєстрованих користувачів, в цій моделі є лише одне поле (окрім первинного ключа) типу BigInteger (оскільки телеграм айди являють собою дуже великі числа) яке має бути унікальним в таблиці

Та друга модель Shelter таблиці shelters, в якій будуть зберігатись всі існуючі та відомі сховища міста Київ з інформацію про них в наступних полях:

- district, address, type, owner, type_of_ownership, ramp – рядкові поля з інформацією про район, адресу, тип, власника, форму власності та наявність пристосування для доступу осіб з інвалідністю та інших маломобільних груп населення (пандусу) відповідно.
- coordinates – гео-просторове поле типу POINT з координатами сховища

					ІАЛЦ.467200.003 ПЗ	Арк.
						33
Зм.	Арк.	№ докум.	Підпис	Дата		

Після опису цих моделей напишемо асинхронну функцію, яка буде створювати таблиці в базі даних відповідно до описаних моделей (рис. 3.13).

```

20 class User(Base):
21     __tablename__ = "users"
22
23     id: Mapped[int] = mapped_column(Integer, primary_key=True)
24     user_id: Mapped[int] = mapped_column(BigInteger, unique=True)
25
26
12 usages  Ignasha5451
27 class Shelter(Base):
28     __tablename__ = "shelters"
29
30     id: Mapped[int] = mapped_column(Integer, primary_key=True)
31     district: Mapped[str] = mapped_column(String)
32     address: Mapped[str] = mapped_column(String)
33     coordinates: Mapped[WKTElement] = mapped_column(Geometry("POINT"))
34     type: Mapped[str] = mapped_column(String)
35     owner: Mapped[str] = mapped_column(String)
36     type_of_ownership: Mapped[str] = mapped_column(String)
37     ramp: Mapped[str] = mapped_column(String)
38
39
2 usages  Ignasha5451
40 async def async_main():
41     logger.info("Create connection with database.")
42     async with engine.begin() as conn:
43         await conn.run_sync(Base.metadata.create_all)
44     logger.info("Tables created successfully")
45

```

Рис. 3.13 – Моделі таблиць та асинхронна функція створення таблиць

Тепер, описавши моделі та створивши за їх допомогою необхідні таблиці в базі даних, можна описати необхідні операції з ними:

- create_user_instance – ця асинхронна функція буде записувати айді користувача до таблиці users, якщо той захоче підписатись на повідомлення про зміну статусу повітряної тривоги (рис. 3.14);

```

2 usages  Ignasha5451
14 async def create_user_instance(user_id: int) -> bool:
15     try:
16         async with async_session() as session:
17             user = await session.scalar(select(User).filter(User.user_id == user_id))
18             if not user:
19                 session.add(User(user_id=user_id))
20                 await session.commit()
21                 logger.info(f"User with id {user_id} created.")
22                 return True
23             else:
24                 logger.info(f"User with id {user_id} already exists.")
25                 return False
26     except Exception as e:
27         logger.error(f"Error creating user with id {user_id}: {e}")

```

Рис. 3.14 – Функція create_user_instance

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		34

- `read_user_instances` – ця асинхронна функція буде зчитувати айді всіх користувачів з таблиці `users` при зміні статусу повітряної тривоги в місті Київ, щоб повідомити всіх підписаних на розсилку користувачів (рис. 3.15);

```

2 usages  ▲ Ignasha5451
30 async def read_user_instances() -> list:
31     try:
32         async with async_session() as session:
33             query = await session.scalars(select(User))
34             logger.info(f"Read all registered users.")
35             return query
36     except Exception as e:
37         logger.error(f"Error reading user instances: {e}")

```

Рис. 3.15 – Функція `read_user_instances`

- `delete_user_instance` – ця асинхронна функція буде видаляти айді користувача з таблиці `users`, якщо той захоче відписатись від повідомлень про статус повітряної тривоги (рис. 3.16);

```

2 usages  ▲ Ignasha5451
40 async def delete_user_instance(user_id: int) -> bool:
41     try:
42         async with async_session() as session:
43             user = await session.scalar(select(User).filter(User.user_id == user_id))
44             if user:
45                 await session.delete(user)
46                 await session.commit()
47                 logger.info(f"User with id {user_id} deleted.")
48                 return True
49             else:
50                 logger.info(f"User with id {user_id} not found.")
51                 return False
52     except Exception as e:
53         logger.error(f"Error deleting user with id {user_id}: {e}")

```

Рис. 3.16 – Функція `delete_user_instance`

- `insert_data` – ця асинхронна функція проходить по списку сховищ в форматі словників, обробляє поле координатів, конвертуючи їх з строки в формат WKT (Well-Known Text) для точки (`POINT(lon lat)`) (рис. 3.17);

```

2 usages  Ignasha5451
56 async def insert_data(df_records):
57     try:
58         async with async_session() as session:
59             async with session.begin():
60                 for record in df_records:
61                     lat, lon = re.split( pattern: r"\s*\s*\s*", record["coordinates"].strip())
62                     record["coordinates"] = WKTElement(f"POINT({lon} {lat})")
63                     instance = await session.scalar(select(Shelter).filter(
64                         *criteria: Shelter.district == record["district"],
65                         Shelter.address == record["address"],
66                         Shelter.coordinates == record["coordinates"],
67                         Shelter.type == record["type"],
68                         Shelter.owner == record["owner"],
69                         Shelter.type_of_ownership == record["type_of_ownership"],
70                         Shelter.ramp == record["ramp"]
71                     ))
72                     if not instance:
73                         session.add(Shelter(**record))
74                     await session.commit()
75                     logger.info("Data inserted successfully.")
76     except Exception as e:
77         logger.error(f"Error inserting data: {e}")

```

Рис. 3.17 – Функція `insert_data`

- `find_nearest_shelters` – ця асинхронна функція сортує базу даних сховищ відносно наданої геопозиції, використовуючи спеціальну функцію з розширення `geoalchemy2` до `SQLAlchemy` (яка здебільшого створена для взаємодії з розширенням `PostGIS` до `PostgreSQL`) (рис. 3.18).

```

2 usages  Ignasha5451
85 async def find_nearest_shelters(user_lat, user_lon):
86     try:
87         async with async_session() as session:
88             user_location = WKTElement(f"POINT({user_lon} {user_lat})")
89             query = await session.scalars(
90                 select(Shelter).order_by(functions.ST_Distance(*args: Shelter.coordinates, user_location)).limit(5)
91             )
92             logger.info(f"Found 5 nearest shelters.")
93             return query
94     except Exception as e:
95         logger.error(f"Error finding nearest shelters: {e}")

```

Рис. 3.18 – Функція `find_nearest_shelters`

Зараз потрібно написати функцію, яка буде зчитувати дані з наданої `xlsx` таблиці в форматі списку словників, та згодом зможуть бути записані в базу даних, ця функція була реалізована в файлі `data_reading.py` (рис. 3.19).

```

2 usages  Ignasha5451
8  def xlsx_read():
9      file_path = "database/data/data.xlsx"
10     try:
11         df = pd.read_excel(file_path)
12         columns = {
13             "Район": "district",
14             "Адреса": "address",
15             "Координати ": "coordinates",
16             "Тип": "type",
17             "Власник": "owner",
18             "Форма власності": "type_of_ownership",
19             "Наявність пристосування для доступу осіб з інвалідністю та інших маломобільних груп населення (пандусу)": "ramp"
20         }
21         df = df.rename(columns=columns)
22         df_records = df.to_dict(orient="records")
23         logger.info(f"Read all shelters data from {file_path}")
24         return df_records
25     except Exception as e:
26         logger.error(f"Error reading shelters data from {file_path}: {e}")
27

```

Рис. 3.19 – Функція `xlsx_read`

3.3.3 Реалізація отримання та сповіщення про статус тривоги

Перед написанням самого боту з усіма обробниками повідомлень, треба ще реалізувати отримання поточного статусу тривоги та повідомлення зареєстрованих користувачів про його зміну. Для цього було написано дві асинхронних функції:

- `region_status`, яка робить запит до API з використанням секретного токена (рис. 3.20);

```

5  MAIN_URL = "https://api.ukrainealarm.com/api/v3"
6
7
8  2 usages  Ignasha5451
9  async def region_status(region_id):
10     response_url = f"{MAIN_URL}/alerts/{region_id}"
11     headers = {
12         "accept": "application/json",
13         "Authorization": settings.alert_api_token.get_secret_value()
14     }
15     async with aiohttp.ClientSession() as session:
16         async with session.get(response_url, headers=headers) as response:
17             return await response.json()

```

Рис. 3.20 – Функція `region_status`

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		37

- alert_status_sending, яка за допомогою функції region_status отримає поточний статус повітряної тривоги, перевіряє, чи змінився її стан з минулого її звертання, та відправляє сповіщення всім зареєстрованим користувачам, якщо він змінився (рис. 3.21).

```

116 previous_data = None
117
118
119 1 usage  ▲ Ignasha5451
119 async def alert_status_sending() -> None:
120     global previous_data
121     try:
122         current_data = await region_status(31)
123         if current_data != previous_data:
124             if previous_data:
125                 query = await read_user_instances()
126                 if not current_data[0]["activeAlerts"]:
127                     for user in query:
128                         await bot.send_message(user.user_id, text="🟢 Тривога закінчилась!")
129                         await asyncio.sleep(0.05)
130             else:
131                 buttons = [
132                     [
133                         types.InlineKeyboardButton(text="Знайти найближче укриття!",
134                                                    callback_data="nearest_shelter")
135                     ]
136                 ]
137                 keyboard = types.InlineKeyboardMarkup(inline_keyboard=buttons)
138                 for user in query:
139                     await bot.send_message(
140                         user.user_id,
141                         text="🔴 Почалась тривога! Прямуйте до найближчого укриття та бережіть себе!",
142                         reply_markup=keyboard
143                     )
144                     await asyncio.sleep(0.05)
145                 previous_data = current_data
146     except Exception as e:
147         logger.error(f"Error while sending response to Alert API: {e}")

```

Рис. 3.21 – Функція alert_status_sending

3.3.4 Обробники відправлених телеграм боту повідомлень

Спочатку потрібно створити декілька об'єктів класів (рис. 3.22), які будуть необхідні для роботи бота, серед них об'єкти наступний класів:

- Bot – об'єкт цього класу буде безпосередньо мати в собі бота та для його створення необхідно йому передати секретний токен
- Dispatcher – об'єкт цього класу буде використовуватися для обробки різних подій, таких як повідомлення, команди та інші дії, пов'язані з ботом.

					ІАЛЦ.467200.003 ПЗ	Арк.
						38
Зм.	Арк.	№ докум.	Підпис	Дата		

- Nominatim – об’єкт цього класу буде використовуватись для декодування, а саме для отримання назви міста з відправленої геопозиції
- AsyncIOScheduler – об’єкт цього класу буде використано для асинхронного планування задач, а саме для фонового постійного аналізу повітряної тривоги та сповіщення підписаних користувачів.

```

31 bot = Bot(token=settings.telebot_token.get_secret_value())
32 dp = Dispatcher()
33 geolocator = Nominatim(user_agent="kyiv_detect")
34 scheduler = AsyncIOScheduler()

```

Рис. 3.22 – Створення необхідних об’єктів класів

Тепер треба зрозуміти, які саме обробники нам знадобляться, для цього всього-на-всього треба проаналізувати поставлену задачу та зрозуміти, що треба для її реалізації, й щоб кінцеве рішення було зручним в використанні для користувачів.

По-перше, в будь-якому боті повинен бути обробник команди /start, бо саме ця команда відправляється при запуску бота і вона повинна видавати головну інформацію про бота, щоб було зрозуміло про його функціонал.

По-друге, у бота повинна бути можливість підписуватись та відписуватись від розсилки про зміну статусу повітряної тривоги, тож для цього виділимо команду /alert.

По-третє, бот точно повинен вміти надавати список з найближчих укриттів, для цього потрібно буде реалізувати два обробники: команди /nearest та відправленої геопозиції.

В-четверте, ще буде дуже корисним та зручним, якщо бот буде обробляти команду /help з подальшим наданням інформації про доступні для використання команди.

Отже, як мінімум нам знадобиться 5 обробників, а згодом можливо згодом з’ясується, що потрібно ще декілька додаткових

					ІАЛЦ.467200.003 ПЗ	Арк.
						39
Зм.	Арк.	№ докум.	Підпис	Дата		

Для обробки команди /start та /help напишемо функції start та info відповідно (рис. 3.23), які будуть у відповідь на команди відправляти повідомлення START_MESSAGE та HELP_MESSAGE (рис. 3.24).

```

38 @dp.message(CommandStart())
39 async def start(message: types.Message) -> None:
40     await message.reply(text=START_MESSAGE)
41
42
43 @dp.message(Command("help"))
44 async def info(message: types.Message) -> None:
45     await message.reply(text=HELP_MESSAGE)

```

Рис. 3.23 – Функції-обробники start та info команд /start та /help

```

1  START_MESSAGE = """
2  Привіт!
3  Я бот-помічник 🤖 для відстежування статусу тривоги 🚨 та допомоги в відстежуванні найближчих укриттів 📍.
4  Інструкція 📖: /help
5  """
6
7  HELP_MESSAGE = """
8  Цей бот має для використання наступні команди 📖:
9  /alert 🚨 – надає можливість підписатись та відписатись від повідомлення про зміну статусу повітряної тривоги
10 /nearest 📍 – при відправці геопозиції в межах Києва надає перелік найближчих до неї укриттів
11 /help 📖 – надає перелік доступних для використання команд бота з описом їх функціональності
12 """

```

Рис. 3.24 – Повідомлення, які відправляються у відповідь на команди /start та /help

Далі реалізуємо обробку команди /alert, на яке буде відправлятись повідомлення з питанням, що бажає зробити користувач, підписатись чи відписатись від повідомлень про зміну статусу тривоги. Також до повідомлення буде прикріплена клавіатура з кнопками «Підписатись!» та «Відписатись!».

```

48 @dp.message(Command("alert"))
49 async def alert(message: types.Message) -> None:
50     buttons = [
51         [
52             types.InlineKeyboardButton(text="Підписатись!", callback_data="subscribe"),
53             types.InlineKeyboardButton(text="Відписатись!", callback_data="unsubscribe")
54         ]
55     ]
56     keyboard = types.InlineKeyboardMarkup(inline_keyboard=buttons)
57     text = ("Ви хочете підписатись чи відписатись на повідомлення про тривоги по місту Київ?\n\n"
58           "P.S. Ви завжди зможете змінити свій вибір!")
59     await message.answer(text=text, reply_markup=keyboard)

```

Рис. 3.25 – Функція-обробник alert команди /alert

					ІАЛЦ.467200.003 ПЗ	Арк.
						40
Зм.	Арк.	№ докум.	Підпис	Дата		

Тепер напишемо обробники команди /nearest та відправленої боту геопозиції. Для цього напишемо функції `get_location_from_command` та `get_location_from_button`, які будуть звертатись до допоміжної функції `get_location` (рис. 3.26). Також напишемо обробку відправлених боту координат, яка використовуючи об'єкт класу `Nominatim` перевіряє, чи знаходиться людина в Києві, після чого виводить п'ять найближчих до цілі сховищ з затримкою в 0.05 с, бо у Telegram Bot API є обмеження на кількість відправлених повідомлень в секунду (рис. 3.27).

```

62  async def get_location(message: types.Message) -> None:
63      await message.reply(text=GEOLOCATION_SHARING,
64                          reply_markup=types.ReplyKeyboardMarkup(
65                              keyboard=[[types.KeyboardButton(text="Поділитись геопозицією", request_location=True)],
66                              resize_keyboard=True))
67
68
69  @dp.message(Command("nearest"))
70  async def get_location_from_command(message: types.Message) -> None:
71      await get_location(message)
72
73
74  new *
75  @dp.callback_query(F.data == "nearest_shelter")
76  async def get_location_from_button(callback: types.CallbackQuery) -> None:
77      await get_location(callback.message)

```

Рис. 3.26 – Функції-обробники які використовують `get_location`

```

74  @dp.message(F.location)
75  async def handle_location(message: types.Message) -> None:
76      latitude = message.location.latitude
77      longitude = message.location.longitude
78      location = geolocator.reverse(query=(latitude, longitude), language="uk")
79      await message.reply(text=f"Ви зараз знаходитесь за адресою:\n{location.address}")
80      if "Київ" in re.findall(pattern=r"\b\w+\b", location.address):
81          query = await find_nearest_shelters(latitude, longitude)
82          for instance in query:
83              geometry = wkb.loads(bytes(instance.coordinates.data))
84              await bot.send_location(message.chat.id, latitude=geometry.y, longitude=geometry.x)
85              await bot.send_message(message.chat.id, text=SHELTER_INFO.format(
86                  district=instance.district,
87                  address=instance.address,
88                  type=instance.type,
89                  owner=instance.owner,
90                  type_of_ownership=instance.type_of_ownership,
91                  ramp=instance.ramp
92              ))
93              await asyncio.sleep(0.05)
94          await bot.send_message(message.chat.id, text=SHELTER_FEEDBACK)
95      else:
96          await message.reply(text=OUTSIDE_KYIV)

```

Рис. 3.27 – Функція-обробник `handle_location` відправленої геопозиції

					ІАЛЦ.467200.003 ПЗ	Арк.
						41
Зм.	Арк.	№ докум.	Підпис	Дата		

Також треба не забути написати обробку натискання на кнопки «Підписатись!» та «Відписатись!» з команди /alert, ще обов’язково потрібно перевіряти чи не є користувач вже підписаним та відписаним, щоб не виконувати запит на запис даних в базу даних просто так (рис. 3.28).

```

99  @dp.callback_query(F.data.count("subscribe") == 1)
100  async def callback_sub(callback: types.CallbackQuery) -> None:
101      if not callback.data.startswith("un"):
102          await callback.answer(text='Ви натиснули кнопку "Підписатись!"')
103          res = await create_user_instance(callback.message.chat.id)
104          await callback.message.answer(text="Ви успішно підписались!" if res else "Ви вже підписані!")
105      else:
106          await callback.answer(text='Ви натиснули кнопку "Відписатись!"')
107          res = await delete_user_instance(callback.message.chat.id)
108          await callback.message.answer(text="Ви успішно відписались!" if res else "Ви і не підписані!")

```

Рис. 3.28 – Функція-обробник натискання на кнопки «Підписатись!» та «Відписатись!»

Тепер залишилось лише написати функцію, яка буде запускати всі складові проєкту включно з самим ботом (рис. 3.29), для цього була реалізована функція bot_run, яка виконує наступні дії:

- спочатку викликає функцію async_main, яка створює потрібні таблиці в базі даних
- потім функції xlsx_read та insert_data для зчитування даних про сховища з xlsx таблиці та подальшого їх запису в відповідну таблицю бази даних
- після чого, використовуючи об’єкт класу AsyncIOScheduler, було створено й запущено фонову задачу, яка буде кожну хвилину перевіряти, чи змінився статус тривоги, та повідомляти про це підписаних користувачів
- на останок запускається сам бот.

```

2 usages  Ignasha5451
150  async def bot_run() -> None:
151      await async_main()
152      df_records = xlsx_read()
153      await insert_data(df_records=df_records)
154      scheduler.add_job(alert_status_sending, trigger="interval", minutes=1)
155      scheduler.start()
156      await dp.start_polling(bot)

```

Рис. 3.29 – Функція bot_run

					ІАЛЦ.467200.003 ПЗ	Арк.
						42
Зм.	Арк.	№ докум.	Підпис	Дата		

Також вважається гарним тоном робити так, щоб весь проєкт запускався з одного файлу в головній директорії проєкту, тому для цього було створено файл main.py, в якому були вказані базові налаштування логуювання та запущений проєкт за допомогою функції bot_run (рис. 3.30).

```

1  import asyncio
2  import logging
3
4  from telegram_bot.core import bot_run
5
6  if __name__ == "__main__":
7      logging.basicConfig(
8          level=logging.INFO,
9          format="%(asctime)s - %(levelname)s - %(module)s - %(message)s",
10         handlers=[
11             logging.StreamHandler(),
12             logging.FileHandler("log/bot.log")
13         ]
14     )
15     logger = logging.getLogger(__name__)
16     try:
17         logger.info("Bot start.")
18         asyncio.run(bot_run())
19     except Exception as e:
20         logger.error(f"Error while bot running: {e}")
21

```

Рис. 3.30 – Файл main.py

					ІАЛЦ.467200.003 ПЗ	Арк.
						43
Зм.	Арк.	№ докум.	Підпис	Дата		

Висновок до розділу 3

В цьому розділі було детально описано процес розробки телеграм-бота, який відповідає темі дипломного проєкту – «Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної тривоги». Процес розробки був розглянутий починаючи від отримання унікального токена від офіційного бота Telegram з ім'ям BotFather, який є обов'язковим для підключення свого бота до Telegram Bot API та взаємодії з ним, закінчуючи написанням main.py файлу, який повністю запускає проєкт.

Отримання токена від Telegram передбачає реєстрацію бота в системі Telegram Bot API, в якій було закріплено унікальний токен за ботом. Цей токен ідентифікує бота та дозволяє розробнику взаємодіяти й налаштовувати його, після чого він зможе взаємодіяти з користувачами.

Крім токена від Telegram, для виконання деяких функцій бота, наприклад, отримання актуальних статусів тривог, необхідно мати ключ доступу до відповідного API. Цей ключ надається після заповнення та відправки запиту на відповідному сайті та дозволяє боту отримувати необхідну інформацію про актуальний статус повітряної тривоги.

Ще для реалізації функціоналу пошуку найближчих укриттів було необхідно отримати саму базу даних сховищ, що було зроблено за допомогою відправленого запиту на електронну поштову скриньку для запитів на доступ на публічну інформацію в київській міській державній адміністрації.

Таким чином, отримавши необхідні токени з базою даних сховищ та написавши код, ми завершили розробку телеграм-бота для сповіщення користувачів про зміну статусу тривоги та допомоги в пошуку в найближчих укриттів.

					ІАЛЦ.467200.003 ПЗ	Арк.
						44
Зм.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 4. ОГЛЯД ТА ТЕСТУВАННЯ РОЗРОБЛЕНОГО ТЕЛЕГРАМ-БОТА

4.1 Тестування ініціалізації бота

Після закінчення написання коду дипломного проєкту «Telegram-бот для пошуку найближчих укриттів у м. Київ під час повітряної тривоги» проведемо його тестування з повним оглядом реалізованого функціоналу.

При першому запуску проєкту в першу чергу в нас повинні створитись таблиці в базі даних та зчитатись дані з `xlsx` файлу, після чого записатись у відповідну таблицю бази даних. Виконаємо декілька запитів в БД, щоб впевнитись що все правильно працює.

В першу чергу під'єднавшись до бази даних виконаємо певний запит, щоб впевнитись, що таблиці були створені. Як можемо побачити (рис. 4.1) в нас з'явилися таблиці «shelters» та «users» (таблиця `spatial_ref_sys` є системною й з'являється при підключенні розширення PostGIS)

```
mydatabase=# \dt
              List of relations
Schema |      Name      | Type  | Owner
-----+-----+-----+-----
public | shelters       | table | ignatiy
public | spatial_ref_sys | table | postgres
public | users          | table | ignatiy
(3 rows)
```

Рис. 4.1 – Запит на відображення всіх таблиць бази даних

Також виконавши запит:

```
SELECT COUNT(*) FROM shelters;
```

отримаємо кількість записів в таблиці й саме так впевнимось, що всі сховища з `xlsx` файлу було успішно перенесено (рис. 4.2).

```
mydatabase=# SELECT COUNT(*) FROM shelters;
count
-----
    3560
(1 row)
```

Рис. 4.2 – Запит в БД на відображення кількості записів в таблиці shelters

Ще виконавши запит:

```
SELECT * FROM users;
```

зможем впевнитись, що таблиця users від самого початку є пустою (рис. 4.3).

```
mydatabase=# SELECT * FROM users;
id | user_id
----+-----
(0 rows)
```

Рис. 4.3 – Запит в БД на відображення всіх записів в таблиці users

4.2 Тестування базових команд /help та /start

При першому зверненні до будь-якого бота Telegram внизу біля поля вводу є кнопка «Розпочати» щоб автоматично відправити команду /start та почати спілкування.

Також в будь-який момент є можливість зупинити роботу бота та тоді замість кнопки «Розпочати» буде вже кнопка «Перезапустити».

В моєму ж випадку, оскільки я багато разів вже тестував бота при його розробці, буде саме кнопка «Перезапустити» (рис. 4.4). При її натисканні, як і очікується, відправляється команда /start на що отримується шаблонна відповідь (рис. 4.5). Також відразу можна побачити зручне меню, яке було створено у BotFather при створенні бота (рис. 4.6).

					ІАЛЦ.467200.003 ПЗ	Арк.
						46
Зм.	Арк.	№ докум.	Підпис	Дата		

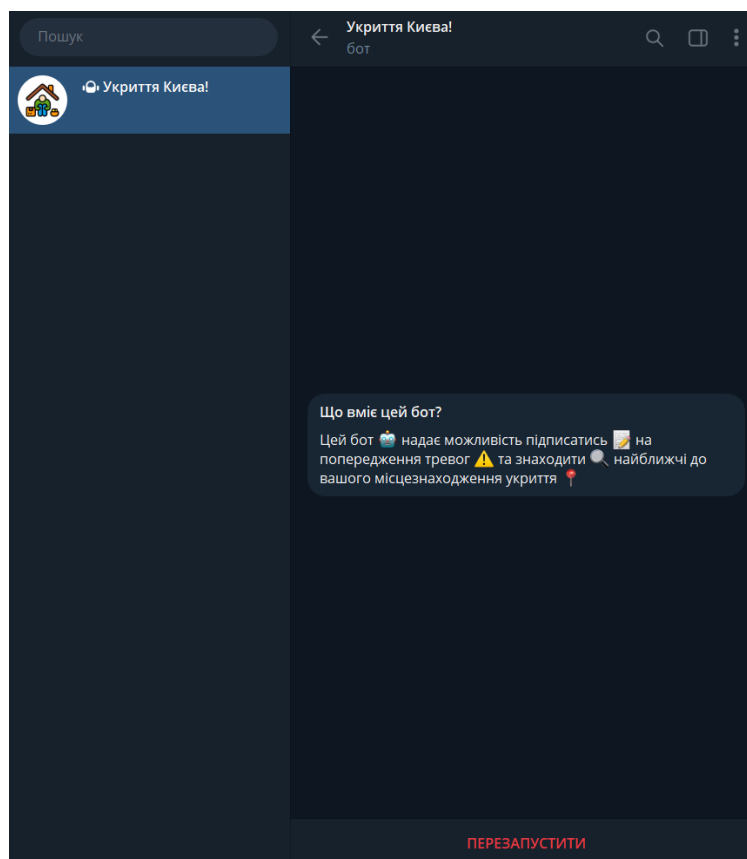


Рис. 4.4 – Початковий екран бота

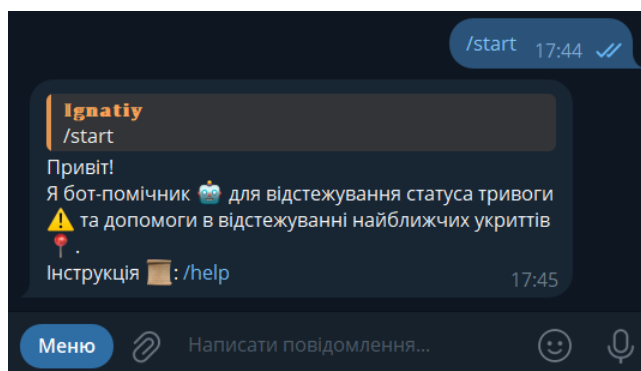


Рис. 4.5 – Запуск бота з наступним надсиланням команди /start

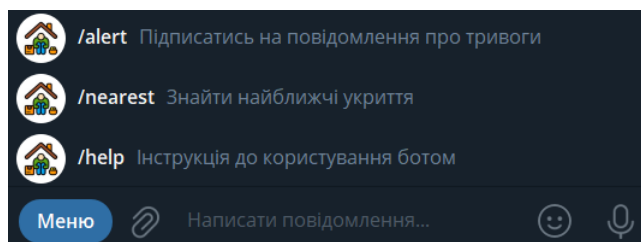


Рис.4.6 – Налаштоване меню для зручного використання команд бота

					ІАЛЦ.467200.003 ПЗ	Арк.
						47
Зм.	Арк.	№ докум.	Підпис	Дата		

Далі, натиснувши на команду /help у відповіді бота, автоматично відправимо її від свого обличчя й отримаємо відповідь від бота з коротким описом всіх доступних для використання команд (рис. 4.7).

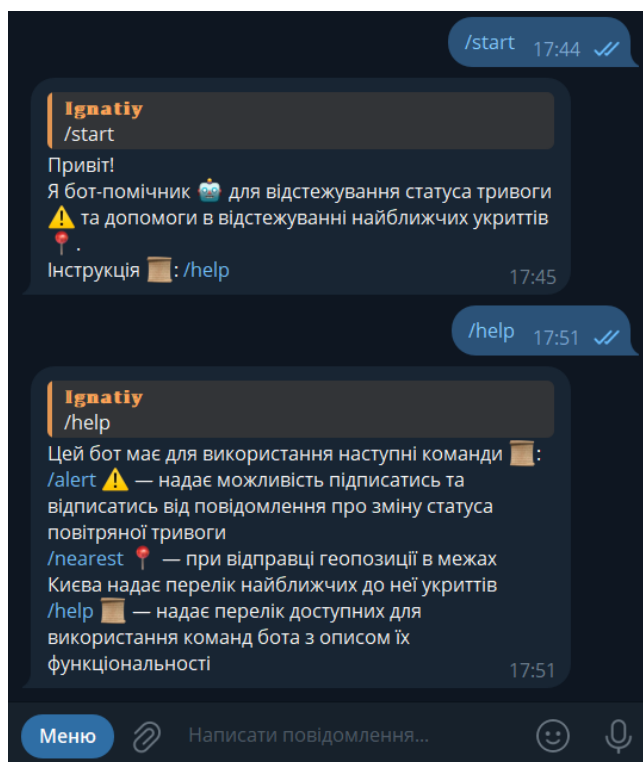


Рис. 4.7 – Надсилення та отримання відповіді на команду /help

4.3 Тестування команди /alert

Далі скористаємось командою /alert на що отримаємо пропозицію підписатись або відписатись від отримання повідомлень про зміну статусу тривоги. Далі буде чотири сценарії взаємодії з кнопками під повідомленням бота (рис. 4.8).

- Якщо натиснути кнопку «Відписатись!», коли ви ще не підписані, то ви отримаєте відповідне повідомлення про це та нічого більше не відбудеться;
- При натисканні кнопки «Підписатись!», якщо ви ще не підписані, ваш унікальний телеграм айді буде записаний в таблицю users бази даних та ви отримаєте відповідне повідомлення про це;

					ІАЛЦ.467200.003 ПЗ	Арк.
						48
Зм.	Арк.	№ докум.	Підпис	Дата		

Якщо натиснути кнопку «Підписатись!», коли ви вже підписані, то відбудеться все ідентично першому сценарію;

При натисканні кнопки «Відписатись!», якщо ви вже підписані, ваш унікальний телеграм айді буде видалений з таблиці users бази даних та ви отримаєте відповідне повідомлення про це.

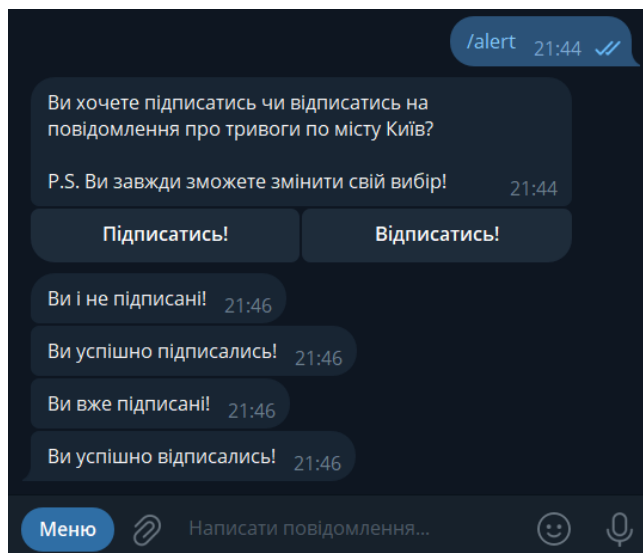


Рис. 4.8 – Чотири можливих сценарії натискання на кнопки при виконанні команди /alert

Також ще раз натиснемо на кнопку «Підписатись!» й повторно проведемо запит до бази даних для відображення всіх записів в таблиці users, щоб впевнитись в тому, що там з'являється новий запис (рис. 4.9).

```
mydatabase=# SELECT * FROM users;
 id | user_id
----+-----
  1 | 410415937
(1 row)
```

Рис. 4.9 – Запит в БД для виводу всіх записів в таблиці users

4.4 Тестування команди /nearest

Далі, поки чекаємо на початок повітряної тривоги, протестуємо ще команду /nearest, при її використанні бот відправляє повідомлення, в якому є інструкція, як можна відправити йому ваше розташування, для цього є два шляхи, а саме включити геопозицію на телефоні та або натиснути на кнопку «Поділитись геопозицією», або зробити це вручну через інтуїтивно зрозумілий інтерфейс Telegram (рис. 4.10).

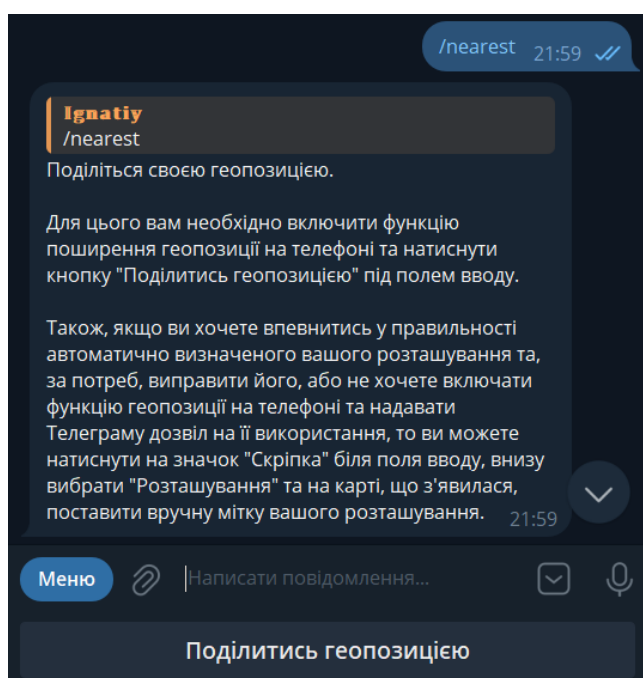


Рис. 4.10 – Реакція бота на команду /nearest

Також треба зазначити, що при ручній відправці розташування можна на карті двигати та корегувати мітку. У відповідь на геопозицію буде відправлена адреса відправленої мітки та наведено п'ять найближчих укриттів, якщо ця мітка знаходиться в межах міста Київ (рис. 4.11, рис. 4.12, рис. 4.13, рис. 4.14, рис. 4.15, рис. 4.16, рис. 4.17).

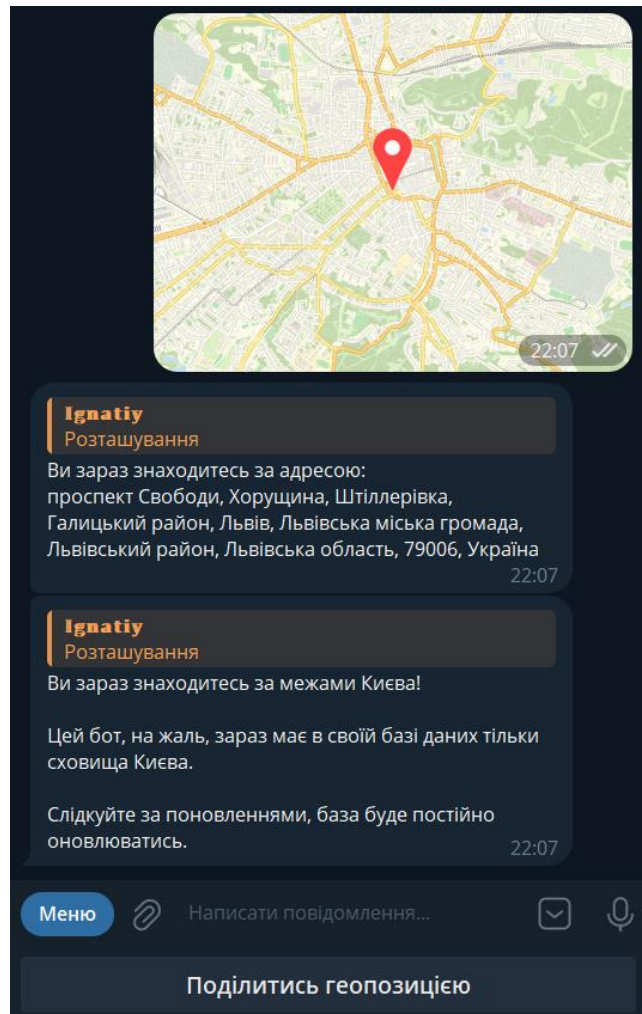


Рис. 4.11 – Реакція бота на мітку за межами Києва

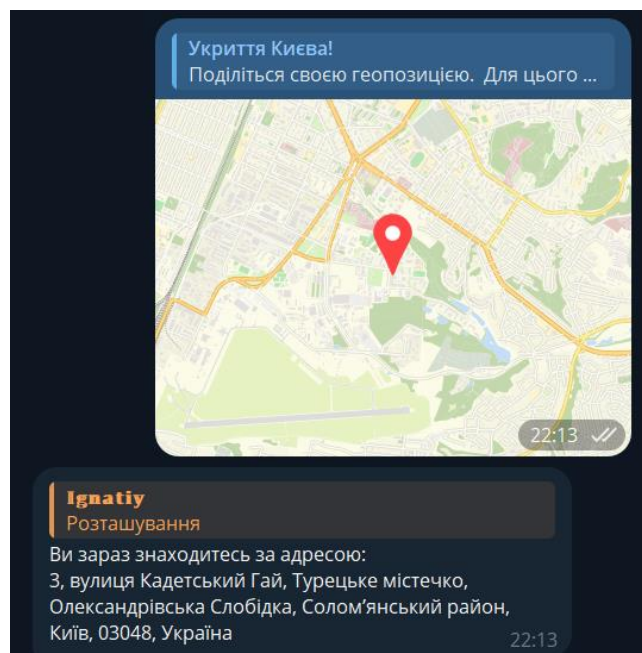


Рис. 4.12 – Реакція бота на мітку в межах Києва, ідентифікована адреса

					ІАЛЦ.467200.003 ПЗ	Арк.
						51
Зм.	Арк.	№ докум.	Підпис	Дата		

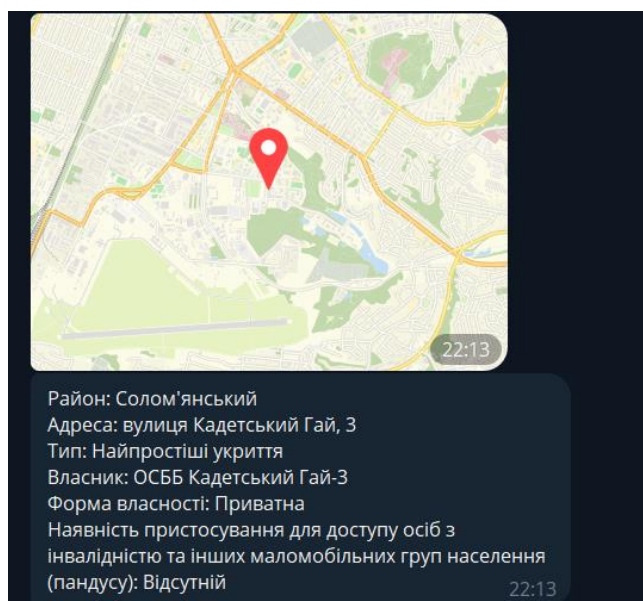


Рис. 4.13 – Реакція бота на мітку в межах Києва, запропоноване перше укриття

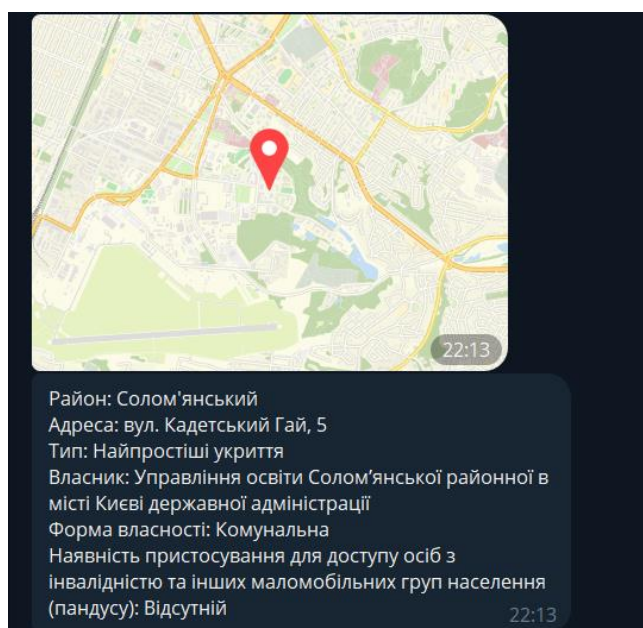


Рис. 4.14 – Реакція бота на мітку в межах Києва, запропоноване друге укриття

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		52

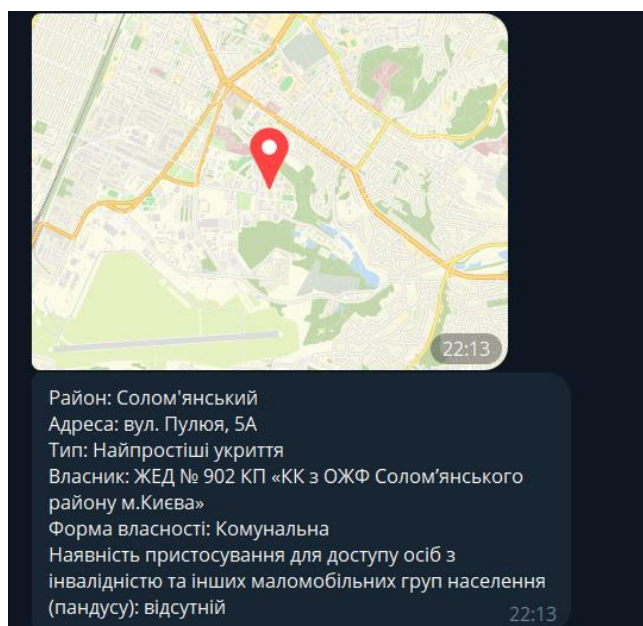


Рис. 4.15 – Реакція бота на мітку в межах Києва, запропоноване третє укриття

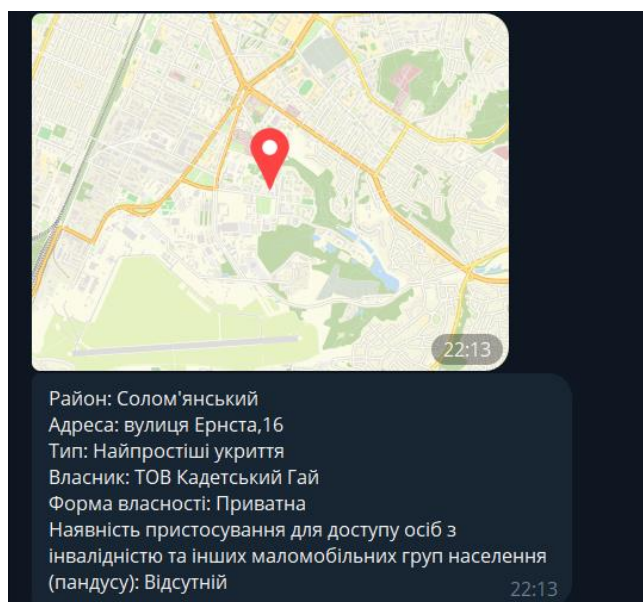


Рис. 4.16 – Реакція бота на мітку в межах Києва, запропоноване четверте укриття

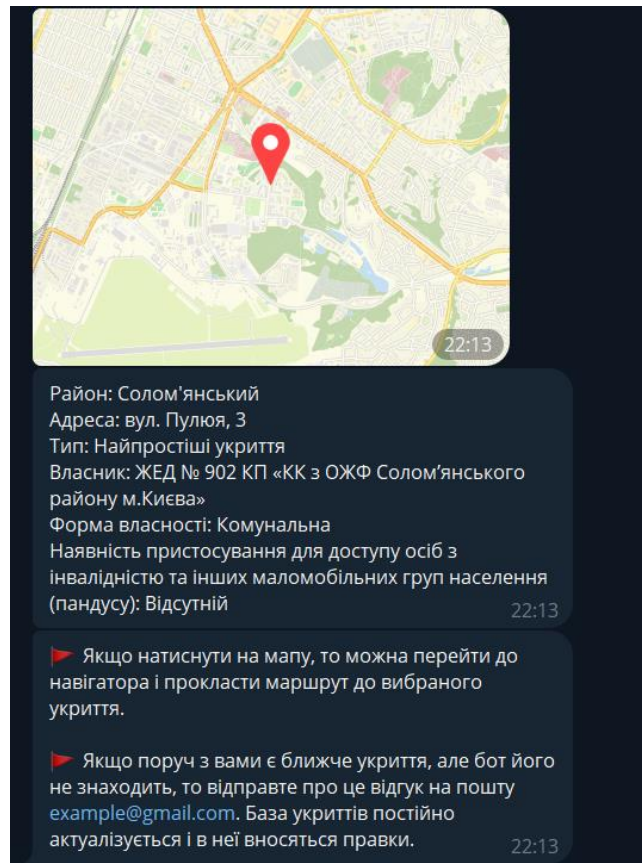


Рис. 4.17 – Реакція бота на мітку в межах Києва, запропоноване п'яте укриття з загальними порадами

4.5 Тестування отримання сповіщення про зміну статусу тривоги

Тепер, дочекавшись початку повітряної тривоги, можемо побачити відповідне повідомлення надіслане ботом автоматично з рекомендацією знайти укриття та відповідною кнопкою (рис. 4.18). Кнопка «Знайти найближче укриття!» поводить себе абсолютно ідентично з командою /nearest, бо використовують один і той самий обробник.

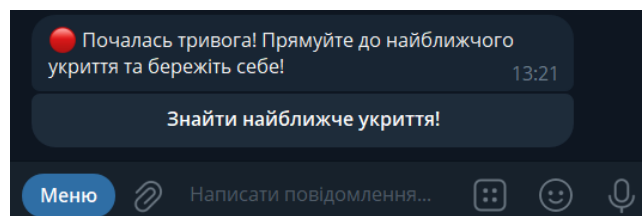


Рис. 4.18 – Повідомлення бота про початок повітряної тривоги

					ІАЛЦ.467200.003 ПЗ	Арк.
Зм.	Арк.	№ докум.	Підпис	Дата		54

Також, як можемо побачити, після завершення тривоги теж прийшло відповідне повідомлення про це (рис. 4.19).

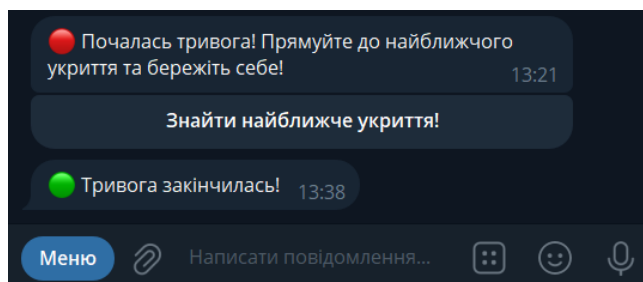


Рис. 4.19 – Повідомлення бота про закінчення повітряної тривоги

Ще проведемо перевірку коректності повідомлень скориставшись додатком «Київ цифровий» (рис. 4.20), як можемо побачити час повідомлень спіпадає.

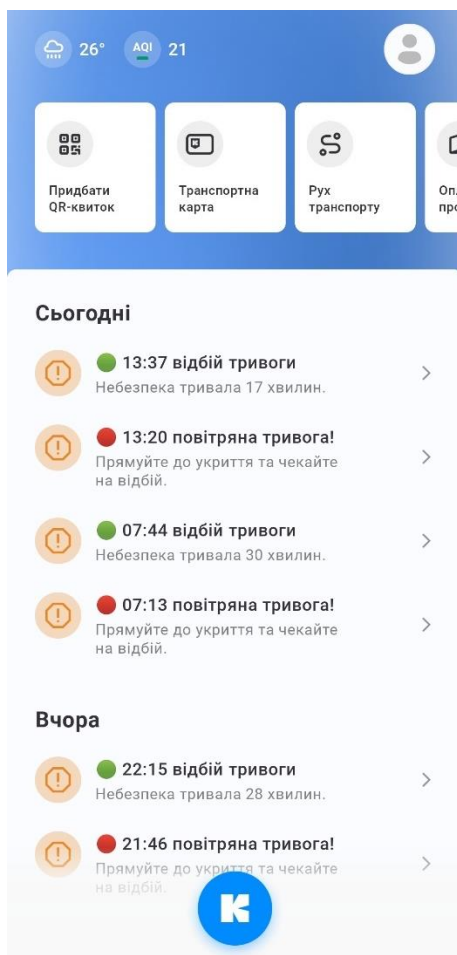


Рис. 4.20 – Інтерфейс додатку «Київ цифровий»

					ІАЛЦ.467200.003 ПЗ	Арк.
						55
Зм.	Арк.	№ докум.	Підпис	Дата		

Висновок до розділу 4

В цьому розділі було детально розглянуто весь розроблений функціонал телеграм бота з усіма можливими сценаріями поведінки користувачів. Проєкт було протестовано в повному обсязі, всі функції виконуються так, як первинно й задумувалось, та без будь-яких помилок, численні скріншоти це підтверджують. Проєкт є завершеним та не вимагає жодних технічних правок.

Готовий телеграм бот є дуже простим в розгортанні та зручним в використанні, що дозволить мешканцям Києва просто та швидко реагувати на змін стану повітряної тривоги та зберегти свої життя.

					ІАЛЦ.467200.003 ПЗ	Арк.
						56
Зм.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВОК

У даному дипломному проєкті мною було розроблено Телеграм бот, який допоможе користувачам, а саме мешканцям міста Київ, просто та швидко отримувати сповіщення про зміну статусу тривоги та реагувати на неї, знаходити найближчі укриття до їх поточного місцезнаходження.

У першому розділі я детально розглянув переваги та недоліки існуючих месенджерів, в результаті їх аналізу, та дивлячись на зростаючу популярність Telegram, в тому числі серед українців, вирішив бота розробляти саме під цю платформу. Також мною було проведено дослідження аналогічних ботів, створених з ідентичним функціоналом, розібрав їх переваги та недоліки, і відштовхуючись від цього розробив план для вирішення поставленої задачі.

В другому розділі мною було проведено аналіз існуючих технологій, які підходять для вирішення поставленої задачі. На основі виявлених переваг для розробки була вибрана мова програмування Python, бо в мене є великий опит розробки з її використанням, база даних PostgreSQL, через наявність розширення PostGIS для роботи з геопросторовими даними, а ще бібліотека Aiogram для більш зручної взаємодії з Telegram Bot API, за рахунок її початкового створення під асинхронність. Також було проведено дослідження джерел, з яких можна брати дані для реалізації проєкту.

У третьому розділі я детально описав всі етапи процесу розробки телеграм бота, починаючи від отримання необхідних токенів та створення бази даних й закінчуючи запуском проєкта через один файл.

Четвертий розділ було присвячено огляду та тестуванню функціоналу розробленого бота з усіма можливими сценаріями використання.

В результаті виконання дипломного проєкту я отримав вдалий програмний продукт, який повністю відповідає всім поставленим вимогам на початку розробки завдання.

					ІАЛЦ.467200.003 ПЗ	Арк.
						57
Зм.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Загальні відомості про Telegram [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Telegram>
2. Загальні відомості про MTProto [Електронний ресурс] – Режим доступу до ресурсу: <https://core.telegram.org/mtproto>
3. Обмеження груп і каналів Telegram: вичерпний огляд [Електронний ресурс] – Режим доступу до ресурсу: <https://teleteg.com/blogs/telegram-group-and-channel-limits>
4. Надсилання файлів на стероїдах — і багато іншого [Електронний ресурс] – Режим доступу до ресурсу: <https://telegram.org/blog/files-on-steroids>
5. Групові відеодзвінки [Електронний ресурс] – Режим доступу до ресурсу: <https://telegram.org/blog/group-video-calls>
6. HTML5 ігри для Telegram [Електронний ресурс] – Режим доступу до ресурсу: <https://core.telegram.org/bots/games>
7. Особливості Telegram ботів [Електронний ресурс] – Режим доступу до ресурсу: <https://core.telegram.org/bots/features>
8. Корисні функції Telegram [Електронний ресурс] – Режим доступу до ресурсу: <https://maxnet.ua/en/blog/20-i-1-korisna-mozhливist-telegram-vidkrijte-dlya-sebe-novi/>
9. Переваги та недоліки месенджера Telegram [Електронний ресурс] – Режим доступу до ресурсу: <https://www.profolus.com/topics/advantages-disadvantages-of-telegram-messenger/>
10. Що таке бот Telegram? [Електронний ресурс] – Режим доступу до ресурсу: <https://www.doprax.com/app-ideas/what-is-a-telegram-bot/>
11. Телеграм-бот «Укриття!» [Електронний ресурс] – Режим доступу до ресурсу: <https://stfalcon.com/uk/portfolio/telegram-bot-shelter>

					ІАЛЦ.467200.003 ПЗ	Арк.
						58
Зм.	Арк.	№ докум.	Підпис	Дата		

12. Основні бібліотеки мов програмування для Telegram Bot API [Електронний ресурс] – Режим доступу до ресурсу: <https://core.telegram.org/bots/samples>
13. Загальні відомості Python [Електронний ресурс] – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Python_\(programming_language\)](https://en.wikipedia.org/wiki/Python_(programming_language))
14. Загальні відомості про «збирання сміття» в комп'ютерних науках [Електронний ресурс] – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Garbage_collection_\(computer_science\)](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science))
15. Загальні відомості про JavaScript [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/JavaScript>
16. Загальні відомості про ECMAScript [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/ECMAScript>
17. Загальні відомості про Node.js [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Node.js>
18. Загальні відомості про Go [Електронний ресурс] – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Go_\(programming_language\)](https://en.wikipedia.org/wiki/Go_(programming_language))
19. Технічна документація Aiogram [Електронний ресурс] – Режим доступу до ресурсу: <https://aiogram.dev/>
20. Технічна документація Asyncio [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.python.org/3/library/asyncio.html>
21. Технічна документація python-telegram-bot [Електронний ресурс] – Режим доступу до ресурсу: <https://python-telegram-bot.org/>
22. Технічна документація pyTelegramBotAPI [Електронний ресурс] – Режим доступу до ресурсу: <https://pytba.readthedocs.io/en/latest/>
23. Загальні відомості про бази даних [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Database>
24. Загальні відомості про SQL [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/SQL>

					ІАЛЦ.467200.003 ПЗ	Арк.
						59
Зм.	Арк.	№ докум.	Підпис	Дата		

25. Відмінності між SQL і NoSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/difference-between-sql-and-nosql/>
26. Загальні відомості про MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/MySQL>
27. Загальні відомості про PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/PostgreSQL>
28. Загальні відомості про SQLite [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/SQLite>
29. Форма для запиту на доступ до Ukraine Alert API [Електронний ресурс] – Режим доступу до ресурсу: <https://api.ukrainealarm.com/>
30. Додаток «Повітряна тривога» [Електронний ресурс] – Режим доступу до ресурсу: https://stfalcon.com/uk/portfolio/air_alert
31. Офіційний портал Києва [Електронний ресурс] – Режим доступу до ресурсу: <https://kyivcity.gov.ua/>
32. Документація Ukraine Alert API [Електронний ресурс] – Режим доступу до ресурсу: <https://api.ukrainealarm.com/swagger/index.html>
33. Офіційний ресурс PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.postgresql.org/>
34. Офіційний ресурс PostGIS [Електронний ресурс] – Режим доступу до ресурсу: <https://postgis.net/>

ДОДАТОК 1

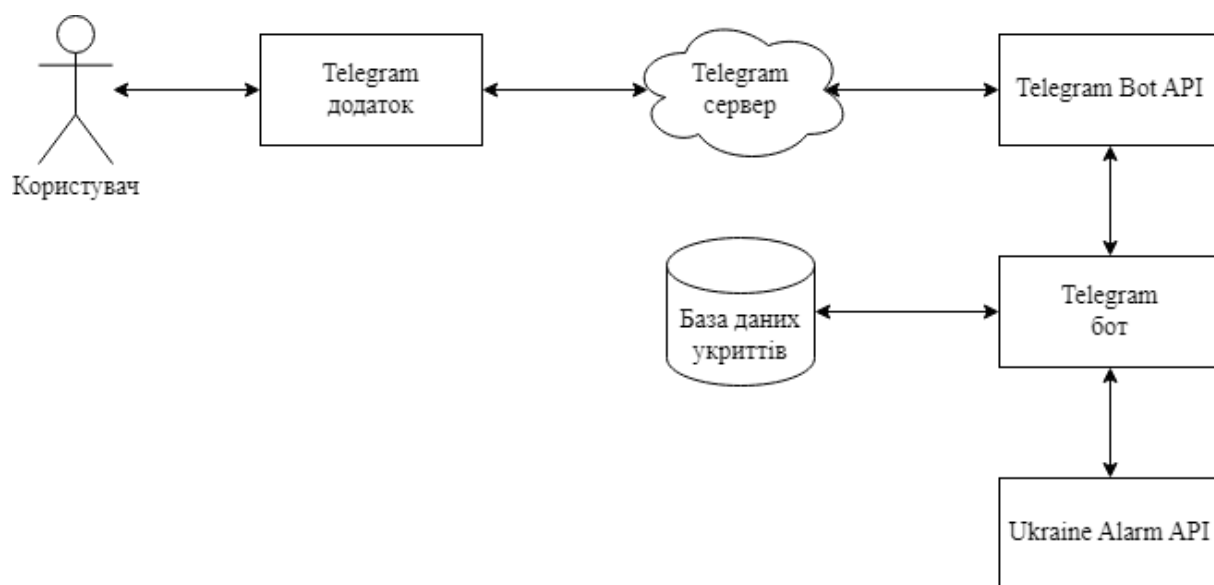
Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної
тривоги

Структура системи (Структурна схема)

ІАЛЩ.467200.004 Д1

Аркушів 1

Київ 2024



					ІАЛЦ.467200.004 Д1						
Зм.	Арк.	№ докум.	Підпис	Дата	Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної тривоги Структура системи (Структурна схема)	Літ.	Аркуш	Аркушів			
Розробив		Молочко І.М.									
Перевірив		Васильєва М.Д.					1	1			
Н. Контр.		Іваніщев Б.В.									
Затвердив											
						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-04					

ДОДАТОК 2

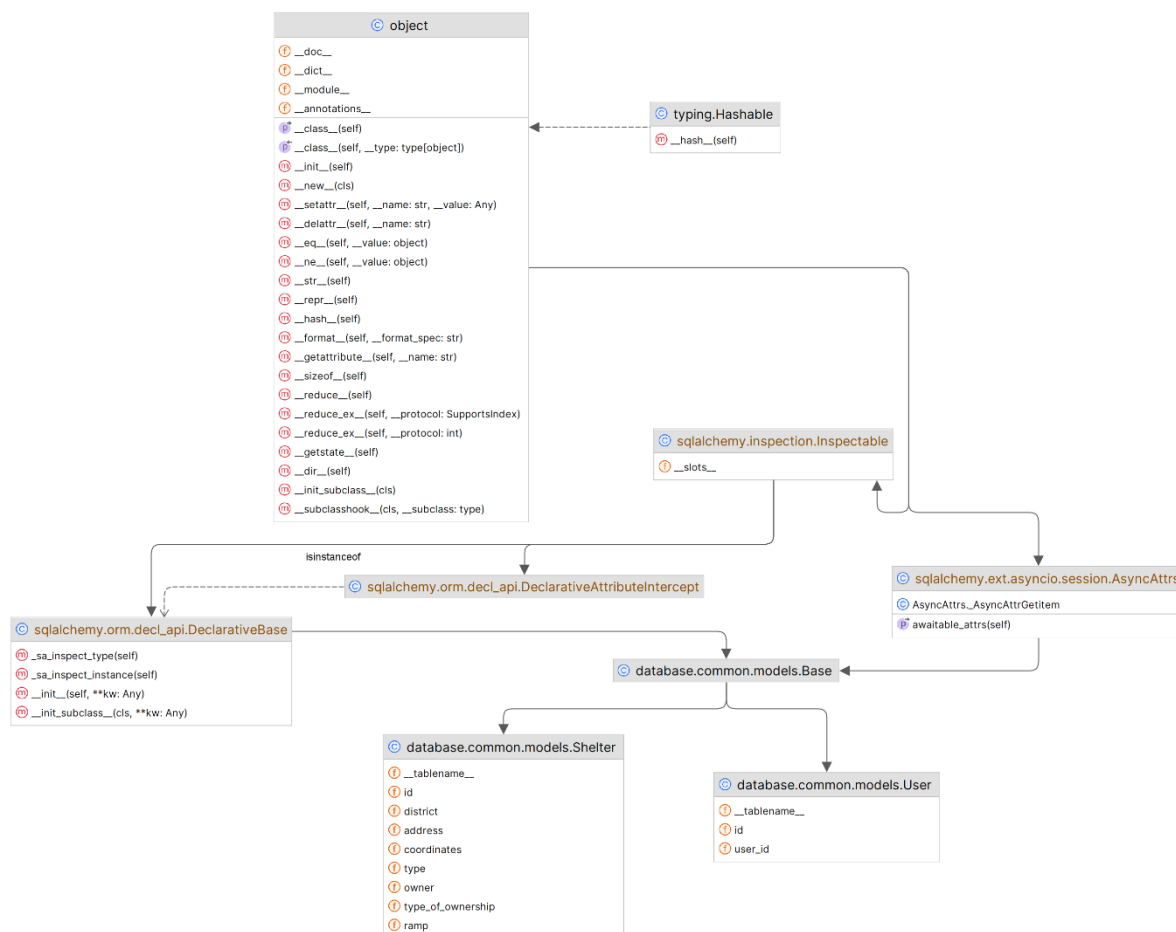
Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної
тривоги

Функціональна схема (діаграма класів)

ІАЛЩ.467200.004 Д2

Аркушів 1

Київ 2024



					ІАЛЦ.467200.004 Д2			
Зм.	Арк.	№ докум.	Підпис	Дата	Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної тривоги Функціональна схема (діаграма класів)			
Розробив	Молочко І.М.							
Перевірів	Васильєва М.Д.							
Н. Контр.	Іваніщев Б.В.							
Затвердив								
					Літ. Аркуш Аркушів 1 1 КПІ ім. Ігоря Сікорського, ФІОТ, ІО-04			

ДОДАТОК 3

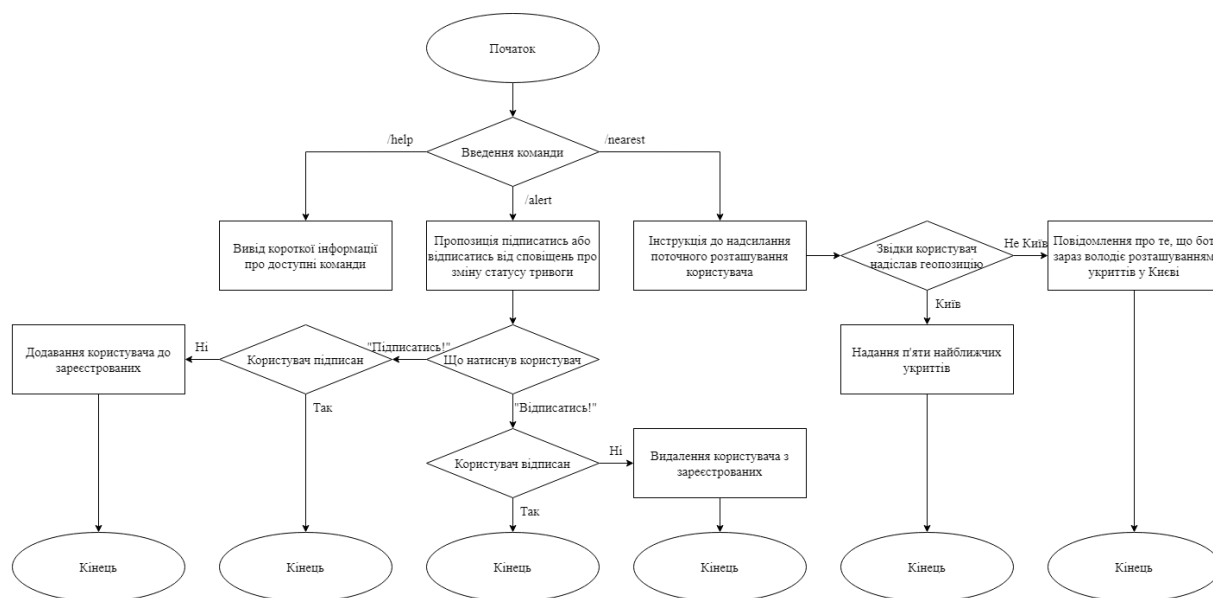
Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної
тривоги

Алгоритм дій програми (Принципова схема)

ІАЛЦ.467200.004 ДЗ

Аркушів 1

Київ 2024



					ІАЛЦ.467200.004 ДЗ						
Зм.	Арк.	№ докум.	Підпис	Дата	Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної тривоги Алгоритм дій програми (Принципова схема)	Літ.	Аркуш	Аркушів			
Розробив	Молочко І.М.										
Перевірив	Васильєва М.Д.						1	1			
						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-04					
Н. Контр.	Іваніщев Б.В.										
Затвердив											

ДОДАТОК 4

Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної
тривоги

Лістинг програмного коду

ІАЛЩ.467200.004 Д4

Аркушів 7

Київ 2024

main.py

```
import asyncio
import logging

from telegram_bot.core import bot_run

if __name__ == "__main__":
    logging.basicConfig(
        level=logging.INFO,
        format="%(asctime)s - %(levelname)s - %(module)s - %(message)s",
        handlers=[
            logging.StreamHandler(),
            logging.FileHandler("log/bot.log")
        ]
    )
    logger = logging.getLogger(__name__)
    try:
        logger.info("Bot start.")
        asyncio.run(bot_run())
    except Exception as e:
        logger.error(f"Error while bot running: {e}")
```

config.py

```
import os

from pydantic import BaseModel, SecretStr
from dotenv import load_dotenv

load_dotenv()

class Settings(BaseModel):
    telebot_token: SecretStr
    alert_api_token: SecretStr

settings = Settings(
    telebot_token=os.getenv("TELEBOT_TOKEN"),
    alert_api_token=os.getenv("ALERT_API_TOKEN"),
)
```

alarm_api/core.py

```
import aiohttp

from config import settings

MAIN_URL = "https://api.ukrainealarm.com/api/v3"
```

					ІАЛЦ.467200.004 Д4			
Зм.	Арк.	№ докум.	Підпис	Дата	Телеграм-бот для пошуку найближчих укриттів у м. Київ під час повітряної тривоги Лістинг програмного коду	Літ.	Аркуш	Аркушів
Розробив		Молочко І.М.					1	7
Перевірів		Васильєва М.Д.						
Н. Контр.		Іваніщев Б.В.				КПІ ім. Ігоря Сікорського, ФІОТ, ІО-04		
Затвердив								

```

async def region_status(region_id):
    response_url = f"{MAIN_URL}/alerts/{region_id}"
    headers = {
        "accept": "application/json",
        "Authorization": settings.alert_api_token.get_secret_value()
    }
    async with aiohttp.ClientSession() as session:
        async with session.get(response_url, headers=headers) as response:
            return await response.json()

```

database/common/models.py

```

import logging

from sqlalchemy.orm import DeclarativeBase, Mapped, mapped_column
from sqlalchemy.ext.asyncio import AsyncAttrs, async_sessionmaker,
create_async_engine
from sqlalchemy import Integer, BigInteger, String
from geoalchemy2 import Geometry
from geoalchemy2.elements import WKTElement

logger = logging.getLogger(__name__)

engine =
create_async_engine(url="postgresql+asyncpg://ignatij:qwerty12345@localhost
:5432/mydatabase")

async_session = async_sessionmaker(engine)

class Base(AsyncAttrs, DeclarativeBase):
    pass

class User(Base):
    __tablename__ = "users"

    id: Mapped[int] = mapped_column(Integer, primary_key=True)
    user_id: Mapped[int] = mapped_column(BigInteger, unique=True)

class Shelter(Base):
    __tablename__ = "shelters"

    id: Mapped[int] = mapped_column(Integer, primary_key=True)
    district: Mapped[str] = mapped_column(String)
    address: Mapped[str] = mapped_column(String)
    coordinates: Mapped[WKTElement] = mapped_column(Geometry("POINT"))
    type: Mapped[str] = mapped_column(String)
    owner: Mapped[str] = mapped_column(String)
    type_of_ownership: Mapped[str] = mapped_column(String)
    ramp: Mapped[str] = mapped_column(String)

async def async_main():
    logger.info("Create connection with database.")
    async with engine.begin() as conn:
        await conn.run_sync(Base.metadata.create_all)
    logger.info("Tables created successfully")

```

					ІАЛЦ.467200.003 Д4	Арк.
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

database/data/data_reading.py

```
import logging

import pandas as pd

logger = logging.getLogger(__name__)

def xlsx_read():
    file_path = "database/data/data.xlsx"
    try:
        df = pd.read_excel(file_path)
        columns = {
            "Район": "district",
            "Адреса": "address",
            "Координати ": "coordinates",
            "Тип": "type",
            "Власник": "owner",
            "Форма власності": "type_of_ownership",
            "Наявність пристосування для доступу осіб з інвалідністю та інших маломобільних груп населення (пандусу)": "ramp"
        }
        df = df.rename(columns=columns)
        df_records = df.to_dict(orient="records")
        logger.info(f"Read all shelters data from {file_path}")
        return df_records
    except Exception as e:
        logger.error(f"Error reading shelters data from {file_path}: {e}")
```

database/utils/database_operations.py

```
import logging
import re

from sqlalchemy import select
from geoalchemy2.elements import WKTElement
from geoalchemy2 import functions

from database.common.models import async_session, User, Shelter

logger = logging.getLogger(__name__)

async def create_user_instance(user_id: int) -> bool:
    try:
        async with async_session() as session:
            user = await session.scalar(select(User).filter(User.user_id == user_id))
            if not user:
                session.add(User(user_id=user_id))
                await session.commit()
                logger.info(f"User with id {user_id} created.")
                return True
            else:
                logger.info(f"User with id {user_id} already exists.")
                return False
    except Exception as e:
        logger.error(f"Error creating user with id {user_id}: {e}")
```

					ІАЛЦ.467200.003 Д4	Арк.
						3
Зм.	Арк.	№ докум.	Підпис	Дата		

```

async def read_user_instances() -> list:
    try:
        async with async_session() as session:
            query = await session.scalars(select(User))
            logger.info(f"Read all registered users.")
            return query
    except Exception as e:
        logger.error(f"Error reading user instances: {e}")

async def delete_user_instance(user_id: int) -> bool:
    try:
        async with async_session() as session:
            user = await session.scalar(select(User).filter(User.user_id ==
user_id))
            if user:
                await session.delete(user)
                await session.commit()
                logger.info(f"User with id {user_id} deleted.")
                return True
            else:
                logger.info(f"User with id {user_id} not found.")
                return False
    except Exception as e:
        logger.error(f"Error deleting user with id {user_id}: {e}")

async def insert_data(df_records):
    try:
        async with async_session() as session:
            async with session.begin():
                for record in df_records:
                    lat, lon = re.split(r"\s*,\s*|\s+",
record["coordinates"].strip())
                    record["coordinates"] = WKTElement(f"POINT({lon}
{lat}))")
                    instance = await session.scalar(select(Shelter).filter(
                        Shelter.district == record["district"],
                        Shelter.address == record["address"],
                        Shelter.coordinates == record["coordinates"],
                        Shelter.type == record["type"],
                        Shelter.owner == record["owner"],
                        Shelter.type_of_ownership ==
record["type_of_ownership"],
                        Shelter.ramp == record["ramp"]
                    ))
                    if not instance:
                        session.add(Shelter(**record))
                    await session.commit()
                    logger.info("Data inserted successfully.")
    except Exception as e:
        logger.error(f"Error inserting data: {e}")

async def find_nearest_shelters(user_lat, user_lon):
    try:
        async with async_session() as session:
            user_location = WKTElement(f"POINT({user_lon} {user_lat}))")

```

					ІАЛЦ.467200.003 Д4	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

```

        query = await session.scalars(

select(Shelter).order_by(functions.ST_Distance(Shelter.coordinates,
user_location)).limit(5)
        )
        logger.info(f"Found 5 nearest shelters.")
        return query
    except Exception as e:
        logger.error(f"Error finding nearest shelters: {e}")

```

telegram_bot/core.py

```

import asyncio
import logging
import re

from aiogram import Bot, Dispatcher, types, F
from aiogram.filters import CommandStart, Command
from geopy.geocoders import Nominatim
from apscheduler.schedulers.asyncio import AsyncIOScheduler
from shapely import wkb

from config import settings
from database.data.data_reading import xlsx_read
from database.common.models import async_main
from database.utils.database_operations import (
    read_user_instances,
    create_user_instance,
    delete_user_instance,
    find_nearest_shelters,
    insert_data
)
from telegram_bot.messages import (
    START_MESSAGE,
    HELP_MESSAGE,
    SHELTER_INFO,
    GEOLOCATION_SHARING,
    OUTSIDE_KYIV,
    SHELTER_FEEDBACK
)
from alarm_api.core import region_status

bot = Bot(token=settings.telebot_token.get_secret_value())
dp = Dispatcher()
geolocator = Nominatim(user_agent="kyiv_detect")
scheduler = AsyncIOScheduler()
logger = logging.getLogger(__name__)

@dp.message(CommandStart())
async def start(message: types.Message) -> None:
    await message.reply(text=START_MESSAGE)

@dp.message(Command("help"))
async def info(message: types.Message) -> None:
    await message.reply(text=HELP_MESSAGE)

```

					ІАЛЦ.467200.003 Д4	Арк.
						5
Зм.	Арк.	№ докум.	Підпис	Дата		

```

@dp.message(Command("alert"))
async def alert(message: types.Message) -> None:
    buttons = [
        [
            types.InlineKeyboardButton(text="Підписатись!",
callback_data="subscribe"),
            types.InlineKeyboardButton(text="Відписатись!",
callback_data="unsubscribe")
        ]
    ]
    keyboard = types.InlineKeyboardMarkup(inline_keyboard=buttons)
    text = ("Ви хочете підписатись чи відписатись на повідомлення про
тривоги по місту Київ?\n\n"
        "P.S. Ви завжди зможете змінити свій вибір!")
    await message.answer(text=text, reply_markup=keyboard)

async def get_location(message: types.Message) -> None:
    await message.reply(text=GEOLOCATION_SHARING,
        reply_markup=types.ReplyKeyboardMarkup(

keyboard=[[types.KeyboardButton(text="Поділитись геопозицією",
request_location=True)]],
        resize_keyboard=True))

@dp.message(Command("nearest"))
async def get_location_from_command(message: types.Message) -> None:
    await get_location(message)

@dp.message(F.location)
async def handle_location(message: types.Message) -> None:
    latitude = message.location.latitude
    longitude = message.location.longitude
    location = geolocator.reverse((latitude, longitude), language="uk")
    await message.reply(text=f"Ви зараз знаходитесь за
адресою:\n{location.address}")
    if "Київ" in re.findall(r"\b\w+\b", location.address):
        query = await find_nearest_shelters(latitude, longitude)
        for instance in query:
            geometry = wkb.loads(bytes(instance.coordinates.data))
            await bot.send_location(message.chat.id, latitude=geometry.y,
longitude=geometry.x)
            await bot.send_message(message.chat.id,
text=SHELTER_INFO.format(
                district=instance.district,
                address=instance.address,
                type=instance.type,
                owner=instance.owner,
                type_of_ownership=instance.type_of_ownership,
                ramp=instance.ramp
            ))
            await asyncio.sleep(0.05)
            await bot.send_message(message.chat.id, text=SHELTER_FEEDBACK)
    else:
        await message.reply(text=OUTSIDE_KYIV)

```

					ІАЛЦ.467200.003 Д4	Арк.
						6
Зм.	Арк.	№ докум.	Підпис	Дата		

```

previous_data = None

async def alert_status_sending() -> None:
    global previous_data
    try:
        current_data = await region_status(31)
        if current_data != previous_data:
            if previous_data:
                query = await read_user_instances()
                if not current_data[0]["activeAlerts"]:
                    for user in query:
                        await bot.send_message(user.user_id, text="⚡
Тривога закінчилась!")
                        await asyncio.sleep(0.05)
            else:
                buttons = [
                    [
                        types.InlineKeyboardButton(text="Знайти
найближче укриття!",
callback_data="nearest_shelter")
                    ]
                ]
                keyboard =
types.InlineKeyboardMarkup(inline_keyboard=buttons)
                for user in query:
                    await bot.send_message(
                        user.user_id,
                        text="⚡ Почалась тривога! Прямуйте до
найближчого укриття та бережіть себе!",
                        reply_markup=keyboard
                    )
                    await asyncio.sleep(0.05)
                previous_data = current_data
    except Exception as e:
        logger.error(f"Error while sending response to Alert API: {e}")

async def bot_run() -> None:
    await async_main()
    df_records = xlsx_read()
    await insert_data(df_records=df_records)
    scheduler.add_job(alert_status_sending, "interval", minutes=1)
    scheduler.start()
    await dp.start_polling(bot)

```

					ІАЛЦ.467200.003 Д4	Арк.
						7
Зм.	Арк.	№ докум.	Підпис	Дата		