МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

першого (бакалаврського) рівня вищої освіти

на тему: **«Розробка чат-боту моніторингу погодних умов для месенджера Telegram»**

Виконав: студент 4 курсу групи Іт-41

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)

Шаповал Роман Олександрович
(Прізвище та ініціали)

Керівник: к.т.н., в.о. доцента Падюка Р.І.
(Прізвище та ініціали)

ДУБЛЯНИ-2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА
ІНФОРМАЦІЙНИХ ТЕХНЕОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Перший (бакалаврський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А. М. Тригуба
«____» _____2023 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Шаповалу Роману Олександровичу

1. Тема роботи: «Розробка чат-боту моніторингу погодних умов для месенджера Telegram»

Керівник роботи Падюка Роман Іванович, в.о. доцента
затверджені наказом по університету від 27.11.2023 року № 641/к-с.

2. Строк подання студентом роботи 10.06.2024 р.
3. Вихідні дані до роботи: *вимоги до проектування чат-ботів; методика проектування інформаційних систем; технічне завдання на проектування чат-бота .*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
1. Аналіз предметної області.
2. Постановка задачі та планування проекту
3. Проектування чат-боту чат-боту моніторингу погодних умов.
4. Охорона праці та безпека в надзвичайних ситуаціях
Висновки та пропозиції.
Список використаної літератури.
5. Перелік ілюстраційного матеріалу (з точним зазначенням обов'язкових креслень): *Огляд платформ з підтримки чат-ботів, аналіз існуючих технологій зі створення чат-ботів, етапи створення чат-бота і пов'язані із ними технології, структурна схема проектованої інформаційної системи, сценарії використання бота та загальний вигляд робочих вікон*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3	*Падюка Р.І., в.о. доцента кафедри ІТ*		
4	*Городецький І.М., доцент кафедри фізики, інженерної механіки та безпеки виробництва*		

7. Дата видачі завдання 30 листопада 2023 р.

Календарний план

№ з/п	Назва етапів дипломного проекту	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	*30.11.23-02.02.24*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*03-28.02.24*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*01.03-30.04.24*	
4.	*Написання розділу «Охорона праці»*	*01-15.05.24*	
5.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу*	*15-30.05.24*	
6.	*Завершення роботи вцілому*	*01 - 10.06.24*	

Студент _____ Шаповал Р.О.
(підпис)

Керівник роботи _____Падюка Р.І.
(підпис)

УДК 004.9 : 631.1

Розробка чат-боту моніторингу погодних умов для месенджера Telegram. Шаповал Р.О. Кафедра ІТ – Дубляни, Львівський НУП, 2024.

Кваліфікаційна робота: 58 с. текст. част., 32 рис., 2 табл., 10 арк. ілюстраційного матеріалу, 22 джерел.

Здійснено загальний огляд чат-ботів та їх функцій. Було проаналізовано сучасні тенденції та можливості використання чат-ботів у різних сферах діяльності. Зокрема, проведено детальний аналіз аналогічних рішень для отримання метеорологічних даних на платформі Telegram.

Обгрунтовано вибір засобів реалізації проекту та середовища розробки. На основі проведеного аналізу було обрано відповідні технології та інструменти для розробки чат-бота.

Зареєстровано бот на платформі Telegram та вибрано платформу керування API отримання погодних даних. Реалізовано клієнтську частину чат-бота та відлагоджено її роботу. У рамках реалізації проекту було створено функціональний чат-бот, який успішно взаємодіє з користувачами через інтерфейс Telegram.

Розроблено заходи щодо охорони праці.

ЗМІСТ

ВСТУП

У сучасному світі, де темпи життя постійно прискорюються, своєчасний доступ до інформації стає все більш важливим. Одним із найбільш запитаних типів даних є метеорологічні показники, які дозволяють людям планувати свої дії, ураховуючи погодні умови. Враховуючи зростаючу популярність месенджерів, зокрема Telegram, розробка чат-ботів для моніторингу погодних умов стає актуальною і затребуваною темою досліджень.

Чат-боти є програмними агентами, які імітують спілкування з користувачами через текстовий інтерфейс. Вони інтегруються в популярні платформи обміну повідомленнями і надають користувачам швидкий і зручний доступ до різноманітної інформації та послуг. Зокрема, чат-боти для метеорологічного моніторингу можуть надавати інформацію про поточну погоду, прогноз на кілька днів вперед, сповіщати про погодні аномалії та багато іншого.

Основною метою даної кваліфікаційної роботи є розробка чат-боту для месенджера Telegram, який буде забезпечувати користувачів актуальною та точною інформацією про погодні умови. У роботі розглядаються основні підходи та технології, що використовуються для створення чат-ботів, а також особливості взаємодії з API метеорологічних служб, зокрема OpenWeatherMap.

Розробка такого чат-боту передбачає вирішення ряду завдань, включаючи аналіз вимог до системи, проектування архітектури чат-боту, вибір відповідних технологій та інструментів, реалізацію функціоналу збору та обробки даних про погоду, а також тестування і впровадження рішення.

Отже, кваліфікаційна робота має на меті розробку ефективного та зручного інструменту для отримання інформації про погодні умови, що допоможе користувачам Telegram бути завжди в курсі змін погоди і відповідно планувати свої дії.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз ринку месенджерів

Функція соціальних мереж змінилася від платформи для спілкування до простору для самопідтвердження та просування. З'являється помітна тенденція, оскільки люди поступово переходять від соціальних мереж до більш зручних програм для обміну повідомленнями, де їхня конфіденційність захищена, вимагаючи знати свій номер телефону чи ім'я користувача, щоб зв'язатися з ними.

Месенджер — це форма спілкування через мобільну програму чи веб-службу, яка дозволяє обмінюватися миттєвими повідомленнями.

Месенджери еволюціонували з простих платформ для спілкування до значущих джерел інформації для українців. За останні 18 місяців, під час конфлікту, частота читання українцями новин з каналів Telegram і Viber зросла в чотири рази. Таким чином, популярність національних телеканалів і месенджерів як джерел інформації в Україні досягла порівнянного рівня – 43% і 41% відповідно.

Українці продовжують віддавати перевагу Viber як додатку для обміну повідомленнями, зберігаючи його статус найпопулярнішого вибору. Останні дані компанії показують, що у 2022 році українці витратили приголомшливі 48,1 мільярда хвилин на дзвінки у Viber, що на 44% більше, ніж у попередньому році. Дивно, але незважаючи на триваючу війну та міграцію людей за кордон з іноземними телефонними номерами, обсяг повідомлень, надісланих через Viber, залишився на вражаючому рекордному рівні – 97,5 мільярда, що відповідає цифрам, досягнутим у 2021 році.

Інший месенджер - WhatsApp є найпопулярнішим месенджером у світі, хоча спочатку він навіть не був месенджером. Це була програма, яка показувала статус контактів: хто знаходиться в мережі, хто телефонує, хто зайнятий.

Згодом додаток перетворився на вдосконалену альтернативу sms і перший месенджер, синхронізований з контактами в телефоні.

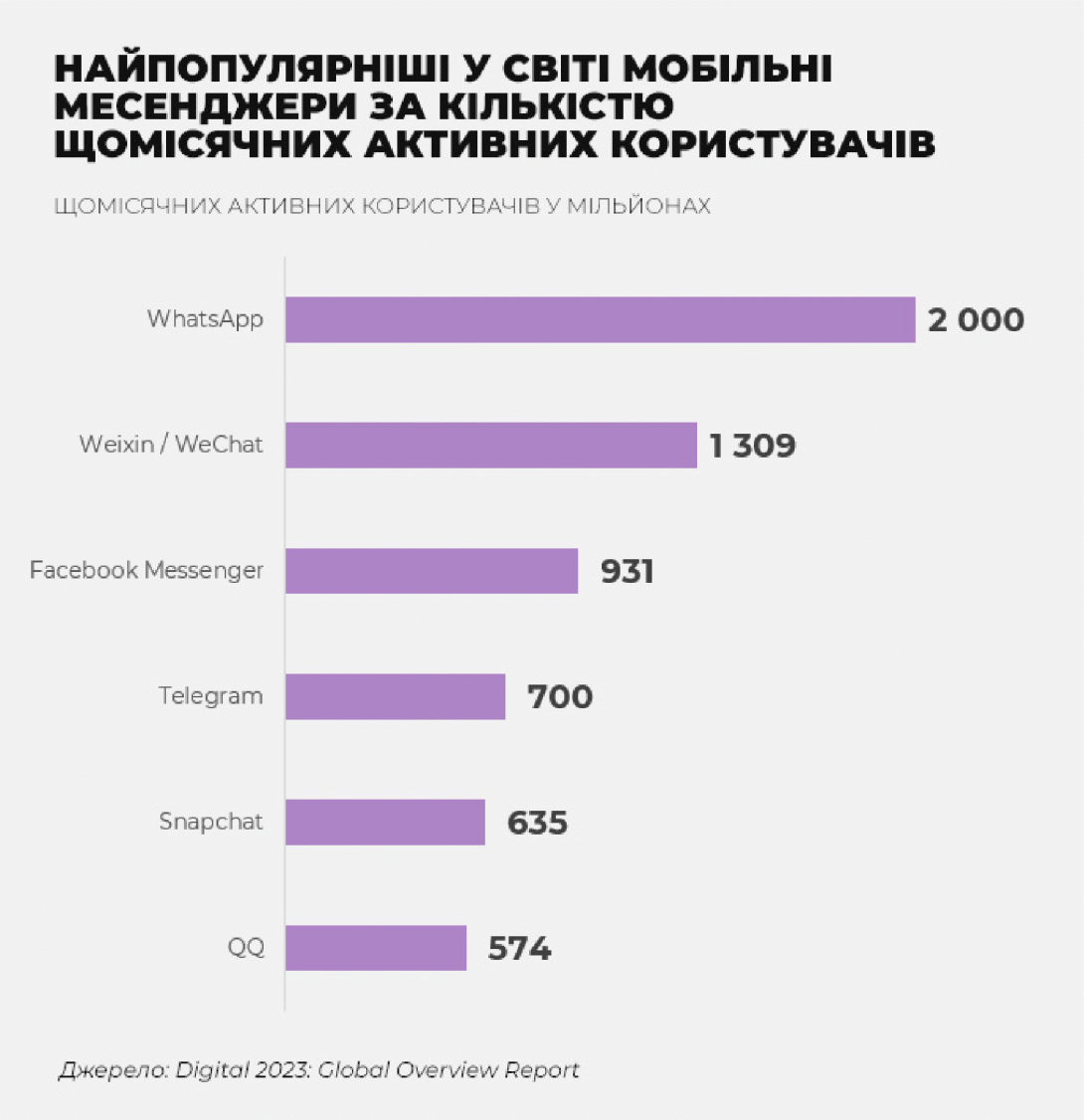


Рис. 1.1 Найпопулярніші у світі мобільні месенджери

З 2022 року спостерігається неабиякий сплеск популярності Telegram в Україні. Фактично месенджер перевершив Facebook за кількістю користувачів. З 2019 по 2022 рік Telegram став найбільш завантажуваним додатком для обміну повідомленнями в Європі. Зараз цією платформою користуються понад 700 мільйонів людей у всьому світі.

Вважаючи втіленням безпечного обміну повідомленнями, Signal заслужив підтримку не кого іншого, як Едварда Сноудена, поважного

американського програміста, який працював як на ЦРУ, так і на Агентство національної безпеки США.

Threema, популярний додаток для обміну повідомленнями в Європі, є конкурентом Signal. На відміну від Signal, Threema вимагає одноразового платежу в розмірі приблизно 3 доларів. У цьому месенджері велика увага приділяється забезпеченню безпеки користувачів. Після першого доступу до програми користувачам пропонується провести пальцем по екрану, щоб створити власний унікальний ідентифікатор.

Silent Phone – це ще одна платформа обміну повідомленнями, яка пропонує свої послуги. Хоча саму програму можна завантажити безкоштовно, вона доступна лише для тих, хто підписався на підписку. Найдоступніший варіант підписки починається від трохи більше ніж 100 доларів на рік. В обмін на цю плату користувачам надаються надійні криптографічні засоби, які захищають передачу голосу, тексту, відеоповідомлень і файлів розміром до 100 мегабайт.

Позиціонуючи себе як месенджер, який надає пріоритет конфіденційності, Confide гарантує, що ваші повідомлення будуть зашифровані та захищені від знімків екрана. За допомогою цієї платформи ви можете брати участь у бесідах, не турбуючись про те, що вони будуть збережені чи поширені з іншими. Після прочитання всі повідомлення зникають у забуття, щоб їх ніколи не було відновлено.

Позиціонуючи себе як програму без відстеження, Wickr пропонує унікальну функцію, яка дозволяє користувачам повністю видаляти повідомлення, роблячи їх непоправними. Користувачі мають право вирішувати тривалість, протягом якої їхні повідомлення залишаються доступними для одержувачів, і навіть мають можливість «відкликати» надіслані повідомлення.

У 2015 році Джейсон Сітрон і Станіслав Вишневський, двоє програмістів із Кремнієвої долини, об’єдналися, щоб заснувати Discord. Ця платформа спочатку була розроблена з наміром покращити спілкування між геймерами, коли вони грають у комп’ютерні ігри.

1.2 Історичні аспекти розвитку та основні властивості чат-ботів

Поява цифрових засобів зв'язку та комп'ютерних мережевих технологій, у тому числі Інтернету, революціонізувала спосіб віддаленої взаємодії людей ще до можливості передавати великі обсяги мультимедійних даних. До 1990-х років було прийнято надсилати короткі текстові повідомлення, наприклад телеграми, через спеціалізовані телеграфні мережі, такі як Телекс, або надсилати рукописні чи друковані листи поштою. Ці методи вимагали оплати за послугу, що робило непрактичним щоденне надсилання повідомлень, не кажучи вже про кілька повідомлень на день. Крім того, час, необхідний одержувачу для отримання текстового повідомлення, коливався від кількох годин до кількох днів.

На початку розвитку комп'ютерних мереж, коли були представлені та введені в експлуатацію основні мережеві технології, однією з перших функцій цих мереж був обмін короткими текстовими повідомленнями. Ця можливість стала можливою завдяки мережевій поштовій системі Netmail у глобальній комп'ютерній мережі Fidonet [1]. Навіть на цьому ранньому етапі деякі люди визнали потенціал для автоматизації надсилання певних типів повідомлень. Вони розробили програми-роботи, які могли взаємодіяти з іншими абонентами мережі в повністю автоматизований спосіб. Ці програми, згодом названі «ботами», служили різним цілям, зокрема:

Поява чат-ботів в онлайн-спілкуванні революціонізувала спосіб надання допомоги та інформації користувачам. У минулому офлайнові режими лише текстового спілкування не сприяли активному розвитку цих продуктів. Однак із запровадженням режимів онлайн-спілкування чат-боти стали важливим інструментом для відповіді на поширені запитання (FAQ), надання статистичної інформації за запитом, пропозиції загальної інформації для нових користувачів і доставки регулярних навчальних інформаційних бюлетенів.

Концепція чат-ботів вперше з'явилася в 1988 році з розробкою Internet Relay Chat (IRC), системи, яка дозволяла великій кількості користувачів

спілкуватися одночасно у віртуальних кімнатах, відомих як канали IRC. Цей протокол набув величезної популярності, залучаючи мільйони користувачів у всьому світі, які брали участь у текстовому онлайн-спілкуванні. Саме в цих чатах відбувся дебют чат-ботів. Незважаючи на те, що вони комп'ютерні програми, вони імітували поведінку звичайних користувачів, непомітно вписуючись у середовище чату.

Цікаво, що через поширеність граматичних помилок і присутність недосвідчених або недбалих користувачів у цих чатах траплялися випадки, коли справжніх користувачів приймали за ботів, тоді як ботів, поки не було виявлено їх справжню природу, можна було прийняти за людей. Це підкреслює ефективність чат-ботів у імітації людської взаємодії та їх здатність легко інтегруватися в онлайн-розмови.

Тепер давайте перемістимо нашу увагу з обговорення цільових платформ на інший важливий аспект використання чат-ботів: їх схожість з людьми. Чат-боти можна розділити на два типи: відкриті та секретні. Відкриті чат-боти – це ті, у яких користувачі з самого початку усвідомлюють, що вони взаємодіють із ботом, а не з реальною людиною. У цих випадках зазвичай немає необхідності в тому, щоб бот забезпечував якісне спілкування. Навіть вимогливі користувачі, які знають, що спілкуються з ботом, не переймаються якістю його відповідей; вони просто хочуть досягти бажаного результату. З іншого боку, секретні чат-боти вимагають від своїх розробників більш складної інтелектуальної основи. Мета — зробити відповіді бота максимально схожими на людину. У науковій літературі, що стосується штучного інтелекту, існує концепція, відома як тест Тюрінга. Цей тест передбачає сліпий текстовий обмін між людиною та комп'ютерною програмою чат-бота з метою унеможливлення для людини визначення, з якою людиною вона спілкується чи машиною. Варто зазначити, що під час коротких, неконкретних розмов із високоякісними програмними продуктами може бути досить складно визначити, чи спілкується хтось із ботом. Як правило, це вимагає використання конкретних запитань, які вимагають точних, конкретних відповідей, отриманих за допомогою таких

процесів, як побудова аналогії, узагальнення, аналіз вільно формалізованих ситуацій та образне мислення. Тому, щоб розробити ботів найвищої якості, розробникам вкрай важливо прагнути відтворити ці когнітивні властивості людського мозку. Однак така реплікація стає непотрібною при створенні простих відкритих ботів, які служать виключно для надання користувачам різноманітної корисної інформації.

Потенціал ботів для використання зловмисниками очевидний. Завдяки допомозі ботів зловмисники можуть ефективно атакувати безліч IRC-каналів, завдання, яке було б непрактичним виконати вручну. Це значно підвищує ймовірність успішної атаки, особливо при використанні методів соціальної інженерії, які передбачають використання перетину психології та інформаційних технологій для маніпулювання користувачами та отримання несанкціонованого доступу до комп'ютерних систем. Крім того, спеціалісти з неетичної реклами можуть використовувати можливості чат-ботів для розповсюдження великих обсягів рекламного контенту серед більш широкої аудиторії. Однак важливо зазначити, що чат-боти для IRC також виконують корисні функції для суспільства, як зазначено вище.

Після ери IRC з'явилися програми персональних месенджерів як наступний крок у розвитку комунікаційних технологій. Відтоді ці програми стали домінуючою силою на ринку, постійно розширюючи свої можливості, включаючи такі функції, як голосові дзвінки через Інтернет і відеозв'язок. Першими піонерами в цій галузі були ICQ, QIP та інші, які проклали шлях до WhatsApp, Viber і Telegram. Варто зазначити, що деякі з цих платформ тепер включають чат-ботів, що робить їх ще більш універсальними та динамічними. Давайте заглибимося у світ чат-ботів і вивчимо їх функціональні можливості.

У минулому ICQ, інтернет-пейджер, який широко використовувався на настільних комп'ютерах, не мав такого ж рівня вдосконалення для мобільних систем. Сьогодні його можна вважати застарілим, особливо в українській інтернет-спільноті, оскільки його місце зайняли нові та сучасніші продукти. Варто зазначити, що під час свого піку це програмне забезпечення широко

використовувало програми-ботів. Однак із занепадом месенджера в цілому немає жодного стимулу їх розвивати.

б) QIP, популярна російська програма, схожа на ICQ, широко використовувалася серед української інтернет-спільноти в 2000-х і на початку 2010-х років. Головним чином це було тому, що він підтримував протоколи, відмінні від ICQ.

с) З перших днів Skype мав можливість надсилати лише текстові повідомлення. Однак через маркетингову стратегію виробника це не було широко пов'язане з обміном текстом серед загальної бази користувачів. Як наслідок, лише відносно невелика кількість людей насправді використовує цю функцію. Отже, чат-боти не використовуються для Skype.

Програма для обміну повідомленнями Viber, спочатку розроблена для смартфонів, набула величезної популярності, що призвело до розробки версії для ПК, яка широко використовується сьогодні. Цей ізраїльський продукт став лідером у сфері текстового спілкування в Україні, дозволяючи не лише друзям і знайомим, а й керівникам вищого рівня та їхнім підлеглим зручно обмінюватися повідомленнями. Програмне забезпечення містить ряд рекламних продуктів і чат-ботів, які виконують різні функції в системі, включаючи рекламу, розповсюдження інформації та допоміжні послуги. Варто зазначити, що більшість користувачів сприймають Viber-ботів в першу чергу як рекламний інструмент і залучаються до них лише в разі потреби.

Viber, програмний продукт, схожий на WhatsApp, колись був популярним, особливо в західних країнах. Однак після того, як його придбала Facebook, йому було важко не відставати від внутрішньої конкуренції з боку Facebook Messenger. Варто зазначити, що хоча WhatsApp є найпоширенішим месенджером у світі, він не має можливості створювати ботів, що робить його непридатним для цілей цього проекту. Версію WhatsApp Business, яка дозволяє створювати ботів, можна не враховувати через її обмежену поширеність.

Основна перевага Facebook Messenger полягає в його повній інтеграції з обліковим записом користувача Facebook. Це означає, що будь-які дії в

програмі Messenger негайно відображаються в обліковому записі користувача Facebook, що робить його неймовірно зручним для тих, хто активно використовує соціальну мережу. Варто зазначити, що боти широко використовуються у Facebook Messenger як засіб спілкування з користувачами. Однак через величезну популярність платформи та відносно низький рівень IT-навичок серед пересічного користувача Facebook для створення цих ботів використовуються прості конструктори чатів, такі як ChattyPeople, Botsify, Chatfuel, FlowXO та BeepBoop. Важливо визнати, що ці інструменти розробки мають властиві обмеження, такі як неможливість робити тонкі налаштування та обмеження функціональності тим, що надають розробники конструктора.

Додаток Telegram пропонує безліч можливостей, окрім простого обміну текстовими повідомленнями, включаючи організацію каналів Telegram і використання надійних алгоритмів шифрування, які, як вважається, є непроникними для державного стеження. Його зручний інтерфейс та інші функції зробили його ідеальною платформою для розповсюдження програмного забезпечення чат-ботів. На відміну від Viber, боти в Telegram не зосереджені лише на рекламі; натомість вони часто служать допоміжним, службовим або освітнім цілям, тому більшість користувачів їх добре сприймає. Таким чином, цей месенджер і надалі слугуватиме основою для майбутнього розвитку.

Хоча й існують альтернативні способи обміну текстом, вони не набули значної популярності серед широкої публіки. Тому не рекомендується використовувати їх при розробці чат-бота для моніторингу метеорологічних умов.

1.3 Дослідження існуючих програм-аналогів для відстежування метеорологічних умов

Перш ніж розпочинати будь-яку розробку, необхідно врахувати наявність на ринку вже існуючих або легкодоступних програмних продуктів,

які служать подібним цілям, зокрема програмних продуктів, призначених для моніторингу метеорологічних умов. По-перше, люди можуть отримати інформацію про погоду через відповідні веб-сайти. Веб-сайти, які найчастіше використовуються, можна визначити, виконавши пошук у Google, як показано на малюнку 1.1.

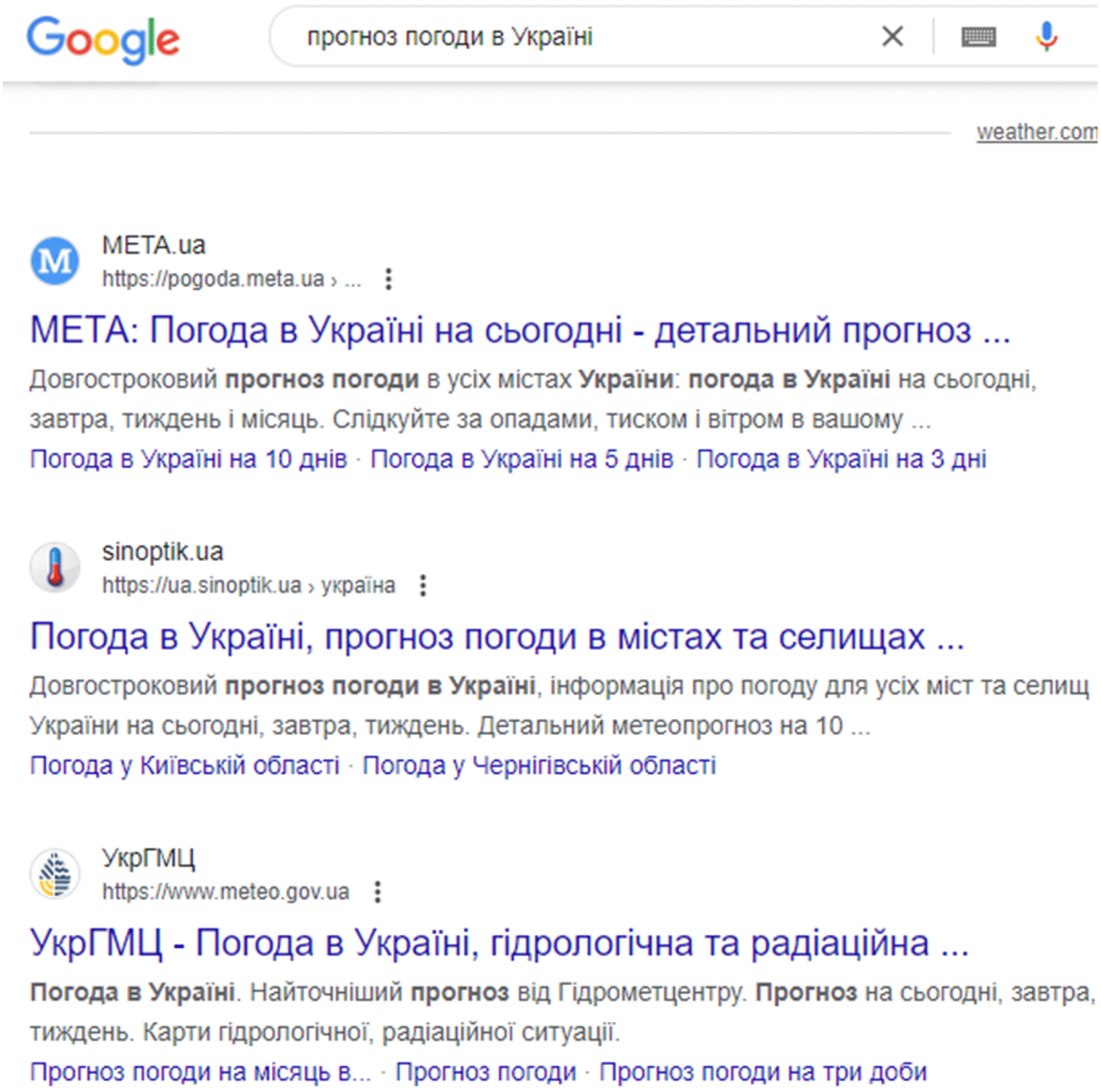


Рис. 1.1 Найпопулярніші веб-портали для відстежування погоди в Україні

Загальновідомо, що для використання будь-яких веб-сайтів потрібен доступ до Інтернету. Однак багато мобільних операторів пропонують

необмежений і безкоштовний доступ до месенджерів на різних тарифних планах. Веб-сайти не тільки часто мають захаращені інтерфейси непотрібними опціями, але й бомбардують користувачів рекламою, чого не можна сказати про Telegram. Крім того, використання бота в популярному месенджері може значно спростити завдання отримання інформації про погоду для звичайного користувача, навіть для тих, хто не має глибоких знань в IT. Пам'ятаючи про це, давайте дослідимо доступні варіанти програмного забезпечення – метеорологічні боти.

Одним із варіантів для вивчення є чат-бот «Прогноз одягу», який надає миттєві рекомендації щодо одягу на основі поточних погодних умов у певному місці (див. рис. 1.2).

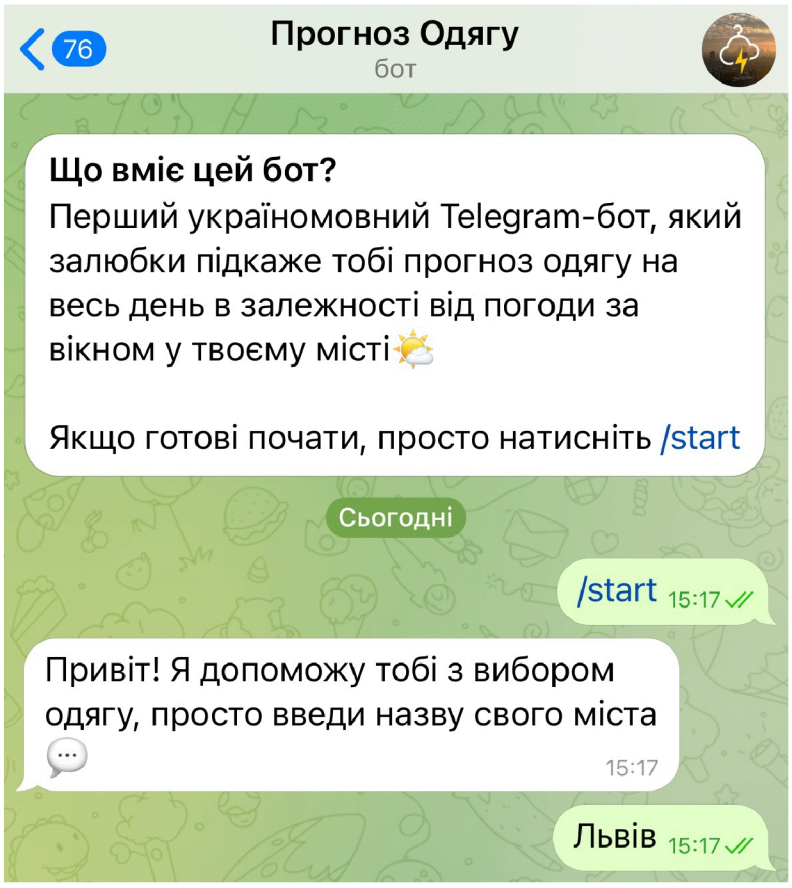


Рис. 1.2 Чат-бот «Прогноз одягу» для месенджера Telegram

У цього рішення є кілька недоліків, зокрема неможливість визначити конкретні метеорологічні умови та можливість надати інформацію лише про відповідний одяг. Крім того, бракує вибору мови.

Однак доступне більш комплексне рішення у вигляді чат-бота «Погодник». Перевага цього чат-бота полягає в тому, що користувачі можуть обирати мову спілкування, зокрема українську. Крім того, він надає користувачам детальну інформацію про погоду, як показано на малюнку 1.3.

Рис. 1.3 Чат-бот «Погодник» для месенджера Telegram

Втім, логіка роботи цього боту є дещо надмірною, адже у відповідь на перший запит із назвою міста «Львів» бот видає забагато уточнюючих питань щодо локації, що може відштовхнути потенційного користувача. Також бот створює підписку на отримання погодних даних за розкладом, що може розцінюватися деякими користувачами як спам.

Рис. 1.2 Чат-бот «Meteobot» для месенджера Telegram

З Meteobot пов'язано кілька недоліків. По-перше, бракує можливості вільного вибору мов, зокрема української. Крім того, є деякі недоліки в тексті, який використовується у власних повідомленнях. Наприклад, на початковому екрані бота, як показано на малюнку 1.4, неправильно написана фраза «вибери найближчого».

Після ретельного розгляду обмежень існуючих програмних продуктів, зокрема у сфері моніторингу метеорологічних умов за допомогою чат-бота, стає очевидним, що розробка спеціального чат-бота є дуже доцільним напрямком дій.

2. ПОСТАНОВКА ЗАДАЧ ТА ПЛАНУВАННЯ ПРОЕКТУ

2.1. Мета та задачі проекту

Метою створення цього бота є надання користувачам зручного та швидкого доступу до актуальної інформації про погодні умови та прогнози погоди на 10 днів для будь-якого міста за допомогою Telegram. Завдяки інтеграції з API OpenWeatherMap, бот може оперативно надавати точні дані про поточну температуру, опис погоди, вологість та швидкість вітру, а також детальний прогноз погоди на наступні дні. Це дозволяє користувачам планувати свої щоденні активності, подорожі та заходи, беручи до уваги майбутні погодні умови.

Для виконання цього проекту потрібно виконати наступні завдання:

- Проаналізувати предметну область, а саме ринок месенджерів, а також здійснити огляд веб сервісів для отримання даних про метеорологічні умови;
- Здійснити загальний огляд чат-ботів та їх функцій. Проаналізувати аналоги чат-ботів для отримання метеорологічних даних на платформі Telegram;
- Здійснити вибір засобів реалізації проекту та середовища розробки;
- Зареєструвати бот на платформі Telegram та вибрати платформу керування API отримання погодних даних;
- Реалізувати клієнтську частину чат-бота та відлагодити її роботу.

2.2 Вибір засобів реалізації проекту

Щоб розпочати створення програмного продукту, необхідно зробити огляд програмних і апаратних ресурсів, необхідних для виконання проекту. Основна мета цього сегменту — обґрунтувати методи, використані при

розробці майбутнього бота Telegram, який слугуватиме платформою для доставки оновлених прогнозів погоди.

Python. Python, розроблений Гвідо ван Россумом у 1990 році, є об'єктно-орієнтованою мовою програмування, яка працює на високому рівні. Всупереч поширеній думці, назва цієї мови віддає данину не виду рептилій, а радше відомому британському комедійному шоу «Летючий цирк Монті Пайтона» [7].

При створенні мови програмування він черпав натхнення з багатьох інших мов програмування, включаючи ABC, C, C++, Java, Lips, Fortran, Miranda, Icon, Modula-3 і Smalltalk. Ці мови принесли численні переваги мові Python, серед яких деякі з ключових переваг:

Мова програмування пропонує відкритий вихідний код, зручний синтаксис, зручні рішення для математичних задач, велику бібліотеку модулів і портативність програми.

Python, мова програмування, сумісна з широким спектром операційних систем, включаючи Microsoft Windows, різні системи UNIX, iOS, MacOS, Symbian, Android тощо. Спільнота Python навіть надає постійну підтримку для старих версій, таких як Windows 95, Windows 98 і Windows ME.

Jython, віртуальна машина Java, також підтримується у версіях 2.3 і раніших.

Мова програмування Python виділяється серед більшості інших мов програмування завдяки своєму унікальному набору функцій, які зазвичай не зустрічаються в інших місцях:

- Виконання та негайне відображення результатів на екрані відбувається в інтерактивному режимі, коли користувач вводить певні вирази з клавіатури.
- Python вирізняється своїм елегантним дизайном, величезною потужністю та ретельним плануванням, що робить його надзвичайно відмінною мовою програмування порівняно з іншими, особливо завдяки його можливостям об'єктно-орієнтованого програмування.

- Функціональне програмування є ще одним важливим аспектом Python.
- Програми Python структуровані у формі модулів, які можна комбінувати для створення пакетів. Крім того, Python дозволяє використовувати пакети, написані іншими мовами програмування.
- Шляхом самоаналізу можна отримати вичерпні знання про внутрішній склад будь-якого об'єкта.
- Реалізація обробки винятків дозволяє ефективно керувати помилками та винятковими ситуаціями.

Серед важливих концепцій програмування Python – ітератори, генератори та декоратори.

Мова програмування Python пропонує привабливу функцію у своїй великій стандартній бібліотеці. Ця бібліотека містить широкий спектр модулів, які полегшують такі завдання, як створення HTTP-серверів, обробка мережевих протоколів, робота з регулярними виразами, керування мультимедійними форматами, проведення модульного тестування тощо. Крім того, доступні численні модулі розширення для задоволення конкретних потреб, включаючи бази даних, веб-розробку, обробку текстів, чисельні методи та різноманітні інші завдання.

У Python існує велика колекція графічних бібліотек, спеціально створених для обробки різних типів графічних даних. Примітно, що існують навіть бібліотеки, спеціально створені для роботи з графікою в іграх.

Однак, як і інші мови програмування, Python має свої обмеження, зокрема:

- відносно нижча швидкість виконання;
- відсутність статичної типізації;
- неможливість модифікації вбудованих класів;
- глобальне блокування інтерпретатора.

Вибір цієї мови програмування базувався на її простоті, широкому спектрі можливостей і доступності різноманітних корисних модулів.

BotFather. Сама назва дає зрозуміти мету цього бота, який був розроблений творцями Telegram для нагляду та регулювання інших ботів. За своєю суттю цей бот відповідає за низку основних функцій, таких як керування підключеними ботами, реєстрація нових ботів у системі, впровадження оновлень і налаштування різноманітних налаштувань. [6].

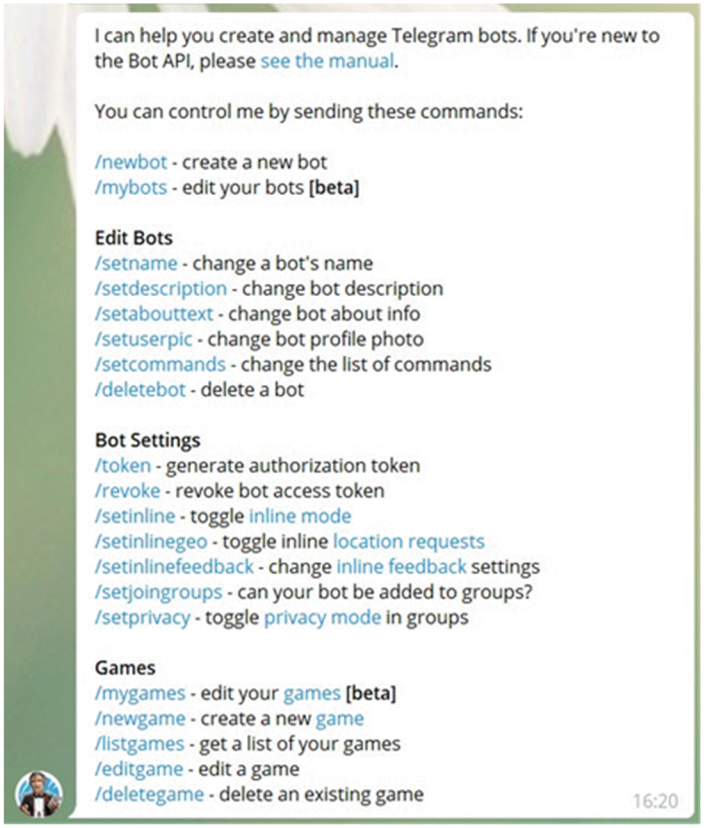


Рис. 2.1 – Команди для налаштування бота BotFather

BotFather дозволяє зареєструвати кілька нових ботів, за умови, що ім’я кожного бота є окремим. Крім того, ця програма дає змогу налаштувати зовнішній вигляд вашого чат-бота, змінивши основне зображення та опис бота. Крім того, у вас є можливість налаштувати бота за допомогою різноманітних команд (рис. 2.1) [8].

Створення бота можна здійснити лише за три прості кроки. Перший крок включає в себе мозковий штурм і вибір відповідного імені для бота, яке буде не тільки ідентифікувати його, але й буде видно в чатах і контактах.

Відповідно до другого кроку потрібно створити ім’я користувача, яке відповідає головній вимозі бути повністю відмінним. Недотримання цього критерію призведе до того, що BotFather відхилить ваш запит на створення бота та запропонує вам вибрати інше ім’я. Дозволяється використовувати будь-яку комбінацію літер, цифр і символу підкреслення з латинського алфавіту[19].

Після успішного завершення ви отримаєте токен, який надасть вам повний контроль над ботом. Крім того, використання цієї програми дозволяє створювати, редагувати та налаштовувати бот-ігри.

Telegram Bot API. Протокол шифрування, який використовує Telegram, — MTProto. Цей конкретний протокол, також відомий як Telegram API, забезпечує зв’язок із сервером. Telegram Bot API був спеціально розроблений для створення ботів, спрощення процесу написання ботів і звільнення програмістів від необхідності розбиратися в тонкощах протоколу шифрування MTProto. Зв’язок із сервером відбувається через простий інтерфейс HTTPS, що пропонує спрощену версію Telegram API[11].

Для отримання оновлень від бота доступні два методи: webhook і long pulling. За допомогою webhook сервер Telegram миттєво інформується про будь-які оновлення, а при long pulling програма надсилає запити на сервери Telegram для перевірки оновлень. Принцип отримання оновлень чат-бота проілюстровано на рисунку 2.2.

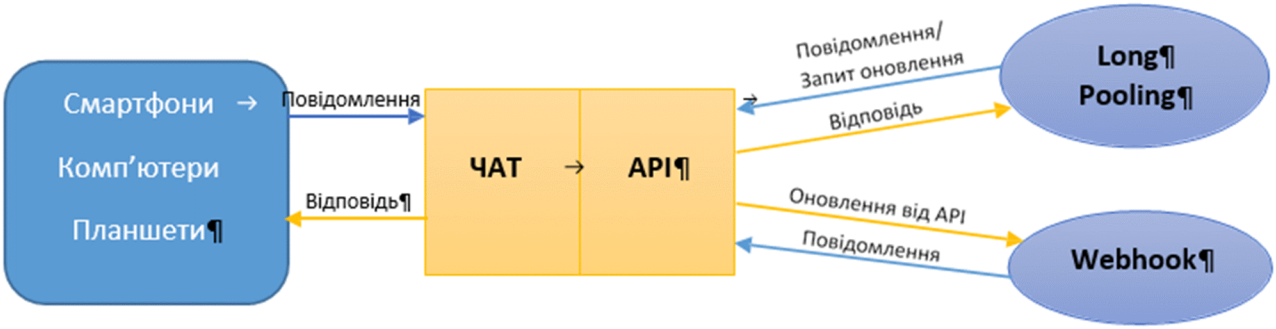


Рис. 2.2 – Принцип оновлення чат-боту на платформі Telegram

Щоб відрізнити кожного бота в Telegram, під час його створення генерується відмітний токен. Цей маркер має форму унікальної комбінації символів, як-от 566689:ABCD534uritv-lkj19W3g5y567mt99, і служить ідентифікатором для бота.

Після отримання токена стає можливим передавати запити до API Telegram Bot. Ці запити мають відповідати певним критеріям: вони мають надсилатися через захищене з'єднання HTTPS і мати такий формат: https://api.telegram.org/bot<token>/method_name[11].

Обидва методи POST і GET підтримуються Telegram Bot API. Що стосується передачі параметрів, то доступно 4 підходи.

- Перший, це "application/x-www-form-urlencoded".
- Наступний - application/json не може бути використаний для завантаження файлів.
- Використання типу вмісту multipart/form-data обмежено лише завантаженням файлів.
- І останній підхід – це звичайний URL-запит.

Відповідь JSON на запит включатиме два поля за замовчуванням: логічне поле під назвою «ок» і рядкове поле під назвою «description». Якщо значення поля «ок» є істинним, результат буде описано в полі «description», яке буде легко зрозумілим. Результат виконання буде знайдено в полі "result". Однак, якщо значення логічного поля є false, це вказує на помилку під час запиту, а опис помилки буде надано в полі «description». Крім того, поле "error_code" міститиме конкретний код помилки.

Існує безліч методів API, які легко доступні для використання, серед них можна виділити наступні:

- Функція "getUpdates" дозволяє отримувати оновлення.
- Функція setWebhook використовується для асоціації URL-адреси домену, де працює бот.
- За відправку повідомлення відповідає функція «sendMessage».

- Функція «sendLocation» дозволяє передавати географічні координати.
- Функція getFile надасть завантажений файл.

Більшість ботів працюють за умовчанням, наданим серверами Telegram. Однак, якщо вам потрібна розширена функціональність, у вас є можливість перемикатися на власну конфігурацію за потреби.

Доступ до вхідних оновлень можна отримати за допомогою методу getUpdates, який використовує тривале опитування.

Використовуючи метод setWebhook, ви маєте можливість отримувати оновлення через веб-хук, який спрямовується на вказану URL-адресу. Щоразу, коли з'являється оновлення, пов'язане з вашим ботом, Telegram надсилає запит POST, який містить файл JSON, що містить оновлення. У разі невдачі запиту Telegram припинить подальші спроби надіслати оновлення після певної кількості спроб. Важливо розуміти, що під час обробки веб-хуку оновлення, отримані через метод getUpdates, не передадуться.

Якщо ви хочете видалити вебхук, просто скористайтеся методом deleteWebhook. Після цього ви можете відновити отримання оновлень за допомогою функції getUpdate.

Об'єкти JSON використовуються для представлення всіх типів у відповідях API ботів. Ці типи охоплюють різні категорії:

Тип, відомий як «User», служить для визначення окремого користувача або бота Telegram.

- Типом «Chat» позначаються чати
- За обробку повідомлень відповідає тип «Message»
- Поле «MessageId» використовується для позначення ідентифікатора повідомлення за допомогою цілого числа.
- Інше поле, «Animation», відповідає за обробку анімацій, таких як GIF або відео H.264/MPEG-4 AVC без звуку.

- Нарешті, поле «Audio» представляє аудіофайли, визначені як музика.

За допомогою API бота ви маєте можливість використовувати форматування тексту, зокрема жирний шрифт, курсив, підкреслення та закреслення. Більше того, ви також можете змінювати форматування посилань, покращуючи їх зовнішній вигляд і мінімізуючи їхній розмір, вставляючи їх у будь-який бажаний текст.

Крім того, Telegram API Bot пропонує додаткові методи, сумісні із запитами GET і POST. Важливо зазначити, що ці методи не чутливі до регістру. Ось кілька прикладів цих методів:

- Команда "getMe" використовується для перевірки автентичності бота під час тестування.
- Щоб відключити бота від сервера Bot API і запустити його локально, скористайтеся командою «logOut».
- Використовуйте метод "close", щоб безпечно перенести бота з одного локального сервера на інший, переконавшись, що він все ще може отримувати оновлення.
- Функція "sendMessage" використовується для надсилання текстових повідомлень.
- Якщо вам потрібно переслати повідомлення будь-якого типу, використовуйте метод "forwardMessage".
- Щоб надіслати фотографії, скористайтеся функцією "sendPhoto".
- Метод sendAudio використовується для передачі аудіофайлів, зокрема музичних файлів, які можуть бути у форматі .MP3 або .M4A.

- Функція "sendDocument" використовується для передачі різних типів файлів.

Є кілька доступних методів для надсилання різних типів медіафайлів через бота. Метод «sendVideo» дозволяє передавати відеофайли у форматі mp4,

а метод «sendAnimation» спеціально розроблений для надсилання анімацій у форматі GIF або H.264/MPEG-4 AVC без звуку. Боти також можуть використовувати метод «sendVoice» для надсилання аудіофайлів, які представлені користувачеві як голосові повідомлення. Нарешті, метод «sendLocation» дозволяє боту надсилати певні точки на карті.

За передачу телефонних даних відповідає метод sendContact.

Бот також має можливість працювати у вбудованому режимі, що дозволяє користувачам отримати доступ до нього в будь-якому чаті за допомогою спеціальних маркерів, таких як @gif, @sticker, @vid тощо (рис. 2.2).

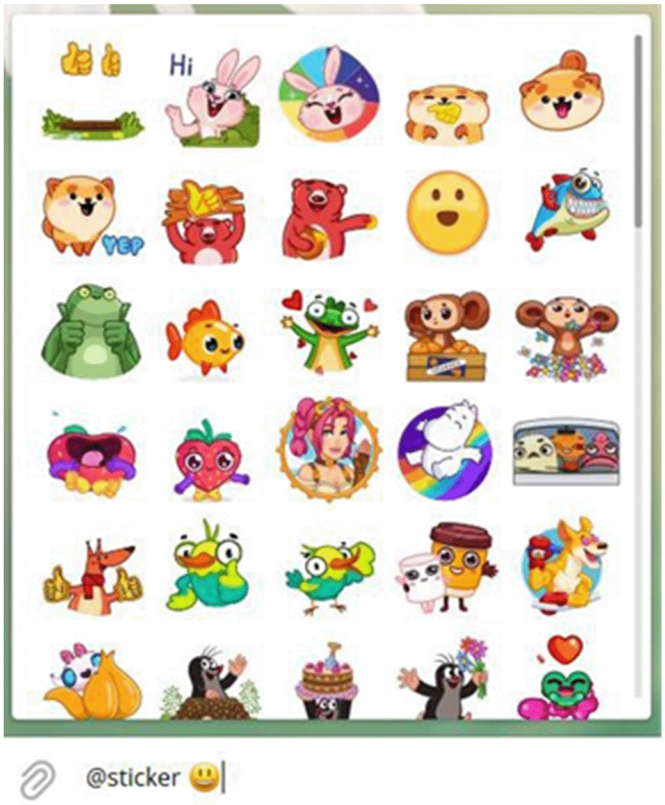


Рис. 2.3 Приклад виклику і вмісту для @sticker bot

Якщо потрібно розширити функціональність бота, у вас є можливість включити ігри HTML5. У ці ігри можна грати індивідуально або в змагальній обстановці з іншими користувачами, за умови, що ігровий бот викликаний у груповому чаті. Щоб додати ігри до бота, просто скористайтеся командою /newgame через бота BotFather. Відомі ігрові боти, які є чудовими прикладами, включають @gamebot і @gamee[10].

2.2 Огляд можливостей середовища розробки

Підтримуючи різні операційні системи, включаючи Windows, MacOS і Linux, редактор вихідного коду Visual Studio Code забезпечує баланс між простотою та можливостями. [13]

Деякі з мов програмування та технологій, які широко використовуються в IDE включають наступні: JavaScript, TypeScript, Node.js, C++, C#, Java, Python, PHP, Go, .NET і Unity.

Щоб покращити роботу користувачів із програмними програмами, VS Code містить налагоджувач, інструменти репозиторію Git, підсвічування синтаксису, інструменти рефакторингу та IntelliSense (функція, яка надає пропозиції автозаповнення на основі введених даних).

Microsoft представила IDE VS Code на конференції Blink 28 травня 2015 року. Згодом, 19 жовтня 2015 року, VS Code став доступним для використання за ліцензією MIT.

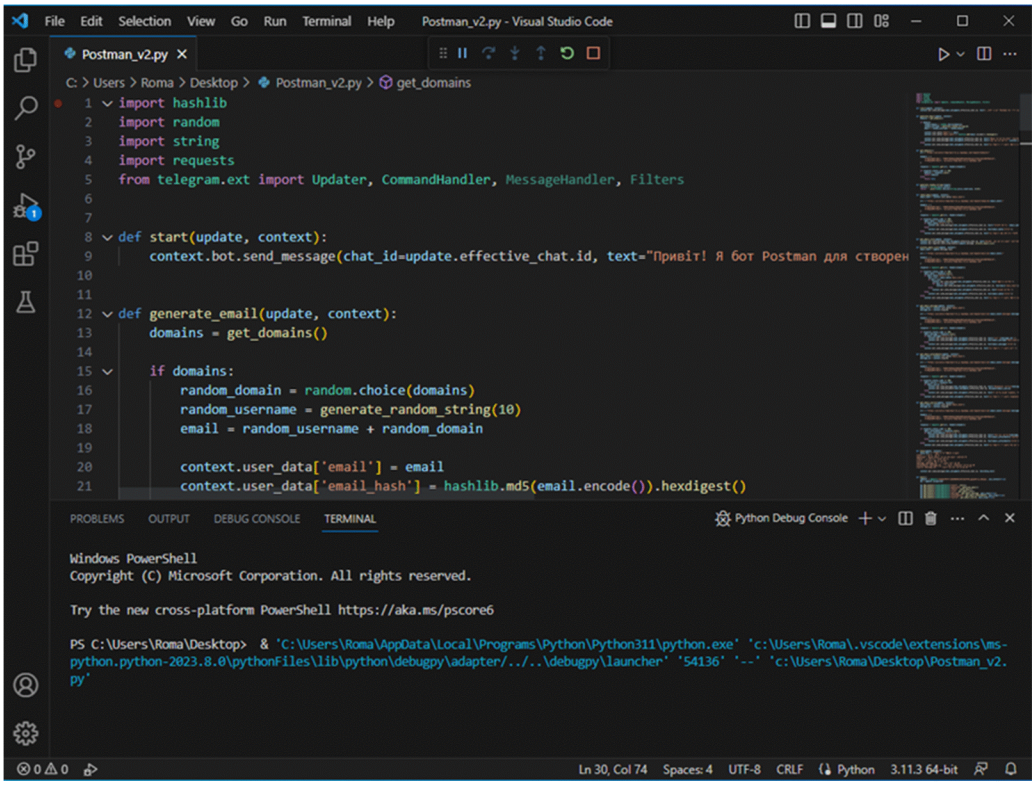


Рисунок 2.4. - Інтерфейс IDE Visual Studio Code.

2.3. Опис основних бібліотек для розробки бота

Requests. Бібліотека запитів Python широко використовується для створення HTTP-запитів. Ця бібліотека пропонує простий і організований API, що спрощує процес створення запитів. У результаті розробники можуть зосередитися на максимальному використанні даних і ефективній взаємодії з додатними службами.

Для створення запитів використовуються спеціальні команди, такі як GET і POST. Коли виконується команда GET, генерується запит на отримання даних із сервера. Ці дані представлені у форматі JSON, що виявляється дуже корисним при аналізі вмісту файлу. З іншого боку, команда POST відповідає за передачу певних даних на сервер[12].

Щоб отримати глибше розуміння, важливо зрозуміти концепцію JSON. JSON служить способом зберігання та обміну інформацією, використовуючи формат файлу з розширенням .json, який складається виключно з легко читаного тексту. Цей конкретний формат виявляється вигідним при передачі даних від сервера до клієнта[17].

Синтаксис JSON відносно простий і складається з двох основних об’єктів.

- Ключі завжди повинні бути у вигляді рядка.
- Значення можуть бути представлені в різних формах, таких як масиви, рядки, числа тощо, що забезпечує гнучкість і зручність.

Крім того, ця бібліотека пропонує гнучкість для зміни запитів відповідно до ваших конкретних вимог. Наприклад, ви можете змінити текст повідомлення та рядки запиту, налаштувати заголовки та змінити автентифікацію. Крім того, як дані, що передаються на сервер, так і дані, отримані з сервера, проходять ретельну перевірку.

Щоб ініціювати процес, клієнт передає необхідні дані в API сайту, дотримуючись зазначеного формату, зазначеного в документації сайту. Ці дані містять конкретну інформацію, до якої бажає отримати доступ клієнт, яка

зберігається на серверах відповідного сайту. Як тільки сервер отримує запит, він негайно генерує відповідь, яка потім надсилається назад клієнту (рис. 2.5).

```
<Response [200]>
```

Рис. 2.5 – Відповідь від сервера

Щоб отримати файл із даними, необхідно вказати конкретне розширення даних, наприклад JSON, щоб сервер надіслав відповідну відповідь (рис. 2.6).


```
{'base': 'stations',
 'clouds': {'all': 40},
 'cod': 200,
 'coord': {'lat': 50.4501, 'lon': 30.5241},
 'dt': 1622627676,
 'id': 703448,
 'main': {'feels_like': 14.36,
          'humidity': 65,
          'pressure': 1017,
          'temp': 15.1,
          'temp_max': 15.95,
          'temp_min': 14.75},
 'name': 'Kyiv',
 'sys': {'country': 'UA',
         'id': 2029227,
         'sunrise': 1622598663,
         'sunset': 1622656865,
         'type': 2},
 'timezone': 10800,
 'visibility': 10000,
 'weather': [{'description': 'scattered clouds',
              'icon': '03d',
              'id': 802,
              'main': 'Clouds'}],
 'wind': {'deg': 30, 'gust': 0, 'speed': 8.05}}
```


Рис. 2.6 –Приклад JSON файлу від сервера

Щоб ініціювати запит, першим кроком є імпорт модуля Requests. Після цього вам потрібно буде створити змінну, яка містить команду для створення запиту та посилання на API потрібного джерела даних (рис. 2.7).

```
r = requests.get(
    f"http://api.openweathermap.org/data/2.5/weather?q={message.text}&appid={open_weather_token}&units=metric"
)
```

Рис. 2.7 – Здійснення запиту

У деяких випадках може знадобитися включити дані в рядок запиту URL-адреси. Щоб досягти цього, ви можете передати дані як словник, використовуючи значення params як аргумент ключового слова (рис. 2.8).

```
payload = {'key1': 'value1', 'key2': 'value2'}
r = requests.get(
    f"http://api.openweathermap.org/data/2.5/weather?q={payload}&appid={open_weather_token}&units=metric"
)
```

Рис. 2.8 – Надсилання даних до URL

В рамках цього модуля процес декодування запитів відбувається автоматично. Це означає, що коли ініціюється запит, Requests використовує

методи аналізу для декодування відповіді, враховуючи інформацію, надану заголовками HTTP.

Якщо є потреба працювати з даними JSON, до складу модуля входить зручний декодер JSON У разі невдачі декодування буде створено виняток. Наприклад, якщо відповідь містить недійсний файл JSON або файл без вмісту, спроба модуля вирішити проблему може призвести до того самого винятку - ValueError: No JSON object could be decodedjson.JSONDecodeError.

Важливо мати на увазі, що успішний запит не гарантує успішної відповіді. У разі невдалої відповіді може бути надано JSON із описом конкретної помилки чи помилок. Тому вкрай необхідно ретельно вивчити відповідь сервера та переконатися, що файл JSON містить саме цю інформацію.

Якщо є випадки, коли необхідно отримати необроблену відповідь безпосередньо від сервера, можна використати команду .raw. Також важливо переконатися, що для значення потоку встановлено значення True (див. рис. 2.9).

```
r = requests.get(
    f"http://api.openweathermap.org/data/2.5/weather?q={message.text}&appid={open_weather_token}&units=metric", stream=True
)
print(r.raw)
```

Рис. 2.9 – Команда отримання необробленої відповіді

У модулі Requests існує класифікація основнх помилок і винятків. Якщо виникне проблема, пов’язана з мережею, у запитах буде виникнути виняток ConnectionError. У випадку, якщо HTTP-запит видає невдалий код статусу, виникне помилка HTTPError. Якщо відведений час очікування закінчиться, буде спрацьовувати виняток Timeout. Коли запит перевищує заздалегідь визначене порогове значення для максимальної кількості перенаправлень, буде створено виняток TooManyRedirects. У всіх випадках, коли Requests використовує імітацію, Requests.exceptions.RequestException повертається як виняток [18].

Бібліотека **python-telegram-bot** є популярним інструментом для створення ботів у Telegram на мові програмування Python. Вона забезпечує високорівневий інтерфейс для взаємодії з API Telegram, спрощуючи розробку

ботів з різноманітними можливостями, включаючи обробку повідомлень, команд, клавіатур, мультимедіа тощо. Розглянемо детальніше основні компоненти цієї бібліотеки:

Updater є основним компонентом, який забезпечує отримання оновлень від сервера Telegram та розподіл цих оновлень до відповідних обробників. Він діє як головний цикл подій, постійно опитуючи сервер Telegram для отримання нових повідомлень, команд та інших подій, які відбуваються в боті.

Dispatcher є частиною Updater і відповідає за маршрутизацію отриманих оновлень до відповідних обробників. Це дозволяє чітко розділяти різні типи подій та визначати, яка функція буде викликана у відповідь на конкретну подію.

CommandHandler є спеціалізованим обробником, який дозволяє обробляти команди, що вводяться користувачами. Команди починаються зі слешу (/) і можуть бути будь-яким словом або набором символів, які не містять пробілів.

CallbackContext є допоміжним класом, що забезпечує доступ до контексту оновлення. Він містить корисну інформацію, таку як аргументи команд, стан користувача та дані чату. CallbackContext передається у функції-обробники, що дозволяє легко взаємодіяти з користувачем та відповідати на його дії (рис. 2.10).

```
def weather(update: Update, context: CallbackContext) -> None:
    args = context.args
    # Використання аргументів для подальшої обробки
```

Рис. 2.10 Приклад використання CallbackContext

MessageHandler є універсальним обробником для обробки текстових повідомлень, що не є командами. Він дозволяє обробляти всі типи повідомлень, включаючи звичайні текстові повідомлення, фотографії, відео тощо. Filters є потужним інструментом для визначення типів повідомлень, які будуть

оброблятися MessageHandler. За допомогою фільтрів можна визначити, які саме повідомлення повинні викликати відповідні обробники (рис. 2.11).

```
from telegram.ext import MessageHandler, Filters

text_handler = MessageHandler(Filters.text & ~Filters.command, handle
dispatcher.add_handler(text_handler)
```

Рис. 2.11 Приклад створення MessageHandler

Бібліотека також підтримує створення та обробку клавіатур. ReplyKeyboardMarkup використовується для створення клавіатур, які будуть прикріплені до чатів, а InlineKeyboardMarkup дозволяє створювати інтерактивні клавіатури, що відображаються безпосередньо в повідомленнях (рис. 2.12).

```
from telegram import ReplyKeyboardMarkup

reply_keyboard = [['Option 1', 'Option 2'], ['Option 3', 'Option 4']]
markup = ReplyKeyboardMarkup(reply_keyboard, one_time_keyboard=True)

update.message.reply_text('Choose an option:', reply_markup=markup)
```

Рис. 2.12 Приклад створення клавіатури

Бібліотека python-telegram-bot надає багатий набір інструментів для створення функціональних та інтерактивних ботів у Telegram. Використовуючи компоненти Updater, Dispatcher, CommandHandler, MessageHandler, та інші, розробники можуть легко обробляти різні типи подій, взаємодіяти з користувачами та забезпечувати високу інтерактивність ботів. Це робить бібліотеку надзвичайно потужною і гнучкою для різних сценаріїв використання, від простих чат-ботів до комплексних систем автоматизації та інтеграції.

3. ПРОЕКТУВАННЯ ЧАТ-БОТУ МОНІТОРИНГУ ПОГОДНИХ УМОВ ДЛЯ МЕСЕНДЖЕРА TELEGRAM

3.1 Реєстрація чат боту на платформі Telegram

Щоб розпочати розробку програми, необхідно завершити процес реєстрації в призначеному чат-боті під назвою «BotFather». Процес реєстрації починається з виконання команди "/Newbot", після чого вводиться бажане ім'я чату. Обов'язково, щоб назва чату закінчувалася на «Bot» або «_bot». Коли всі вказані умови будуть виконані, BotFather згенерує ключ, який є унікальною послідовністю символів, яка надає доступ до HTTP API Telegram Bot, а також URL-адресу для доступу до чат-бота. На малюнку 3.1 наведено ілюстративний приклад процесу реєстрації

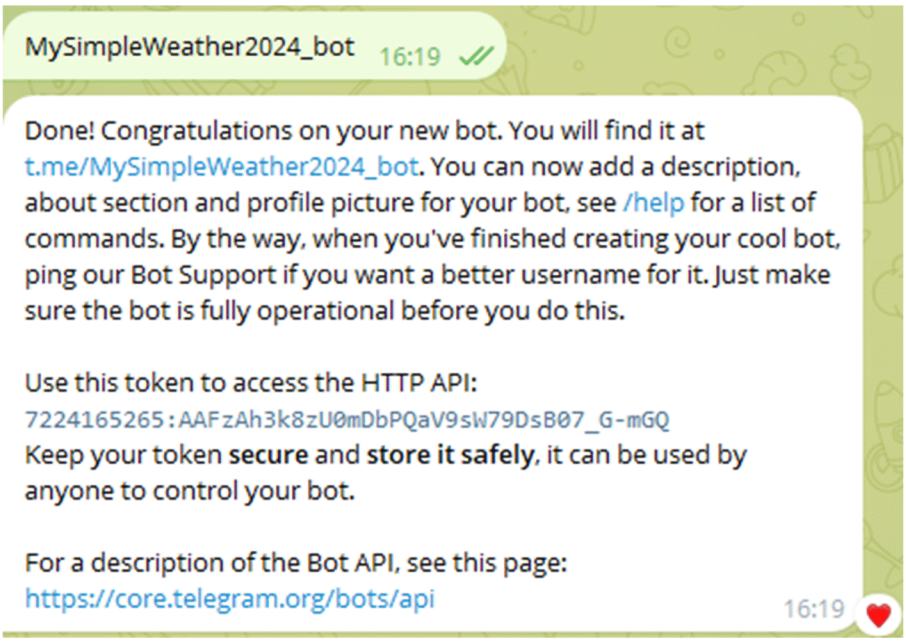


Рисунок 3.1. – Реєстрація чат боту в програмі «BotFather»

Виконавши цей крок, можна або завершити свою взаємодію з цим ботом, або персоналізувати його вигляд. Це включає зміну опису бота, фотографії профілю, списку команд та інших елементів. Щоб приступити до створення боту для моніторингу погодних умов, необхідно отримати дані з

надійного джерела. У цьому випадку для цієї мети було обрано сайт OpenWeathermap.org (рис. 3.3).

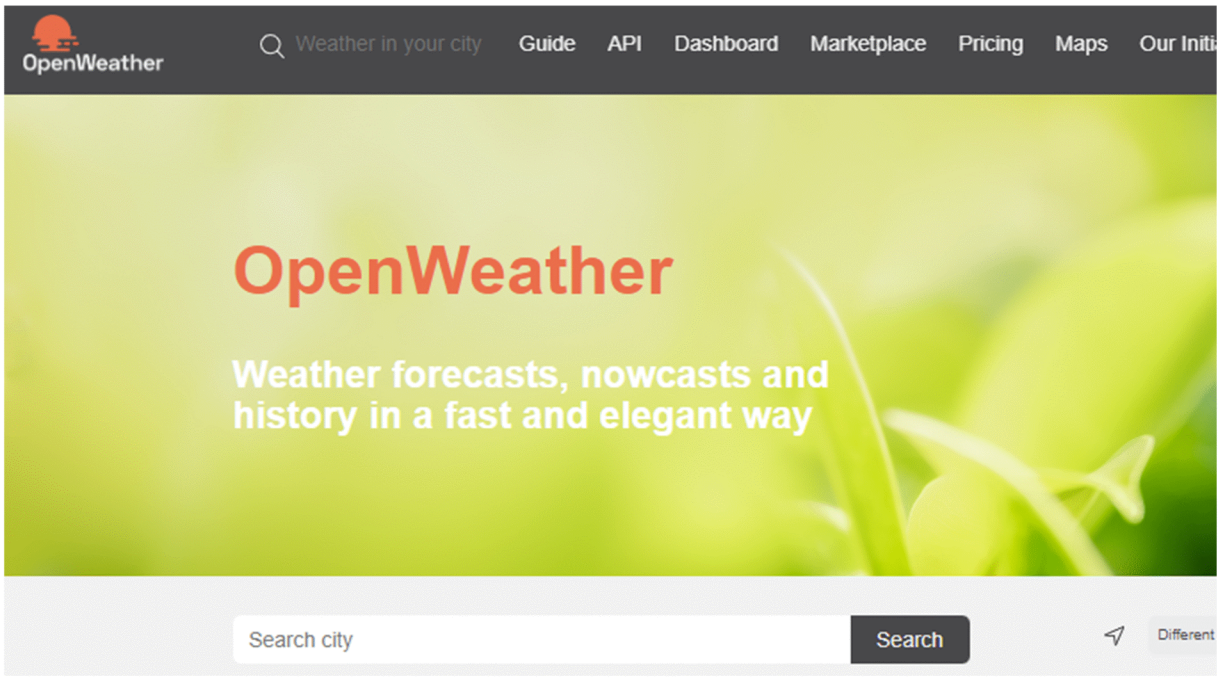


Рисунок 3.2. – Загальний вигляд порталу OpenWeathermap.org

Щоб отримати доступ до даних про погоду, що зберігаються на серверах цього сервісу, необхідно отримати ключ API від сайту. Це можна зробити, зареєструвавшись і вибравши тарифний план, який найкраще відповідає вашим потребам (див. рис. 3.3).

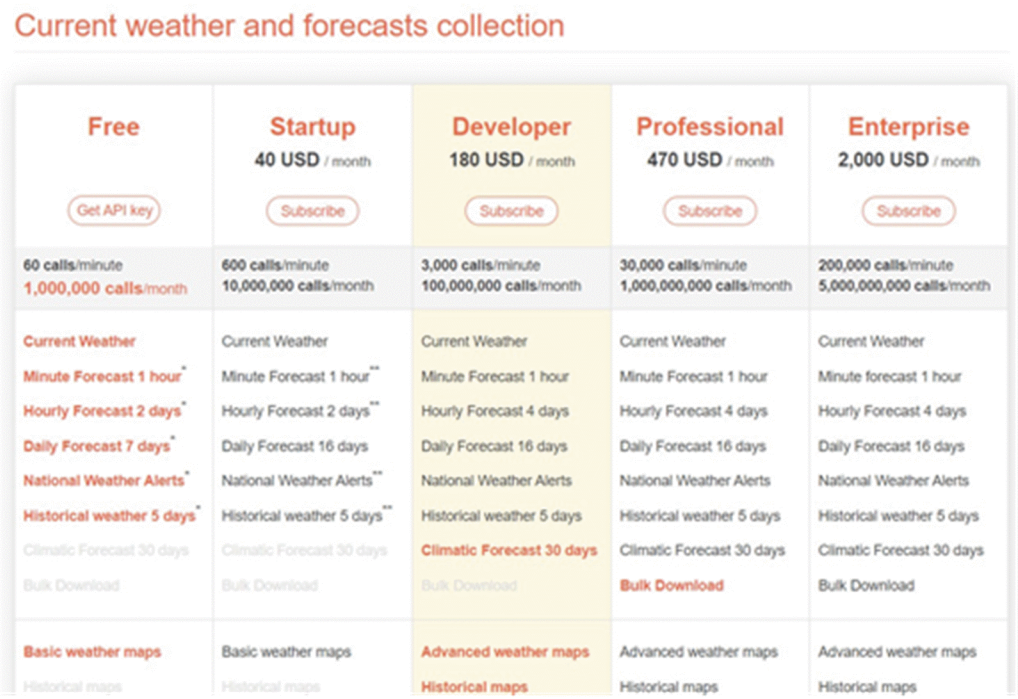


Рис. 3.3 – Тарифні плани OpenWeather

Після вибору тарифного плану API-ключ буде закріплено за вашим особистим кабінетом і стане активним максимум через 2 години. Ключ API матиме такий формат: 6edc3f14684bf6c92eab0d56b25d230a.

На веб-сайті доступний документ, який містить усю необхідну інформацію для використання API сайту. Основною функцією, яку необхідно використовувати, є виклик API, як показано на малюнку 3.4. Реалізувавши цю функцію, користувачі можуть відправити запит на сайт для отримання даних про погоду. Однак важливо зазначити, що цей запит буде оброблено, лише якщо надано як місто, так і ключ API.

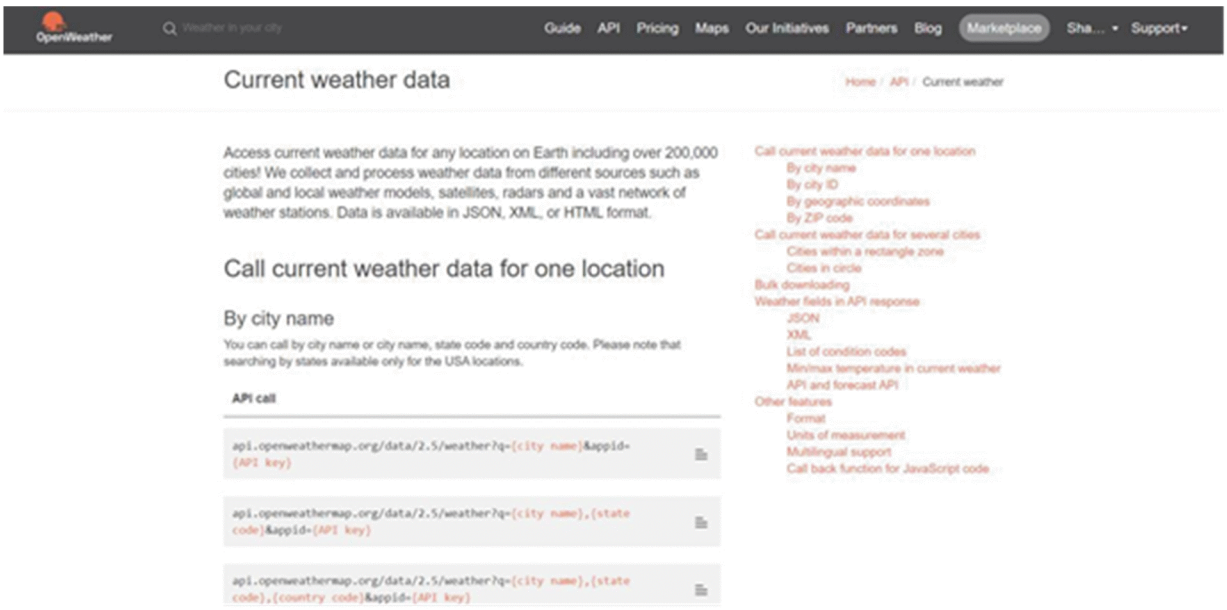


Рис. 3.4 – Функція API call на порталі OpenWeather

3.2 Особливості використання сервісу, який надає дані для моніторингу погодних умов

Робота з API OpenWeatherMap складається з кількох ключових кроків, від отримання API-ключа до здійснення запитів і обробки відповідей. Нижче наведено детальний покроковий опис цього процесу. Щоб отримати поточну погоду для конкретного міста, необхідно надіслати HTTP-запит до URL http://api.openweathermap.org/data/2.5/weather, вказавши назву міста, ваш API-ключ, одиниці вимірювання температури (наприклад, Цельсій), та мову відповіді. Для отримання прогнозу погоди на 10 днів спочатку слід отримати координати міста, а потім зробити запит до API One Call за URL https://api.openweathermap.org/data/3.0/onecall, використовуючи отримані координати, API-ключ та інші необхідні параметри .

```python
import requests

city_name = "Kyiv"
api_key = "YOUR_API_KEY"

# Отримання координат міста
url_city = f"http://api.openweathermap.org/data/2.5/weather?q={city_name}&appid={api_key}"
response_city = requests.get(url_city)
data_city = response_city.json()
lat = data_city['coord']['lat']
lon = data_city['coord']['lon']

# Отримання прогнозу погоди
url_forecast = f"https://api.openweathermap.org/data/3.0/onecall?lat={lat}&lon={lon}&exclu
response_forecast = requests.get(url_forecast)
data_forecast = response_forecast.json()

print(data_forecast)
```

Рис. 3.5 – Приклад запиту для отримання прогнозу погоди на 10 днів

Отримавши відповідь від API у форматі JSON, необхідно обробити ці дані, витягнувши корисну інформацію про погоду, таку як температура, опис погоди, вологість та швидкість вітру. Ці дані можна форматувати у зручний для користувача вигляд. Наприклад, для поточної погоди можна створити функцію, яка витягує відповідні дані з JSON-об'єкта та формує текстовий звіт, який буде відправлений користувачу через Telegram (рис.3.6).

```python
def get_weather(city_name, api_key):
    url = f"http://api.openweathermap.org/data/2.5/weather?q={city_name}&appid={api_ke
    response = requests.get(url)
    data = response.json()

    if response.status_code == 200:
        temperature = data['main']['temp']
        weather_description = data['weather'][0]['description']
        humidity = data['main']['humidity']
        wind_speed = data['wind']['speed']

        weather_report = (f"Погода в місті {city_name}:\n"
                          f"Температура: {temperature}°C\n"
                          f"Опис: {weather_description}\n"
                          f"Вологість: {humidity}%\n"
                          f"Швидкість вітру: {wind_speed} м/с")
        return weather_report
    else:
        return "Не вдалося отримати дані про погоду."

city_name = "Kyiv"
api_key = "YOUR_API_KEY"
print(get_weather(city_name, api_key))
```

Рис. 3.6. Функція обробки даних поточної погоди

Для отримання прогнозу погоди на 10 днів необхідно створити функцію, яка спочатку отримує координати міста через запит до API поточної погоди, а потім надсилає запит до API One Call, використовуючи отримані координати. Отримані дані прогнозу у форматі JSON повинні бути оброблені для витягнення ключових параметрів, таких як денна та нічна температура, та опис погоди для кожного дня. Ця інформація формується у детальний звіт про прогноз погоди на 10 днів і відправляється користувачу через Telegram.

3.3 Опис реалізації клієнтської частини чат-боту

Після завершення необхідних процедур для отримання ключа та токена можна приступати до впровадження бота. Початковий крок полягає в підключенні необхідних бібліотек, а саме Requests, Datetime і Aiogram. Після успішного підключення цих бібліотек можна розпочати розробку бота Telegram.

Після активації користувачем кнопки /start негайно надсилається повідомлення із запитом на вказівку міста. Це дозволяє боту відображати

актуальні погодні умови, температуру та іншу пов’язану інформацію. Функція, відповідальна за захоплення початкового повідомлення, запускає подальшу відповідь бота, як показано на малюнку 3.7.

```python
# Обробник команди /start
@dp.message(Command(commands=['start']))
async def send_welcome(message: types.Message):
    await message.reply("Привіт! Я бот для моніторингу погодних умов. Напиши наз
```

Рис. 3.7 – Фрагмент коду обробника команди /start

Після того, як користувач вводить місто, програма зберігає дані за певний проміжок часу та надсилає запит на веб-сайт OpenWeathermap.org для отримання даних (рис. 3.8). Ілюстрацію процесу запиту наведено на рис. 3.8.

```python
def get_weather(city_name):
    url = f'http://api.openweathermap.org/data/2.5/weather?q={city_name}
    response = requests.get(url)
    data = response.json()
    return data
```

Рис. 3.8 – Функція отримання погодних даних

Після завершення цього процесу дані, що надійшли з OpenWeathermap представлені у форматі JSON (рис. 3.9) і потребують належного форматування для забезпечення розуміння пересічним користувачем. Крім того, вкрай важливо визначити та витягти найціннішу інформацію з набору цих даних.

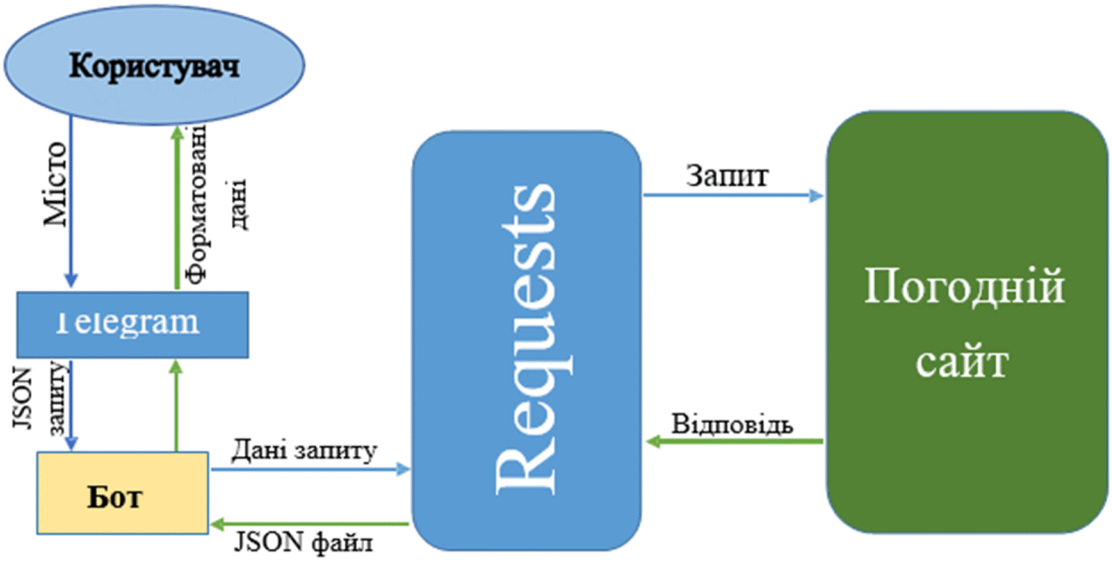


Рис. 3.9 – Опис принципу отримання даних та використання Requests

Таблиця 3.1 Опис даних отриманого JSON-файлу

base	stations	
clouds	'all': 100	
cod	200	
coodr	lat	49.8383
	lon	24.0232
dt	1717584493	
id	804	
main	feels_like	16.52
	humidity	81
	pressure	1014
	temp	16.68
	temp_max	16.68
	temp_min	16.68
name	Lviv	
sys	country	UA
	id	702550
	sunrise	1717553881
	sunset	1717612032
	Type-	
timezone	10800	
visibility	10000	
weather	description	хмарно
	icon	04d
	id	804
	main	Clouds
wind	dec	306
	gust	6.49
	speed	3.23

Для таблиці потрібні лише деякі дані - вологість, тиск, температура, назва, схід, захід сонця, погода: в основному. Вони відповідають однойменним змінним.

Оскільки дані надаються у вигляді словника, а погодні умови певного міста описуються детально, ми обрали спосіб отримання даних через ключ у словнику (рис. 3.10)

```
async def send_weather_info(message: types.Message):
    city_name = message.text
    weather_data = get_weather(city_name)

    if weather_data['cod'] == 200:
        weather_main = weather_data['weather'][0]['main']
        weather_description = weather_data['weather'][0]['description']
        temperature = weather_data['main']['temp']
        humidity = weather_data['main']['humidity']
        pressure = weather_data['main']['pressure']
        sunrise = datetime.datetime.fromtimestamp(weather_data['sys']['sunrise']).strftime('%H:%M')
        sunset = datetime.datetime.fromtimestamp(weather_data['sys']['sunset']).strftime('%H:%M')
        wind_speed = weather_data['wind']['speed']
        date_time = datetime.datetime.now().strftime('%d-%m-%Y %H:%M:%S')
```

Рис. 3.10. Функція для відправлення погодних даних у чат користувача

Запит робиться для конкретного міста, яке вказується змінною «city_name». Дані про температуру зберігаються в змінній «temperature». Змінна "weather_description" надає інформацію про поточні погодні умови. Вологість повітря представлена змінною «humidity», а дані про тиск зберігаються у змінній «pressure». Змінна «wind_speed» відображає швидкість вітру. Щоб точно представити час сходу та заходу сонця, використовувалися змінні «sunrise» і «sunset». Для обробки даних про час була використана бібліотека datetime, оскільки дані представлені у форматі Unix Timestamp, який представляє кількість секунд з 1 січня 1970 року [15].

Щоб врахувати непередбачувані погодні явища, як-от торнадо, пилові бурі та вулканічний попіл, програма включила заходи захисту від помилок, які потребують введення користувача.

У разі введення користувачем неправильного міста бот негайно видасть повідомлення про помилку та запропонує користувачеві перевірити правильність наданих даних (рис. 3.11).

```
    await message.reply(response_message, parse_mode="Markdown")
else:
    await message.reply('❌ Не вдалося отримати дані про погоду. Перевірте
```

Рис. 3.11. Перевірка правильності назви міста

Для зручності користувача було введено функцію форматування для полегшення відображення інформації про погоду (рис. 3.12).

```
response_message = (f'{emoji} *Погода в місті {city_name} на {date_time}:*\n'
                    f'🌡️ Температура: {temperature}°C\n'
                    f'💧 Вологість: {humidity}%\n'
                    f'🌬️ Атмосферний тиск: {pressure} hPa\n'
                    f'💨 Швидкість вітру: {wind_speed} м/с\n'
                    f'🌅 Схід сонця: {sunrise}\n'
                    f'🌇 Захід сонця: {sunset}')
```

Рис. 3.12 Код відображення інформації про погоду

У відповідь на повідомлення користувача в першому рядку міститься поточна дата і час. Після цього форматуються дані про погодні умови, температуру, вологість та інші важливі фактори. Кожен рядок супроводжується власним унікальним емодзі.

```
def choose_emoji(weather_main):
    emojis = {
        'Clear': '☀️',
        'Clouds': '☁️',
        'Rain': '🌧️',
        'Thunderstorm': '⛈️',
        'Snow': '❄️',
        'Mist': '🌫️',
        'Fog': '🌫️',
        'Haze': '🌫️',
        'Dust': '🌫️',
        'Sand': '🌫️',
        'Ash': '🌫️',
        'Squall': '🌬️',
        'Tornado': '🌪️'
    }
    return emojis.get(weather_main, '🌫️')
```

Рис. 3.13 - Функція для вибору емодзі відповідно до погоди

Людське сприйняття надає перевагу візуальній інформації через її легкість обробки. Як зазначалося раніше, кожен тип погоди пов’язаний із певним емодзі, що додатково підкреслює перевагу візуального сприйняття у передачі інформації (рис. 3.13).

Після завершення процесу розробки був отриманий повнофункціональний бот. Цей бот служить для надання оновленої інформації про погоду для певного міста. Щоб розпочати взаємодію з ботом, просто введіть /start або під час початкового запуску натисніть відповідну кнопку. Після цього бот запропонує ввести потрібне місто, для якого збиратиметься інформація про погоду (див. рис. 3.14).

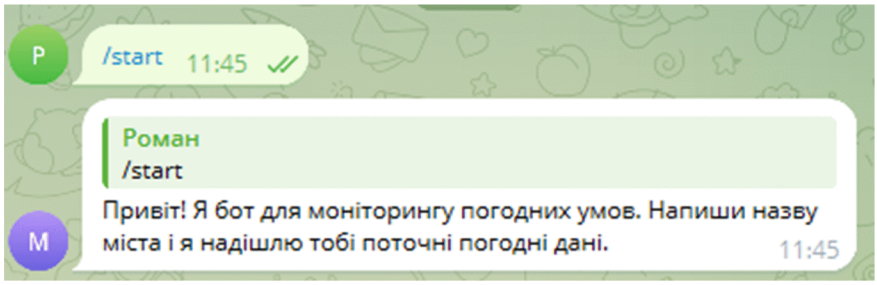


Рис. 3.14 Відповідь бота на введення команди /start

Після того, як місто введено користувачем, миттєво буде представлено вичерпну розбивку даних про погоду (рис. 3.15).

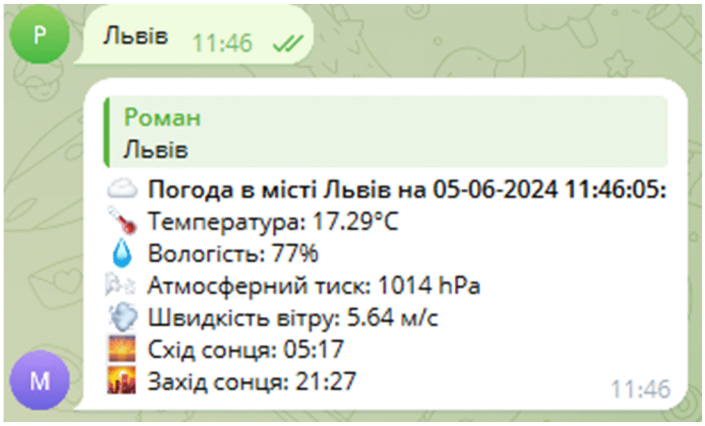


Рис. 3.15 Відповідь бота після введення назви міста

У випадку, якщо назва міста, введена для інформації про погоду, є неточною, бот негайно повідомить користувача про помилку та порадить перевірити дані (рис. 3.16).

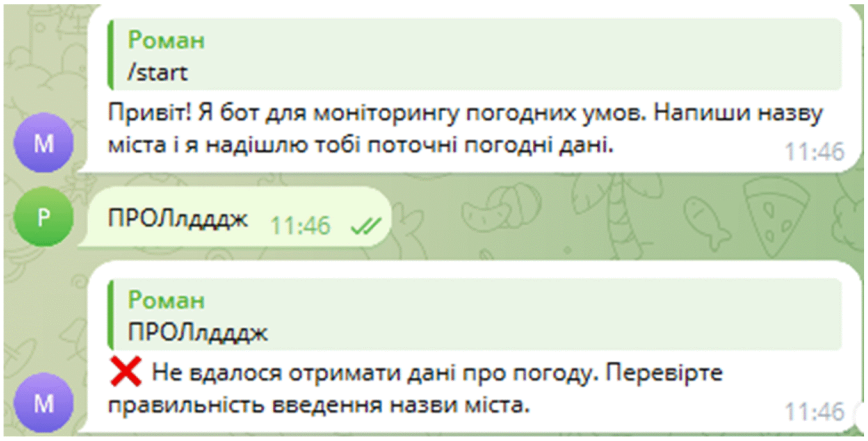


Рис. 3.16 Відповідь бота після неправильного введення назви міста

Крім того, були проведені експерименти для вивчення взаємодії користувача з програмним продуктом. З отриманих результатів видно, що обмін повідомленнями між роботом і месенджером Telegram також відбувається у формі файлу JSON (рис. 3.16).


```
{'coord': {'lon': 24.0232, 'lat': 49.8383}, 'weather': [{'id': 804, 'main': 'Clouds', 'description': 'хмарно',
'feels_like': 16.52, 'temp_min': 16.68, 'temp_max': 16.68, 'pressure': 1014, 'humidity': 81, 'sea_level': 1014,
3.23, 'deg': 306, 'gust': 6.49}, 'clouds': {'all': 100}, 'dt': 1717584493, 'sys': {'country': 'UA', 'sunrise':
702550, 'name': 'Lviv', 'cod': 200}
```


Рис. 3.16. Результати перевірки взаємодії користувача з ботом.

Після отримання даних надсилається запит на сайт Openweathermap, вказаний у коді. Після успішного зв’язку з цим ресурсом інформація надсилається у форматі JSON (рис. 3.16). Як можна побачити на рисунку, що код запиту повертає значення 200, що означає успішний доступ.

4. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Структурно-функціональний аналіз виробничого процесу та розроблення моделі травмонебезпечних ситуацій

У зображеннях процесів формування, виникнення аварій та виробничих травм усі випадкові події, що утворюють конкретну аварійну ситуацію, пов'язані між собою причинно-наслідковими зв'язками.

Метод логічного моделювання потенційних аварій, травм та катастроф відкриває можливість розробити досконалу систему управління ОП виробництва, яка базується на оперативному пошуку виробничих небезпек, їх глибокому аналізі й терміновому прийнятті заходів для усунення потенційних небезпек ще до виникнення травмонебезпечних та катастрофічних ситуацій. Деякі небезпечні ситуації в табл. 4.1.

Працівники, що обслуговують електрообладнання вениляційної системи, зобов'язані знати Правила безпечної експлуатації електроустановок споживачів відповідно до займаної посади або роботи, як вони виконують, і мати відповідну групу з електробезпеки [21,22].

Працівники, що порушили вимоги Правил безпечної експлуатації електроустановок, усуваються від роботи і несуть відповідальність (дисциплінарну, адміністративну, кримінальну) згідно з чинним законодавством. Такі працівники не допускаються до робіт в електроустановках без позачергової перевірки знань вимог правил безпечної експлуатації електроустановок.

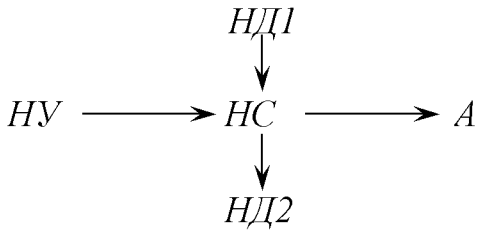
Забороняється допускати до роботи в електроустановках осіб, які не пройшли навчання і перевірку знань Правил безпечної експлуатації електроустановок.

Працівнику, який пройшов перевірку знань Правил безпечної експлуатації електроустановок, видається посвідчення встановленої форми.

Таблиця 4.1. Моделювання процесів формування та виникнення травмонебезпечних і аварійних ситуацій

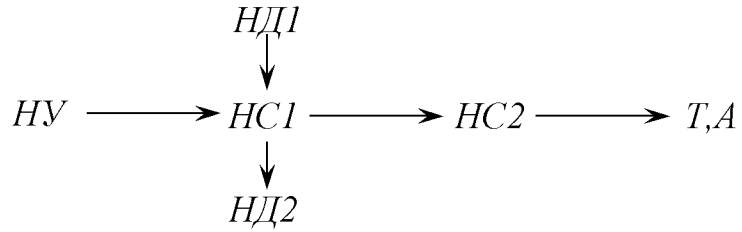
Вид робіт	Виробнича небезпека			Можливі наслідки	Заходи запобігання небезпечним ситуаціям
	Небезпечна умова (НУ)	Небезпечна дія (НД)	Небезпечна ситуація (НС)		
Використання механічної вентиляції	Оператор не перевірив обладнання НУ	Пошкоджений трубопровід мережі НД1 Закупорений трубопровід шланга НД2	Відмова вентиляційної системи (двигуна) НС	Аварія	Розвісити плакати, провести інструктажі із експлуатації обладнання системи

Модель процесу:



Використання електронних пристроїв регулювання	Пошкоджена ізоляція провідників з’єднання НУ	Пробій на корпус НД1 Коротке замикання НД2	Ураження людини електричним струмом НС1 Виведення обладнання із ладу НС2	Травма Аварія	Заміна провідників, установлення захисного обладнання (запобіжників, захист від ураження людини струмом) тощо

Модель процесу:



Посвідчення про перевірку знань працівника є документом, який засвідчує право на самостійну роботу в електроустановках на зазначеній посаді за фахом.

4.2. Вимоги техніки безпеки під час роботи обладнання та протипожежні заходи

Вимоги правил техніки безпеки перед початком роботи. Для початку роботи пов'язаної з вентиляцією вимикають рубильники або автоматичні вимикачі щита низької напруги, запирають шафу і вивішують попереджувальні плакати. Також повинні бути основні захисні засоби до яких належать такі, ізоляція яких надійно захищає від робочої напруги мережі і за допомогою яких можна дотикатися до струмопровідних частин, що перебувають під напругою, без небезпеки ураження електричним струмом (інструмент з ізольованими ручками, ізолюючі струмовимірювальні кліщі, діелектричні рукавиці).

Вимоги правил техніки безпеки під час роботи. Виконавши ці операції, надівають діелектричні рукавиці і за допомогою покажчика напруги перевіряють відсутність напруги на всіх фазах. Потім, приєднавши один кінець переносного заземлення до заземлюючого пристрою, накладають його на струмоведучі частини. Після цього остаточно приступають до роботи.

Вимоги правил техніки після закінчення роботи. Після закінчення роботи системи перед її вимиканням необхідно виконати такі технічні операції: перевірити надійність кріплення, зняти переносні тимчасові заземлення, відімкнути щит низької напруги і зняти плакати з техніки безпеки; якщо тимчасове переносне заземлення встановлене на лінії, його також треба зняти тощо [21].

Протипожежні заходи на об'єкті. Для запобігання пожеж на об'єкті розроблено організаційні, експлуатаційні, технічні режимного характеру, пожежно-евакуаційні, профілактичні заходи. До організаційних заходів

відносяться правила розміщення машин, що обслуговують приміщення, обладнання, матеріалів з дотримання певних проходів, не допускається захаращення приміщень, проходів і т.д.

4.3. Розрахунок штучного заземлення

Вибір штучного заземлення проводиться в залежності від характеру ґрунту і способу забивання стержнів [21]. Розраховуємо заземлюючий контур підстанції напругою 10/0,4 кВ з глухозаземленою нейтраллю. Характер ґрунту – чорнозем з $\rho=2\cdot10^4$ Ом·см. Кліматична зона – IV (K_c – 1,2, K_n – 1,5). Струм замикання на землю в мережі становить 50 А.

В відповідності з діючими правилами, опір заземлюючого пристрою повинен становити

$$R = \frac{125}{I_з} = \frac{125}{50} = 2,5 Ом , \qquad (4.1)$$

де $I_з$ – струм замикання на землю, А.

Приймаємо 3 Ом. Контур заземлення розміщуємо в ряд з а = 5 м, l = 2,5 м. В якості стержневого заземлювача приймаємо кутникові сталь 50х50х5 мм, а протяжного – пластинчасту сталь 40х4 мм.

Опір одиночного стержня становить:

$$R_o = 0.00318\rho \cdot K_c, Ом \qquad (4.2)$$

де K_c – коефіцієнт сезонності для стержневого заземлювача (K_c – 1,2).

$$R_o = 0.00318 \cdot 2 \cdot 10^4 \cdot 1.2 = 76.32 Ом .$$

Число стержнів приймаємо 15. При цьому коефіцієнт використання стержневих заземлювачів становить η_c = 0,7. Опір всіх стержнів розтікання струму становить:

$$R_c = \frac{R_o}{n \cdot \eta_c}, Ом , \qquad (4.3)$$

де n – число стержнів, шт.

$$Rc = \frac{76.32}{15 \cdot 0.7} = 7.3Ом .$$

Довжина протяжного заземлювача становить l = 35 м (3500 см); приймаємо t = 50 см, b = 0,4 см. Опір протяжного заземлювача становить:

$$R_{пр} = \frac{0,366}{l} \cdot \rho \cdot 2 \cdot \lg \frac{2 \cdot l^2}{t \cdot b}, Ом \qquad (4.4)$$

$$R_{пр} = \frac{0,366}{3500} \cdot 1,2 \cdot 10^4 \cdot 2 \cdot \lg \frac{2 \cdot 3500^2}{0,4 \cdot 50} = 3,2Ом$$

Коефіцієнт використання протяжного заземлювача η_n = 0,71. Дійсний опір протяжного заземлення становить:

$$R_n = \frac{R_{пр}}{\eta_n} = \frac{3,2}{0,71} = 4,5Ом \qquad (4.5)$$

Опір всього заземлюю чого пристрою становить:

$$R_u = \frac{R_c \cdot R_n}{R_c + R_n} = \frac{4.5 \cdot 7.3}{4.5 + 7.3} 2.78 < 3Ом \qquad (4.6)$$

Отже, число стержнів вибрано вірно.

4.4. Захист цивільного населення

Забезпечення захисту населення і території у разі загрози та виникнення надзвичайних ситуацій є одним з найважливіших завдань не лише підприємства, але й цілої держави. Актуальність проблеми забезпечення природо-техногенної безпеки населення і території зумовлена тенденціями зростання втрат людей і шкоди територіям, що спричиняються небезпечними природними явищами, промисловими аваріями і катастрофами.

Інженерний захист проводиться з метою виконання вимог ІТЗ із питань забудови міст, розміщення ПНО, будівлі будинків, інженерних споруд та інше.

Медичний захист проводиться для зменшення ступеня ураження людей, своєчасного надання допомоги постраждалим та їх лікування, забезпечення епідеміологічного благополуччя в районах надзвичайних ситуацій.

Біологічний захист включає своєчасне виявлення чинників біологічного зараження, їх характеру і масштабів, проведення комплексу адміністративно-господарських, режимно-обмежувальних і спеціальних протиепідемічних та медичних заходів.

Радіаційний і хімічний захист включає заходи щодо виявлення і оцінки радіаційної та хімічної обстановки, організацію і здійснення дозиметричного та хімічного контролю, розроблення типових режимів радіаційного захисту, забезпечення засобами індивідуального захисту, організацію і проведення спеціальної обробки.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Здійснено загальний огляд чат-ботів та їх функцій. Було проаналізовано сучасні тенденції та можливості використання чат-ботів у різних сферах діяльності. Зокрема, проведено детальний аналіз аналогічних рішень для отримання метеорологічних даних на платформі Telegram. Це дозволило визначити основні вимоги до функціональності та зручності використання чат-бота, а також виявити переваги та недоліки існуючих рішень.

Здійснено вибір засобів реалізації проекту та середовища розробки. На основі проведеного аналізу було обрано відповідні технології та інструменти для розробки чат-бота. Зокрема, для реалізації проекту було використано мову програмування Python, фреймворк Aiogram для створення бота та API OpenWeatherMap для отримання даних про погоду. Вибір цих інструментів було обґрунтовано їх зручністю, гнучкістю та широкими можливостями для інтеграції.

Зареєстровано бот на платформі Telegram та вибрано платформу керування API отримання погодних даних. Було успішно створено обліковий запис бота на платформі Telegram, отримано токен доступу та налаштовано всі необхідні параметри для його роботи. Для отримання метеорологічних даних обрано платформу OpenWeatherMap, яка забезпечує доступ до надійних та актуальних даних про погодні умови.

Реалізовано клієнтську частину чат-бота та відлагоджено її роботу. У рамках реалізації проекту було створено функціональний чат-бот, який успішно взаємодіє з користувачами через інтерфейс Telegram. Бот здатний надавати інформацію про поточну погоду, прогноз на кілька днів вперед, а також відповідати на інші запити користувачів. Особлива увага приділена зручності та простоті використання чат-бота, що забезпечує позитивний користувацький досвід.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Telegram [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Telegram.

2. Топ месенджерів в україні та світі 2023 [Електронний ресурс] Режим доступу до ресурсу: https://marketer.ua/ua/top-messengers-in- ukraine-and-the-world/.

3. Чат-бот [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Чат-бот.

4. ТОП-40 популярних телеграм-ботів в Україні: фінанси, шопінг і відпочинок [Електронний ресурс] – Режим доступу до ресурсу: https://psm7.com/uk/news/top-40-boty-telegram-v-ukraine.html.

5. Погодник [Електронний ресурс] – Режим доступу до ресурсу: https://pogodnik.com/uk/about.

6. Python [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Python.

7. Requests: HTTP for Humans™ [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python-requests.org/en/master/.

8. Нетребенко А. О. Чат-боти соціальних мереж як інструмент для керування інформаційним середовищем. Матеріали XII науково-технічної конференці. Збірник тез 10.12.2021, ДУТ, м. Київ 38 - 39 с.

9. Нетребенко А. О. Чат-боти у соціальних мережах як інструмент для надання інформаційних послуг. Збірник матеріалів іх всеукраїнської науково-практичної конференції 30.11.2021, К., ІІТЗН НАПН, м. Київ 116 – 118 с.

10. Openweathermap [Електронний ресурс] – Режим доступу: https://openweathermap.org/api.

11. Python Programming Language Documentation [Електронний ресурс] – Режим доступу: https://www.python.org/doc/.

12. Facebook Messenger Platform [Електронний ресурс] – Режим доступу: https://developers.facebook.com/docs/messenger-platform/.

13. Telegram Bot API. [Електронний ресурс] – Режим доступу: https://core.telegram.org/bots/ap

14. Chatbot Conversations to deliver $8 billion in Cost savings by 2022 [Електронний ресурс]. – Режим доступу: https:// www. juniperresearch. com/analystxpress/july-2017/chatbot-conversations-to-deliver-8bn-cost-saving.

15. Global number of mobile messaging users 2016-2022. [Електронний ресурс]. – Режим доступу:https://www.statista.com/statistics/483255/number-of-mobile-messaging-users-worldwide/.

16. JSON: [Електронний ресурс]. – Режим доступу: https://www.json.org/json-ru.html

17. Messenger Platform [Електронний ресурс]. – Режим доступу: https://developers.facebook.com/docs/messenger-platform/.

18. Введення в JSON [Електронний ресурс] / JSON – Режим доступу: https://www.json.org/json-ua.html

19. Марк Лутц, Learning Python :2019, 720с

20. Огляд протоколу HTTP : [Електронний ресурс]. – Режим доступу: https:// developer. mozilla.org /ru/docs/ Web/HTTP/Overview

21. Пістун І. П., Тимочко В.О., Городецький І. М.. Березовецький А. П. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Ч. ІІ. Львів: Тріада плюс, 2011. 224 с.

22. Пістун І. П., Тимочко В.О., Городецький І. М.. Березовецький А. П. Охорона праці (гігієна праці та виробнича санітарія): навч. посібн. / за ред. І.П. Пістуна. Ч. ІІ. Львів: Тріада плюс, 2011. 224 с.

ДОДАТКИ

Код Telegram бота моніторингу погодних умов для месенджера Telegram

```
import datetime
import requests
import asyncio
from aiogram import Bot, Dispatcher, types
from aiogram.filters import Command
from aiogram.types import Message, BotCommand

# Токен бота з BotFather
BOT_TOKEN = '7224165265:AAFzAh3k8zU0mDbPQaV9sW79DsB07_G-mGQ'

# Ключ доступу до OpenWeatherMap API
OPENWEATHERMAP_API_KEY = '6edc3f14684bf6c92eab0d56b25d230a'

# Функція для отримання погодних даних за містом
def get_weather(city_name):
    url = f'http://api.openweathermap.org/data/2.5/weather?q={city_name}&appid={OPENWEATHERMAP_API_KEY}&units=metric&lang=ua'
    response = requests.get(url)
    data = response.json()
    return data


# Функція для вибору емодзі відповідно до погоди
```

```
def choose_emoji(weather_main):
    emojis = {
        'Clear': '☀️',
        'Clouds': '☁️',
        'Rain': '🌧️',
        'Thunderstorm': '⛈️',
        'Snow': '❄️',
        'Mist': '🌫️',
        'Fog': '🌫️',
        'Haze': '🌫️',
        'Dust': '🌫️',
        'Sand': '🌫️',
        'Ash': '🌫️',
        'Squall': '💨',
        'Tornado': '🌪️'
    }
    return emojis.get(weather_main, '🌫️')

# Функція для відправлення погодних даних у чат користувача
async def send_weather_info(message: types.Message):
    city_name = message.text
    weather_data = get_weather(city_name)

    if weather_data['cod'] == 200:
        weather_main = weather_data['weather'][0]['main']
        weather_description = weather_data['weather'][0]['description']
        temperature = weather_data['main']['temp']
        humidity = weather_data['main']['humidity']
```

```
            pressure = weather_data['main']['pressure']
            sunrise = datetime.datetime.fromtimestamp(weather_data['sys']['sunrise']).strftime('%H:%M')
            sunset = datetime.datetime.fromtimestamp(weather_data['sys']['sunset']).strftime('%H:%M')
            wind_speed = weather_data['wind']['speed']
            date_time = datetime.datetime.now().strftime('%d-%m-%Y %H:%M:%S')

            emoji = choose_emoji(weather_main)

            response_message = (f'{emoji} *Погода в місті {city_name} на {date_time}:*\n'
                                f'🌡️ Температура: {temperature}°C\n'
                                f'💧 Вологість: {humidity}%\n'
                                f'💪 Атмосферний тиск: {pressure} hPa\n'
                                f'🌬️ Швидкість вітру: {wind_speed} м/с\n'
                                f'🌅 Схід сонця: {sunrise}\n'
                                f'🌇 Захід сонця: {sunset}')

            await message.reply(response_message, parse_mode="Markdown")
        else:
            await message.reply('❌ Не вдалося отримати дані про погоду. Перевірте правильність введення назви міста.')

    # Ініціалізація бота та диспетчера
    bot = Bot(token=BOT_TOKEN)
    dp = Dispatcher()
```

```
# Обробник команди /start
@dp.message(Command(commands=['start']))
async def send_welcome(message: types.Message):
    await message.reply("Привіт! Я бот для моніторингу погодних умов. Напиши назву міста і я надішлю тобі поточні погодні дані.")

# Обробник повідомлень з текстом (назва міста)
@dp.message(lambda message: True)
async def get_weather_info(message: types.Message):
    await send_weather_info(message)

async def main():
    # Установка команд бота
    await bot.set_my_commands([
        BotCommand(command="/start", description="Запустити бота"),
    ])

    # Запуск полінгу
    await bot.delete_webhook(drop_pending_updates=True)
    await dp.start_polling(bot)

if __name__ == '__main__':
    import logging
    logging.basicConfig(level=logging.INFO)
    asyncio.run(main())
```