ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРААВ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти бакалавр

на тему: **«Розроблення погодного чат-боту на мові Python»**

Виконав: здобувач вищої освіти
за освітньо професійною програмою
Інформаційні управляючі системи
спеціальності 126 Інформаційні
системи та технології
освітнього ступеня бакалавр
групи 126ІСТ_бд_2020
Гришко М.С.
Керівник: Одарущенко О.Б.
Рецензент: Брикун О.М.

Полтава – 2024 року

ВСТУП

Погода завжди мала суттєвий вплив на суспільство. Клімат, погодні умови та природні явища можуть мати різноманітний вплив, починаючи від вибору одягу на наступний день чи планування відпустки у найближчому майбутньому, і закінчуючи кількістю жертв, що можуть бути зумовлені стихійними лихами. Зміни погоди можуть також впливати на самопочуття окремої особи, спричиняючи збільшення або зменшення тиску, викликаючи алергічні реакції або маючи вплив на психічний стан. Крім того, прогноз погоди має прямий вплив на економіку. Без належного використання метеорологічної інформації в енергетиці, будівництві, авіації, судноплавстві, промисловості та сільському господарстві можуть виникнути значні збитки. Гідрометеорологічні катастрофи можуть серйозно пошкодити економіку.

Прогноз погоди стає все більш актуальною проблемою в епоху зростаючої цифрової зв'язаності і доступу до інформації через соціальні мережі та месенджери, що робить цю інформацію доступною для широкого кола людей. Створення телеграм-ботів для аналізу прогнозу погоди може відігравати ключову роль у забезпеченні громадського інтересу та підвищенні свідомості про погодні умови. Наприклад, авіація надзвичайно залежить від точності прогнозу погоди. Для авіаційної безпеки необхідні довгострокові прогнози погоди на 5, 10 днів та навіть місяць. Жоден літак не здійснює польотів без детальної інформації про погодні умови по маршруту, а також про місця призначення та запасні аеродроми. Морський та річковий транспорт також сильно залежать від погодних умов. Від штормів до туману, небезпечні погодні умови можуть призвести до аварій. Зростання популярності соціальних мереж та месенджерів, таких як Telegram, вносить свій внесок у зручний доступ до цієї інформації. Створення телеграм-бота для аналізу прогнозу погоди на основі метеорологічних даних може стати важливим кроком у забезпеченні громадського інтересу.

Актуальність теми. У сучасному інформаційному суспільстві, де погодні умови можуть суттєво впливати на наші плани та діяльність, швидкий і зручний

доступ до актуальної метеорологічної інформації стає все більш важливим. Разом з тим, месенджери, такі як Telegram, стали невід'ємною частиною нашого щоденного спілкування. Саме на перетині цих тенденцій і зародилася ідея мого проекту - створення погодного чат-боту для Telegram.

Мета роботи – є розробка погодного чат-боту на мові Python для месенджера Telegram, який надаватиме користувачам актуальну інформацію про поточні погодні умови та прогнози в заданому регіоні за допомогою зручного інтерактивного інтерфейсу.

Об'єкт дослідження кваліфікаційної роботи – методи і інструментальні засоби розроблення інтерактивного погодного чат-боту на мові Python, інтегрованого з месенджером Telegram, для більш зручного отримання погодних даних користувачами.

Предмет дослідження кваліфікаційної роботи – процес розробки погодного чат-боту на мові Python.

Методи дослідження: системний аналіз (розділи 1, 2), моделювання (розділи 2, 3).

Інформаційна база, що використовувалася: публікації в періодичних виданнях, веб-ресурси.

Практична значущість: Розроблено та впроваджено функціональний погодний чат-бот для месенджера Telegram, реалізований на мові програмування Python. Це забезпечує користувачам зручний спосіб отримання актуальної інформації про погоду та прогнози безпосередньо в популярному месенджері без необхідності встановлювати додаткові застосунки.

Робота складається зі вступу, трьох розділів, висновків та списку літератури. Обсяг роботи становить 52 сторінки, 1 таблиця, 19 рисунків, 2 додатки.

РОЗДІЛ 1
АНАЛІЗ ТА ОБРАННЯ ТЕХНОЛОГІЙ РОЗРОБКИ ТЕЛЕГРАМ БОТУ

1.1 Месенджери

Історія розвитку інтернет-сервісів для спілкування почалася з чатів - програм обміну повідомленнями з обмеженим функціоналом, які спеціалізувалися лише на обміні текстовими повідомленнями. Проте з часом і прогресом, який постійно рухається вперед, цього стало недостатньо, і на зміну прийшли відомі нам месенджери. Ці програми значно розширили свій функціонал та можливості, зробивши їх більш зручними завдяки новому, зрозумілому інтерфейсу. Проте згодом на зміну прийшли соціальні мережі, які додали можливість створювати групи з розважальним контентом, новинами та іншим. Розробники месенджерів вирішили зібрати ці можливості в месенджерах, зробивши їх досить функціональними. Останнім часом можна спостерігати, як месенджери вийшли на новий рівень популярності. Прикладами таких програм є Telegram, Viber, WhatsApp, Discord, Facebook Messenger та деякі інші.

Найпопулярнішими соціальними мережами в усьому світі станом на 2024 рік за кількістю активних користувачів, за даними сайту Statista.com (рис. 1.1), є Facebook, YouTube, WhatsApp, Facebook Messenger та Instagram [1]. Ці мережі є справжнім хабом активності мільярдів користувачів з усього світу, забезпечуючи платформу для спілкування, спільного перегляду відео, обміну фотографіями та відео, а також сприяючи взаємодії та віртуальному спілкуванню.

Ці мережі не лише об'єднують людей, а й стають важливим каналом для реклами, впливу та розповсюдження інформації. Вони створюють можливості дляспілкування, розваг та навчання, ставлячи зв'язок між людьми на новий рівень.

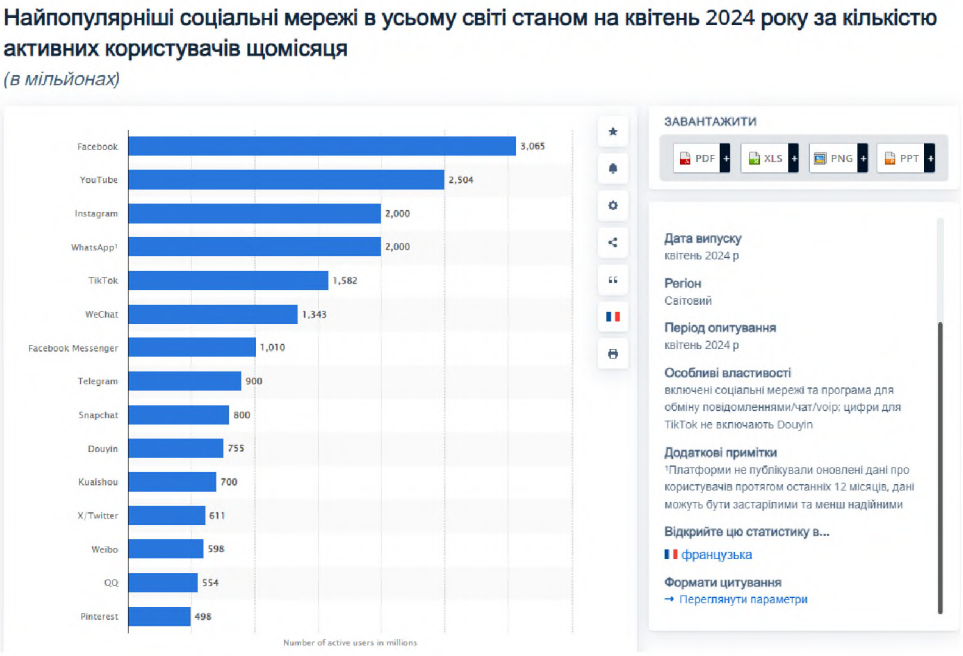

Рисунок 1.1 – Сайт statista.com

В Україні ситуація з популярністю месенджерів трохи відрізняється від глобальної. Згідно з даними компанії InMind, у 2024 році найпопулярнішими месенджерами в Україні є:

– Viber (73,6%);
– Facebook Messenger (42,7%);
– Telegram (31,6%);
– WhatsApp (25,3%);
– Skype (19,2%).

Зокрема, месенджер Telegram заслуговує особливої уваги. Це безкоштовний клауд-месенджер з підтримкою текстових повідомлень, медіафайлів та голосового спілкування. Однією з головних особливостей є його висока безпека, забезпечена протоколом MTProto (рис. 1.2), розробленим спеціально для Telegram. Також він відрізняється швидкістю доставки повідомлень, приватністю, відсутністю реклам і підписок та можливістю створення секретних чатів з шифруванням end-to-end. Однак Telegram не підтримує відеодзвінки. Протокол MTProto був спеціально розроблений для Telegram з метою найвищого рівня безпеки та захисту від взлому.

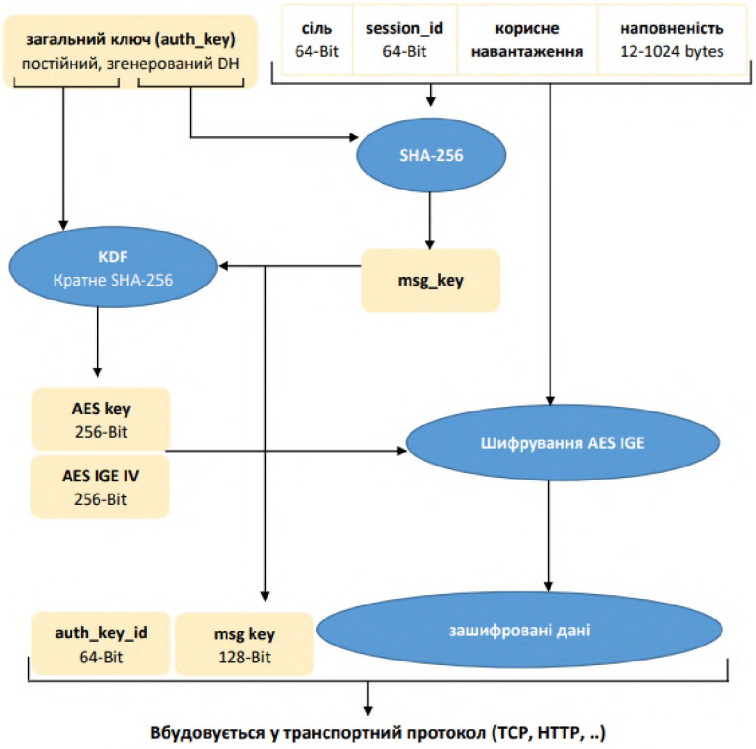

Рисунок 1.2 – Принцип роботи протоколу шифрування MTProto

Також важливо відзначити, що Telegram надає можливість користувачам створювати секретні чати, які є важливими для тих, хто прагне забезпечити конфіденційність своїх розмов. У такому типі чатування, крім основного протоколу, застосовується протокол типу end-to-end, що означає, що всі повідомлення шифруються ключами, які існують лише на двох пристроях користувачів. Тому такі повідомлення неможливо перехопити або розшифрувати, навіть співробітники Telegram не мають доступу до них. Повідомлення цього типу не можна переслати іншим користувачам, і вони не зберігаються на серверах Telegram. Проте, такий тип чатування має свої недоліки, такі як неможливість відкрити чат на іншому пристрої та відсутність захисту від знімка екрану з відкритим секретним чатом - користувач отримає лише повідомлення про те, що співрозмовник зробив знімок екрану.

Месенджер доступний на всіх відомих платформах, таких як, наприклад, Windows, Android, iOS, Linux та інші. Також є можливість використання веб версії Telegram під назвою Telegram Web. Програма підтримує всі популярні мови світу.

1.2 Вибір та порівняння сайтів, що надають API ключі

На сьогоднішній день існує значна кількість сервісів, які дозволяють отримати API ключі для доступу до різноманітних даних та послуг. Ці сервіси відрізняються у способі реалізації та наданні інформації користувачам. Наприклад, серед зарубіжних вебресурсів можна відзначити такі як OpenWeatherMap (рис. 1.3).

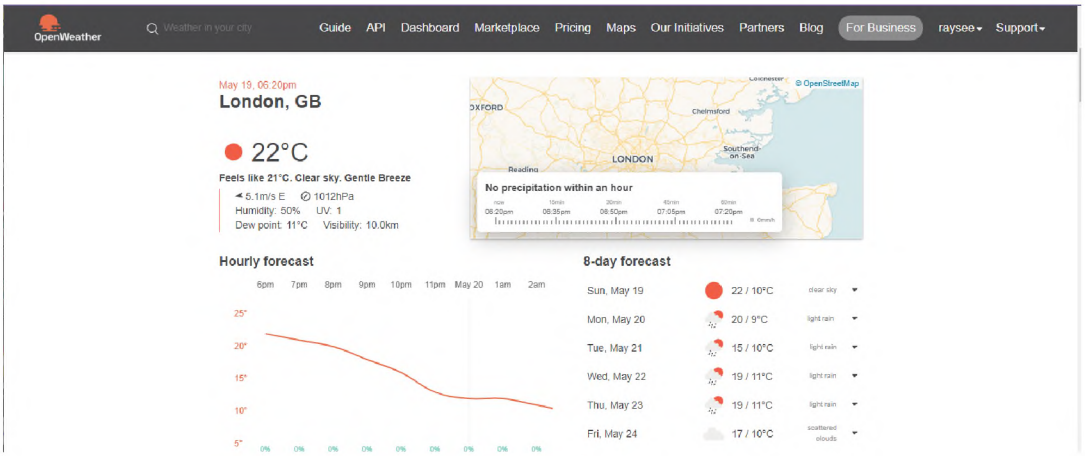

Рисунок 1.3 – Сайт OpenWeatherMap

Англомовний ресурс, який надає API для погодних даних, Weatherstack (рис. 1.4), що пропонує простий у використанні погодний API.

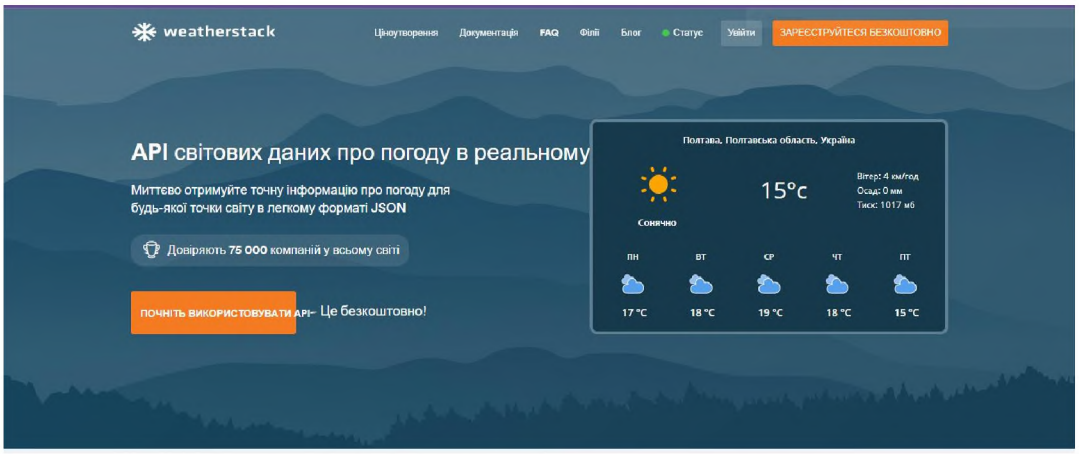

Рисунок 1.4 – Сайт Weatherstack

Weatherbit (рис. 1.5) є ще одним важливим джерелом для отримання високоякісних даних про поточну погоду. Цей сервіс надає розгорнуту інформацію про погодні умови в реальному часі, включаючи температуру, вологість, швидкість вітру, атмосферний тиск та інші метеорологічні параметри.

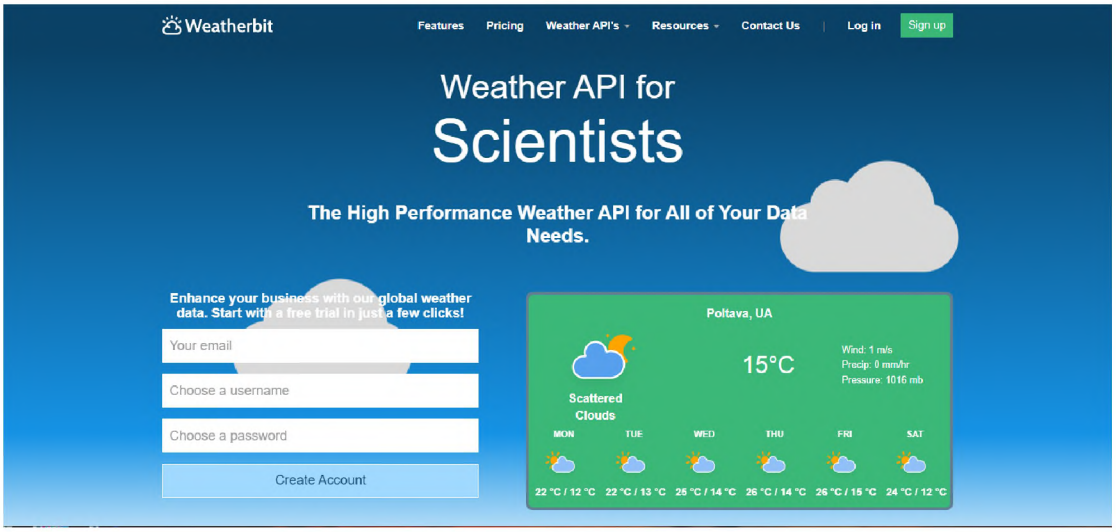

Рисунок 1.5 – Сайт Weatherbit

Також популярною є платформа, Tomorrow.io (рис. 1.6) Ці сервіси дозволяють інтегрувати погодні дані у ваші додатки та веб-сайти, пропонуючи різні плани та можливості в залежності від ваших потреб.

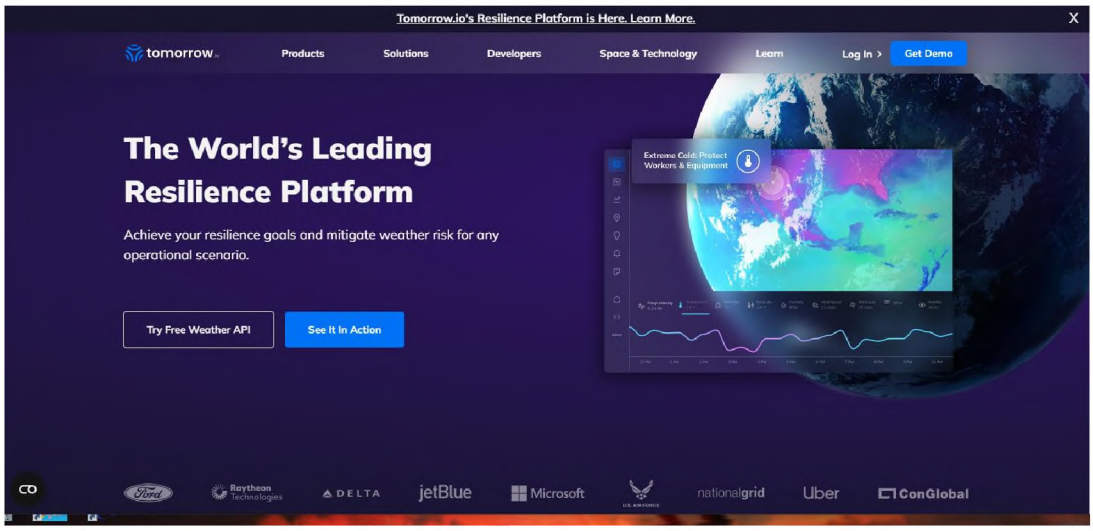

Рисунок 1.6 – Сайт Tomorrow.io

Порівняння сервісів OpenWeatherMap, Weatherstack, Weatherbit та Tomorrow.io базується на кількох ключових критеріях, таких як точність прогнозу, доступні функції, ціноутворення та зручність використання.

OpenWeatherMap переваги:

- широкий набір даних: поточна погода, прогнози, історичні дані;
- безкоштовний план з обмеженням кількості запитів;
- підтримка великої кількості мов;
- добре документоване API.

Недоліки:

- точність прогнозу може варіювати залежно від місцевості;
- в безкоштовному плані обмежені функції та кількість запитів.

Weatherstack переваги:

- легка інтеграція і швидка відповідь арі;
- безкоштовний план з основними функціями;
- детальні дані про поточну погоду, історичні дані та прогнози.

Недоліки:

- менше функцій порівняно з іншими сервісами;
- точність прогнозу може бути меншою в деяких регіонах.

Weatherbit переваги:

- високоякісні дані про поточну погоду, прогнози та історичні дані;
- розширений безкоштовний план з більшою кількістю запитів;
- інформація про якість повітря та інші метеорологічні параметри.

Недоліки:

- інтерфейс арі може бути складнішим для початківців;
- можливі обмеження на кількість запитів в безкоштовному плані.

Tomorrow.io переваги:

- високоточні дані з використанням інноваційних технологій, таких як iot;
- детальні прогнози та аналіз погоди;

– інформація про вплив погоди на різні галузі, наприклад, авіацію та транспорт.

Недоліки:

– вищі ціни порівняно з іншими сервісами;

– вільний план може бути обмеженим.

Залежно від потреб, найкращий вибір може варіюватися: OpenWeatherMap найкраще підходить для тих, хто шукає широкий набір функцій і гарну документацію при помірній точності. Weatherstack підходить для простих завдань з невеликою кількістю запитів та легкістю інтеграції. Weatherbit хороший вибір для користувачів, які потребують детальних метеорологічних даних та розширеного безкоштовного плану. Tomorrow.io найкращий для високоточних прогнозів і аналізу погоди з використанням новітніх технологій, але дорожчий.

Загалом, OpenWeatherMap забезпечує оптимальне поєднання функціональності, доступності та простоти використання, що робить його привабливим варіантом для використання.

1.3 Чат-боти

Останнім часом чат-боти стають все більш популярними в відомих месенджерах. Термін "чат-бот" був вперше використаний Майклом Маулдіном у 1994 році, коли він створив першого в світі вебробота Julia, щоб описати програми, здатні спілкуватися з людьми завдяки штучному інтелекту.

З того часу розвиток таких програм не припинився. Зараз чат-боти визначаються як комп'ютерні програми, що базуються на нейромережах і використовують машинне навчання, що дозволяє їм покращувати свої відповіді на запити користувачів. Завдяки цим функціям, чат-боти можуть спілкуватися як в аудіо, так і в текстовому форматі [2]. Оглянувши сучасні чат-боти, можна дійти висновку, що вони є універсальними інструментами для різноманітних завдань –

від розваг і спілкування до надання послуг та вирішення складних математичних задач.

Сьогодні існує багато різних чат-ботів із різним призначенням та функціональністю. Їх можна розділити на такі категорії:

- боти для моніторингу;
- боти для роботи;
- боти для навчання;
- боти для фінансів;
- музичні та книжкові боти;
- боти для шопінгу;
- боти для відпочинку.

Усі ці боти створені на основі штучного інтелекту, причому деякі з них використовують машинне навчання, що дозволяє їм надавати користувачам більш актуальну інформацію. Це особливо корисно для ботів, які ведуть діалог безпосередньо з користувачем.

Є також чат-боти, які не використовують машинне навчання. В таких програмах функції заздалегідь запрограмовані, і введення даних відбувається за допомогою тексту або кнопок управління. Ці чат-боти зазвичай мають обмежені можливості і можуть виконувати лише певні завдання в межах своєї програмованої логіки. Вони часто використовуються для простих завдань, таких як надання довідкової інформації, виконання базових команд або обробка стандартних запитів. Такі чат-боти є ефективними в ситуаціях, де вимагається стабільна і передбачувана реакція на користувацькі запити без необхідності у складному аналізі тексту чи контексту.

Ще однією перевагою чат-ботів без використання машинного навчання є їх простота і надійність. Оскільки всі можливі сценарії та відповіді заздалегідь програмуються, ці боти зазвичай працюють без збоїв і мають високу швидкість.Вони ідеально підходять для ситуацій, коли потрібна швидка реакція на стандартні запити або коли важлива кожна секунда у взаємодії з користувачем, наприклад під час обслуговування клієнтів.

1.4 Огляд реалізацій телеграм ботів

Серед існуючих реалізацій можна виділити наступні додатки:«pogodnik» (рис. 1.7) та Телеграмм бот «Погода» (рис. 1.8).

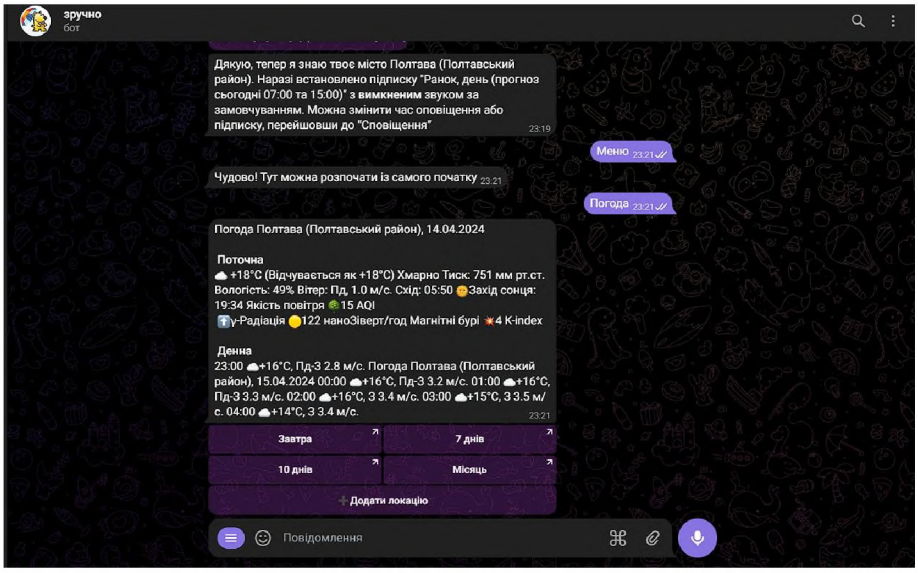

Рисунок 1.7 – Телеграм бот «pogodnik»

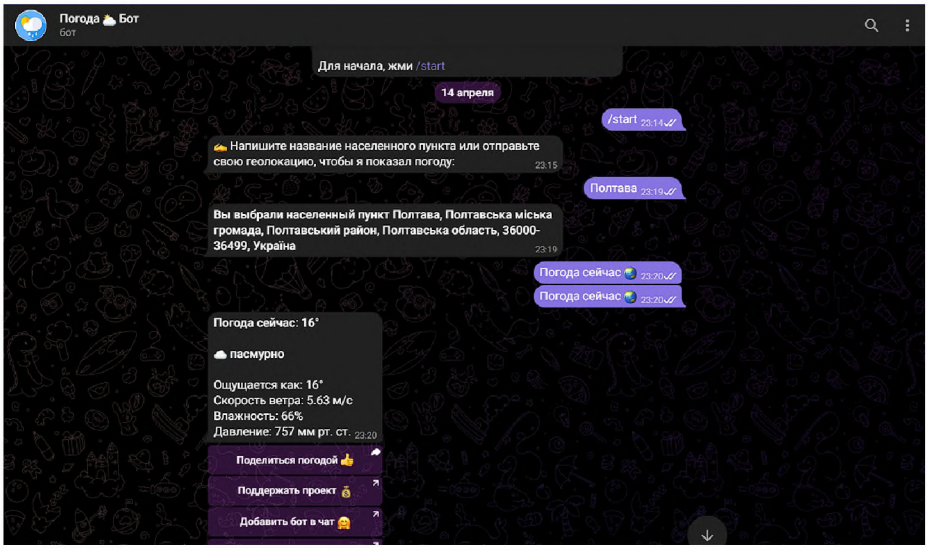

Рисунок 1.8 – Телеграмм бот «Погода»

Зазначені програми мають схожі реалізації та надають інтерфейс керування у вигляді меню, яке пристосоване до української мови. Додаток «pogodnik» додатково підтримує можливість зміни мови інтерфейсу на англійську. Серед переваг цих програм можна виділити автоматичні сповіщення, що

підтримуються Телеграм-ботом «Погода» (рис. 1.9), які є досить зручною функцією для користувача. Крім того, додаток «pogodnik» має гнучкі налаштування, що дозволяють використовувати інформери за допомогою яких можна отримати більше інформації про погоду (рис. 1.10).

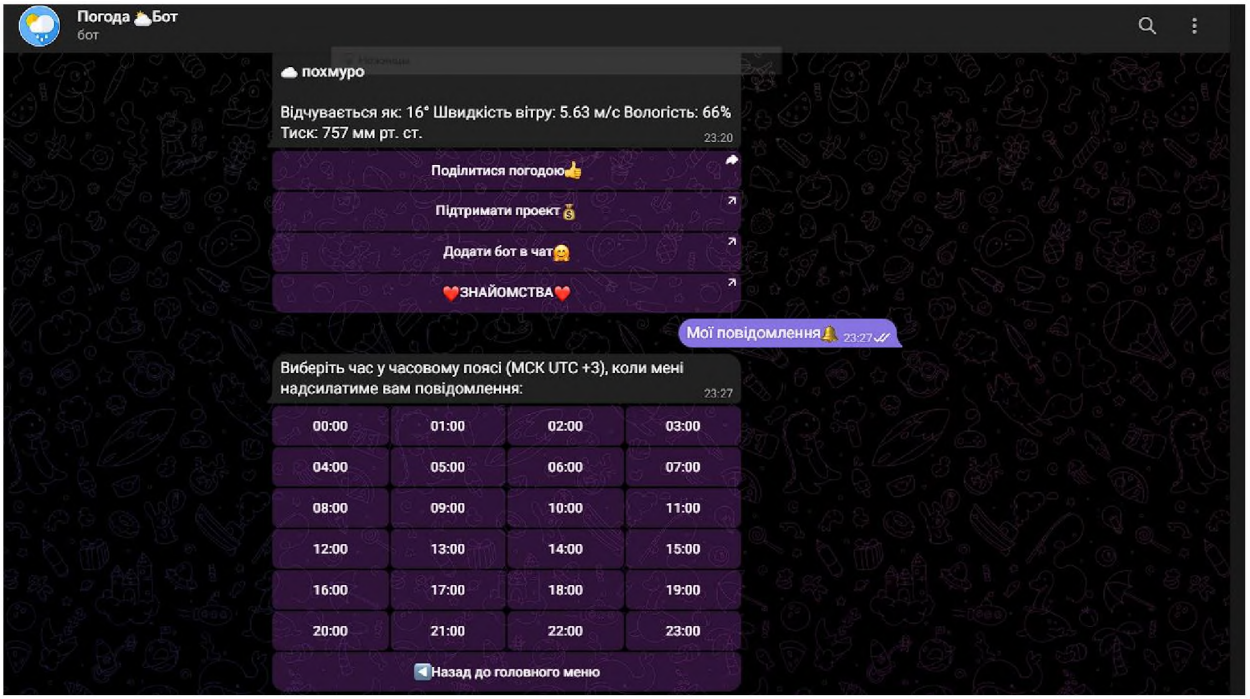

Рисунок 1.9 – Функція сповіщення Телеграмм бота «Погода»

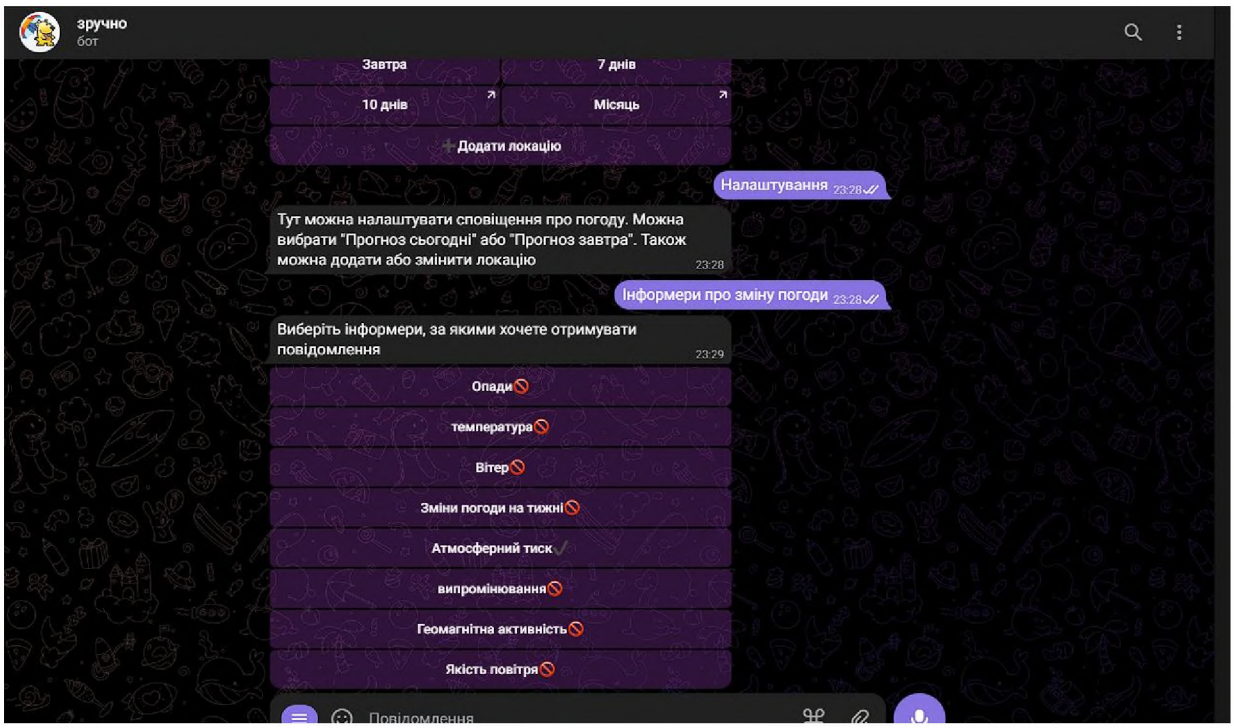

Рисунок 1.10 – Налаштування додатку «pogodnik»

Серед переваг реалізації телеграм бота " Daily Weather " можна відзначити наступне:

– доступність в реальному часі чат-бот, побудований на базі api openweathermap, надає актуальну інформацію про погоду у реальному часі, що дозволяє користувачам завжди бути в курсі останніх прогнозів;

– широкий функціонал api openweathermap надає різноманітну інформацію про погоду, таку як температура, вологість, швидкість вітру, тиск тощо. це дозволяє чат-боту надавати користувачам детальний та повний прогноз на будь-який час та місцезнаходження;

– гнучкість та налаштування api openweathermap дозволяє налаштовувати запити згідно потреб користувачів. чат-бот може адаптуватися до різних форматів запитів та надавати інформацію у зручній для користувача формі;

– надійність та точність openweathermap має репутацію надійного джерела погодної інформації. api забезпечує високу точність прогнозів, що дозволяє користувачам отримувати достовірну інформацію про погоду на потрібний час і місце;

– масштабованість openweathermap має широку географічну покриття та можливість отримання інформації про погоду в будь-якій точці світу. Це робить чат-бот погоди через API OpenWeatherMap відмінним інструментом для користувачів з різних куточків планети.

1.5 Опис задачі

Метою цієї роботи є створення телеграм-бота, який буде отримувати дані про погоду з веб-сервісу OpenWeatherMap. Вимоги до функціоналу бота включають:

– запуск бота та обробка команд /start і /help для вітання користувача і надання інструкцій;

– функція отримання погоди для введеного користувачем міста;

– використання api openweathermap для отримання даних про погоду;

– обробка отриманих даних і відправлення їх користувачеві у вигляді повідомлення.

Додатково в коді здійснюється:

– використання емодзі для опису погоди;

– виведення інформації про максимальну і мінімальну температуру на день;

– використання української мови для відображення погоди та інструкцій для користувача.

Цей бот створює зручний спосіб отримання інформації про погоду в різних містах через популярний месенджер Telegram.

У дипломній роботі будуть описані кроки відтворення кожного сценарію використання бота, включаючи пошук прогнозу погоди за назвою міста, зміну збереженого міста та отримання прогнозу на певний день.

Детально будуть розглянуті аспекти інтеграції бота з існуючими погодними сервісами, такими як OpenWeatherMap чи другими популярними джерелами даних, і розробка інтерфейсів взаємодії з користувачем через платформу Telegram.

Окрема увага буде приділена методам забезпечення точності та актуальності наданої ботом інформації про погоду. Це включає стратегії моніторингу оновлень даних, використання надійних API для отримання прогнозів та адаптацію до змін у метеорологічних умовах.

Цей бот не лише спрощує доступ до погодної інформації через Telegram, але й забезпечує можливість користувачам швидко отримувати актуальні дані безпосередньо на свої мобільні пристрої. Використання інтерактивних можливостей месенджера сприяє зручному і ефективному взаємодії з ботом, що є ключовим фактором в його популярності серед різних категорій користувачів.

РОЗДІЛ 2
ВИБІР І ОБГРУНТУВАННЯ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Обрані технології розробки

Перед тим як приступати до розробки програмного продукту, потрібно спочатку описати програмні та апаратні засоби, які будуть використовуватися для реалізації проєкту. Головна мета цього розділу полягає в обґрунтуванні засобів реалізації майбутнього Telegram-бота для інформування про погоду.

Щодо вибраних технологій розробки, Python є основною мовою програмування для цього проєкту. Python - це об'єктно-орієнтована мова програмування високого рівня, яка була розроблена Гвідо ван Россумом у 1990 році. Назва мови походить не від імені змії, а від британського комедійного серіалу "Літаючий Цирк Монті Пайтона" [3].

Python був сформований під впливом різноманітних мов програмування, таких як ABC, C, C++, Java, інші. Його переваги включають:

- відкритий код;
- зрозумілий синтаксис;
- зручність у розв'язанні математичних задач;
- наявність великої кількості модулів;
- портативність між різними операційними системами;

Python є платформонезалежною мовою програмування, що підтримується на більшості відомих операційних систем, включаючи Microsoft Windows, UNIX системи такі як Linux, iOS, MacOS, Android та інші. Навіть застарілі версії операційних систем, такі як Windows 95 і Windows 98, підтримують роботу з Python.

Спільнота Windows ME продовжує активну підтримку Python, починаючи з версії Python 2.3 і старіших. Це дозволяє розробникам використовувати Python на різних платформах із збереженням сумісності зі старішими операційними системами.

У мові програмування Python можна виділити такі можливості, які часто відсутні в інших мовах програмування:

– інтерактивний режим, коли користувач може вводити вирази з клавіатури, що виконуються та виводяться результати негайно;

– об'єктно-орієнтоване програмування, що є визначальною особливістю мови python і реалізоване елегантно, потужно і добре продумано, але може відрізнятися від інших мов програмування;

– функціональне програмування;

– модулі та пакети, що дозволяють створювати програми у вигляді модулів та збирати їх у пакети, а також використовувати пакети, написані на інших мовах програмування.

Інтроспекція, що дозволяє отримати всю інформацію про внутрішню структуру будь-якого об'єкта:

– обробка винятків;

– ітератори;

– генератори;

– декоратори.

Однією з переваг мови програмування Python є багата стандартна бібліотека, яка містить модулі для написання HTTP-серверів. У Python також існує розширена база графічних бібліотек, які придатні для роботи з різними типами графічних даних. Навіть існують бібліотеки, спеціалізовані на графіці для ігор. Проте, як і в інших мовах, у Python є певні недоліки:

– низька швидкодія;

– відсутність статичної типізації;

– неможливість модифікації вбудованих класів.

Глобальне блокування інтерпретатора. Обрано Python для цього проєкту через його простоту, багатий функціонал та наявність великої кількості корисних модулів, що дозволяють ефективно вирішувати різноманітні завдання, пов'язані з обробкою даних, автоматизацією процесів та інтеграцією з іншими системами.

Python відомий своєю активною та великою спільнотою розробників, що створює відкриті бібліотеки та фреймворки для різноманітних завдань. Наприклад, для реалізації Telegram-бота для інформування про погоду можна використати бібліотеки, такі як python-telegram-bot, які спрощують роботу з API Telegram та полегшують розробку функціоналу бота.

Крім того, Python має багато інструментів для тестування коду, що допомагає забезпечити надійність та якість програмного продукту. Можна використати бібліотеки, такі як pytest, для написання автоматизованих тестів і перевірки правильності роботи бота.

Також, Python є популярним вибором для обробки даних та машинного навчання. Це означає, що в майбутньому бот може бути покращений за допомогою алгоритмів прогнозування погоди на основі історичних даних та сучасних методів машинного навчання.

Загалом, обрання Python для розробки Telegram-бота для інформування про погоду виправдовується його гнучкістю, простотою та широким функціоналом, що сприяє швидкій і якісній розробці програмного продукту.

2.2 Бібліотеки для розробки

Бібліотека datetime у Python надає класи для обробки дат і часу. Вона дозволяє виконувати операції з датами і часом, форматувати їх, обчислювати різницю між датами, а також працювати з часовими зонами. Бібліотека datetime є важливим інструментом для роботи з часом у Python, забезпечуючи широкі можливості для різноманітних завдань, пов'язаних з обробкою дат і часу. Ось основний функціонал, який надає бібліотека datetime:

1. Класи бібліотеки datetime:

```
date: для роботи з датами (рік, місяць, день).
time: для роботи з часом (година, хвилина, секунда, мікросекунда).
datetime: для роботи з датами та часом одночасно.
timedelta: для обчислення різниці між датами або часом.
```

```
tzinfo: для роботи з часовими зонами.
```

2. Створення об'єктів за допомогою конструкторів класів наприклад:

```
from datetime import date, time, datetime, timedelta

my_date = date(2024, 5, 23)
my_time = time(14, 30, 0)

my_datetime = datetime(2024, 5, 23, 14, 30, 0)

my_timedelta = timedelta(days=5, hours=3)
```

3. Операції з об'єктами. Додавання та віднімання об'єктів timedelta до об'єктів date та datetime:

```
new_date = my_date + my_timedelta
new_datetime = my_datetime - my_timedelta
```

4. Обчислення різниці між датами:

```
delta = my_date - date(2024, 5, 18)
```

5. Форматування за допомогою методів strftime та strptime:

```
formatted_date = my_datetime.strftime('%Y-%m-%d %H:%M:%S')
parsed_datetime = datetime.strptime('2024-05-23 14:30:00', '%Y-%m-%d %H:%M:%S')
```

6. Отримання поточної дати та часу:

```
now = datetime.now()
today = date.today()
```

7. Підтримка часових зон через бібліотеку pytz:

```
from datetime import datetime
import pytz

utc = pytz.utc
local_tz = pytz.timezone('Europe/Kiev')

utc_dt = datetime.now(utc)
local_dt = utc_dt.astimezone(local_tz)
```

У коді Telegram-бота використовується бібліотека datetime для форматування часу сходу і заходу сонця та обчислення тривалості дня:

```
from datetime import datetime
sunrise_timestamp = data['city']['sunrise']
sunset_timestamp = data['city']['sunset']
sunrise = datetime.fromtimestamp(sunrise_timestamp).strftime('%H:%M:%S')
```

```
sunset = datetime.fromtimestamp(sunset_timestamp).strftime('%H:%M:%S')
day_length = datetime.fromtimestamp(sunset_timestamp - sunrise_timestamp).strftime('%H:%M:%S')
```

Ми використовуємо datetime.fromtimestamp для перетворення Unix-часу (кількість секунд з 1 січня 1970 року) на об'єкт datetime. Потім застосовуємо strftime для форматування часу в зрозумілий людині формат година, хвилина, секунда. Це дозволяє зручно відображати час сходу і заходу сонця, а також тривалість дня в повідомленнях Telegram-бота.

Бібліотека Requests у Python - це простий, але потужний інструмент для виконання HTTP-запитів. Вона забезпечує зручний і інтуїтивно зрозумілий інтерфейс для взаємодії з вебсерверами та отримання даних з Інтернету. Ось кілька ключових функцій та можливостей бібліотеки Requests:

1. Виконання HTTP-запитів, здійснення GET, POST, PUT, DELETE запитів до веб-серверів.
2. Налаштування параметрів запитів, встановлення заголовків, параметрів запиту, файлів для відправки, аутентифікації.
3. Отримання відповіді від сервера HTML-сторінок, JSON-даних, зображень.
4. Робота з сесіями, підтримка сесій для збереження кукісів, автентифікації та інших параметрів між запитами.
5. Обробка статус кодів, автоматична обробка статус кодів HTTP-відповідей (наприклад, 200 для успішного запиту, 404 для відсутності сторінки тощо).
6. Обробка помилок, які можуть виникнути під час виконання запитів (наприклад, помилки з'єднання, часові витрати тощо).
7. Скачування файлів з Інтернету за URL-адресою.
8. Встановлення таймаутів для запитів для уникнення блокування.
9. SSL-підтримка, requests автоматично встановлює безпечне з'єднання (SSL) з веб-серверами за допомогою TLS-шифрування для забезпечення безпеки під час передачі даних через Інтернет.

10. Параметри URL-адреси requests дозволяє легко робити запити з параметрами URL-адреси, вказуючи їх у вигляді словника або рядка параметрів.

11. Керування редиректами, бібліотека може автоматично слідувати за редиректами від веб-сервера або дозволяти контролювати цей процес вручну.

12. Постійність з'єднання requests може зберігати відкрите з'єднання з сервером для кількох запитів за один раз, що дозволяє підвищити швидкість взаємодії з сервером.

13. Збереження cookie бібліотека підтримує автоматичне зберігання та використання cookie для автентифікації та збереження стану сесії між запитами.

14. Proxy-сервери з requests можна встановлювати з'єднання через HTTP або SOCKS проксі-сервери для анонімного або безпечного доступу до веб-серверів.

15. Бібліотека підтримує кешування HTTP-відповідей для збереження ресурсів та підвищення швидкодії, зокрема під час великої кількості повторних запитів до одного ресурсу.

Ці функції роблять бібліотеку Requests дуже корисним інструментом для взаємодії з веб-серверами та отримання різних типів даних з Інтернету. Вона широко використовується в різних сферах програмування, від веб-розробки до наукових досліджень та автоматизації завдань.

Бібліотека telebot є інструментом для створення Telegram-ботів у Python. Вона надає зручний і простий спосіб інтеграції з API Telegram, дозволяючи легко налаштовувати та керувати ботами. Ось деякі ключові функції та можливості бібліотеки telebot:

1. Створення та запуск бота Створення об'єкту бота за допомогою класу TeleBot і передача йому токену вашого бота. Запуск бота за допомогою методу polling або webhook.

2. Обробка повідомлень визначення обробників для різних типів повідомлень (текстові повідомлення, команди, фотографії тощо) за допомогою декораторів.

3. Відправлення повідомлень та використання методів для відправлення текстових повідомлень, медіафайлів, стікерів, аудіофайлів тощо.

4. Відповіді на повідомлення користувачів, відправлення текстових повідомлень у відповідь на отримані повідомлення від користувачів.

5. Робота з клавіатурою, створення власних клавіатур для взаємодії з користувачем (клавіатура з кнопками, inline-клавіатура тощо).

6. Взаємодія з API Telegram, можливість виконання різних запитів до API Telegram для отримання додаткової інформації або зміни налаштувань бота.

7. Робота з розсилкою, відправлення розсилок повідомлень користувачам.

8. Обробка помилок, які можуть виникнути під час виконання різних операцій (наприклад, невірний запит до API або втрата з'єднання).

Бібліотека telebot надає зручний та потужний інтерфейс для створення різноманітних Telegram-ботів у Python. Її простота використання та багатий функціонал дозволяють швидко розробляти та розширювати функціональність ботів [7].

2.3 BotFather

BotFather, це бот, розроблений командою Telegram, призначений для управління іншими ботами. Основні функції BotFather включають управління ботами, прив'язаними до облікового запису, реєстрацію нових ботів, їх оновлення та налаштування. За допомогою BotFather можна зареєструвати безліч нових ботів, за умови, що їхні імена будуть унікальними. Також за допомогою цього інструмента можна налаштувати зовнішній вигляд чат-бота, змінити головне зображення, опис чи налаштувати його команди (рис. 2.8) [5].

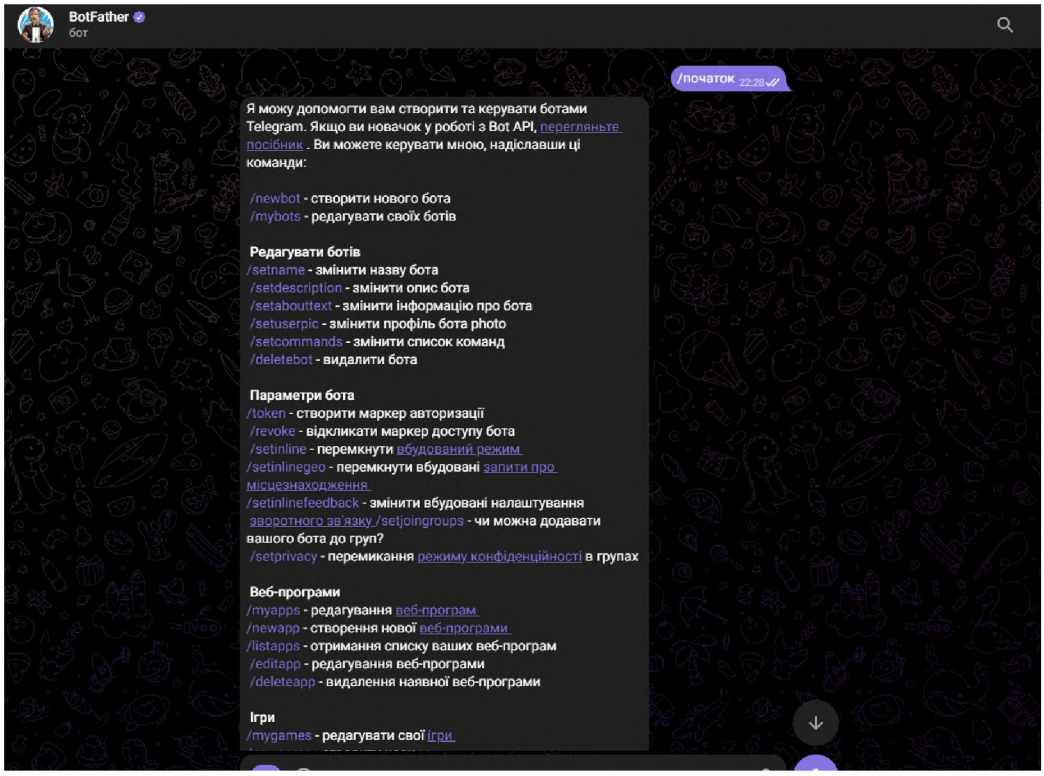

Рисунок 2.8 – Команди налаштування бота

Створити бота можливо за 3 дії:

1. Вигадати ім’я бота, яким він буде названий, також це ім’я буде відображатися в чатах та контактах.

2. Вигадати ім’я користувача, головною умовою якого буде унікальність цього імені. В іншому випадку BotFather не дозволить створити бота і висуне прохання змінити ім’я. Допускаються всі букви та цифри латинського алфавіту і символ підкреслення.

3. Якщо все добре, буде наданий токен, за допомогою якого можна повністю управляти ботом. Окрім того, з допомогою цієї програми можна створювати ігри в боті, редагувати їх, та налаштовувати.

2.4 Telegram Bot API

Telegram використовує протокол шифрування MTProto, який також відомий як Telegram API. Цей протокол дозволяє здійснювати зв'язок з сервером.

Для розробки ботів було створено Telegram Bot API, яке спрощує процес написання ботів, звільняючи програміста від необхідності знати всі деталі роботи протоколу MTProto. Зв'язок з сервером відбувається через простий HTTPS-інтерфейс, який надає просту версію Telegram API [6].

Існують два способи отримання оновлень від бота:

– webhook: сервери telegram сповіщаються про наявність будь-якихоновлень;

– long polling: сервери Telegram отримують запити на перевірку наявності оновлень від програми.

При створенні кожному боту надається унікальний токен, який зазвичай має вигляд, 7099601327:AAGROr2MEVthdcczkvDtcBtqViZ2R_WNKm0. Цей токен потрібний для ідентифікації бота в Telegram.

Після отримання токену можна використовувати Telegram Bot API для відправлення запитів. У Telegram Bot API допускаються POST та GET запити. Для передачі параметрів існують чотири способи:

– application/x-www-form-urlencoded;

– application/json (не використовується для завантаження файлів);

– multipart/form-data (використовується саме для завантаження файлів);

– URL-запит.

Відповідь на запит повертається у форматі JSON, де зазвичай містяться два поля: bool-поле ok та string-поле description. Якщо значення поля ok рівне true, то результат запиту буде описаний в полі description. Дані в цьому полі зазвичай легко читаються. Результат виконання запиту міститься у полі result. У випадку, якщо значення булевого поля дорівнює false, це означає, що при запиті сталася помилка, а опис помилки міститься у полі description. Крім того, у полі error_code зазвичай міститься код помилки.

Приклади доступних методів API включають:

– getUpdates – отримання оновлень;

– setWebhook – прив'язка URL домену, де працює бот;

– sendMessage – відправлення повідомлення;

– sendLocation – відправлення координат;

– getFile – отримання завантаженого файлу.

Більшість ботів використовують стандартну конфігурацію, яка зберігається на серверах Telegram. Однак, якщо потрібно розширити функціонал, завжди можна перейти на власну конфігурацію.

Для отримання оновлень можна використовувати метод getUpdates, який забезпечує доступ до вхідних оновлень шляхом довготривалого опитування.

Використовуючи метод setWebhook, можна отримувати оновлення через веб-хук, який адресується за вказаною URL-адресою. Коли з'являються оновлення бота, Telegram надсилає POST-запит, в якому міститься JSON-файл з оновленнями. У випадку невдалих спроб надіслання оновлень, Telegram припиняє спроби після певної кількості.

Для видалення веб-хуку використовується метод deleteWebhook. Після цього можна повернутись до отримання оновлень за допомогою методу getUpdates.

Всі типи представлені у вигляді JSON-об'єктів і використовуються в відповідях API бота. Ось деякі з найбільш важливих типів:

– user – визначає користувача або бота в telegram;

– chat – позначає чат;

– message – відповідає за повідомлення;

– messageid – ідентифікатор повідомлення, представлений у вигляді цілого числа.

– animation – відповідає за анімації, такі як gif або h.264/mpeg-4 avc відео без звуку;

– audio – аудіофайл, також відомий як музика;

– voice – представляє голосові повідомлення;

– video – цей тип визначає відеофайл;

– videonote – представляє відео повідомлення;

– contact – відповідає за телефонний контакт, тобто, номер телефону;

– location – відповідальний за передачу даних про місце знаходження особи.

API бота має підтримку форматування тексту, включаючи жирний шрифт, курсив, підкреслення та закреслений текст. Також можна налаштувати форматування посилань. В Telegram API Bot є інші методи, які підтримують GET та POST запити і не враховують регістр. Деякі з них:

– getMe – перевіряє бота на автентичність;
– logOut – відключає бота від сервера Bot API та запускає його локально;
– close – потрібен для безпечного переміщення бота між серверами;
– sendMessage – надсилає текстові повідомлення;
– forwardMessage – пересилає будь-які типи повідомлень;
– sendPhoto – відправляє фотографії;
– sendAudio – передає аудіофайли;
– sendDocument – надсилає будь-які файли;
– sendVideo – відправляє відеофайли;
– sendAnimation – надсилає анімації;
– sendVoice – передає аудіофайли як голосові повідомлення;
– sendLocation – відправляє точки на карті;
– sendContact – передає телефонні дані.

Бот також може працювати у вбудованому режимі, доступному в будь-якому чаті за допомогою відмітки бота, наприклад: @gif, @sticker, @vid та інші (рис. 2.9, 2.10).

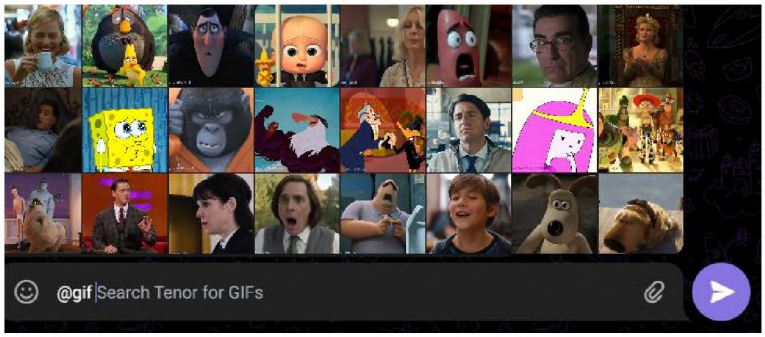

Рисунок 2.9 – Приклад виклику та вміст бота @gif

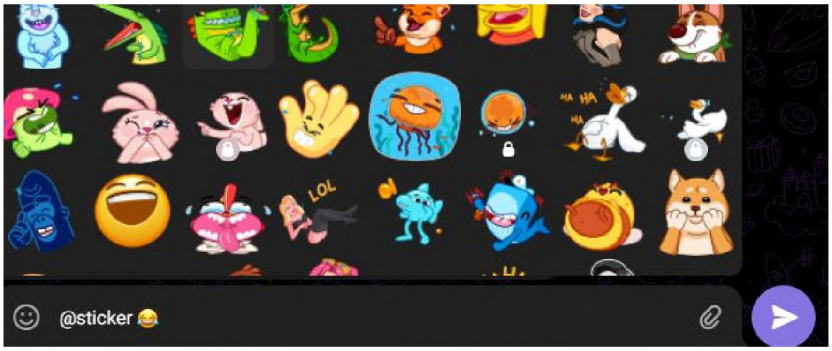

Рисунок 2.10 – Приклад виклику та вміст бота @sticker

Цей режим дозволяє використовувати бота для швидкого доступу до певних функцій чи сервісів прямо в чаті без необхідності переходити в окремий додаток або інтерфейс. Наприклад, користувачі можуть викликати бота для отримання гіфок, стікерів або відео безпосередньо у своїх повідомленнях, що робить взаємодію з контентом більш зручною та швидкою.

2.5 Середовище розробки

PyCharm – це інтегроване середовище розробки для Python. Це середовище розробки надає графічний налагоджувач, засоби для аналізу коду, інструмент для запуску юніт-тестів (рис. 2.11). PyCharm розроблена компанією JetBrains на основі IntelliJ IDEA. Це середовище підтримується на MacOS X, Windows, Linux [4].

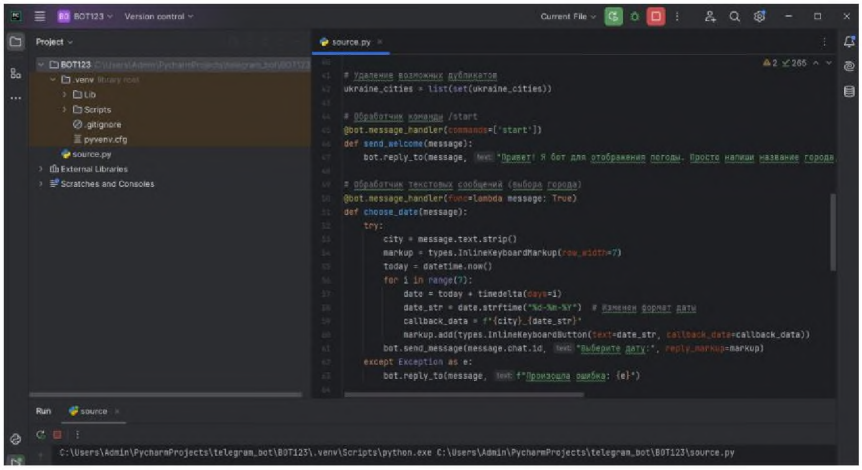

Рисунок 2.11 – Приклад інтерфейсу середовища PyCharm

Основні можливості середовища PyCharm:

1. Інтелектуальне автодоповнення пайчарм надає розширену функціональність автодоповнення, що допомагає швидко завершувати код, пропонуючи варіанти методів, атрибутів та інших об'єктів, які доступні у поточному контексті.

2. Аналіз коду інтегровані інструменти аналізу коду допомагають виявляти можливі помилки та підказувати оптимізації в коді. Це допомагає забезпечити якість коду та покращує його ефективність.

3. Інтеграція з іншими інструментами пайчарм легко інтегрується з іншими інструментами розробки, такими як системи контролю версій (наприклад, Git), віртуальні середовища (наприклад, virtualenv) та інші.

4. Налагодження та тестування: PyCharm має потужний налагоджувач та засоби для тестування, що дозволяють відстежувати виконання коду, знаходити та виправляти помилки, а також запускати тести для перевірки працездатності коду.

5. Рефакторинг коду: PyCharm надає потужні інструменти для рефакторингу, що дозволяють швидко та безпечно змінювати структуру коду. Це включає перейменування змінних, методів, класів, а також витягування методів та інші операції, які спрощують підтримку та розвиток коду.

6. Підтримка різних мов програмування: Окрім Python, PyCharm підтримує багато інших мов програмування, таких як JavaScript, HTML, CSS, SQL та інші. Це робить його зручним інструментом для розробки веб-додатків та інших багатомовних проектів.

7. Інструменти для роботи з базами даних: Вбудований клієнт баз даних у PyCharm дозволяє легко підключатися до різних баз даних, виконувати SQL-запити, переглядати структуру баз даних та редагувати дані безпосередньо з середовища розробки.

8. Підтримка рефакторингу коду: Інструменти рефакторингу у PyCharm дозволяють автоматизувати процеси покращення структури коду, включаючи перейменування змінних, екстракцію функцій, оптимізацію імпортів та інше.

РОЗДІЛ 3

ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Отримання токену та API ключа

Кожен новий бот вимагає процедури реєстрації в системі та отримання унікального токену. Тому потрібно скористатися сервісом під назвою BotFather (рис. 3.1).

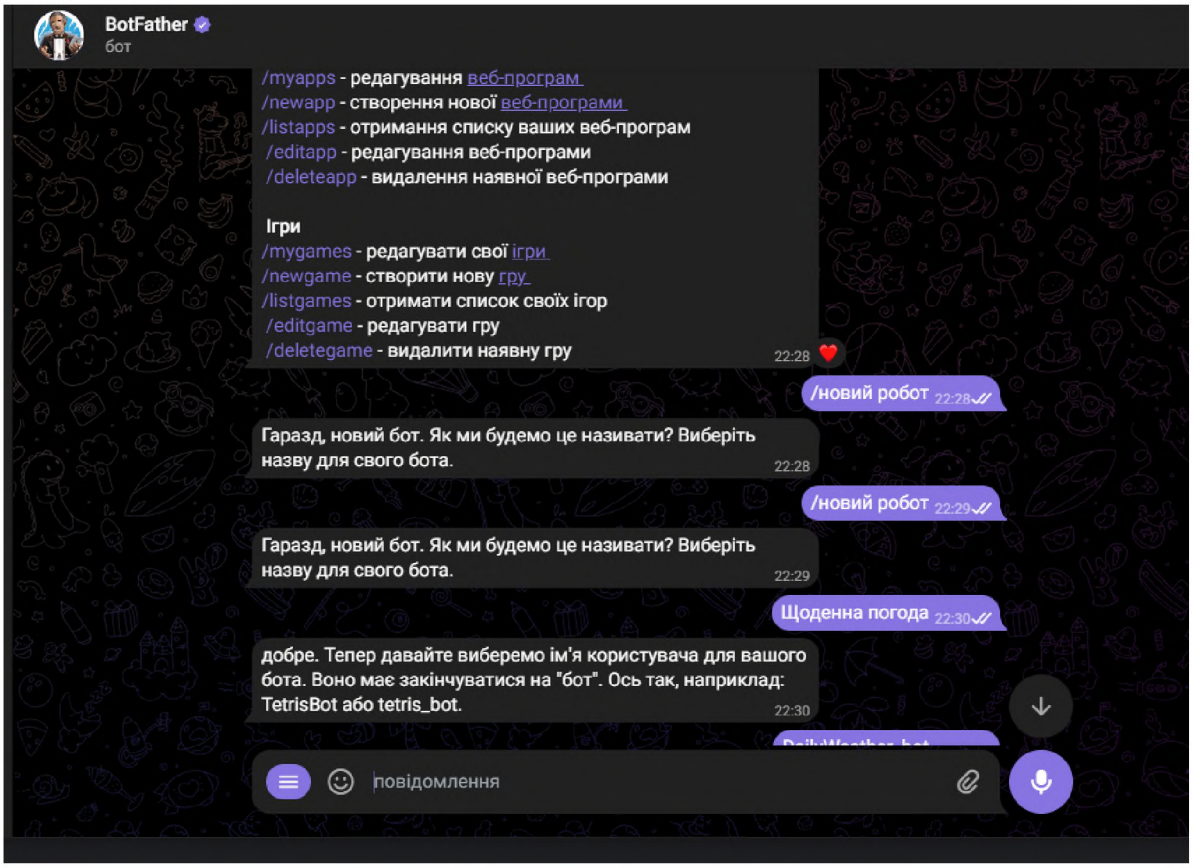

Рисунок 3.1 – Бот BotFather

Вводячи команду /newbot отримуємо вказівки з назви бота, та його ідентифікатору. Головною умовою є те що, ідентифікатор має бути унікальним. Але якщо не дотриматися цієї умови, програма не дозволить створити існуючий ідентифікатор і сповістить про це користувача. У випадку, коли ідентифікатор є унікальним, бот дозволить створення і присвоїть новому боту унікальний токен, який надає доступ до повного управління ботом (рис. 3.2).

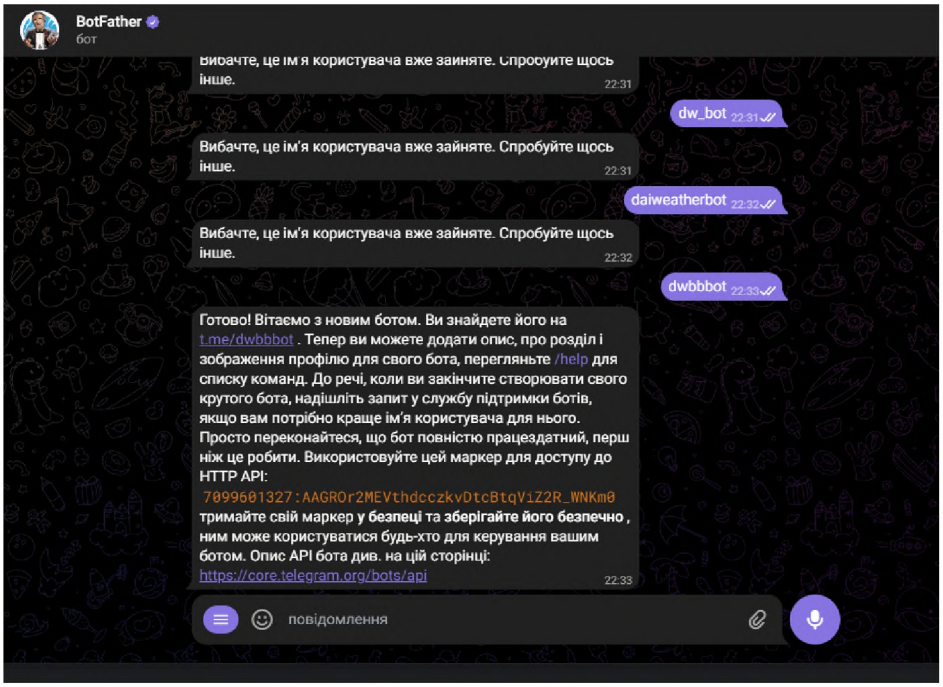

Рисунок 3.2 – Присвоєння унікального токену

Після цього, можна завершити роботу з цим ботом або налаштувати дизайн бота – його опис, фото профілю, список команд та інше.

Для того щоб продовжити створення погодного бота, потрібно отримати API ключ. На цю роль було обрано сайт OpenWeather.org (рис. 3.3).

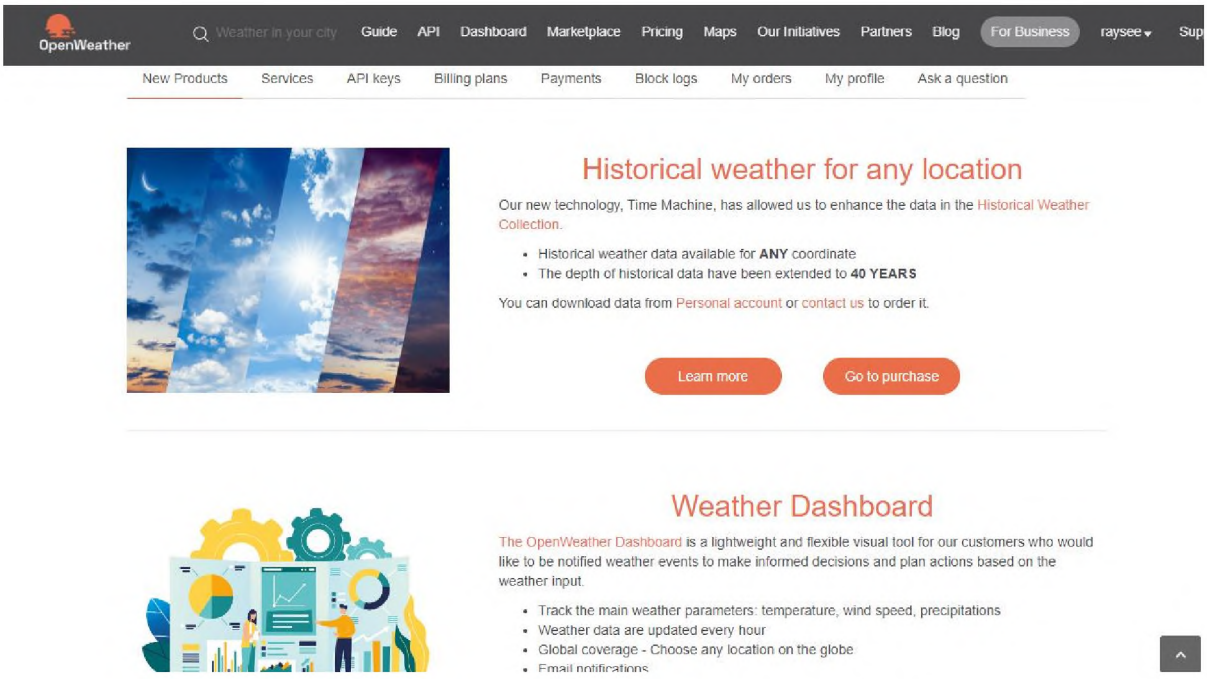

Рисунок 3.3 – Інтерфейс сайту OpenWeather

Щоб отримати API ключ для доступу до даних про погоду, які знаходяться на серверах цього сервісу потрібно зареєструватися та обрати тарифний план який влаштовує (рис. 3.4).

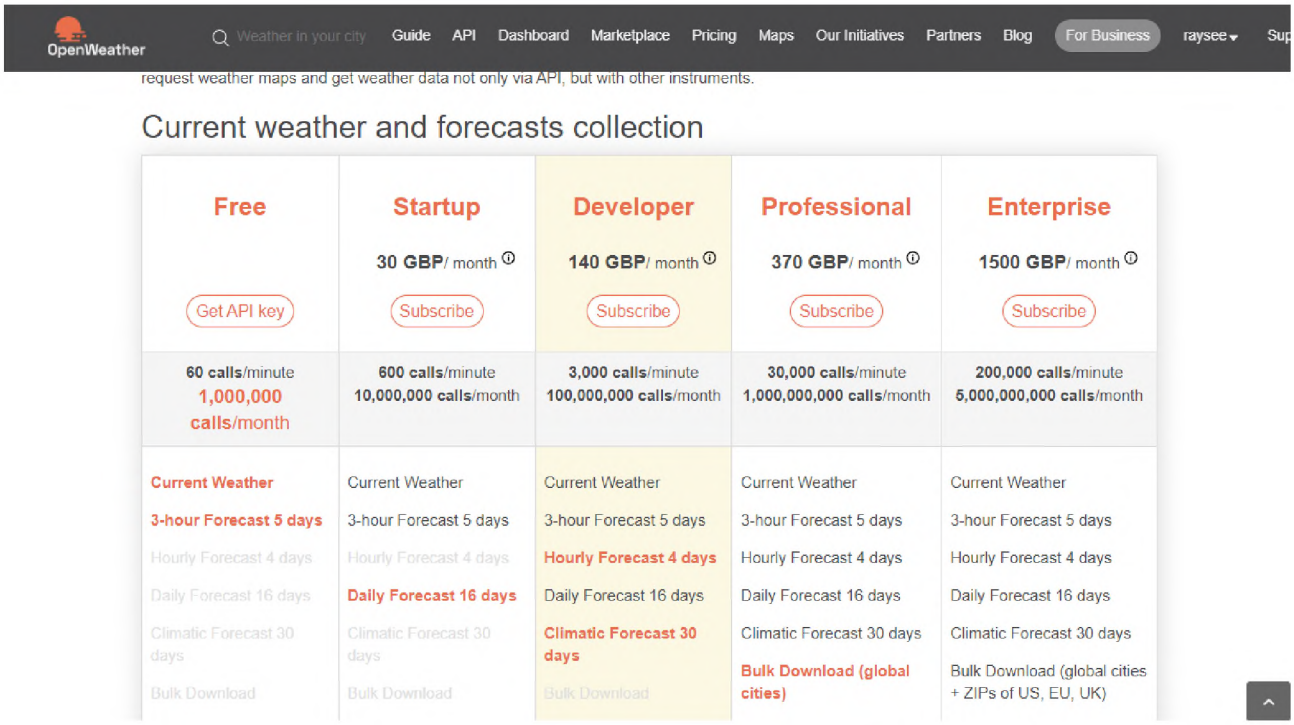

Рисунок 3.4 – Тарифні плани OpenWeather

Після обрання тарифного плану, ваш API ключ буде доступний у особистому кабінеті та активується протягом двох годин. Цей ключ має приблизно такий формат: 79a89c63d529523f1471c89225918fc6. На веб-сайті можна знайти документацію, яка містить необхідну інформацію для використання API сайту. У цьому документі основною функцією є API call (рис. 3.5), яка забезпечує можливість здійснювати запити до сайту для отримання інформації про погодні умови. Запит може бути виконаний лише за умови, що задано принаймні місто та обов'язково API ключ.

Документація також містить приклади коду для різних мов програмування, що дозволяє легко інтегрувати API у ваші додатки. Важливо дотримуватися обмежень тарифного плану щодо кількості запитів, щоб уникнути блокування доступу до API. Це забезпечує надійну роботу вашого додатку і зменшує ризик можливих проблем з доступом до погодних даних.

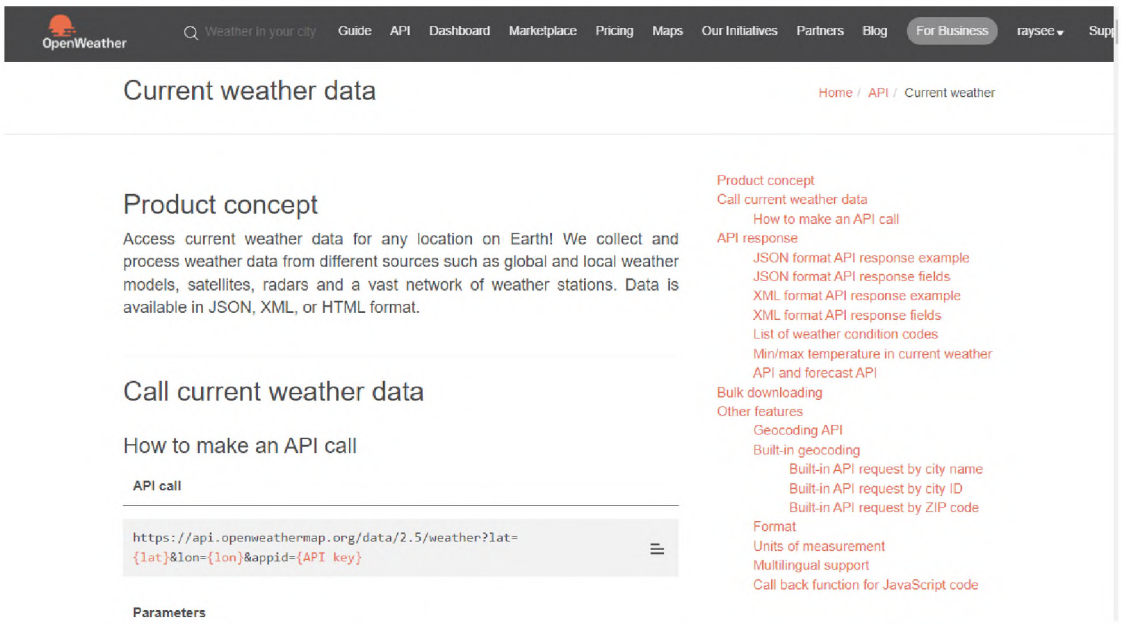

Рисунок 3.5 – Функція API call

На додачу до прикладів коду та рекомендацій щодо обмежень тарифного плану, документація API зазвичай містить інформацію про можливість отримання ключів доступу (API keys), які необхідно використовувати для аутентифікації при кожному запиті до сервісу. Це забезпечує безпеку та контроль за використанням API, дозволяючи проводити моніторинг та управління доступом до сервісу погоди з вашого додатку.

3.2 Реалізація Telegram-бота

Для того, щоб розпочати роботу з Telegram-ботом, необхідно виконати низку підготовчих кроків та імпортувати необхідні бібліотеки. Ця частина коду забезпечує правильну ініціалізацію бота та доступ до потрібних функцій та ресурсів.

Спочатку імпортуються необхідні бібліотеки Python:

```
pythonCopy codeimport requests
import telebot
from telebot import types
from datetime import datetime
```

requests - потужна бібліотека для виконання HTTP-запитів та взаємодії з різними веб-сервісами та API. У нашому випадку вона буде використовуватися для отримання даних про погоду з OpenWeatherMap API.

telebot - офіційна бібліотека pyTelegramBotAPI, яка надає зручний інтерфейс для розробки Telegram-ботів та взаємодії з Telegram Bot API.

types - модуль з бібліотеки telebot, що містить різні типи об'єктів, такі як ReplyKeyboardMarkup для створення кнопок та InlineKeyboardMarkup для вбудованих кнопок у повідомленнях.

datetime - стандартний модуль Python для роботи з датами та часом. Він буде використовуватися для форматування та обробки часових даних, отриманих від API погоди, наприклад, часу сходу/заходу сонця.

Для взаємодії з Telegram Bot API необхідно створити екземпляр класу TeleBot з використанням отриманого токену:

```
pythonCopy codebot = telebot.TeleBot('7099601327:AAGROr2MEVthdcczkvDtcBtqViZ2R_WNKm0')
```

Цей токен є унікальним ідентифікатором бота, який був згенерований під час його створення за допомогою BotFather у Telegram. Він використовується для авторизації та надання доступу до функцій Telegram Bot API.

Для отримання даних про погоду буде використовуватися OpenWeatherMap API - популярний сервіс для отримання метеорологічних даних. Для доступу до цього API потрібно задати API-ключ:

pythonCopy codeAPI_KEY = '79a89c63d529523f1471c89225918fc6'

Цей ключ є унікальним ідентифікатором, який було згенеровано при реєстрації на OpenWeatherMap. Він буде використовуватися для автентифікації під час відправки запитів до OpenWeatherMap API для отримання даних про погоду.

Після завершення цих підготовчих кроків бот ініціалізований та готовий до подальшої реалізації функціоналу. У наступних пунктах буде описано обробку вхідних повідомлень від користувачів, взаємодію з API погоди та формування відповідей.

Розглянемо допоміжну функцію get_weather_emoji(description), яка використовується для вибору відповідного емоджі на основі текстового опису погоди. Ця функція додає візуалізацію до повідомлень про погоду та робить їх більш наочними та зрозумілими для користувачів.

```
pythonCopy codedef get_weather_emoji(description):
    emojis = {
        'ясно': '☀',
        'малохмарно': '⛅',
        'хмарно': '☘',
        'уривчасті хмари': '☁',
        'легкий дощ': '🌦',
        'дощ': '🌧',
        'сильний дощ': '⛈',
        'гроза': '⛈',
        'легкий сніг': '❄',
        'сніг': '❄',
        'туман': '🌫',
    }
    return emojis.get(description.lower(), '')
```

Розглянемо деталі реалізації цієї функції:

1. Функція приймає один аргумент description, який є рядком з текстовим описом погоди, отриманим від OpenWeatherMap API.

2. Всередині функції визначається словник emojis, який містить відповідність між текстовими описами погоди та емоджі, що їх візуалізують. Ключами словника є рядки з описами погоди, а значеннями - відповідні емоджі у форматі Unicode.

3. Далі функція перетворює отриманий опис погоди description у нижній регістр за допомогою методу .lower() для того, щоб пошук емоджі був нечутливим до регістру.

4. Для отримання відповідного емоджі використовується метод .get(key, default) словника emojis. Якщо в словнику є ключ, що відповідає опису погоди, повертається відповідне значення (емоджі). Якщо ж такого ключа немає, повертається порожній рядок '' за замовчуванням.

Ця функція викликається в іншій частині коду, де формується повідомлення про погоду для користувача. Отриманий емоджі додається до тексту повідомлення, роблячи його більш наочним та зрозумілим.

Наприклад, якщо опис погоди від API є «хмарно», функція get_weather_emoji(«хмарно») поверне емоджі, який потім буде включений до текстового повідомлення про погоду.

Такий підхід дозволяє легко розширювати список підтримуваних описів погоди та відповідних емоджі, просто додаючи нові пари до словника emojis. Це робить код більш гнучким та адаптивним до змін у форматі даних, що надходять від API.

У цій частині коду реалізовані спеціальні функції-обробники (handlers), які відповідають за обробку різних команд та повідомлень, що надходять від користувачів Telegram-бота. Вони забезпечують інтерактивну взаємодію з користувачем, обробку введених ним даних та відповідну реакцію бота.

Обробник команд /start та /help:

```
pythonCopy code@bot.message_handler(commands=['start', 'help'])
def send_welcome(message):
    markup = types.ReplyKeyboardMarkup(one_time_keyboard=True, resize_keyboard=True)
    markup.add('Старт')
    bot.send_message(message.chat.id,
                     "Привіт! Я бот для відображення погоди. Просто надішліть мені назву міста або натисніть кнопку 'Старт'.",
                     reply_markup=markup)
```

Ця функція send_welcome(message) викликається, коли користувач вводить команду /start або /help в Telegram-боті. Декоратор @bot.message_handler(commands=['start', 'help']) забезпечує, що ця функція буде викликатися лише для цих двох команд.

Всередині функції виконуються наступні дії:

1. Створюється об'єкт types.ReplyKeyboardMarkup для формування клавіатури з кнопками у відповіді. Параметри one_time_keyboard=True та resize_keyboard=True забезпечують, що клавіатура буде показана лише один раз і підлаштовуватиметься під розмір екрану пристрою користувача.

2. До клавіатури додається кнопка "Старт" за допомогою методу markup.add('Старт').

3. Бот відправляє користувачеві повідомлення з вітанням та інструкцією, як взаємодіяти з ботом. Це повідомлення супроводжується створеною клавіатурою за допомогою параметра reply_markup=markup.

Таким чином, при вводі користувачем команд /start або /help бот відправляє йому вітальне повідомлення і пропонує натиснути кнопку "Старт" або відразу ввести назву міста для отримання прогнозу погоди.

Обробник для кнопки "Старт":

```
pythonCopy code@bot.message_handler(func=lambda message: message.text == 'Старт')
    def start_weather_request(message):
        bot.send_message(message.chat.id, "Введіть назву міста для отримання прогнозу погоди:")
```

Ця функція start_weather_request(message) викликається, коли користувач натискає кнопку "Старт" на клавіатурі. Декоратор @bot.message_handler(func=lambda message: message.text == 'Старт') визначає, що ця функція буде викликатися лише для повідомлень, які містять текст "Старт".

Всередині функції бот просто відправляє користувачеві повідомлення з проханням ввести назву міста для отримання прогнозу погоди.

Обробник для введення назви міста:

```
pythonCopy code@bot.message_handler(func=lambda message: message.text != 'Старт')
    def get_weather(message):
```

Функція get_weather(message) є основним обробником, який викликається, коли користувач вводить будь-який текст, крім "Старт". Декоратор @bot.message_handler(func=lambda message: message.text != 'Старт') забезпечує, що ця функція буде викликатися для всіх повідомлень, які не містять текст "Старт".

Всередині цієї функції міститься логіка для формування HTTP-запиту до OpenWeatherMap API, отримання даних про погоду та формування відповіді для користувача. Детальний опис цієї функції розглядається в наступних пунктах.

Використання декораторів @bot.message_handler з різними умовами дозволяє гнучко визначати обробники для різних типів вхідних повідомлень та команд від користувачів. Наприклад, можна додати обробники для голосових повідомлень, зображень, локацій тощо, задаючи відповідні умови в декораторі.

Загалом, ця частина коду забезпечує початкову взаємодію з користувачем, відображення вітального повідомлення та клавіатури з кнопкою "Старт", а також отримання від користувача необхідних даних (назви міста) для подальшої обробки та отримання прогнозу погоди.

Важливо зазначити, що обробники викликаються у тому порядку, в якому вони визначені в коді. Тому їх порядок має значення, оскільки при співпадінні умов буде викликано перший відповідний обробник.

Як було зазначено раніше, ця частина коду відповідає за взаємодію з API сервісу OpenWeatherMap для отримання даних про погоду для введеного користувачем міста. Вона реалізована у функції get_weather(message).

pythonCopy code@bot.message_handler(func=lambda message: message.text != 'Старт')

```
def get_weather(message):
    city = message.text
    params = {
        'q': city,
        'appid': API_KEY,
        'units': 'metric',
        'lang': 'uk'
    }

    weather_url = 'http://api.openweathermap.org/data/2.5/forecast'

    try:
        response = requests.get(weather_url, params=params)
        data = response.json()

        if data['cod'] == '200':
            # Код для обробки отриманих даних про погоду
            # ...
        else:
```

```
                bot.send_message(message.chat.id, "Вибачте, не можу знайти інформацію про погоду для даного міста.")
        except Exception as e:
            bot.send_message(message.chat.id, "Сталася помилка при отриманні даних про погоду.")
```

Розглянемо детальніше процес взаємодії з OpenWeatherMap API:

Назва міста зберігається у змінній city з повідомлення, отриманого від користувача

```
pythonCopy codecity = message.text
```

Це значення буде використовуватися для формування запиту до API погоди.

Формування параметрів запиту

Створюється словник params, який містить необхідні параметри для запиту до API.

```
pythonCopy codeparams = {
    'q': city,
    'appid': API_KEY,
    'units': 'metric',
    'lang': 'uk'
}
```

Ключами є назви параметрів, а значеннями - відповідні дані. Розглянемо детальніше ці параметри:

1. q: назва міста, для якого потрібно отримати дані про погоду. Це значення отримується з повідомлення користувача.

2. appid: API-ключ для авторизації та доступу до OpenWeatherMap API. Зазвичай цей ключ зберігається як окрема змінна або завантажується з файлу конфігурації.

3. units: одиниці виміру температури та швидкості вітру. У даному випадку, використовується метрична система.

4. lang: мова, якою будуть повернені дані про погоду. У цьому випадку, українська.

Зберігається URL-адреса, на яку буде відправлено запит для отримання прогнозу погоди:

```
pythonCopy codeweather_url = 'http://api.openweathermap.org/data/2.5/forecast'
```

Ця URL-адреса відповідає точці входу для отримання прогнозу погоди з OpenWeatherMap API.

За допомогою бібліотеки requests виконується GET-запит до OpenWeatherMap API з використанням попередньо сформованих параметрів params.

```
pythonCopy coderesponse = requests.get(weather_url, params=params)
```

Відповідь від API зберігається у змінній response. Далі, відповідь перетворюється у формат JSON за допомогою методу response.json() і зберігається у змінній data для подальшої обробки.

```
pythonCopy codedata = response.json()
```

Перевіряється код відповіді від API:

```
pythonCopy codeif data['cod'] == '200':
```

Якщо код 200, що означає успішний запит, виконується подальша обробка отриманих даних про погоду. Інакше, боту надсилається повідомлення про неможливість знайти інформацію про погоду для введеного міста.

```
pythonCopy codeelse:
    bot.send_message(message.chat.id, "Вибачте, не можу знайти інформацію про погоду для даного міста.")
```

У блоці try/except відловлюються будь-які виключення, що можуть виникнути під час виконання HTTP-запиту або обробки відповіді від API.

```
pythonCopy codeexcept Exception as e:
    bot.send_message(message.chat.id, "Сталася помилка при отриманні даних про погоду.")
```

У разі виникнення виключення, користувачеві надсилається повідомлення про помилку при отриманні даних про погоду.

Ця частина коду відповідає за коректне формування запиту до OpenWeatherMap API, відправку цього запиту, отримання та первинну обробку відповіді від API. Якщо запит був успішним, отримані дані про погоду передаються для подальшої детальної обробки, яка буде розглянута в наступних пунктах.

Важливо зазначити, що для коректної роботи цього коду необхідно мати дійсний API-ключ для OpenWeatherMap API. Також використовується бібліотека

requests для виконання HTTP-запитів, тому вона має бути встановлена в робочому середовищі Python.

Крім того, OpenWeatherMap API має певні обмеження на кількість запитів, що можуть бути виконані за певний період часу. Тому в реальних додатках рекомендується впровадити механізми кешування та обмеження частоти запитів, щоб не перевищувати ліміти API.

Після успішного отримання даних про погоду від OpenWeatherMap API, наступним кроком є формування текстового повідомлення з цими даними та відправка його користувачеві. Ця частина реалізована в тому ж блоці коду функції `get_weather(message)`:

```
pythonCopy codeif data['cod'] == '200':
    description = data['list'][0]['weather'][0]['description']
    temperature = data['list'][0]['main']['temp']
    temp_min = min([item['main']['temp_min'] for item in data['list'][:8]])
    temp_max = max([item['main']['temp_max'] for item in data['list'][:8]])
    humidity = data['list'][0]['main']['humidity']
    wind_speed = data['list'][0]['wind']['speed']
    pressure = data['list'][0]['main']['pressure'] * 0.75006
    clouds = data['list'][0]['clouds']['all']
    sunrise_timestamp = data['city']['sunrise']
    sunset_timestamp = data['city']['sunset']
    sunrise = datetime.fromtimestamp(sunrise_timestamp).strftime('%H:%M:%S')
    sunset = datetime.fromtimestamp(sunset_timestamp).strftime('%H:%M:%S')
    day_length = datetime.fromtimestamp(sunset_timestamp -
sunrise_timestamp).strftime('%H:%M:%S')
    emoji = get_weather_emoji(description)

    response_text = (f'🏙 Погода у місті {city}:\n'
                     f'📋 Опис: {description.capitalize()} {emoji}\n'
                     f'🌡 Поточна температура: {temperature}°C\n'
                     f'🔻 Мінімальна температура на сьогодні: {temp_min}°C\n'
                     f'🔺 Максимальна температура на сьогодні: {temp_max}°C\n'
                     f'💧 Вологість: {humidity}%\n'
                     f'💨 Швидкість вітру: {wind_speed} м/с\n'
                     f'🌡 Тиск: {pressure:.2f} мм рт. ст.\n'
                     f'☁ Хмарність: {clouds}%\n'
```

```
            f'🌅 Схід сонця: {sunrise}\n'
            f'🌇 Захід сонця: {sunset}\n'
            f'🕒 Тривалість дня: {day_length}')

    bot.send_message(message.chat.id, response_text)
```

Витягування даних про погоду з відповіді API

З відповіді від OpenWeatherMap API витягуються необхідні дані про погоду, такі як опис, температура, вологість, швидкість вітру, тиск, хмарність, час сходу/заходу сонця тощо. Ці дані зберігаються у відповідних змінних.

Виконуються певні обчислення для отримання додаткових корисних даних:

1. temp_min та temp_max - мінімальна та максимальна температура на поточний день, обчислені з прогнозу на 8 днів вперед.
2. pressure - перетворення значення тиску з гектопаскалів (гПа) у міліметри ртутного стовпчика (мм рт. ст.).
3. sunrise, sunset та day_length - час сходу та заходу сонця, а також тривалість дня, обчислені на основі отриманих з API Unix-часових міток.

Виклик функції get_weather_emoji(description) для отримання відповідного емоджі на основі текстового опису погоди. Це робить повідомлення більш наочним та зрозумілим.

Формується текстове повідомлення response_text з використанням отриманих даних про погоду та емоджі. Це повідомлення складається з рядків, що містять назву міста, опис погоди, температуру, вологість, швидкість вітру, тиск, хмарність, час сходу/заходу сонця та тривалість дня. Використовуються спеціальні символи Unicode для відображення емоджі та градусів Цельсія.

Сформоване текстове повідомлення response_text відправляється користувачеві за допомогою методу bot.send_message(message.chat.id, response_text). Тут message.chat.id - унікальний ідентифікатор чату з користувачем, отриманий з об'єкту message.

Ця частина коду є кульмінацією всього процесу отримання даних про погоду. Вона забезпечує витягування необхідної інформації з відповіді від API,

виконання додаткових обчислень, формування зрозумілого для користувача текстового повідомлення та відправку його назад у Telegram-чат.

Повідомлення містить детальну інформацію про поточну погоду у зазначеному місті, включаючи опис, температуру, вологість, швидкість вітру, тиск, хмарність, час сходу/заходу сонця та тривалість дня. Використання емоджі та зрозумілих одиниць вимірювання робить повідомлення більш наочним та зручним для сприйняття.

Хоча основна функціональність бота полягає в отриманні та відображенні даних про погоду, важливо також передбачити належну обробку помилок та виключних ситуацій, які можуть виникнути під час роботи. Це забезпечує стабільність та надійність бота, а також покращує досвід користувача.

```
pythonCopy codeif data['cod'] == '200':
else:
    bot.send_message(message.chat.id, "Вибачте, не можу знайти інформацію про погоду для даного міста.")
```

У цьому фрагменті коду обробляється ситуація, коли OpenWeatherMap API повертає код відповіді, відмінний від 200 (який вказує на успішний запит). Це може статися, наприклад, якщо користувач ввів некоректну назву міста або виникли проблеми з доступом до API.

У такому випадку бот відправляє користувачеві повідомлення про неможливість знайти інформацію про погоду для введеного міста. Це дозволяє уникнути некоректної поведінки бота та інформує користувача про те, що сталося.

```
pythonCopy codeexcept Exception as e:
    bot.send_message(message.chat.id, "Сталася помилка при отриманні даних про погоду.")
```

Цей блок коду відловлює будь-які виключення, що можуть виникнути під час виконання HTTP-запиту або обробки відповіді від API. Це може статися, наприклад, якщо API тимчасово недоступне, виникли проблеми з мережею або сталася непередбачена помилка в коді.

У разі виникнення виключення користувачеві надсилається загальне повідомлення про помилку при отриманні даних про погоду. Хоча це

повідомлення не містить деталей помилки (що є доцільним з міркувань безпеки), воно інформує користувача про те, що сталося, і дозволяє уникнути зависання бота.

Крім обробки помилок, пов'язаних з API, важливо також передбачити належну обробку інших виключних ситуацій, які можуть виникнути під час роботи бота. Наприклад, обробку некоректних команд або повідомлень від користувача, які не відповідають очікуваному формату.

Для цього можна додати додаткові обробники повідомлень, які будуть відловлювати такі ситуації та надсилати користувачеві відповідні повідомлення з інструкціями або проханням ввести коректні дані.

Ще одним важливим аспектом є ведення журналів (логування) помилок та виключних ситуацій. Це дозволяє відстежувати проблеми, які виникають під час роботи бота, та полегшує їх діагностику та усунення.

Для логування можна використовувати вбудований модуль logging в Python або сторонні бібліотеки, такі як loguru. Журнали можна записувати у файли або відправляти на спеціальний сервіс для централізованого моніторингу та аналізу.

Ось приклад використання модуля logging для реєстрації помилок:

```
pythonCopy codeimport logging

logging.basicConfig(filename='bot.log', level=logging.ERROR)

try:
except Exception as e:
    logging.error(f"Сталася помилка: {e}")
    bot.send_message(message.chat.id, "Сталася помилка при отриманні даних про погоду.")
```

У цьому прикладі налаштовується логування помилок у файл bot.log. Якщо виникає виключення, воно реєструється у журналі за допомогою функції logging.error(), а користувачеві надсилається загальне повідомлення про помилку.

Належна обробка помилок та виключних ситуацій, а також ведення журналів є важливими аспектами при розробці будь-якого додатку, включно з

Telegram-ботами. Це підвищує стабільність та надійність додатку, покращує досвід користувача та полегшує діагностику та вирішення проблем.

Для запуску бота та початку опитування вхідних повідомлень від користувачів викликається метод bot.polling(). Цей метод є частиною бібліотеки pyTelegramBotAPI і забезпечує постійний зв'язок з серверами Telegram для отримання оновлень та повідомлень від користувачів.

```
pythonCopy codebot.polling()
```

Метод polling() запускає безкінечний цикл, під час якого бот постійно перевіряє наявність нових повідомлень і обробляє їх відповідно до реалізованої логіки обробників. У нашому випадку, коли користувач надсилає команду /start або /help, викликається функція send_welcome(message), яка відображає вітальне повідомлення та пропонує ввести назву міста. При введенні назви міста спрацьовує функція get_weather(message), яка виконує запит до OpenWeatherMap API та відправляє користувачеві прогноз погоди.

Зазвичай, виклик bot.polling() розміщується в частині коду, що виконується при безпосередньому запуску скрипту, а не при імпорті як модуль. Це забезпечується за допомогою перевірки, яка гарантує, що цикл опитування запуститься тільки при безпосередньому виконанні скрипту, а не при імпорті в інший Python-модуль.

```
pythonCopy codeif __name__ == "__main__":
    bot.polling()
```

Під час роботи циклу опитування, бот буде очікувати на вхідні повідомлення від користувачів та обробляти їх відповідно до реалізованої логіки. Цикл опитування продовжується до тих пір, поки бот не буде зупинений або не виникне критична помилка.

Для зупинки бота потрібно перервати виконання скрипту. Найпростіший спосіб зробити це - натиснути комбінацію клавіш Ctrl+C у командному рядку, де виконується скрипт. Це призведе до генерування виключення KeyboardInterrupt, яке зупинить виконання циклу опитування та завершить роботу бота.

Також можна реалізувати більш складну логіку завершення роботи бота, наприклад, обробляти певні команди користувача (наприклад, /stop) або події

(наприклад, натискання певної кнопки). У цьому випадку потрібно додати відповідні обробники команд або подій і викликати метод bot.stop_polling() для коректного завершення циклу опитування.

Крім того, важливо пам'ятати про коректне звільнення ресурсів та закриття з'єднань після завершення роботи бота. Це можна зробити за допомогою конструкцій try/finally або використовуючи контекстні менеджери для гарантованого виконання коду звільнення ресурсів незалежно від того, чи виникли якісь виключення під час роботи бота.

Таким чином, виклик bot.polling() є ключовим моментом у запуску та безперервній роботі Telegram-бота, забезпечуючи постійний зв'язок з серверами Telegram та обробку вхідних повідомлень від користувачів відповідно до реалізованої логіки.

3.3 Вартість розробки

Основною метою проєкту є розробка погодного чат-боту на мові Python для месенджера Telegram, який надаватиме користувачам актуальну інформацію про поточні погодні умови та прогнози в заданому регіоні за допомогою зручного інтерактивного інтерфейсу. Проєкт спрямований на покращення якості погодних даних та підвищення зацікавленості у сфері програмування та розробки чат-ботів.

Аналіз потреб ринку показав, що існує значний попит на погодні чат-боти, особливо інтегровані з популярними месенджерами, такими як Telegram. Важливим аспектом є забезпечення точності та актуальності прогнозів, що досягається використанням надійних джерел, таких як OpenWeatherMap API. Також було вивчено законодавчі вимоги щодо обробки та захисту персональних даних користувачів, щоб забезпечити відповідність усім необхідним стандартам безпеки та конфіденційності.

Прогнозування вигод включає визначення економічних вигод від використання чат-боту. Завдяки зручному доступу до актуальної

метеорологічної інформації, чат-бот сприятиме підвищенню зацікавленості до розробки чат-ботів.

Визначення можливих ризиків, таких як зміни ринку, технічні проблеми та зміни в законодавстві, а також розробка планів на випадок виникнення ризиків

Вартість розробки розрахована в таблиці 3.1. Вартість часу, витраченого на розробку, взяті з середніх показників погодинної вартості роботи на тематичних вебресурсах.

Аналіз вимог і планування включає вивчення потреб навчальних закладів та користувачів, розробку технічної документації та детальний план проекту.

Розробка включає програмування, налаштування API, тестування окремих модулів та інтеграцію системи.

Тестування охоплює перевірку функціональності, безпеки, навантаження та інтеграційні тестування.

Дизайн інтерфейсу передбачає створення зручного та привабливого інтерфейсу для користувачів.

Технічна підтримка включає початкову технічну підтримку після запуску чат-боту.

Таблиця 3.1 – Вартість розробки

Етапи розробки	Вартість	Опис
Аналіз вимог і планування	0 грн	Збір вимог, розробка технічного завдання, планування
Розробка	20 год х 720 грн	Програмування та інтеграція чат-боту з OpenWeatherMap API
Тестування	2 год х 100 грн	Функціональне та інтеграційне тестування
Дизайн інтерфейсу	5 год х 500 грн	Розробка та впровадження дизайну користувацького інтерфейсу
Технічна підтримка	200 грн	Першочерговий супровід та налаштування

ВИСНОВОК

У ході виконання дипломної роботи було розглянуто та реалізовано проєкт створення Telegram-бота для отримання прогнозу погоди за допомогою мови програмування Python та середовища розробки PyCharm. Основною метою було створити корисний та зручний інструмент, який надавав би актуальну

інформацію про погодні умови користувачам у реальному часі. У першому розділі роботи було проаналізовано існуючі рішення та підходи до створення чат-ботів, зокрема Telegram-ботів. Було визначено, що бібліотека telebot є однією з найзручніших для використання завдяки її простоті та широкому функціоналу.У другому розділі було розглянуто архітектуру та основні компоненти розробленого бота. Особливу увагу було приділено інтеграції з API сервісу OpenWeatherMap для отримання актуальних даних про погоду. Було описано основні принципи роботи з API, а також методи обробки та форматування отриманих даних. У третьому розділі детально описано процес розробки Telegram-бота на мові Python. Розглянуто основні функції бота, такі як налаштування початкового повідомлення, обробка команд користувача та отримання прогнозу погоди для введеного міста. Наведено приклади коду та пояснено їхню роботу. Розроблений бот успішно виконує поставлені завдання.

Загалом, створений Telegram-бот є ефективним інструментом для отримання погодної інформації, що підтверджується його коректною роботою. Виконана робота демонструє можливості мови програмування Python та різних бібліотек для створення чат-ботів, а також корисність інтеграції з зовнішніми API для отримання актуальних даних.

У ході тестування було встановлено, що бот працює стабільно і коректно обробляє запити користувачів, забезпечуючи точні відповіді в режимі реального часу. Використання бібліотеки telebot спростило процес розробки та дозволило зосередитися на функціональності та інтеграції з API.