НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:
Завідувач кафедри
_____ Сергій СТІРЕНКО
(підпис)
“__” _____2022 р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою “Комп’ютерні системи та мережі”
спеціальності 123 “Комп’ютерна інженерія”

на тему: Чат-бот для визначення прогнозу погоди

Виконав: студент 4 курсу, групи ІО-81
(шифр групи)

Кім Олексій Валерійович
(прізвище, ім’я, по батькові) (підпис)

Керівник Роковий Олександр Петрович
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант (нормоконтроль) проф., д. т. н. Сімоненко В. П.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище та ініціали)(підпис)

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

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

Київ – 2022 р.

АННОТАЦІЯ

Робота над дипломним проєктом полягала у створенні чат-бота на основі месенджеру Telegram, який буде за запитом користувача запитувати дані про погоду з погодного сервісу і відправляти їх користувачеві.

Реалізований проєкт дозволяє отримати інформацію про погоду в поточний момент часу і отримати достатньо точні показники погодних умов у певному місті або в місці знаходження користувача.

ANNOTATION

The work on the diploma project was to create a chat-bot based on the Telegram messenger, which will, at the request of the user, request weather data from the weather service and send it to the user.

The implemented project allows to obtain information about the weather at the current time and to obtain sufficiently accurate indicators of weather conditions in a particular city or location of the user.

Національний технічний університет України
«Київський політехнічний інститут»
ім. Ігоря Сікорського
Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

“Комп’ютерні системи та мережі”

спеціальність 123 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ С.Г. Стіренко
(підпис) (ініціали, прізвище)
“___”_____2022 р.

ЗАВДАННЯ

на бакалаврський дипломний проект студента

Кім Олексія Валерійовича
(прізвище, ім’я, по батькові)

1. Тема проекту (роботи) *Чат-бот для визначення прогнозу погоди*
_____,
керівник проекту (роботи) Роковий Олександр Петрович, к.т.н., доцент_____
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом по університету від “___”_____20__ р. №____

2. Термін здачі студентом закінченого проекту (роботи)
_____

3. Вихідні дані до роботи *наукова та технічна документація про реалізацію окремих модулів системи для вибору найоптимальнішого рішення щодо реалізації поставленої задачі з розробки програмного забезпечення.*_____

4. Зміст пояснювальної записки: *опис предметної області і напрямків дослідження, аналіз та характеристика об’єкту досліджень, аналіз існуючих рішень, опис архітектури системи.*_____

5. Перелік графічного матеріалу (із зазначенням обов’язкових креслеників, плакатів, презентацій тощо):

Принципова схема системи, структурна схема системи, діаграма класів.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	д.т.н., проф. Сімоненко В. П.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	Затвердження теми роботи	06.04.2022-07.04.2022	
2.	Вивчення та аналіз завдання	07.04.2022-13.05.2022	
3.	Написання вступної частини та огляду предметної області	13.05.2022-23.05.2022	
4.	Розробка архітектури системи	23.05.2022-01.06.2022	
5.	Програмна реалізація системи	01.06.2022-06.06.2022	
6.	Виправлення помилок	06.06.2022-08.06.2022	
7.	Оформлення документації дипломної роботи	08.06.2022-10.06.2022	
8.	Передзахист	11.06.2022	
9.	Захист	20.06.2022	

Студент _____ Кім О.В.
(підпис) (ініціали, прізвище)

Керівник проекту _____ Роковий О.П.
(підпис) (ініціали, прізвище)

№ рядка	Формат	Позначення	Найменування	Кільк.	Примітка
			Документація загальна		
	А4	ІАЛЦ.467200.001 ОА	Чат-бот для визначення прогнозу погоди Опис альбому	2	
	А4	ІАЛЦ.467200.002 ТЗ	Чат-бот для визначення прогнозу погоди Технічне завдання	3	
	А4	ІАЛЦ.467200.003 ПЗ	Чат-бот для визначення прогнозу погоди Пояснювальна записка	70	
	А4	ІАЛЦ.467200.004 Д1	Чат-бот для визначення прогнозу погоди Принципова схема	1	
	А4	ІАЛЦ.467200.005 Д2	Чат-бот для визначення прогнозу погоди Структурна схема	1	
	А4	ІАЛЦ.467200.006 Д3	Чат-бот для визначення прогнозу погоди Діаграма класів	1	

					ІАЛЦ.467200.001 ОА		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Кім О. В.			Чат-бот для визначення прогнозу погоди Опис альбому	Літ.	Аркуш 1 / Аркушів 2
Перевірив		Роковий О. П.					
Реценз.						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-81	
Н. контр.		Сімоненко В. П.					
Затвердив							

№ рядка	Формат	Позначення	Найменування	Кільк.	Примітка
	А4	ІАЛЦ.462619.007 Д4	Чат-бот для визначення прогнозу погоди Програмний код	5	

					ІАЛЦ.467200.001 ОА	Арк
						2
Зм.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Областю застосування розроблюваної системи є чат-боти на основі популярних месенджерів.

Метою розробки системи є використання її у повсякденному житті та створення альтернативи іншим ботам такого формату.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврського дипломного проекту, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки постає створення чат-боту для моніторингу інформації про стан погодних умов в одному з популярних месенджерів, який буде доступним і зрозумілим для будь-яких користувачів.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література, технічна документація, публікації в періодичних виданнях та мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

Система, що розроблюється, повинна відповідати поставленим вимогам, що наведені нижче:

- Простий і зрозумілий інтерфейс системи;
- Технічна коректність;
- Актуальна інформація;
- Швидкий доступ до системи;

5.2. Вимоги до інструментального програмного забезпечення

Для розробки проекту на локальному ПК необхідне наступне програмне забезпечення:

- Операційна система Windows або сімейств Linux та MacOS;
- Встановлений месенджер Telegram;
- Ruby версії 2.7.5 чи вище;
- Встановлений Git;

6. ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	06.04.2022-07.04.2022
Вивчення та аналіз завдання	07.04.2022-13.05.2022
Написання вступної частини та огляду предметної області	13.05.2022-23.05.2022
Розробка архітектури системи	23.05.2022-01.06.2022
Програмна реалізація системи	01.06.2022-06.06.2022
Виправлення помилок	06.06.2022-08.06.2022
Оформлення документації дипломної роботи	08.06.2022-10.06.2022
Передзахист	11.06.2022
Захист	20.06.2022

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

ПК – персональний комп’ютер

ПЗ – програмне забезпечення

ОС – операційна система

ООП – об’єктно-орієнтоване програмування

MVP – Model-View-Presenter, шаблон проєктування, похідний від MVC.

MVC – Model-View-Controller, шаблон проєктування в програмуванні.

IDE – Integrated Development Environment, інтегроване середовище розробки

VS Code – Visual Studio Code

API – Application Programming Interface, прикладний програ́мний інтерфейс,

JSON – JavaScript Object Notation, текстовий формат обміну даними

XML – Extensible Markup Language, стандарт побудови мов розмітки ієрархічно структурованих даних для обміну

HTML – HyperText Markup Language, стандартизована мова розмітки документів для перегляду веб-сторінок у браузері

CLI – Command Line Interface, консольна утиліта від Heroku

REST – Representational State Transfer, підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів

ВСТУП

Прогноз погоди – це науково обґрунтоване припущення майбутнього стану погоди у певному місці чи регіоні на певний період. Прогноз погоди складається або розробляється метеорологічними службами за допомогою методів метеорології. Прогнози погоди можуть бути різними, як для повсякденного загального користування так і спеціалізовані, все залежить від цілей, для яких вони були розроблені.

1. Прогнози загального користування містять коротку основну інформацію про температуру, хмарність, вологість, атмосферний тиск, вітер, атмосферні опади та інші явища. Публікуються у ЗМІ, на різних веб-сайтах, озвучуються на телебаченні та радіо.
2. Авіаційні прогнози містять більш детальну характеристику видимості, хмарності, вітру та атмосферних явищ.
3. Морські та річкові прогнози містять більш детальну характеристику вітру, хвилювання та атмосферних явищ.
4. Сільськогосподарські (або агрометеорологічні) прогнози містять більш детальну характеристику атмосферних опадів та температури повітря.

Прогноз синоптичного становища, тобто атмосферних опадів, температури, вологості та полів тиску (геопотенціалу) опирається на закони, описуючі рух атмосфери як рідини, що стискається. Дані закони відносяться до такого розділу фізики як гідродинаміка, вона включає рівняння руху, перенесення вологості, стану газів та збереження маси. Вирішуючи ці рівняння чисельними методами конкретного стану атмосфери, можна отримати числові значення майбутніх полів тиску, температури та вітру. Як початкові дані для підстановки в рівняння є виміряні зараз параметри атмосфери; виміри проходять по всій можливій

висоті. Чисельне інтегрування рівнянь зобов'язує своєю точністю сучасним обчислювальним машинам, які можуть опрацьовувати велику кількість вхідних даних. Тим не менш, для цілей чисельного вирішення рівнянь та швидкої інтерпретації цих рішень необхідно використовувати максимальні можливі ресурси – суперкомп'ютери.

Важко переоцінити важливість прогнозів погоди, адже ними користується як мінімум безліч людей у своєму повсякденному житті чи у побуті, завдяки прогнозам дощів, снігу та сильних вітрів люди можуть краще та ефективніше планувати свій день або відпочинок. Також вони використовуються заради більш глобальних цілей, наприклад штормові попередження наприклад використовуються для захисту життя і майна. Також прогнози атмосферних опадів та температури важливі для сільськогосподарської сфери а тому і для трейдерів на фондових ринках.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Месенджери

Система миттєвого обміну повідомленнями (далі Месенджер) – це телекомунікаційна служба, яка служить для обміну текстовими повідомленнями між ПК та іншими пристроями через комп’ютерні мережі(але найчастіше через інтернет). На початку існування месенджерів це були просто текстові повідомлення, але згодом системи стали більш розвинуті та отримали нові функції, такі як обмін зображеннями, файлами, відео, голосовими повідомленнями та взаємодію між користувачами, тобто малювання або ігри.

Сама концепція спілкування великої кількості користувачів між собою за допомогою електронних пристроїв, які об'єднано в уніфіковану мережу, сягає своїм корінням середини 1960-х. Система обміну даними в режимі, наближеному до реального часу Compatible Time-Sharing System (CTSS) була розроблена в Массачусетському Технологічному інституті (MIT) в 1961 році. В рамках експериментального проекту до 30 користувачів могли одночасно увійти до створеного інформаційного середовища та обмінюватися повідомленнями між собою. До 1965 року ця система встигла стати досить популярним внутрішнім інструментом комунікації в MIT. А коли в 1970-х з’явився протокол peer-to-peer, з’явилася також можливість підключати різних користувачів до одного і того ж комп'ютера, обмінюватися даними, і перемикатися між різними користувачами того самого комп'ютера.

У 90-ті роки, коли масово почали використовуватись месенджери, більшість людей, які мали доступ до інтернету, використовували телефонний Dial-Up модем. Це був дуже повільний (всього 56 кбіт\сек),

дорогий та нестабільний спосіб підключення. Користувачі не заходили до інтернету довше, ніж на одну-дві години на добу, тому що цей вихід блокував телефон та оплачувався щохвилини. Доступ до інтернету з мобільних телефонів був неможливим, а ідея смартфонів тоді видавалася просто фантастикою. Саме з цих причин перше покоління месенджерів було створено лише для персональних комп'ютерів. Тоді месенджери були розраховані на обмін виключно текстовими повідомленнями та були розраховані на дуже повільну швидкисть з’єднання з інтернетом.

Одним з перших месенджерів, які відомі і на сьогоднішній день, став ICQ(також відомий як «аська»), який з'явився у 1996 році. Все почалося з того, що четверо школярів із Ізраїлю створили компанію Mirabilis та почали працювати над програмою для спілкування в Інтернеті та локальних мережах.[25] Створивши програму, вони розіслали її безкоштовно друзям та знайомим. Ті, своєю чергою, приводили в «аську» своїх друзів та знайомих. Кількість користувачів зростала у геометричній прогресії. А через якийсь час невелика та молода компанія випустила корпоративну версію ICQ. Вона стала піонером ринку месенджерів як універсальних додатків, а не систем з обмеженим доступом. Відмінністю цієї програми стали розраховані на багато користувачів чати, підтримували передачу файлів, пошук по базі користувачів і ряд інших функцій, яких раніше не було в більш ранніх месенджерах. AOL придбала Mirabilis та ICQ у 1998 році, а потім перепродала сервіс компанії Digital Sky Technologies у 2010 році. Слідом за «аською» з'явилися AIM та MSN/WLM, а також Gadu-Gadu, QQ, NateOn, Google Talk, Miranda, QIP, Skype та багато інших.

Зараз месенджери оновились і отримали безліч нових функцій і вийшли на новий рівень завдяки тому, що інтернет набагато доступніший. Набрали популярність такі месенджери як Facebook Messenger, Telegram, Discord,

WhatsApp, Viber та інші. Серед молоді найпопулярнішими є Telegram та Discord. Згідно з statista.com маємо такі дані:

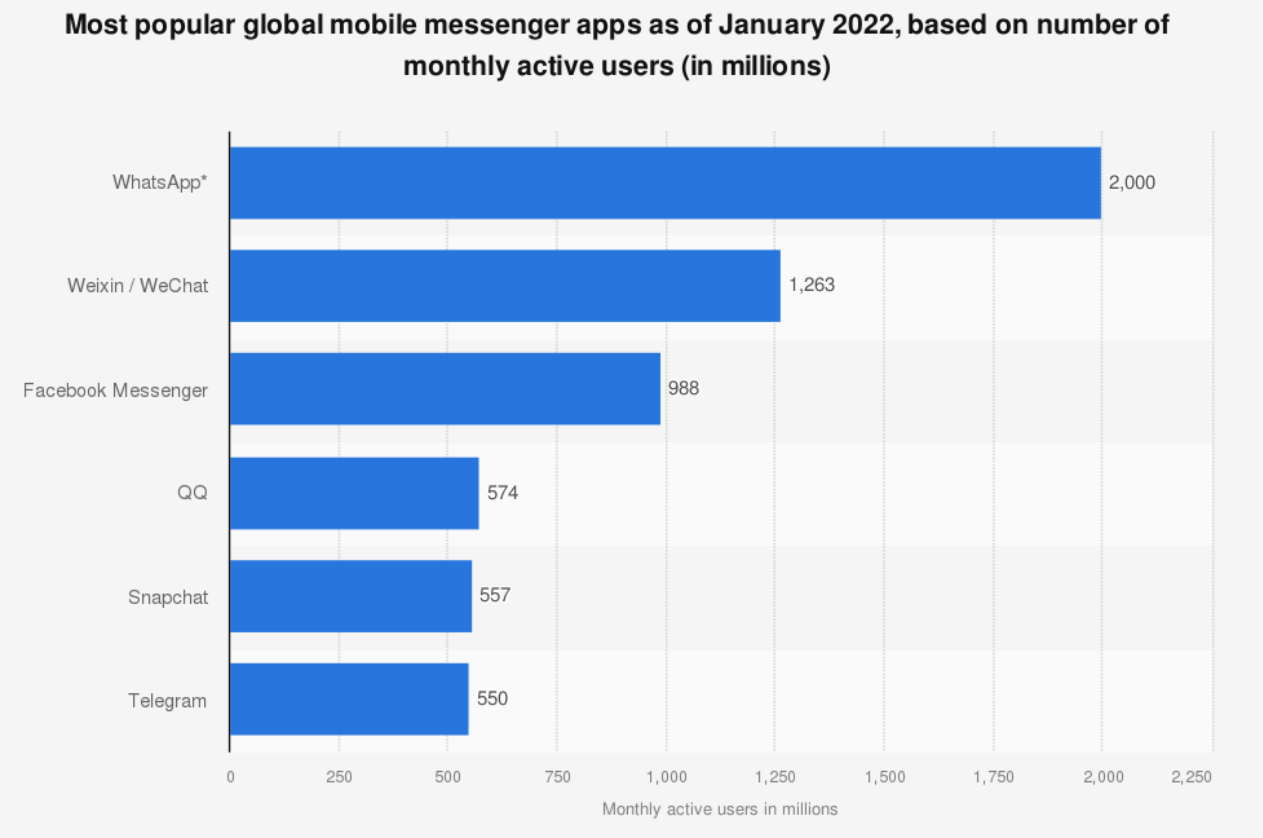


Рис 1.1 – Рейтинг популярності месенджерів за січень 2022 року [1]

На цьому графіку ми можемо побачити кількість щомісячних активних користувачів, згідно з якою складається такий рейтинг:

1. WhatsApp – 2 млрд користувачів
2. Weixin / WeChat – 1,263 млрд користувачів
3. Facebook Messenger – 988 млн користувачів
4. QQ – 574 млн користувачів
5. Snapchat – 557 млн користувачів
6. Telegram – 550 млн користувачів

Проте в Україні цей рейтинг виглядає дещо інакше. У вересні 2021 року Київський міжнародний інститут соціології провів власне всеукраїнське

опитування, у якому було опитано 2002 мешканців усіх регіонів України(крім АР Крим), яким більше 18 років. Згідно з цим опитуванням, рейтинг найпопулярніших месенджерів в Україні виглядає наступним чином:

Використання можливостей для обміну повідомленнями з мобільного телефону

Серед усіх респондентів, що мають мобільний телефон

Мобільний застосунок	%
Viber (Вайбер чи вібер)	73,6
WhatsApp (Вотсап)	25,3
Мессенджер Фейсбуку	42,7
Телеграм	31,6
Твіттер	4,9
Сігнал	3,8
Скайп	11,3
Інстаграм	26,4
Тік Ток	9,1
Інше (ЗАПИШІТЬ)_____	2,6
Нічим з перерахованого	18,7
ВАЖКО СКАЗАТИ/ ВІДМОВА ВІДПОВІДАТИ	0,8

Рис 1.2 – Рейтинг популярності месенджерів в Україні за 2021 рік [2]

Серед усіх цих месенджерів я би виділив Telegram. Telegram – це кросплатформова клауд-система миттєвого обміну повідомленнями, яка дозволяє користувачам обмінюватися текстовими, голосовими та відеоповідомленнями, фотографіями та файлами багатьох форматів(у додатку немає попереднього перегляду, тому сервіс приймає всі, без винятку, типи файлів). Також має функції аудіо- та відеодзвінків або конференцій у чатах та каналах. Telegram доступний на смартфонах та мобільних пристроях, працюючих на базі ОС Android або Apple iOS, ПК та лаптопів з ОС Windows, macOS та операційними системами на основі Linux. Також існує браузерна онлайн-версія месенджера. Програма локалізована

на більшість популярних мов світу, у тому числі і на українську. Цей месенджер має багато переваг над конкурентами, починаючи з відсутності реклами та унікальних стікерів, яких у Телеграмі безліч і всі вони безкоштовні, і закінчуючи конфіденційністю. Так як це клауд-месенджер, усі повідомлення та дані користувачів зберігаються у хмарі, яка може зберігати величезну кількість інформації. Але все ж таки основною особливістю, якої не можуть надати конкуренти Telegram є саме конфіденційність. Спеціально для месенджера було створено протокол MTProto, який передбачає використання одразу декількох протоколів шифрування. Цей протокол забезпечує найвищий, серед усіх інших месенджерів, рівень безпеки та захисту від злому та крадіжки даних. Під час авторизації і аутентифікації користувача використовуються алгоритми RSA-2048, DH-2048 для шифрування, а під час передачі повідомлень протоколу в мережу шифрування відбувається за алгоритмом AES з ключем, відомим клієнту і серверу. Для забезпечення безпеки від перехоплення повідомлень, що пересилаються з боку сервера Telegram, використовується режимі «секретних» чатів (Secret Chats), який з'явився 8 жовтня 2013 року. При цьому режимі шифрування відправник і одержувач використовують спільний лише для них ключ, із застосуванням алгоритму AES-256 у режимі IGE (англ. Infinite Garble Extension) для повідомлень, які пересилаються. Такий метод шифрування називається шифруванням end-to-end. На відміну від звичайного режиму переписки, повідомлення секретних чатів розшифровуються не сервером, а історія листування зберігається лише на двох пристроях, між якими було створено секретний чат.

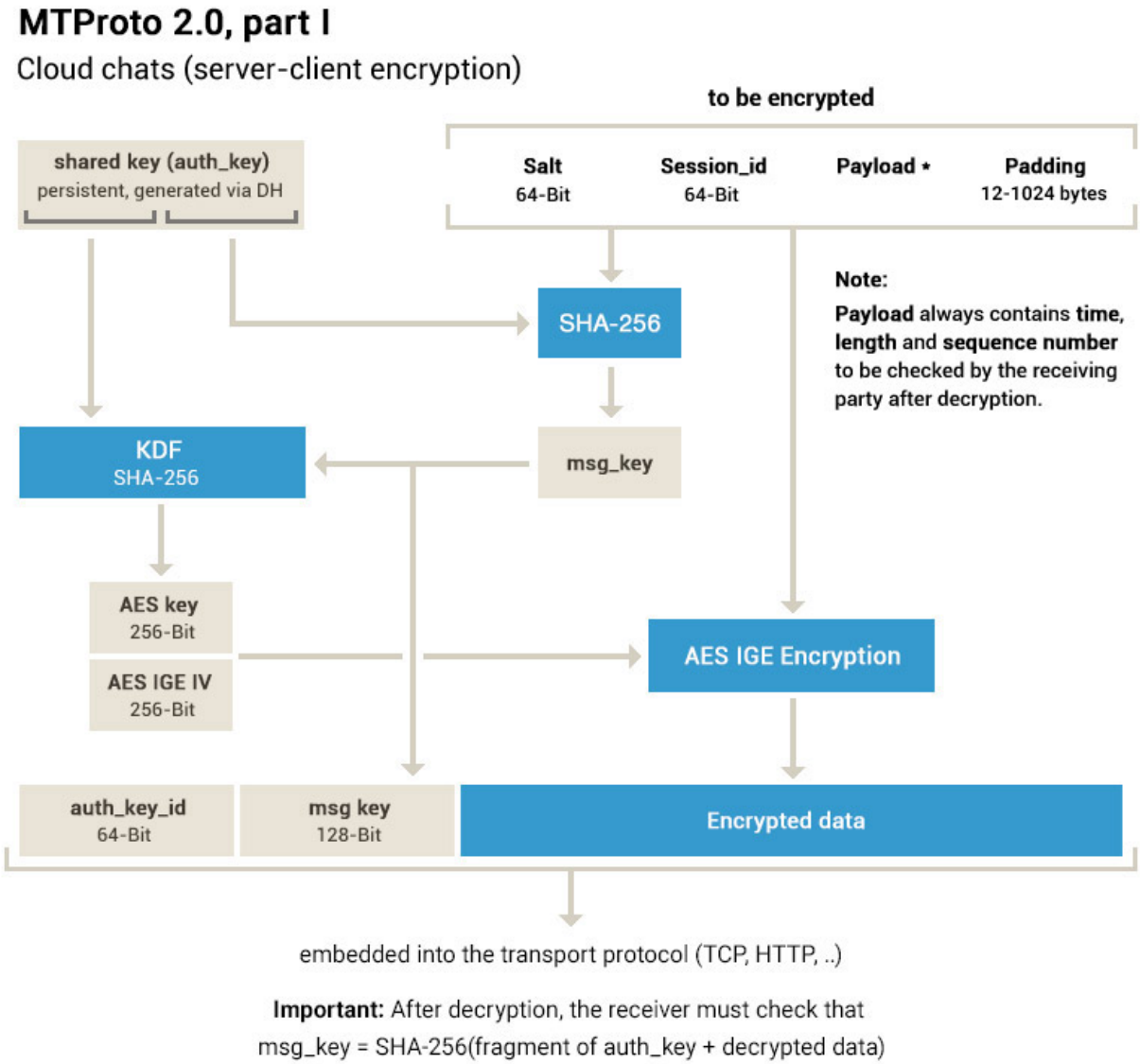


Рис 1.3 – Схема роботи алгоритму MTProto [3]

Нижче буде приведено зображення інтерфейсів різних версій Telegram для різних платформ: для IOS, Windows та Linux.

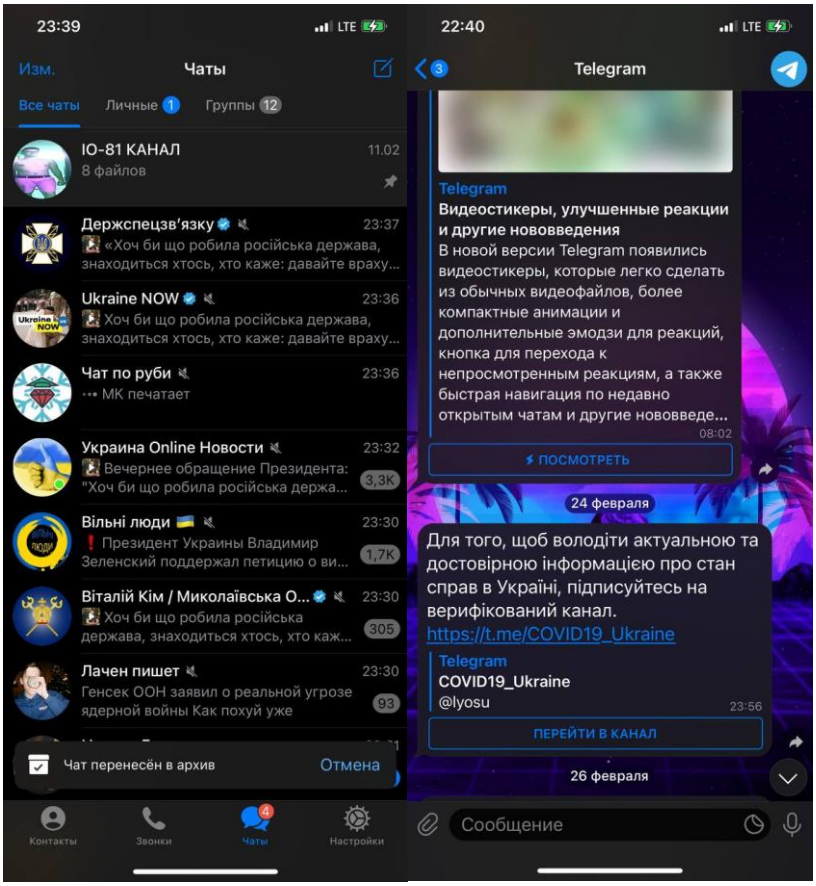


Рис 1.4 – Інтерфейс для смартфона з IOS

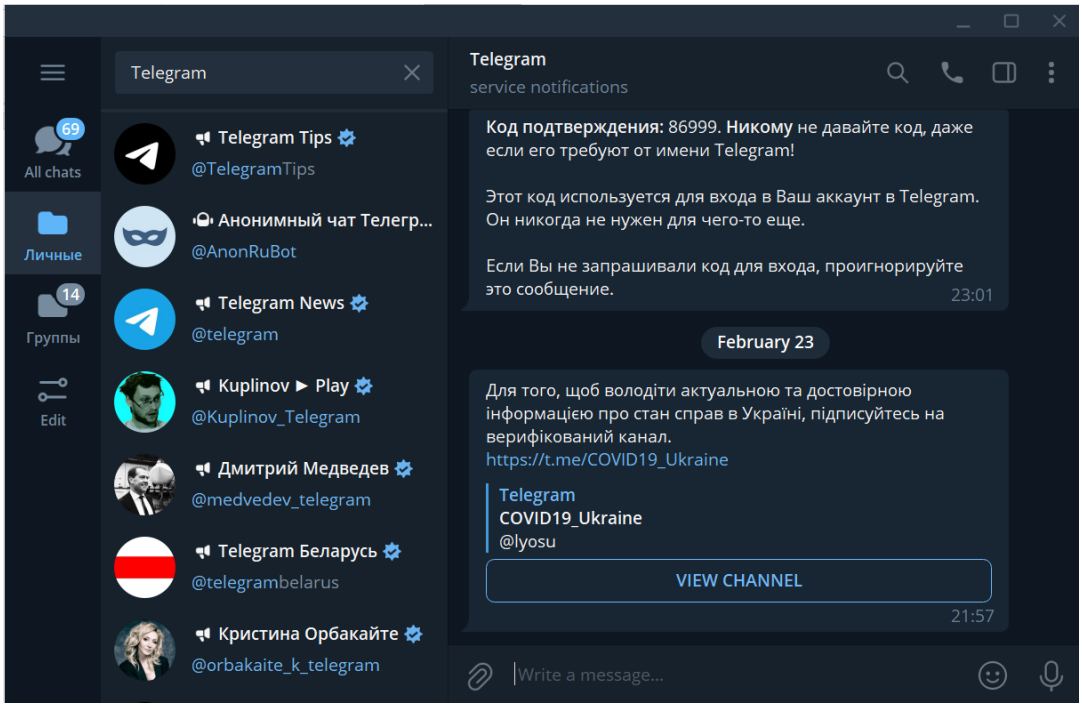


Рис 1.5 – Інтерфейс десктопної версії для пристроїв з ОС Windows

1.2 Чат-боти

Чат-бот − це комп'ютерна програма, за допомогою якої можливо здійснювати комунікацію найчастіше в текстовому форматі, розроблена на основі технологій машинного навчання та нейромереж. Комунікація з такою програмою здійснюється у системі «людина» — «комп’ютер». Програма штучного інтелекту імітує інтерактивну розмову з людиною за допомогою заздалегідь розрахованих, ключових фраз користувача, текстових або ж слухових сигналів. Чат-боти можуть бути використані лише для виконання конкретно визначених завдань, тобто для отримання довідкової інформації, виконання розрахунків або ж задля розваги, тощо.

Перший прототип чат-боту був створений у 1966 році професором Джозефом Вейценбаумом. Програма, розроблена професором була першим у світі віртуальним співрозмовником, вона імітувала діалог з психотерапевтом та реалізовувала техніку активного слухання, а названа була на честь героїні п’єси Бернарда Шоу, Елізи Дулітл, яку навчали мові людей «вищого суспільства». Цей проєкт став успішним, а після нього стало з’являтися ще більше аналогів Елізи.

Рис 1.6 – Інтерфейс програми ELIZA Джозефа Вейценбаума [4]

Саме термін «чатер-бот» (англ. ChatterBot) вперше ввів інформатик та винахідник пошукової системи і веб-порталу Lycos, Майкл Маулдін. Він створив першого веб-бота Julia у 1994 році і використав цей термін, щоб описати розмовні програми, які дають можливість спілкуватися чатерам, тобто, людині і чатботу, який наділено штучним інтелектом.

У наші дні чат-боти складають частину віртуальних помічників, наприклад Google Assistant, Alexa, Siri, Cortana та інші. Вони доступні на вебсайтах багатьох організацій та на платформах обміну миттєвими повідомленнями. Одними з найпопулярніших таких платформ є месенджери Telegram та Viber.

Чат-боти за сферою застосування поділяють на:

- p2p — peer-to-peer або рівний до рівного, для персональних комунікацій (для особистого спілкування)
- b2c — business-to-consumer тобто бізнес для споживача (для підтримки клієнтів на веб-сайті та в мобільних додатках)

Більшість чат-ботів не має налаштувань на машинне навчання, більшість з них пропонує користувачу обрати серед запропонованих, заздалегідь продуманих і підготовлених варіантів. Тобто зазвичай користувач не вводить текст до вікна чату, а отримує інформацію або виконує інші задачі шляхом вибору однієї із запропонованих відповідей або категорій. Натомість більш складні боти можуть надавати можливість вводу тексту і отримування інформації відповідно до запиту користувача. Віртуальні чат-боти мають безліч сценаріїв використання, наприклад вони дозволяють робити онлайн-замовлення, моніторити щось, слухати музику, робити грошові перекази та спілкуватися в режимі онлайн, існують чат-боти поліції, муніципальні чат-боти та чат-боти для працевлаштування. Окремої уваги у цей час заслуговує бот «Народний месник», створений

кіберполіцією. Завдяки цьому боту будь-який власник смартфону з доступом в інтернет може цілком безпечно передати інформацію про розташування, та кількість сил окупантів до рук кіберполіції та збройних сил з метою подальшої ліквідації цих окупантів. Таким чином звичайний бот може суттєво допомогти у веденні бойових дій та у відстоюванні суверенітету.

1.3 Огляд існуючих реалізацій

1.3.1 METAR Info Bot

Даний бот показує інформацію про погодні умови у різних аеропортах світу у форматі METAR, також бот може показати точний час, але тільки у московському часовому поясі. METeorological Aerodrome Report або ж Meteorological Terminal Aviation Routine Weather Report (METAR) – це авіаційний метеорологічний код для передачі інформації про погоду на аеродромі. Також це кодова назва регулярного зведення, складеного в однойменному коді. Ці зведення містять дані про погоду з урахуванням авіаційної специфіки. [30]

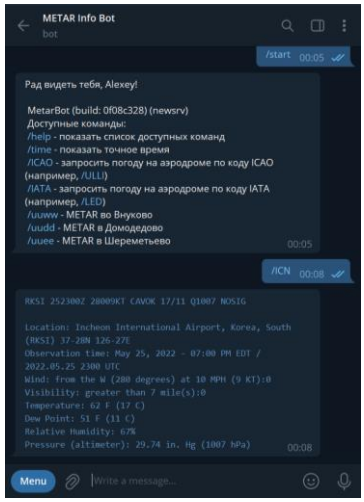


Рис 1.7 – Інтерфейс METAR Info Bot

Щоб отримати дані про погоду у необхідному вам аеропорті. Необхідно ввести унікальний код аеропорту згідно з основними інтернаціональними системами кодування аеропортів (ICAO або IATA) як команду, тобто ввести спеціальний символ «/» і після нього код аеропорту як показано на скріншоті. Підтримує тільки російську мову та не має ніяких налаштувань.

1.3.2 «Просто погода»

Наступний бот може показувати погоду у будь-якій локації, має більший функціонал та більш зручний користувацький інтерфейс, а саме меню з кнопками, завдяки яким здійснюється взаємодія з ботом. Присутня реклама.

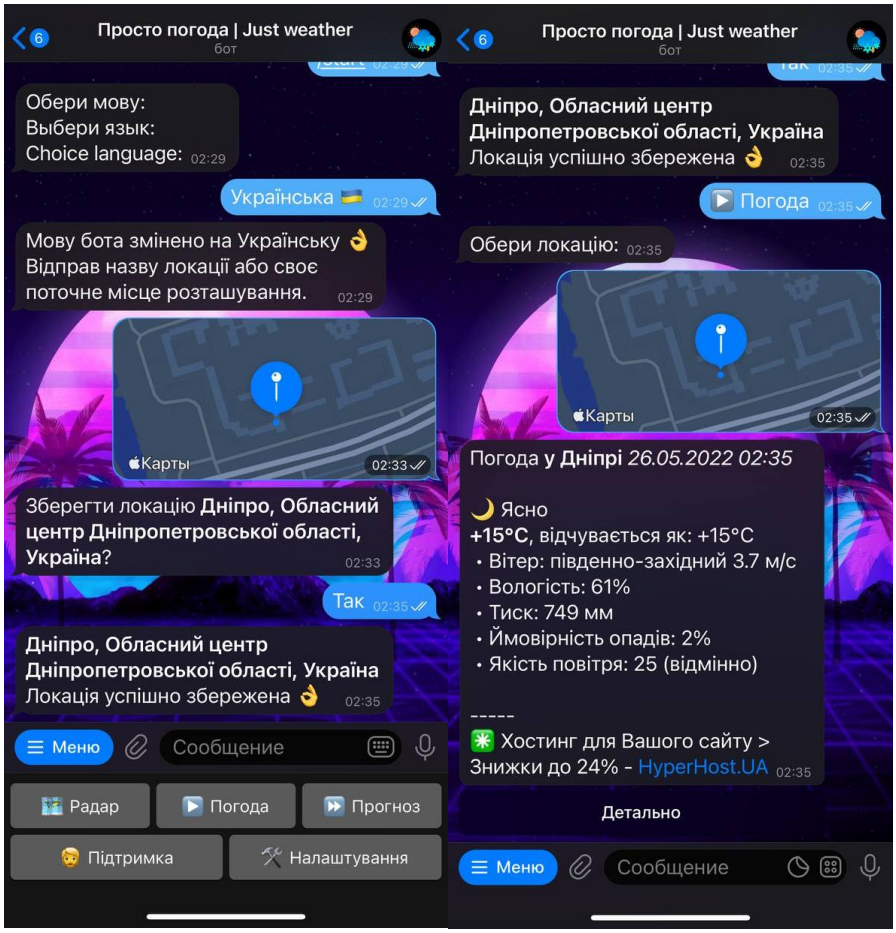


Рис 1.8 – Інтерфейс боту «Просто погода»

«Просто погода» підтримує 3 мови, а саме українську, російську та англійську. Може показувати прогноз як і на поточний день, так і на тиждень вперед. Відображає індекс якості повітря та має платну функцію радару, яка надає доступ до анімованої мапи поточної погоди з прогнозом на годину вперед. Щоб отримати прогноз треба обрати в зручному меню одну з доданих локацій (бот дозволяє зберігати до 5 локацій для швидкого використання), для того щоб додати локацію достатньо прислати її назву або надіслати боту свою поточну геолокацію, після чого вона стане доступною для вибору. Після вибору бот відправить прогноз, це значно простіше ніж у попередньому боті. Також можна налаштувати сповіщення з погодою на сьогоднішній або наступний день. Бот підтримує inline-режим, тобто його можна використовувати в будь-якому чаті. Для цього треба додати бота до чату як звичайного користувача, ввести @SynoptykBot, через пробіл ввести локацію та обрати потрібну з запропонованих. Також можна налаштувати сповіщення з прогнозами для чатів та каналів. Користувачі мають можливість залишити повідомлення автору за допомогою команди /feedback, також існує група підтримки, якщо виникнуть питання.

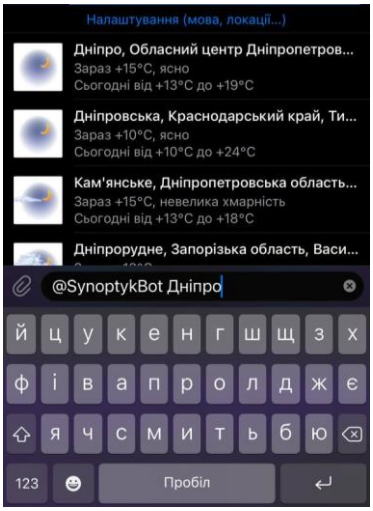


Рис 1.9 – Приклад використання inline-режиму

1.3.3 MeteoBot

MeteoBot надає прогноз у будь-якому місці на сьогоднішній або завтрашній день і також на 5 або 10 днів вперед. Підтримує лише російську та англійську мови. Взаємодія з ботом відбувається за допомогою кнопок у зручному та зрозумілому меню.

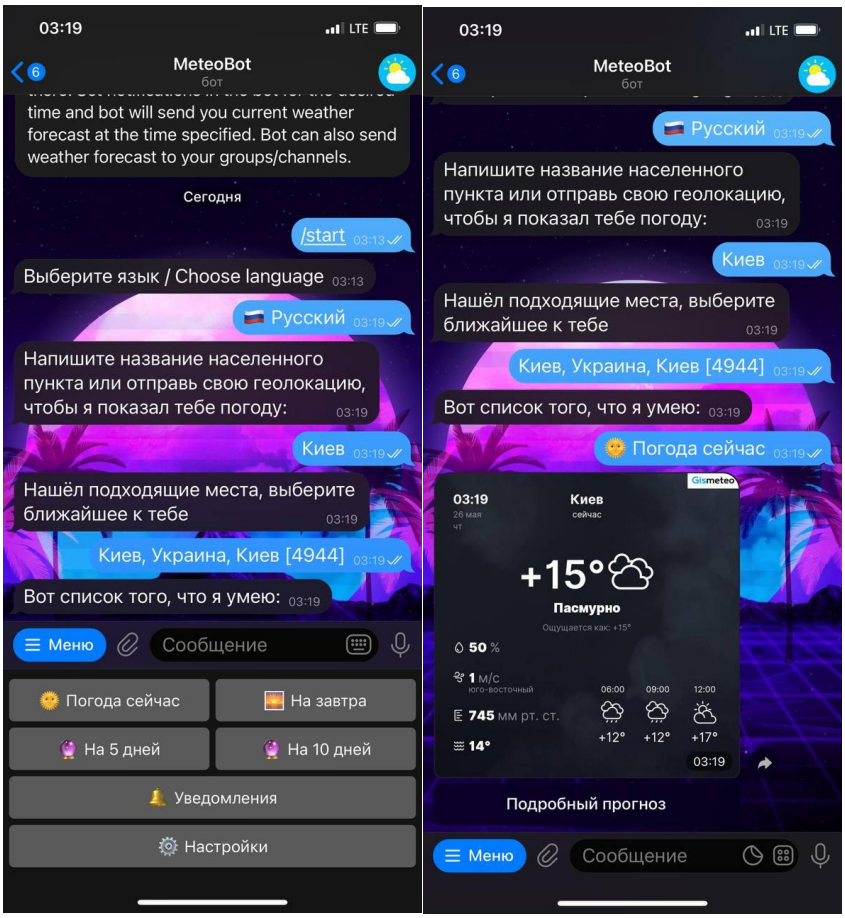


Рис 1.10 – Інтерфейс MeteoBot

Алгоритм отримання прогнозу нам вже відомий. Відправляємо локацію та обираємо потрібну, після чого з’являється меню, в якому можна отримати прогноз на сьогодні або на майбутнє, а ще можна обрати пункти «Сповіщення» та «Налаштування». Можна обрати тип отримуваних сповіщень (прогноз на сьогодні, на завтра, на 5 або 10 днів) і налаштувати самого бота. З меню налаштувань можна змінити місто та мову, додати бота

до чата або каналу, змінити формат часу та одиниці виміру температури, скорості вітру, опадів тощо. Відповідь з прогнозом надається у форматі, легкому для візуального сприйняття, при бажанні можна отримати більш детальний прогноз.

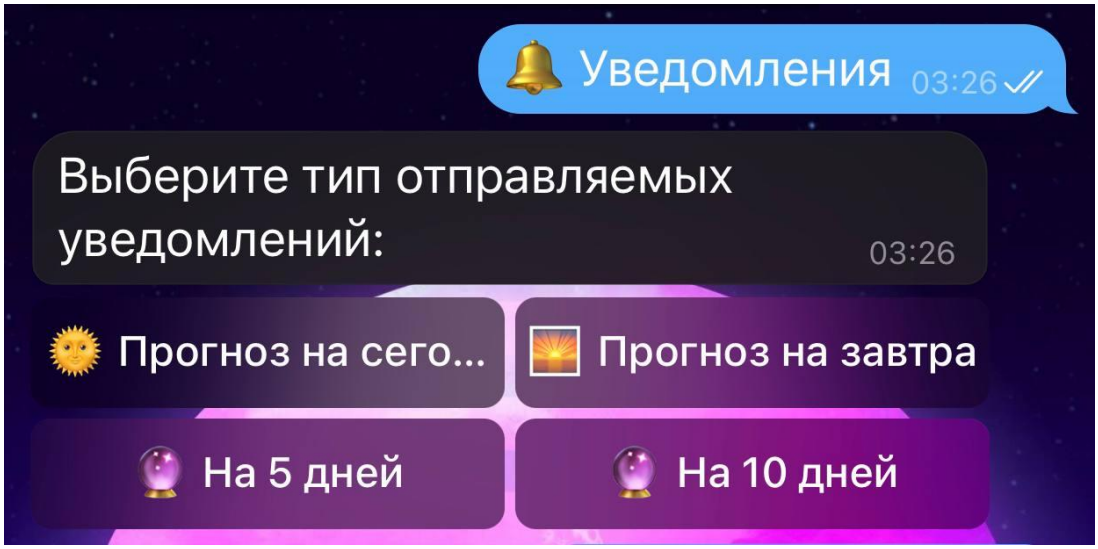


Рис 1.11 – Налаштування сповіщувань MeteoBot

1.3.4 «Погоднік»

Погоднік так само інформує про погоду розсилаючи повідомлення, має зручне та більш просте меню з кнопками, за допомогою яких користувач має можливість здійснювати взаємодію з ботом. У меню можна отримати прогноз (на сьогодні, на завтра, на 7 або 10 днів та на місяць), залишити відгук або відкрити налаштування. У користувача є можливість обрати режим сповіщень, при якому бот буде відправляти по одному сповіщенню про погоду вранці та вдень або ж обрати годину, о котрій користувач хоче отримувати повідомлення. Також налаштування дозволяють змінити місто або мову. Бот підтримує українську, російську та англійську. Інтерфейс дуже простий та зрозумілий для звичайного користувача.

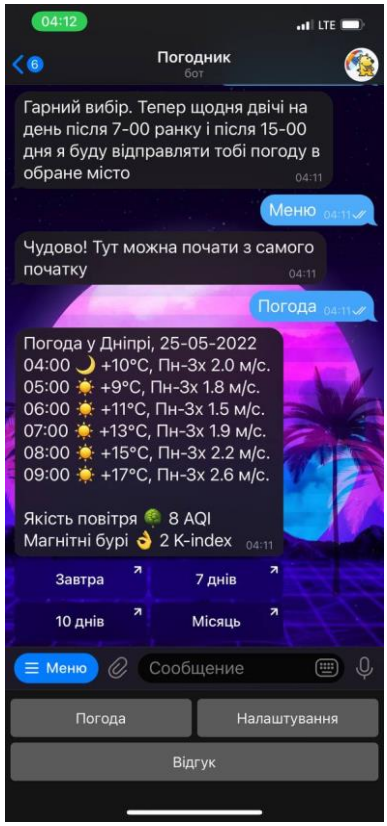


Рис 1.12 – Інтерфейс боту «Погоднік»

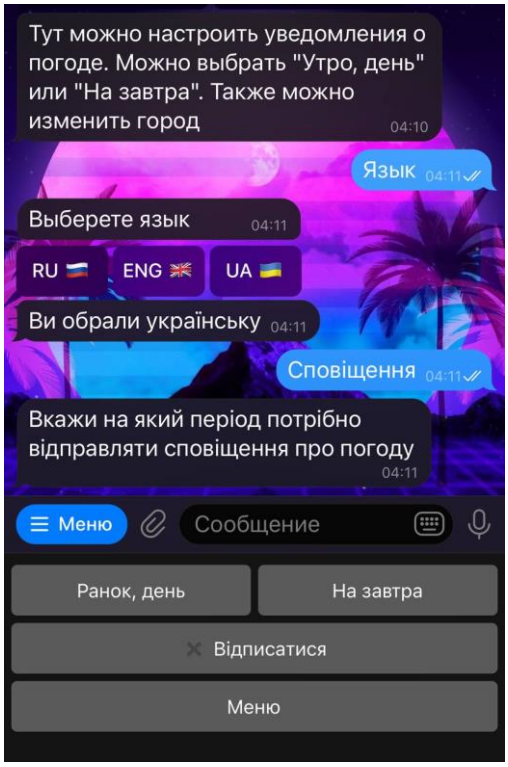


Рис 1.13 – Меню налаштувань боту «Погоднік»

Погоднік має ще пару цікавих функцій, а саме він надсилає сповіщення про геомагнітні бурі та під час дощу сповіщує коли він закінчиться.

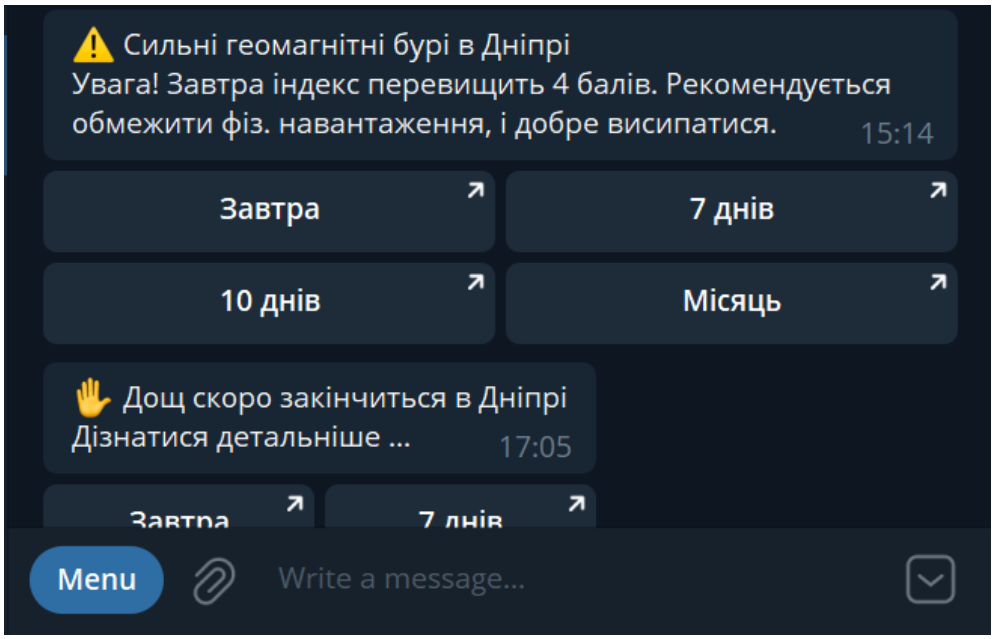


Рис 1.14 – Приклад сповіщувань від «Погоднік»

1.3.5 TheLair Weather

Бот показує детальний прогноз погоди на 5 днів. Взаємодія здійснюється за допомогою спеціальних команд, список яких можна дізнатись написавши команду «/help». Особливістю цього бота є те що крім звичайних даних про погоду (таких як температура, опади, вологість і т.д.) він показує пориви вітру і щільність хмар. Дозволяє у налаштуваннях змінити мову та одиниці виміру для швидкості вітру та температури. Підтримує російську та англійську. Відповідь надає графічним зображенням, яке приведено нижче.

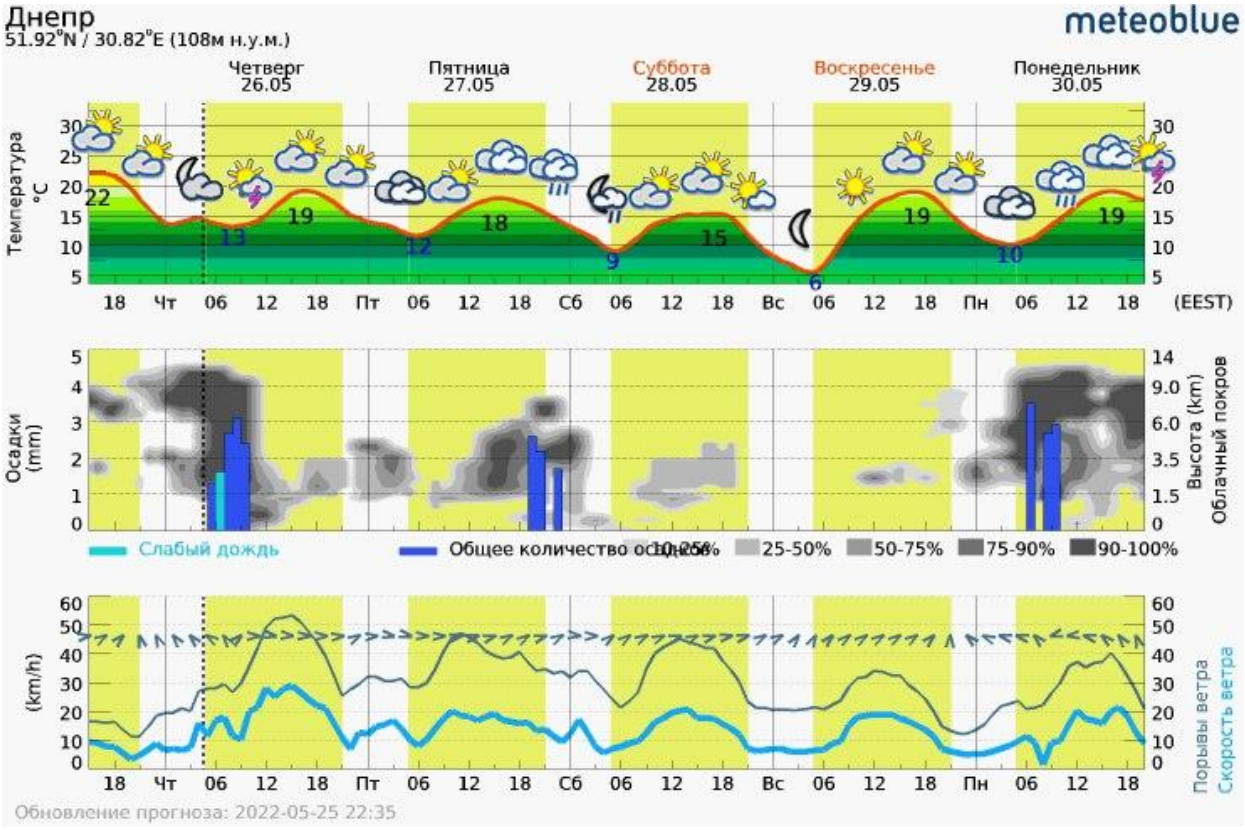


Рис 1.15 – Приклад прогнозу від TheLair Weather

1.3.6 «Прогноз одягу»

«Прогноз одягу» сильно відрізняється від попередніх варіантів реалізацій, подачею інформації користувачу, він надає дані про прогноз погоди не у звичному для багатьох форматі. Бот дуже простий, надає рекомендації щодо одягу зважаючи на стан погоди у обраному місті, не дає можливості якось налаштувати себе. Користуватися ботом можна за допомогою команди «/start», після якої необхідно написати назву міста (рекомендовано писати назву англійською, щоб місто визначалося точніше), у відповідь бот каже що краще сьогодні вдіти. Підтримує лише російську мову і містить рекламу. На даний момент бот припинив підтримку і не працює.

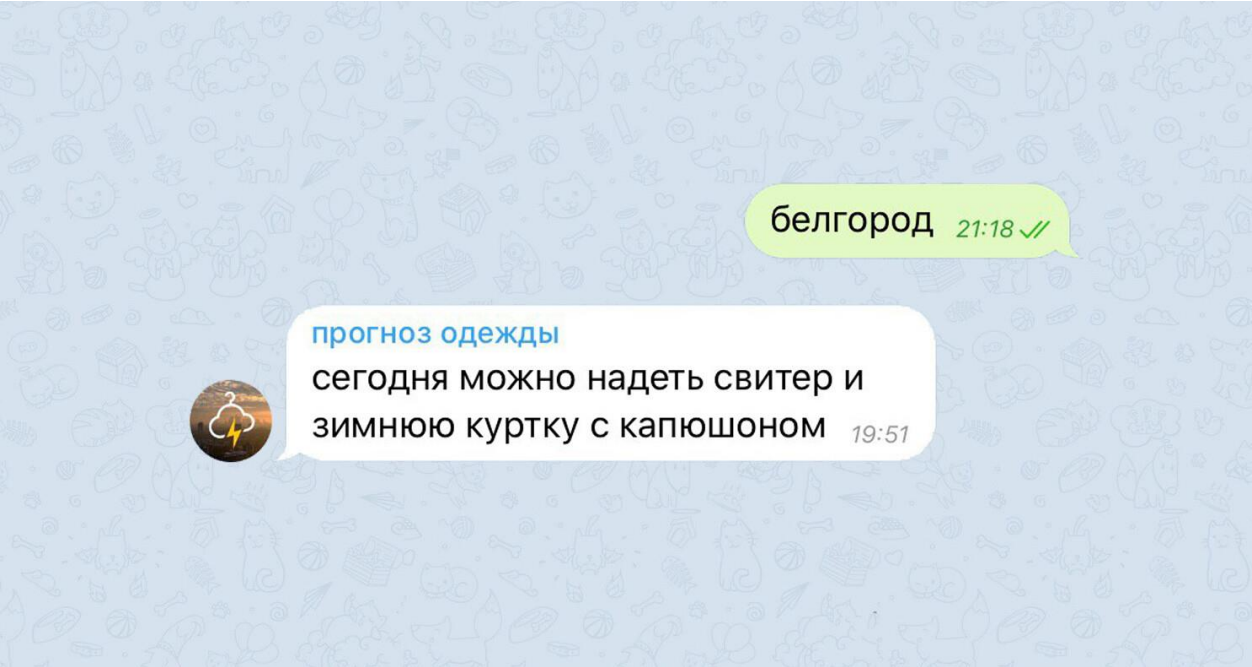


Рис 1.16 – Приклад прогнозу одягу [5]

Бот було розроблено за допомогою конструктора «Botmother», який дозволяє робити прості API-запити. Програма працює за такою схемою:

1. Бот відправляє API-запит до погодного сервісу та отримує у відповідь дані про погоду у місті, яке ввів користувач
2. Розраховує температуру за показником «відчувається як» на кожну годину
3. Визначає середні значення для ранку, дня та вечора
4. По цим значенням обирає одяг з каталогу, складеного на основі спостережень розробника
5. Надсилає повідомлення з прогнозом одягу

Серед недоліків можна відзначити наявність реклами і обмежений функціонал. Важливо ще звернути увагу на те що логіка програми побудована з простих інструментів конструктора, тому має дуже сильно нагромаджену структуру при дуже скромному функціоналі.

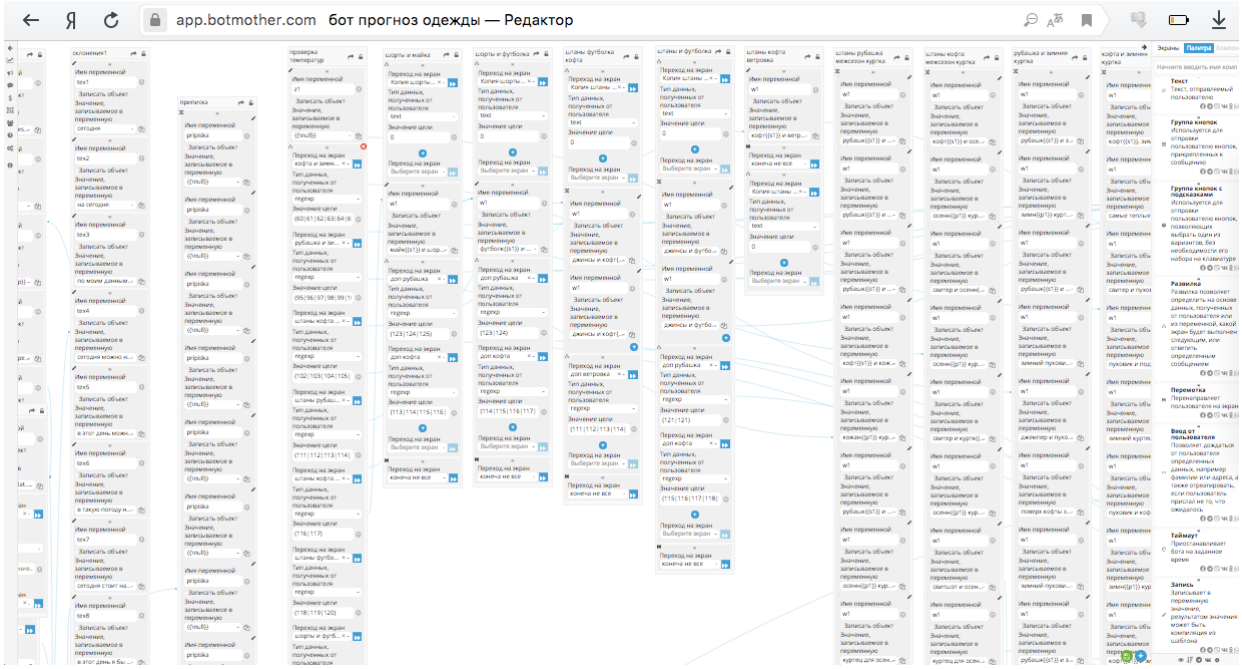


Рис 1.17 – Структура боту «Прогноз одягу» [5]

1.4 Вимоги до програми

1. Сумісність з одним із популярних месенджерів. Програма має бути інтегрована в месенджер, для того щоб користувач мав можливість зручно користуватися ботом прямо у месенджері.
2. Коректність інформації. Програма має надавати актуальну та достатньо точну інформацію про прогноз погоди, які вона буде отримувати з достовірного ресурсу.
3. Сприйняття інформації. Виведені результати мають бути достатньо легкими для сприйняття, щоб звичайний користувач міг без проблем їх читати.
4. Взаємодія з програмою. Інтерфейс управління має бути простим та інтуїтивно зрозумілим, щоб звичайний користувач міг без проблем користуватись функціями бота.

Висновок до розділу 1

Дослідівши предметну область в першому розділі стало зрозуміло, що ідея чат-боту з прогнозом погоди є доволі актуальною. Також, розглянувши та дослідивши існуючі програмні рішення, було сформовано вимоги до майбутнього проєкту чат-боту з інформуванням про стан погоди. Найкращою платформою для цього боту буде саме Telegram, адже більшість продвинутих користувачів, які знають про існування чат-ботів та можуть активно користуватися ними, віддають перевагу саме цьому месенджеру, через його захищеність та конфіденційність.

РОЗДІЛ 2. ВИБІР ІНСТРУМЕНТІВ РЕАЛІЗАЦІЇ

2.1 Вибір мови програмування

2.1.1 Ruby

Ruby – це високорівнева динамічна, інтерпретована та повністю об’єктно-орієнтована мова програмування. Особливістю Ruby є те що майже усе являє собою об’єкт, тобто всі типи даних і класи у Ruby – об'єкти, а змінні містять посилання на ці об’єкти. Кожна функція – це метод. На відміну від доволі поширених мов програмування, присвоєння у Ruby це не передає значення, а копіює посилання на об'єкти. Розробка на Ruby потребує набагато менше часу, тож Ruby ідеально підходить для стартапів. Створення MVP на Ruby значно швидше, ніж на Java. Мовою Ruby було створено такі успішні проєкти як Airbnb, Kickstarter та Github.

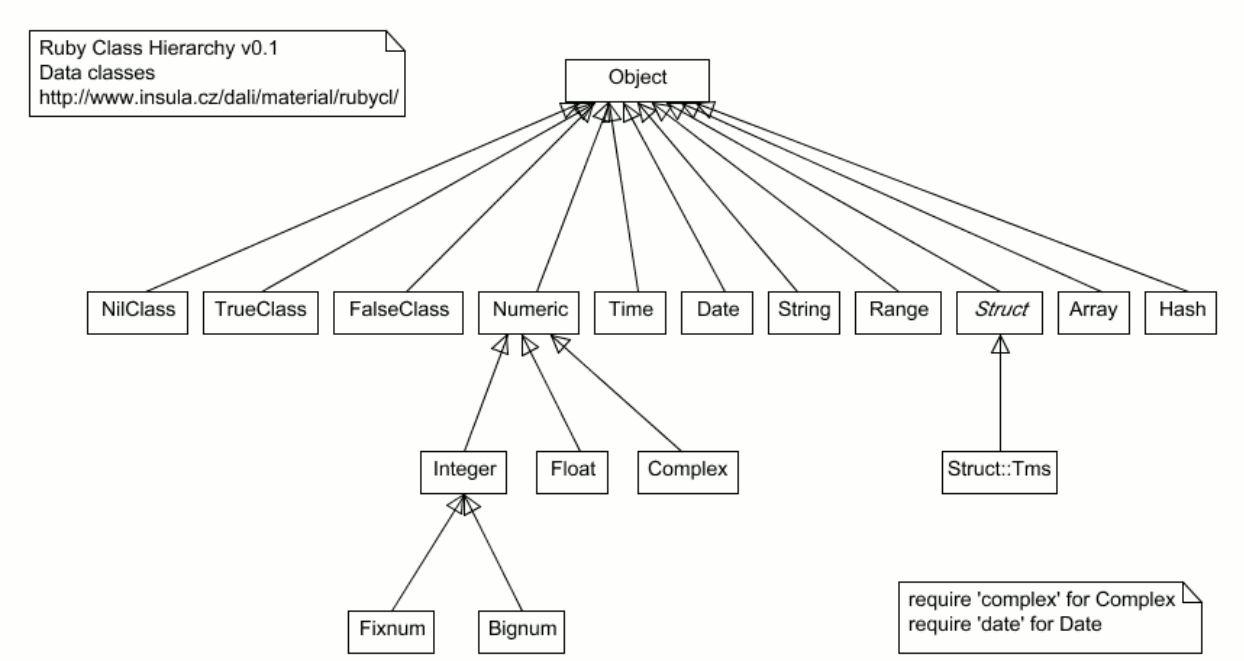


Рис 2.1 – Ієрархія усіх абстрактних класів Ruby [6]

Перша версія Ruby була видана у відкритий доступ у 1995 році. Мова була розроблена японським вченим і програмістом Юкіхіро Мацумото. 24

лютого 1993 року він почав працвати над Ruby, з метою створити об'єктно-орієнтовану, інтерпретовану та легку в розробці мову програмування. Ruby була створена під впливом таких мов як Python та Ада, тож має простий і лаконічний синтакс.

Особливості Ruby:

- Відкритий код.
- Підтримка поліморфізму.
- Мультипарадигмова мова, тобто підтримує об'єктно-орієнтовну, функціональну та процедурну парадигми програмування
- Підтримує блоки коду, які можна використовувати в методах або перетворювати в цикли. Для створення блоку можна використовувати конструкції {…} або do…end, весь код, який знаходиться всередені називається блоком.
- У Ruby вже реалізовано багато функцій та шаблонів програмування. Наприклад не треба заново створювати метод сортування, можна використати один зі створених методів, додавши його до конкретного об'єкту.
- Підтримка динамічної типізації
- MixIn – концепція, заснована в Ruby на основі механізмів модулів, щоб замінити множинну спадковість.
- Наявність великої кількості користувацьких бібліотек (гемів), у тому числі і для розробки телеграм-ботів.

2.1.2 Python

Python – високорівнева динамічна об’єктно-орієнтована мова програмування. Названа на честь британського комедійного шоу «Літаючий цирк Монті Пайтона» та розроблена нідерландським програмістом Гвідо ван Россумом в 1990 році. Python підтримується на усіх відомих платформах, тим паче і на Windows, MacOS, Android, iOS та всіх UNIX системах. Метою створення Python була необхідність у розширюваній скриптовій мова для розподіленої ОС Amoeba. Проєкт розроблявся під впливом таких мов як ABC (розробник Python також був причетний до розробки цієї мови), C та C++. У порівнянні з Ruby, Python не дає можливості модифікувати вбудовані класи, тобто str, int, float та інші, але це ж обмеження, дозволяє Python швидше працювати, так як споживання оперативної пам'яті значно менше.

Існує також певна філософія програмування, вона називається «The Zen of Python». З основними положеннями цієї філософії можна ознайомитись в інтерпретаторі Python, скориставшись командою import this. Тім Пейтерс вважається автором цієї концепції. [10]

Основні можливості Python:

- Підтримка ООП та парадигми функціонального програмування.
- Підтримка модулів (бібліотек або застосунків), модулі підключаються за допомогою оператору import.
- Вбудовані деякі шаблони проєктування (ітератори, генератори, декоратори та інші).
- Багата стандартна бібліотека. У Python вбудовано багато інструментів для роботи з різними мережевими протоколами,

криптографічними алгоритмами, юніт-тестами, розробки кросплатформенних застосунків та багато іншого.

2.1.3 Go

Go (або golang) – компільована багатопоточна мова програмування, яка має вбудовані засоби для віддаленого керування пакунками. Golang було розроблено корпорацією Google. Розробку мови було розпочато у 2007 році, а вперше представили мову у 2009 році. Мова підтримується на операційних системах Android, Linux, Mac OS та Windows.

Проекту Go створювався з метою отримання мови, яка може поєднувати у собі легкість написання коду, швидкість розробки і захищеністю від помилок скриптових мов з високою продуктивністю компільованих. Синтаксис мови було створено на базі звичних елементів мови C з деякими запозиченнями з Python. Проєкт з самого початку розробляється орієнтовним на багатопоточне (або ж паралельне) програмування, тож націлений на ефективну роботу з багатоядерними системами.

Основні можливості Go:

- Підтримка ООП та парадигми функціонального програмування.
- Сувора статична типізація, на відміну від двох попередніх мов.
- Тип string з вбудованою підтримкою юнікоду.
- Автоматичне керування пам'яттю з прибиральником сміття. Оперативна пам'ять ПК звільняється від об'єктів, які програма не буде використовувати у майбутньому.
- Простий та лаконічний синтаксис.

- Перевизначення функцій та методів недоступно з міркувань надійності та ефективності компіляції.
- В Go вбудовані потоки, що забезпечує підтримку паралельного програмування. Потоки взаємодіють через канали та інші вбудовані засоби.

2.2 BotFather

BotFather – це бот, який створили розробники Telegram для реєстрації та керування іншими ботами. При створені та реєстрації кожного бота, йому присвоюється унікальний токен, за допомогою якого можна здійснювати керування ботом та його налаштування. Один користувач може зареєструвати до 20 ботів. Для створення нового бота достатньо вигадати назву та унікальний юзернейм для нього. Після успішної реєстрації, BotFather надає токен для нового бота, завдяки якому можна налаштовувати свого бота. Нижче приведено список команд для керування ботами.

Рис 2.2 – Усі команди BotFather

2.3 Вибір середовища розробки

2.3.1 RubyMine

RubyMine – це IDE для розробки ПЗ мовою Ruby та на RoR від компанії JetBrains. Це середовище розробки було створене на основі іншого середовища для різних мов – IntelliJ IDEA. Підтримує основні бібліотеки для Ruby-додатків. Дуже потужний платний інструмент з багатим функціоналом, має безкоштовний trial-період.

Основні переваги RubyMine:

- Зручний редактор коду з автодоповненням
- Аналіз коду з можливістю миттєвого виправлення
- Окрім Ruby підтримує HTML/HAML, CSS/Sass/Less та JavaScript/CoffeeScript
- Інтеграція з системами контролю версій (наприклад з Git) з більш зручним графічним інтерфейсом.

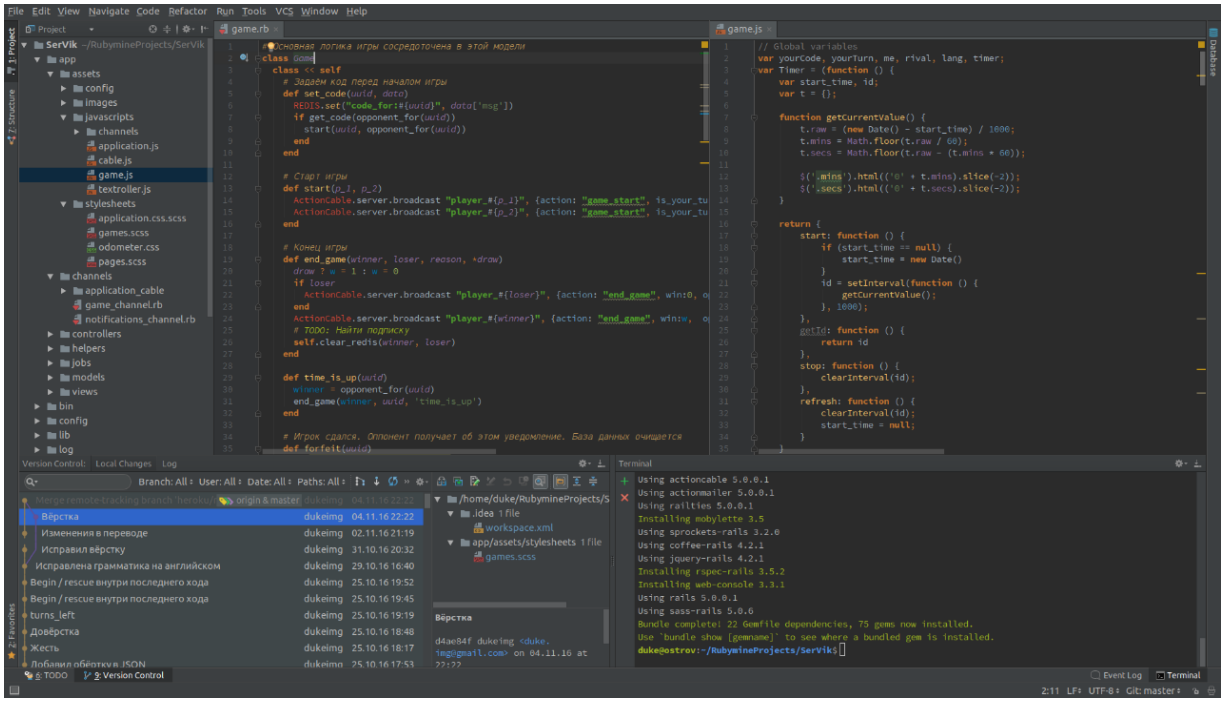


Рис 2.3 – Інтерфейс RubyMine [14]

2.3.2 Visual Studio Code

Visual Studio Code – це безкоштовний текстовий редактор коду, створений компанією Microsoft для Windows, Linux та macOS. Позіціонується як легкий редактор для кросплатформенної розробки веб- та хмарних додатків.

На відміну від IDE, текстові редактори більш швидкі та менш вимогливі до ресурсів, виграють у плані кастомізації та конфігурації під себе та безкоштовні. У VS Code можна встановити необхідні плагіни для того щоб отримати більшість функцій IDE (наприклад автодоповнення).

Особливості VS Code:

- Вбудований відладчик
- Підсвітка синтаксису та засоби для рефакторингу
- Інструменти та зручний інтерфейс для роботи з Git
- Дуже гнучкий в плані кастомізації (безліч користувацьких плагінів та файли конфігурації)

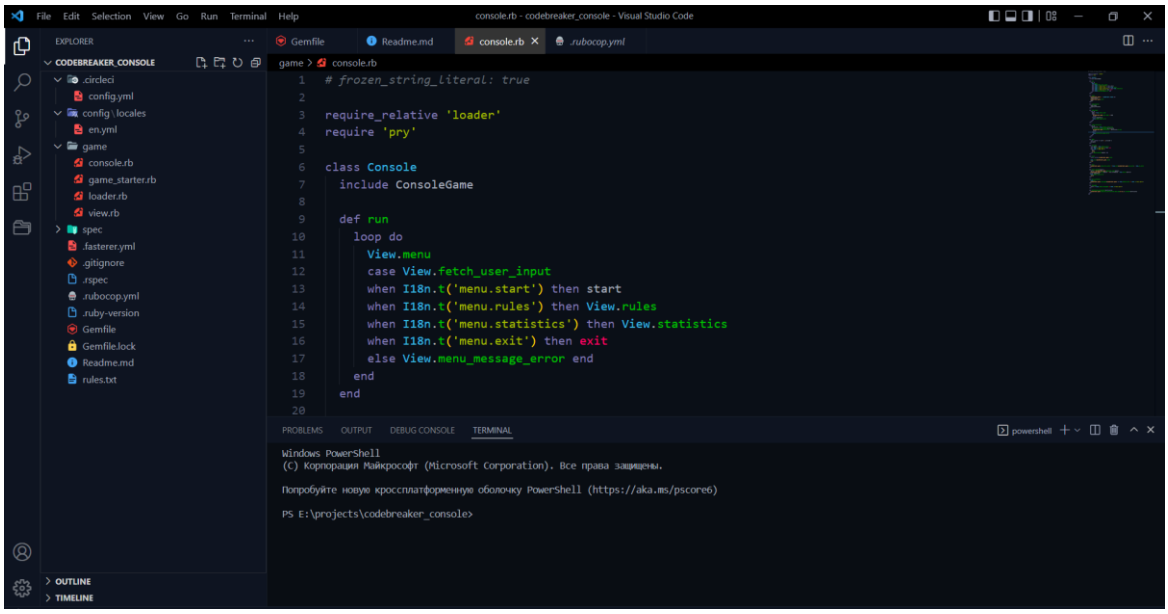


Рис 2.4 – Інтерфейс VS Code

2.4 OpenWeatherMap API

OpenWeatherMap – платний веб-сервіс, який надає API для доступу до даних про актуальний стан погоди або про прогнози. OpenWeatherMap розробляє прогноз погоди та погодні карти, наприклад карти хмарності та опадів, на основі обробленої інформації з погодних станцій. Основна ідеєя OpenWeatherMap – це використання даних з приватних погодних станцій, це допомагає зробити інформацію про погоду набагато точнішою, тож як наслідок, прогнози погоди теж стають набагато точнішими. OpenWeatherMap – це комерційна організація, тож використовує платний API, для надання доступу до даних про поточну погоду, прогнозу, інформації про вітер, хмари, атмосферні опади та тиск. Усі ці дані надаються у JSON, XML та HTML форматах. Є безкоштовна версія API з обмеженим функціоналом, яка надає інформацію тільки про поточний стан погоди. Інформація про поточний стан погоди оновлюються кожні 10 хвилин.

В OpenWeatherMap доступно 200.000 міст, в яких можна дізнатись інформацію про погоду, також ці дані можна отримати за географічними координатами.

Оплативши підписку на цей сервіс, можна отримати такі такі види прогнозів погоди:

- Похвилинний прогноз на 1 годину
- Погодинний прогноз на 4 дні
- Поденний прогноз на 16 днів
- Кліматичний прогноз на 30 днів

Якщо раптом так сталося, що ви маєте у своїй власності приватну метеостанцію, то ви маєте можливість без перешкод підключити її до OpenWeatherMap та відправляти дані зі своєї метеостанції у відповідності до API.

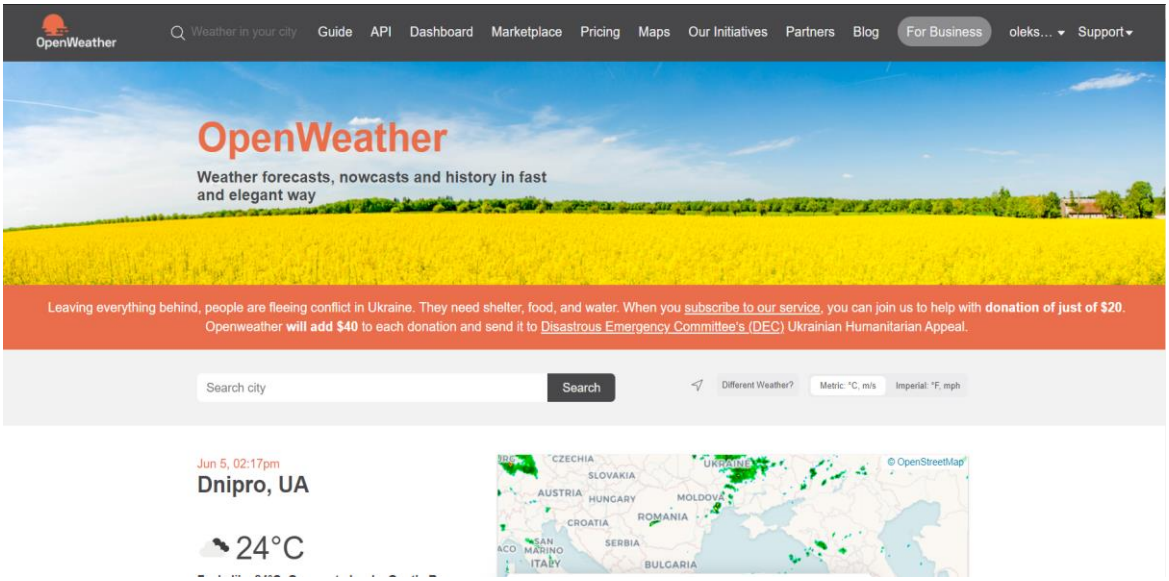


Рис 2.5 – Головна сторінка сайту OpenWeatherMap [17]

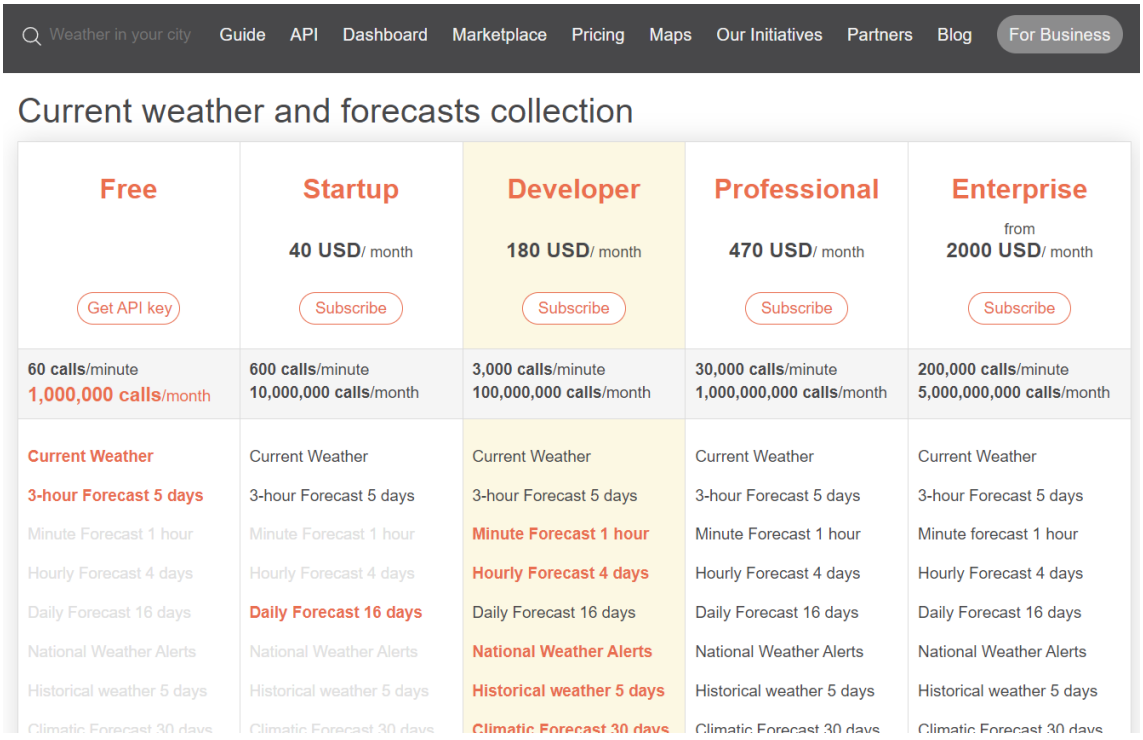


Рис 2.6 – Прайслист OpenWeatherMap [17]

2.5 Heroku

Heroku – це хмарна платформа-хостинг як послуга (PaaS), заснована на керованій контейнерній системі, з інтегрованими службами передачі даних та розвиненою екосистемою для розгортання та запуску додатків, яка підтримує ряд мов програмування. Heroku з'явилась у 2007 році і була однією з першихххмарних платформ. На початку свого існування мала підтримку лише однієї мови програмування – Ruby, але на сьогодні список мов, які підтримує Heroku значно розширився та налічує такі мови як Python, Scala, Golang, Node.js, Java, Clojure, та PHP. Heroku на своїх серверах використовують операційні системи Ubuntu або ж Debian. Платформа надає можливість завантажувати будь-яку програму, не переймаючись налаштуванням та конфігурацією серверної частини.

Heroku – це Platform as a Service, а це означає, що платформа працює як сервіс, тобто надає користувачу певні можливості та функції, доступ до систем та ПЗ. При цьому її інфраструктура повністю прихована, за користувача все роблять співробітники сервісу та більшість процесів автоматизовані. За безпеку, архітектуру та налаштування сервера відповідають спеціалісти платформи. Саме тому Heroku прекрасно підійде для:

- Розміщення додатків та веб-сервісів.
- Спрощення та прискорення циклу розробки.
- Зниження потреби у складній роботі із сервером.
- Роботи з навантаженими програмами.
- Швидкого масштабування проектів.

Додатки, які розміщуються на Heroku, працюють ізольовано від інших – вони укладені в спеціальні контейнери, які називаються диносами

(dynos). Такі контейнери дозволяють створити незалежне легковажне середовище і розгорнути у ньому додаток так, щоб налаштування його середовища не конфліктували з іншими. Одна програма може використовуватися декількома контейнерами, і проєкт може легко масштабуватися під завдання розробника.

Але як і в усіх сервісів у Heroku є нюанси. На ціну впливає кількість ресурсів, які використовує клієнт. Тому не завжди буде доцільно викроистовувати саме Heroku. При роботі з об'ємними хайлоад-проєктами дешевше буде забезпечувати роботу сервера, ніж використовувати цю платформу. Але для невеликого проєкту чи може стартапу Heroku буде чудовим вибором. Адже в Heroku є початковий безкоштовний тариф. Він дає користувачеві 550-1000 годин роботи диносів на місяць. У тарифі доступні два типи процесів і можливість додавати власні домени. Через 30 хвилин без активності сервіс «засинає», цього можна уникнути при виборі іншого тарифу.

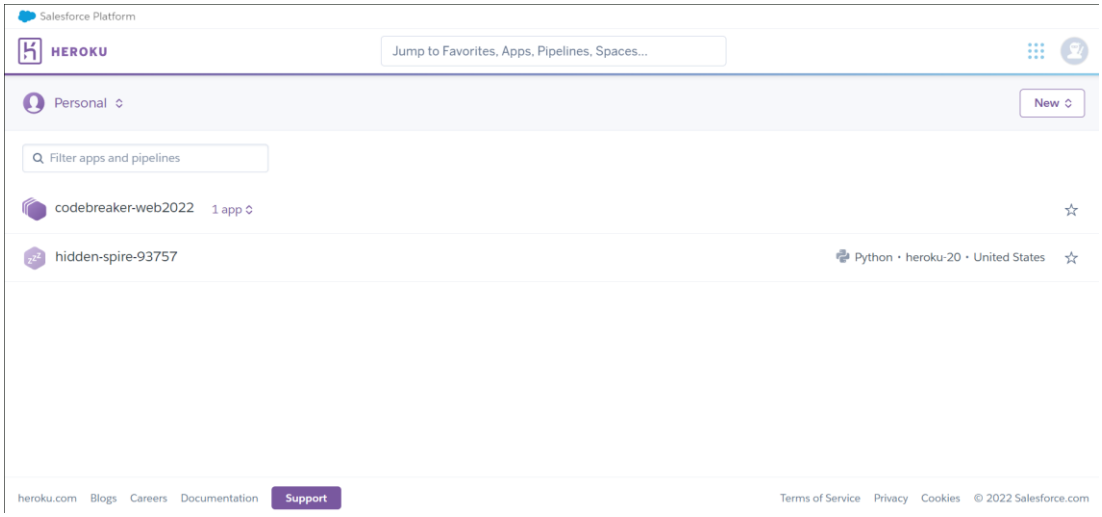


Рис 2.7 – Головна сторінка сайту Heroku [20]

Щоб почати користуватися Heroku треба зареєструватися на сайті Heroku (Рис 2.7) та обрати тариф. Для початку підійде стартовий

безкоштовний тариф, дорогі тарифи знадобляться для високонавантажених програм та комерційного використання. Після цього необхідно завантажити консольну утиліту Command Line Interface (CLI, також відома як Herokuapp) на сайті сервісу. Для її роботи потрібний встановлений Git. Пімля успішної інсталяції, відкрити командний рядок та ввести команду heroku login –i. Ця команда дозволить вам залогінитися, потрібно буде ввести дані вашого облікового запису. Перейти до папки, де зберігається програма, та ввести команду heroku create, після цього Heroku автоматично обробить програму. Також можна задеплоїти свою програму без інсталяції CLI, якщо у вашої програми є репозиторій Git.

1. **Висновок до розділу 2**

Серед трьох розглянутих та досліджених мов програмування, я вирішив все ж таки обрати мову Ruby тому, що я маю досвід програмування на цій мові, так як використовував її на своєму стажуванні, тож мені не прийдеться витрачати багато ресурсів на поглиблене вивчення іншої мови програмування.

Серед середовищ розробки вибір був зроблений на користь VS Code тому, що ця платформа є безкоштовною, менш вимогливою до ресурсів ПК а також має значно більше можливостей індивідуальної кастомізації для більш комфортної роботи з кодом. За рахунок плагінів VS Code значно гнучкіша за будь-який інший текстовий редактор коду або IDE.

РОЗДІЛ 3. ОПИС ЕТАПІВ РОЗРОБКИ

3.1 Підготовка до розробки

Для того щоб створити власний телеграм-бот, спочатку необхідно його зарyєструвати. Щоб зареєструвати бота в Telegram, потрібно скористатися ботом від розробників цього месенджеру – BotFather.

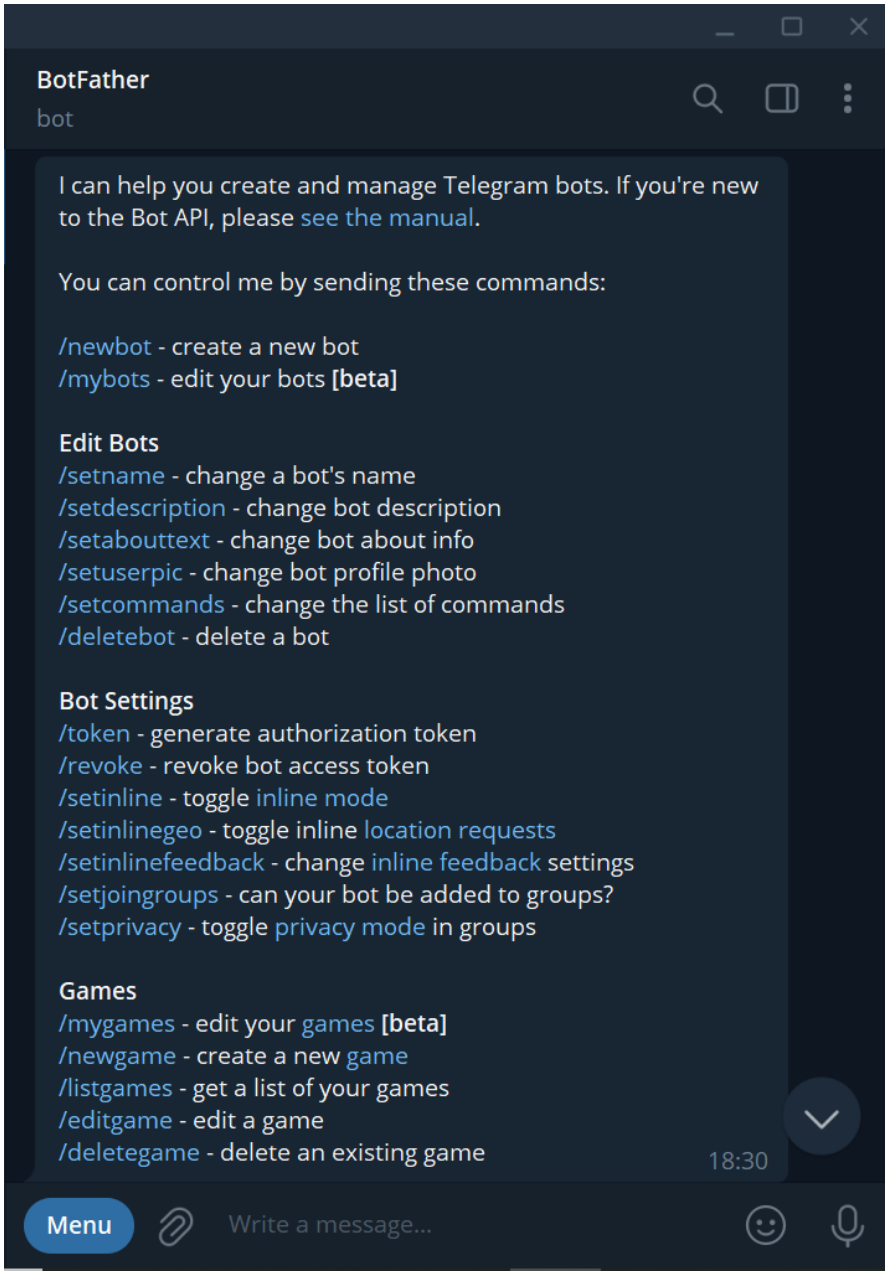


Рис 3.1 – Інтерфейс BotFather

Звернушись до BotFather з командою /newbot, після вибору назви та юзернейму для бота, отримаємо унікальний API-токен, завдяки якому можна буде ідентифікувати нашого бота та звертатися до нього. Приклад токену: *5415464092:AAHEvck9XX48Y7py5g7ODuKnjVmmo-4lTq4*

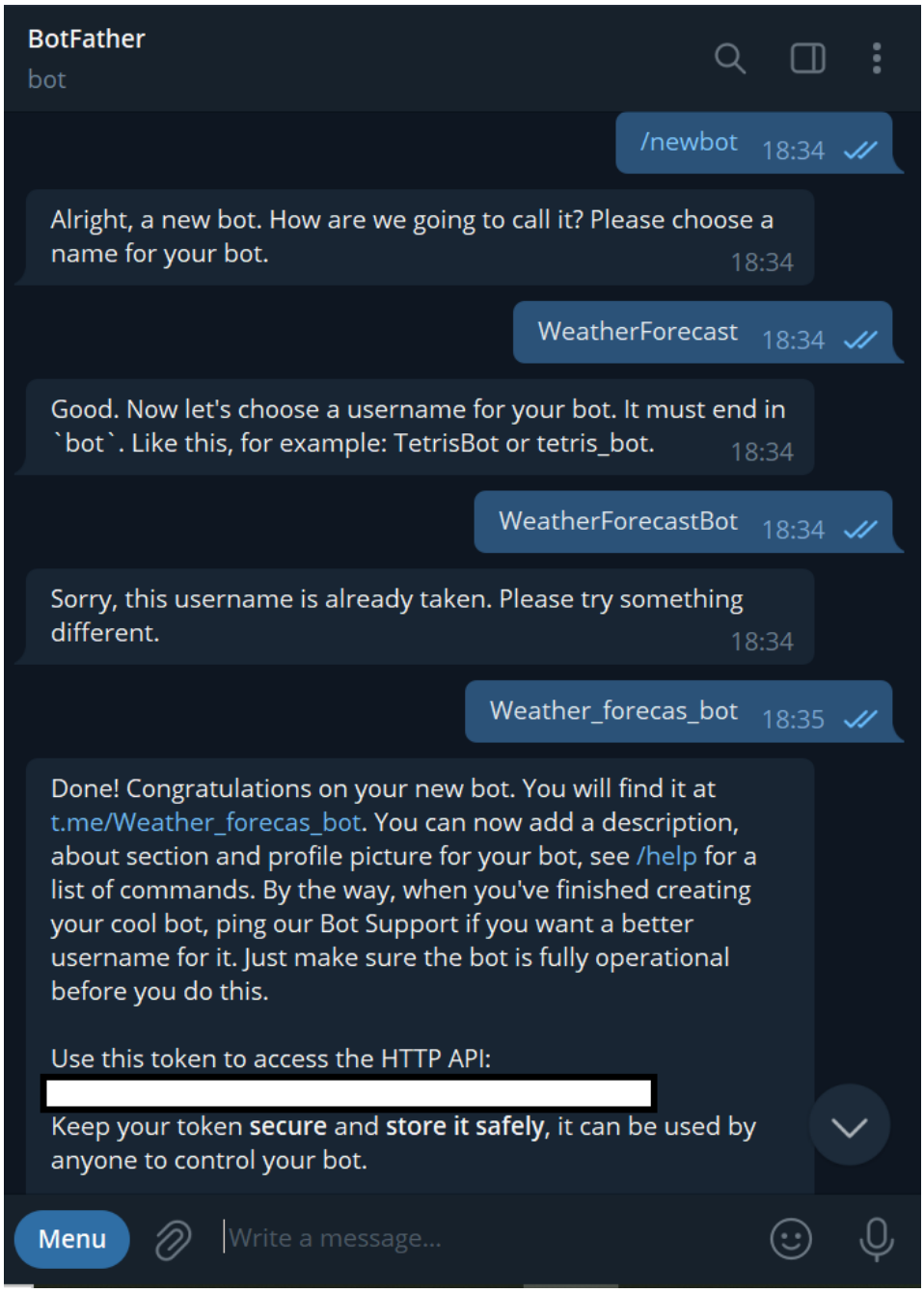


Рис 3.2 – Процес створення та реєстрації боту

BotFather також дає можливість приступити до конфігурації існуючих ботів. Потрібно обрати «аватарку» та короткий опис, який буде зустрічати нових користувачів.

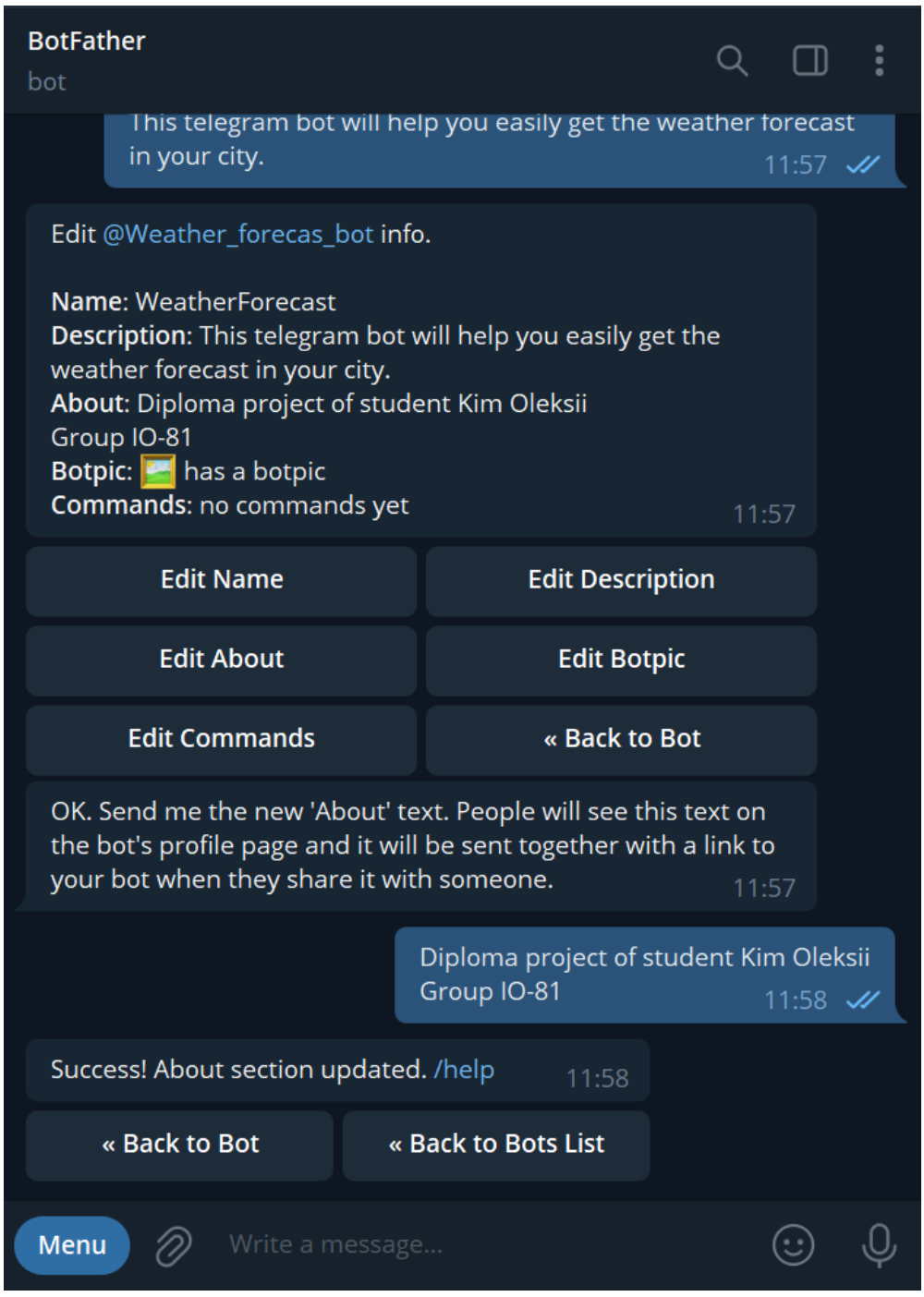


Рис 3.3 – Налаштування нового бота

Після цього можна переходити до наступного етапу розробки бота з прогнозом погоди. Далі треба визначитись звідки саме бот буде брати інформацію про актуальний стан погоди. З цим ми визначились у другому розділі.

Перейдемо на сайт OpenWeatherMap, щоб отримати там API-ключ, за допомогою якого бот буде звертатись до цього сайту і у відповідь буде отримувати прогноз. Для початку роботи з сервісом потрібно авторизуватись та обрати один з тарифних планів, або ж просто користуватись безкоштовно з обмеженим функціоналом. Після цього можна створити API-ключ для доступу до даних про погоду, ключ буде активовано протягом двох годин, а відображатиметься він в особистому кабінеті. Безкоштовний API також має обмеження на запити: максимум 60 запитів на хвилину та 1000000 на місяць.

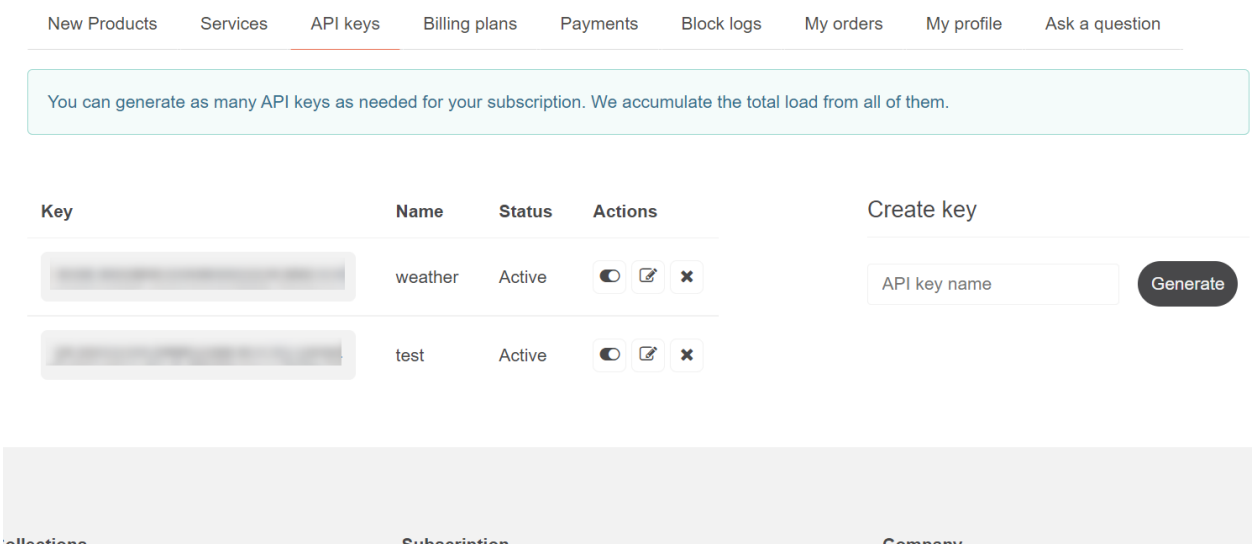


Рис 3.4 – Список особистих API-ключів [17]

Також тут можна вивчити документацію необхідних функцій, ознайомитись з прикладами використання API-запитів, тощо. Сервіс надає усю необхідну інформацію для того щоб швидко та легко почати користуватись ним.

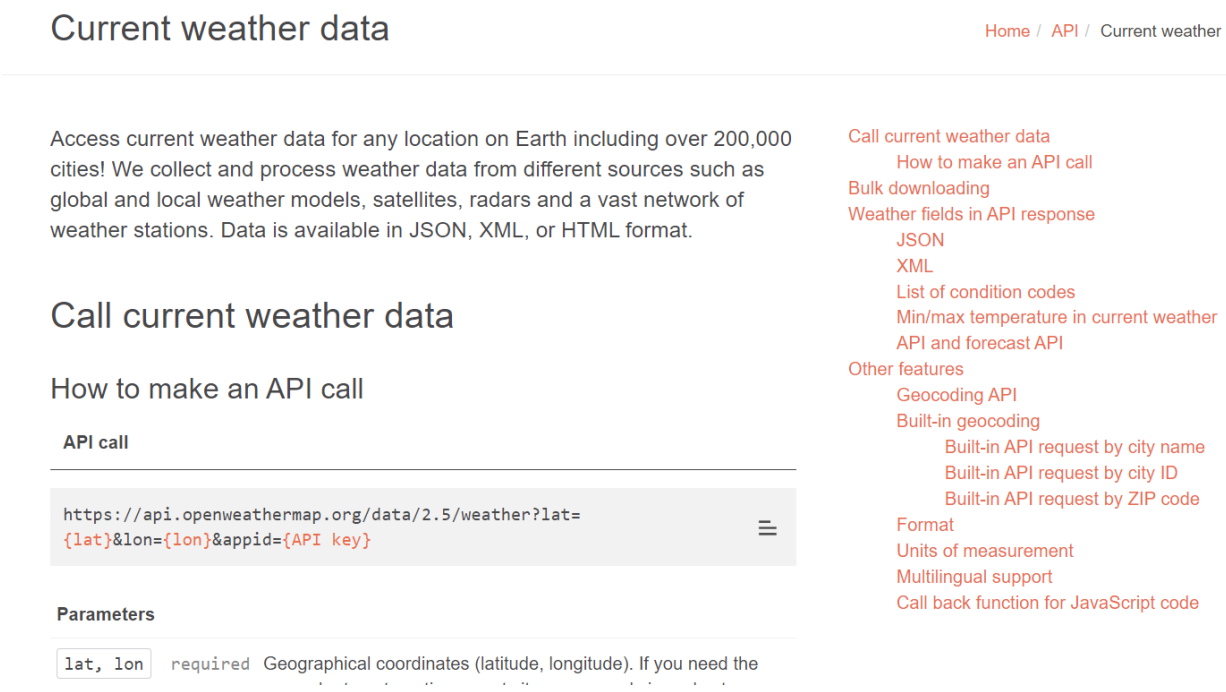


Рис 3.5 – Приклад API-запиту [17]

Тепер, маючи API-ключ для доступу до сервісу OpenWeatherMap та унікальний токен телеграм-бота, можна приступати безпосередньо до розробки.

3.2 Розробка бота

Для початку розробки треба встановити необхідні геми (бібліотеки), а саме rest-client, rubocop та telegram-bot-ruby. Перший гем – це бібліотека для взаємодії з REST-сервисами, саме у цьому проєкті він потрібен для роботи з API, для відправлення запитів та отримання відповіді. Другий гем, RuboCop – це аналізатор коду, який оснований на посібнику зі стилю написання коду Ruby Style Guide. Telegram-bot-ruby – це, як і очевидно із назви, бібліотека для розробки Telegram-ботів на Ruby. Можна встановити кожен з цих гемів в ручну виконавши команду gem install, але набагато зручніше створити файл Gemfile, вписати туди всі геми та джерело, з якого

їх треба скачати, після чого виконати команду bundle install, яка встановить всі прописані у Gemfile геми.

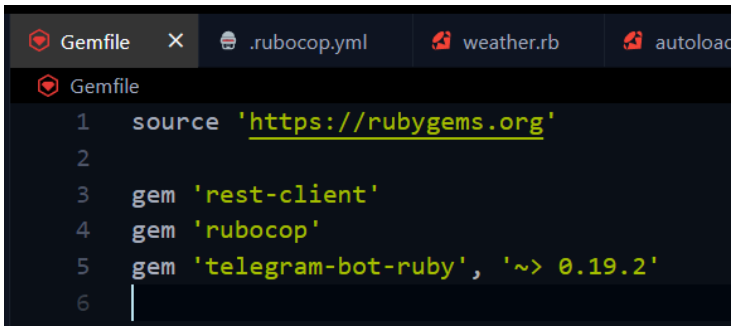


Рис 3.6 – Вміст Gemfile

```
PS C:\Users\alexk\Desktop\WeatherBot> bundle install
Using ast 2.4.2
Using thread_safe 0.3.6
Using descendants_tracker 0.0.4
Using ice_nine 0.11.2
Using axiom-types 0.1.1
Using bundler 2.3.11
Using coercible 1.0.0
Using unf_ext 0.0.8.1 (x64-mingw32)
Using unf 0.1.4
Using domain_name 0.5.20190701
Using dry-inflector 0.2.1
Using faraday-em_http 1.0.0
Using faraday-em_synchrony 1.0.0
Using faraday-excon 1.1.0
Using faraday-httpclient 1.0.1
Using multipart-post 2.1.1
Using faraday-multipart 1.0.3
Using faraday-net_http 1.0.1
Using faraday-net_http_persistent 1.2.0
Using faraday-patron 1.0.0
Using faraday-rack 1.0.0
Using faraday-retry 1.0.3
Using ruby2_keywords 0.0.5
Using faraday 1.10.0
Using ffi 1.15.5 (x64-mingw32)
Using http-accept 1.7.0
Using http-cookie 1.0.4
Using mime-types-data 3.2022.0105
Using mime-types 3.4.1
Using netrc 0.11.0
Using parallel 1.22.1
Using parser 3.1.2.0
Using rainbow 3.1.1
Using regexp_parser 2.3.1
Using rest-client 2.1.0 (x64-mingw32)
Using rexml 3.2.5
Using rubocop-ast 1.17.0
Using ruby-progressbar 1.11.0
Using unicode-display_width 2.1.0
Using rubocop 1.28.2
Using virtus 2.0.0
Using telegram-bot-ruby 0.19.2
Bundle complete! 3 Gemfile dependencies, 42 gems now installed.
Use `bundle info [gemname]` to see where a bundled gem is installed.
```

Рис 3.7 – Встановлення гемів за допомогою bundle install

```
PS C:\Users\alexk\Desktop\WeatherBot> gem install rest-client
Successfully installed rest-client-2.1.0-x64-mingw32
Parsing documentation for rest-client-2.1.0-x64-mingw32
Installing ri documentation for rest-client-2.1.0-x64-mingw32
Done installing documentation for rest-client after 5 seconds
1 gem installed
PS C:\Users\alexk\Desktop\WeatherBot>
```

Рис 3.8 – Встановлення одного гему за допомогою gem install

Створимо новий файл weather.rb, у цьому файлі буде описано клас Weather, який буде працювати з погодою по географічним координатам. Об’єкти цього класу будуть мати 2 характеристики, а саме широту та довготу, щоб реалізувати це необхідно створити метод initialize, це метод конструктор класу, та визначити в ньому ці характеристики. Конструктор класу буде викликано при створенні нового об’єкту цього класу. Також треба додати accessor, щоб можна було зчитувати та записувати широту та довготу.

```ruby
def initialize(lat, lon)
  @lat = lat
  @lon = lon
end

attr_accessor :lat, :lon, :time
```

Рис 3.9 – Конструктор та акцесор класу Weather

Для того щоб отримувати дані з сервісу OpenWeatherMap за координатами, нам знадобляться саме координати, які ми вже визначили, актуальний ча а також наш особистий API-ключ. Нижче оголошено константу з API-ключем, змінну, якій присвоєно метод .new, який створює новий об’єкт класу Time (представляє час і дату в Ruby) з актуальним часом.

API_KEY буде використано у генерації посилання, на яке буде відправлятись API-запит.

```
include OutputHelper

API_KEY = '3b68c45dd                    06b3c9'
time = Time.new
```

Рис 3.10 – фрагмент коду

Також на Рис 3.10 видно підключення модулю OutputHelper. Це модуль, який буде створено пізніше для виводу потрібної інформації про погоду.

```
def weather_url
  "api.openweathermap.org/data/2.5/weather?lat=#{lat}&lon=#{lon}&dt=#{time}&APPID=#{API_KEY}&units=metric"
end

def api_response
  @response_body ||= RestClient.get(weather_url).body
  JSON(@response_body)
end
```

Рис 3.11 – Приватні методи класу Weather

На Рис 3.11 показано два методи, weather_url та api_response. Метод weather_url повертає посилання на яке буде відправлено API-запит, в яке входять координати, які будуть надсилатися користувачем, поточний час та API-ключ від OpenWeatherMap. Другий метод, api_response посилає API-запит на посилання, яке створює метод weather_url, за допомогою гему Rest-client, отримує відповідь та дістає з неї body, після чого конвертує у формат JSON. Також у класі Weather реалізовано генерацію повідомлення з даними про погоду для користувача та методи, які повертають повідомлення в залежності від хмарності та дощових опадів. Методи вибору хмарності та дощовості обирають потрібне значення з хешів, в залежності від отриманих погодних даних. Хеші створені у окремому модулі OutputHelper.

```ruby
def weather_message
  "#{date}
  \nThere is #{select_cloudiness(clouds)} in your location today
  \n🌡Temperature #{temperature}°C, feels like #{feeling_temp}°C
  \n💧Humidity #{humidity}%
  \n🌬Wind speed #{wind} m/s
  \n⏱Atmospheric pressure is #{pressure} hPa
  \n#{select_raininess(rain)}"
end
```

Рис 3.12 – Генерація повідомлення для користувача

```ruby
def select_cloudiness(clouds)
  cloudiness = cloudiness_variations.select { |ico| ico === clouds}.values.first
  cloudiness
end

def select_raininess(rain)
  raininess = raininess_variations.select { |ico| ico === rain}.values.first
  raininess
end
```

Рис 3.13 – Вибір хмарності та дощовості

Перейдемо до файлу output_helper.rb, у якому знаходяться усі допоміжні методи для виводу потрібної користувачеві інформації. Після відправлення API-запиту до сервісу, приходить відповідь у форматі JSON, з прикладом JSON-відповіді можна ознайомитися на сайті OpenWeatherMap.


```json
{
  "coord": {"lon": -122.08, "lat": 37.39},
  "weather": [
    {"id": 800, "main": "Clear", "description": "clear sky", "icon": "01d"}
  ],
  "base": "stations",
  "main": {"temp": 282.55, "feels_like": 281.86, "temp_min": 280.37, "temp_max": 284.26, "pressure": 1023, "humidity": 100},
  "visibility": 10000,
  "wind": {"speed": 1.5, "deg": 350},
  "clouds": {"all": 1},
  "dt": 1560350645,
  "sys": {"type": 1, "id": 5122, "message": 0.0139, "country": "US", "sunrise": 1560343627, "sunset": 1560396563},
  "timezone": -25200,
  "id": 420006353,
  "name": "Mountain View",
  "cod": 200
}
```


Рис 3.14 – Приклад відповіді у форматі JSON [17]

Більшість методів у модулі OutputHelper (Рис 3.15) повертають дані про температуру, вологість, атмосферний тиск, швидкість вітру, хмарність та дощові опади. У цих методах використовується метод dig, який дозволяє працювати з хешами або хеш-подібними даними.

```ruby
def temperature
  api_response.dig('main', 'temp')
end

def humidity
  api_response.dig('main', 'humidity')
end

def clouds
  api_response.dig('clouds', 'all')
end

def wind
  api_response.dig('wind', 'speed')
end

def pressure
  api_response.dig('main', 'pressure')
end

def rain
  api_response.dig('rain', 'rain.1h').to_i
end
```

Рис 3.15 – Фрагмент коду

У цьому ж модулі створено 2 хеша (Рис 3.16). Перший містить у собі повідомлення з іконками про стан дощовості, в залежності від показників, які були отримані від погодного сервісу, а другий – про стан хмарності.

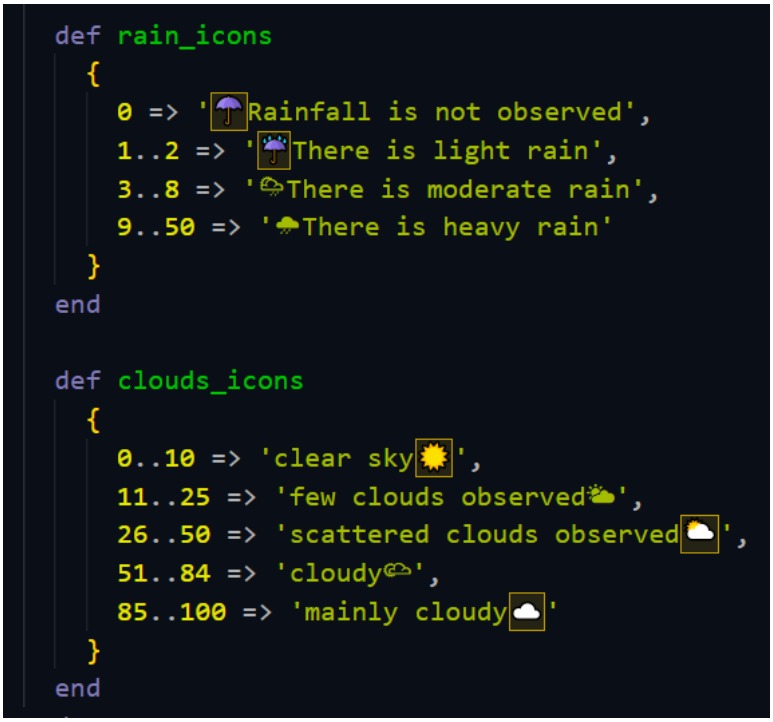


Рис 3.16 – Хеші можливих хмарності та дощовості

Останній метод цього модуля використовує вбудований у Ruby класс Time, для того щоб повертати поточну дату та час згідно з всесвітнім координованим часом (UTC).

```ruby
def date
  date = Time.new.utc
  date
end
```

Рис 3.17 – Метод дати та часу

В окремому файлі weather_city.rb створимо клас WeatherCity, який буде наслідувати від класу Weather. Цей клас, як і його «батько», буде працювати з погодою, тож використання механізму наслідування тут доречне. Відмінностями WeatherCity від Weather є інша генерація

посилання та повідомлення. Метод api_response працює так само як і у класі Weather але вже з іншим посиланням. Нова генерація посилання необхідна для того щоб отримувати інформацію про погоду від сервісу у конкретно вказаному місті, а не за координатами.

```ruby
class WeatherCity < Weather
  include OutputHelper

  def initialize(city)
    @city = city
  end

  attr_accessor :city
```

Рис 3.18 – Конструктор класу WeatherCity

У конструкторі цього класу аргументом буде тільки місто, у якому потрібно дізнатись погоду. На Рис 3.18 також видно підключення модулю OutputHelper.

```ruby
def weather_city_message
  "#{date}
  \nThere is #{select_cloudiness(clouds)} in #{city} today
  \n🌡Temperature #{temperature}°C, feels like #{feeling_temp}°C
  \n💧Humidity #{humidity}%
  \n🌬Wind speed #{wind} m/s
  \n⏱Atmospheric pressure is #{pressure} hPa
  \n#{select_raininess(rain)}"
end
```

Рис 3.19 – Генерація повідомлення у класі WeatherCity

```ruby
def weather_city_url
  "api.openweathermap.org/data/2.5/weather?q=#{city}&APPID=#{API_KEY}&units=metric"
end

def api_response
  @response_body ||= RestClient.get(weather_city_url).body
  JSON(@response_body)
end
```

Рис 3.20 – Генерація посилання та api_response у WeatherCity

В цілому WeatherCity працює так само як і Weather, лише з невеликими відмінностями.

Ознайомившись з документацією гему telegram-bot-ruby, можна приступити до створення класу, у якому буде описано сам бот. Клас Bot буде описано в окремому файлі telegram_bot.rb.

Для початку було створено метод bot, який запускатиме бота з конкретним токеном, аргументом у цей метод передано константу, яка містить сам токен телеграм-бота, який було отримано у 3.1. Константа визначається у класі Bot.

```ruby
def bot
  Telegram::Bot::Client.run(TOKEN) { |bot| return bot }
end
```

Рис 3.21 – Метод запуску боту за токеном

Далі у цьому класі буде описано 2 метода, які будуть використовуватись разом, а саме city_name та city_weather. Кажучи простими словами, метод city_name визначає назву міста, написаного користувачем після команди /weather, а метод city_weather – відправляє у відповідь повідомлення, яке містить інформацію про погоду у цьому місті. Нагадаю що повідомлення генеруються у класі WeatherCity (Рис 3.19) Ознайомитись з кодом цих методів можна на Рис 3.22.

```ruby
def city_name(text)
  text.gsub('/weather', '').strip.tr(' ', '+')
end

def city_weather(message)
  return unless message.text.include? '/weather'

  bot.api.send_message(chat_id: message.chat.id, text: WeatherCity.new(city_name(message.text)).weather_city_message)
end
```

Рис 3.22 – Методи city_name та city_weather

Ретельніше розглянемо ці методи. Метод city_name приймає текст як аргумент, та замінює у рядку «/weather» на пустий рядок, тобто видаляє за допомогою методу gsub. Метод gsub – це метод підстановки, приймає два аргументи. Він знаходить всі екземпляри рядка, що збігаються з першим аргументом цього методу, і замінює їх другим аргументом. Після видалення «/weather» у рядку видаляються всі пробіли за допомогою методу strip та замінюються на плюси за допомогою tr, це потрібно для коректної роботи з API від OpenWeatherMap. Другий метод, тобто city_weather, використовує методи гему telegram-bot-ruby, а саме bot.api.send_message – цьому методу достатньо двох аргументів, хоча їх може бути і більше, за замовченням вони не використовуються. Перший аргумент – це айді чату, у який потрібно прислати повідомлення (не обов'язково чату з користувачем), туди треба передати поточний чат саме з користувачем, щоб бот надіслав відповідь саме тому, хто її запросив. Другий аргумент – це текст повідомлення, конкретно у нашому випадку (Рис 3.22) викликається метод-генератор повідомлення для нового об'єкту класу WeatherCity, аргументом для цього об'єкту виступає результат методу city_weather. Важливо зауважити що метод city_weather спрацьовує лише тоді, коли в тексті повідомлення від користувача міститься команда «/weather», в противному випадку він просто не поверне повідомлення.

Метод для відправки даних про погоду у відповідь на координати – location_weather. Він спрацьовує лише тоді, коли повідомлення від користувача містить дані про його геолокацію. Якщо message.location не дорівнює nil, тобто існує, тоді змінним lat та lon присвоюються значення широти та довготи відповідно. Широту та довготу у собі містить повідомлення з геолокацією, з нього ж ці параметри і беруться. В останньому рядку цього методу викликається вже розглянутий нами метод bot.api.send_message, у перший аргумент якого так само передається айді поточного чату, а у другий метод-генератор повідомлення вже з класу Weather, об'єкту цього класу аргументами вказуються отримані з повідомлення координати, тобто lat та lon (Рис 3.23). Таким чином, поділившись місцем свого розташування з ботом, користувач отримує інформацію про погоду у цьому місці.

```ruby
def location_weather(message)
  if message.location != nil
    lat = message.location.latitude
    lon = message.location.longitude
    bot.api.send_message(chat_id: message.chat.id, text: Weather.new(lat, lon).weather_message)
  end
end
```

Рис 3.23 – Метод location_weather

```ruby
def input_error(message)
  return unless message.text != nil
    unless message.text.include?('/start') || message.text.include?('/location') || message.text.include?('/weather')
      bot.api.send_message(chat_id: message.chat.id, text: "I don't understand.😟\nPlease try the /start to find out how to communicate with me")
    end
end
```

Рис 3.24 – Метод input_error

На Рис 3.24 зображено метод-валідатор input_error. Метою цього метода є повідомлення користувачеві про некоректні запити, та поідомлення про те які запити є коректними. Якщо у повідомленні від користувача є текст (користувач може відправити і геолокацію, тоді у

повідомленні буде тільки локація без тексту), то метод може переходити на наступний етап перевірки. Впевнившись що текст повідомлення існує, метод переходить до перевірки самого тексту: якщо текст містить одну з команд, які відомі боту, то метод повертає повідомлення про некоректність та проханням використати команду для виклику головного меню, щоб отримати там коротку довідку по користуванню ботом.

Щодо реалізації головного меню та кнопок. Вони реалізовані інакше, для цього було використано оператори case та when. За цю реалізацію відповідає метод run. Цей метод буде запускати бота за допомогою методу bot (Рис 3.21) та «прослуховувати» всі повідомлення від користувача за допомогою методу listen, який належить до гему, який ми використовуємо для розробки телеграм-боту. За відповідь користувачеві, в залежності від його повідомлення, відповідає весь інший код у методі run.

```
def run
  bot.listen do |message|
    case message
    when Telegram::Bot::Types::CallbackQuery
      case message.data
      when 'help'
          bot.api.send_message(chat_id: message.from.id, text:
      when 'info'
          bot.api.send_message(chat_id: message.from.id, text:
      when 'commands'
        bot.api.send_message(chat_id: message.from.id, text: "/
      end
    when Telegram::Bot::Types::Message
```

Рис 3.25 – Фрагмент коду методу run

У цьому методі кейс message містить у собі ще 2 кейси, message.data та message.text. Кейс message.text (Рис 3.26), містить у собі інструкції поведінки в залежності від тексту повідомлення користувача. Логіка кейсу працює таким чином: якщо виконується конкретна умова, то виконується визначена для цієї умови дія. Якщо текст повідомлення користувача – це

«/start», то бот відправляє повідомлення з трьома кнопками, тобто викликає меню. Поведінка для кожної кнопки прописана у кейсі message.data (Рис 3.27). Якщо текст повідомлення дорівнює рядку «/location», то бот викличе кнопку для того щоб користувач міг поділитися геолокацією, натиснувши на цю кнопку та підтвердивши свою дію, користувач відправить боту свою геолокацію і спрацює метод location_weather (Рис 3.23). Подальший алгоритм дій відомий – метод дістане з повідомлення координати та поверне повідомлення про стан погоди. Якщо повідомлення буде дорівнювати «/weather», то бот повідомить користувача про те, що він забув ввести назву міста після команди, та попросить спробувати ще раз.

```ruby
when Telegram::Bot::Types::Message
  case message.text
  when '/start'
      kb = [
          Telegram::Bot::Types::InlineKeyboardButton.new(text: 'Help', callback_data: 'help'),
          Telegram::Bot::Types::InlineKeyboardButton.new(text: 'Info', callback_data: 'info'),
          Telegram::Bot::Types::InlineKeyboardButton.new(text: 'Commands List', callback_data: 'commands')
      ]
      markup = Telegram::Bot::Types::InlineKeyboardMarkup.new(inline_keyboard: kb)
      bot.api.send_message(chat_id: message.chat.id, text: "👋Hi #{message.from.first_name}!
        Let's get the weather forecast in your city \nMake a choice", reply_markup: markup)
  when '/location'
      kb = [
          Telegram::Bot::Types::KeyboardButton.new(text: 'Share location', request_location: true)
      ]
      markup = Telegram::Bot::Types::ReplyKeyboardMarkup.new(keyboard: kb)
      bot.api.send_message(chat_id: message.chat.id, text: 'Let me know where are you😉', reply_markup: markup)
  when '/weather'
    bot.api.send_message(chat_id: message.chat.id, text: "You forgot the name of the city.
      \nPlease try again, and don`t forget to include the name of the city after the command!")
  end
end
```

Рис 3.26 – Кейс message.text

Перейдемо до розгляду кнопок головного меню. Як вже було сказано, поведінка меню реалізована завдяки кейсу message.data. Принцип дії той самий, тільки кейс перевіряє стан зворотнього запиту. Якщо користувач нажимає на кнопку «Help», то відправляється запит «help», якому відповідає повідомлення, яке має допомогти користувачу розібратися з керуванням ботом. Якщо натиснуть на кнопку «Info», то у відповідь прийде

коротке повідомлення з інформацією про цей проєкт. Якщо ж користувач обере кнопку «Commands List», то буде відправлено запит «commands», якому відповідає виведення повідомлення зі списком усіх команд, які розуміє бот.

```
case message
when Telegram::Bot::Types::CallbackQuery
  case message.data
  when 'help'
      bot.api.send_message(chat_id: message.from.id,
        text: "You can get information about the weather in any city with /weather <Name of city>\nAlso you can enter
        /location to share your position whith me and get the information about weather")
  when 'info'
      bot.api.send_message(chat_id: message.from.id, text: "This bot was created to help you get weather data in any city or place.
        \nDeveloped by Kim Oleksii, a student of the group IO-81")
  when 'commands'
    bot.api.send_message(chat_id: message.from.id, text: "/weather <name of city> - get the information about the weather in the desired city
      \n/location - call button to share geolocation \n/start - call main menu")
  end
```

Рис 3.27 – Кейс message.data

Після цих кейсів у методі run викликаються методи, які були раніше прописані у класі Bot: location_weather (Рис 3.23), city_weather (Рис 3.22) та метод-валідатор input_error (Рис 3.24). Аргументом у цих трьох методів виступає повідомлення від користувача.

```
location_weather(message)
city_weather(message)
input_error(message)
```

Рис 3.28 – Методи класу Bot

Щоб усі класи та методи працювали коректно, необхідно створити файл, в якому будуть прописані усі залежності. Цим файлом буде файл autoloader.rb.

```ruby
require 'telegram/bot'
require 'rest-client'

require_relative 'bot/telegram_bot'
require_relative 'bot/weather'
require_relative 'bot/weather_city'
require_relative 'bot/helpers/output_helper'
```

Рис 3.29 – Вміст файлу autoloader.rb

Бот вже готовий, залишилось створити для зручності окремий файл main.rb, який буде запускати бота. Важливо не забути під'єднати файл з залежностями, тобто autoloader.rb (Рис 3.29).

```ruby
require_relative 'autoloader'

Bot.new.run
```

Рис 3.30 – Файл main.rb

Тепер запустивши main.rb, буде запускатись весь бот. Проєкт готовий, залишилось розмістити його на Heroku. Перед цим важливо у корні проєкту створити Procfile, який буде вказувати Heroku що саме треба запускати.

Procfile

```
bot: ruby main.rb
```

Рис 3.31 – Вміст Procfile

Простіше за все буде не завантажувати CLI, а створити репозиторій з цим проєктом на GitHub. Адже Heroku підтримує інтеграцію з GitHub і може деплоїти додатки прямо з репозиторія, маючи лише посилання на нього. Тож ініціалізувавши Git репозиторій у папці з проєктом та

виконавши кілька стандартних команд можна легко і швидко впоратись зі створенням репозиторію на GitHub. Після цього можна переходити на сайт Heroku щоб розмістити там свій додаток. Після авторизації треба обрати пункт «New app», обрати для бота унікальну назву та обрати регіон.

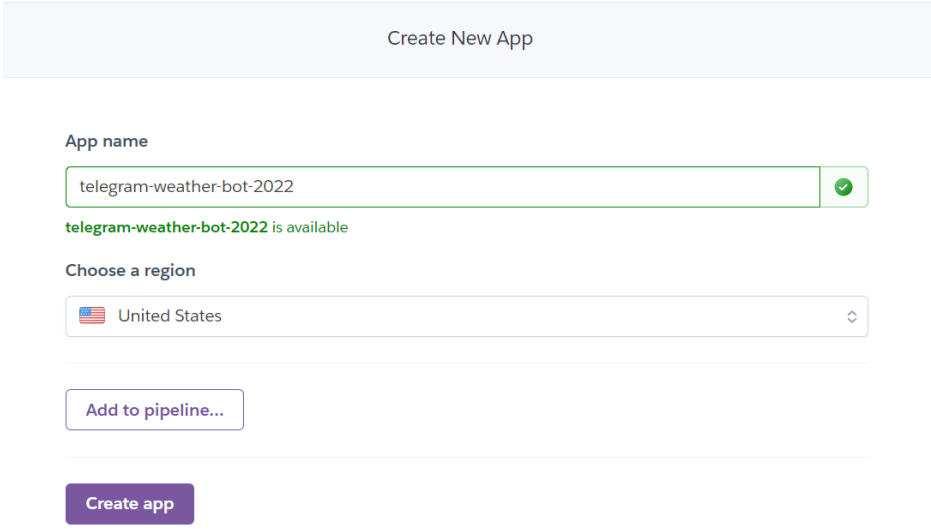


Рис 3.32 – Створення нового додатку на Heroku

Обравши GitHub як метод розгортання, потрібно буде авторизуватися у свій обліковий запис GitHub, та ввести назву репозиторія з проєктом. Heroku видасть список знайдених ваших репозиторіїв, треба обрати потрібний та під'єднатися до нього. Обрати вітку, яку треба розгорнути та натиснути на кнопку «Deploy Branch».

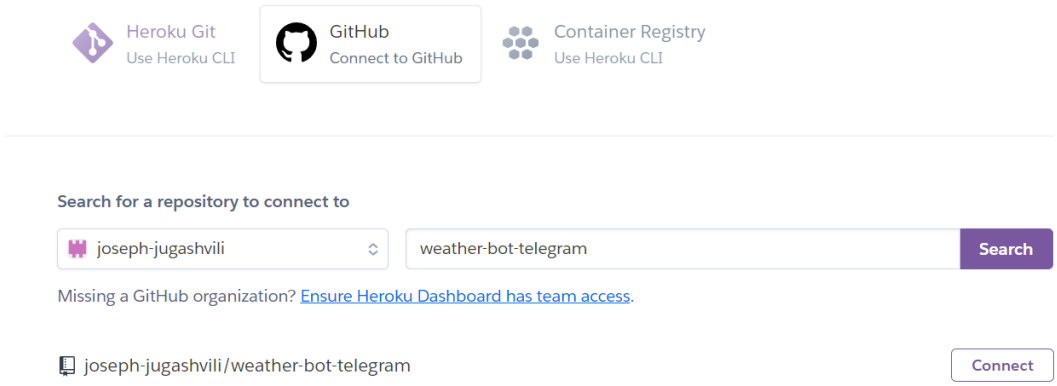


Рис 3.33 – Розгортання проєкту на Heroku

Висновок до розділу 3

У цьому розділі було ретельно описано всі етапи роботи над цим проєктом. В результаті було отримано готовий проєкт на основі месенджеру Telegram та розгорнуто його на платформі Heroku. Під час роботи над ботом було вивчено та освоєно обрані сервіси, а саме Telegram Bot API, OpenWeatherMap та Heroku.

Перевагами готового проєкту є його кросплатформенність та точність отримуваних від бота даних. Бот має зручний користувацький інтерфейс, що дозволяє без проблем користуватись їм незалежно від рівню технічної грамотності.

РОЗДІЛ 4. ОГЛЯД РОЗРОБЛЕНОЇ ПРОГРАМИ

В результаті роботи, описаної у третьому розділі було розроблено повністю функціонуючого чат-бота у месенджері Telegram. У цьому розділі пуде проведено огляд можливостей цього бота.

Щоб почати користуватись ботом потрібно знайти його у месенджері, для цього кожному боту, так само як і користувачу присвоюється унікальний юзернейм. Юзернейм також було обрано у третьому розділі, знайти готового бота можна ввівши в пошук @Weather_forecas_bot.

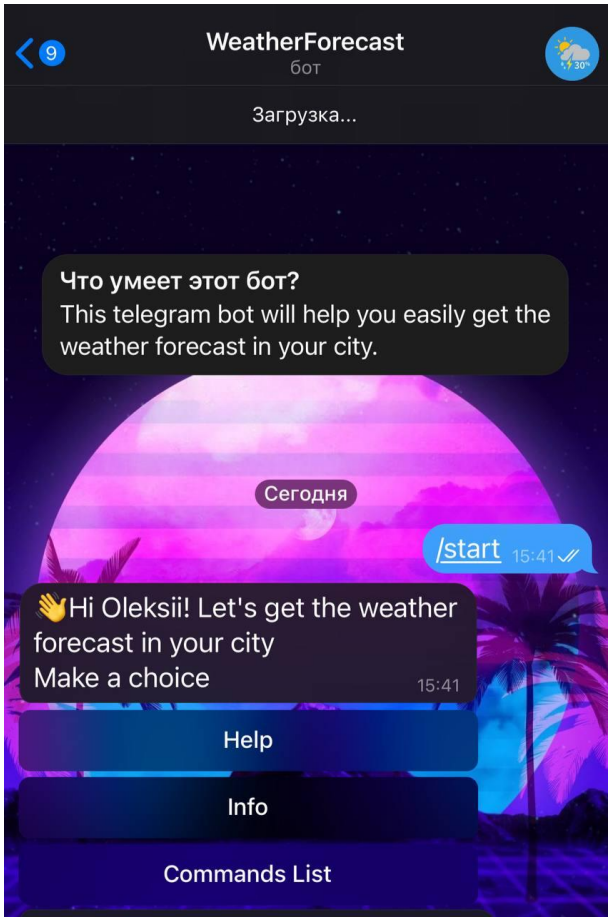


Рис 4.1 – Головне меню бота

Знайшовши та відкривши чат з ботом, телеграм покаже опис бота, який було написано на початку розділу 3, та запропонує натиснути кнопку «Почати», це стандартна процедура для всіх телеграм-ботів. Після

натискання на кнопку, у чат відправляється команда «/start», яка у нашому боті викликає головне меню з вітанням та трьома кнопками.

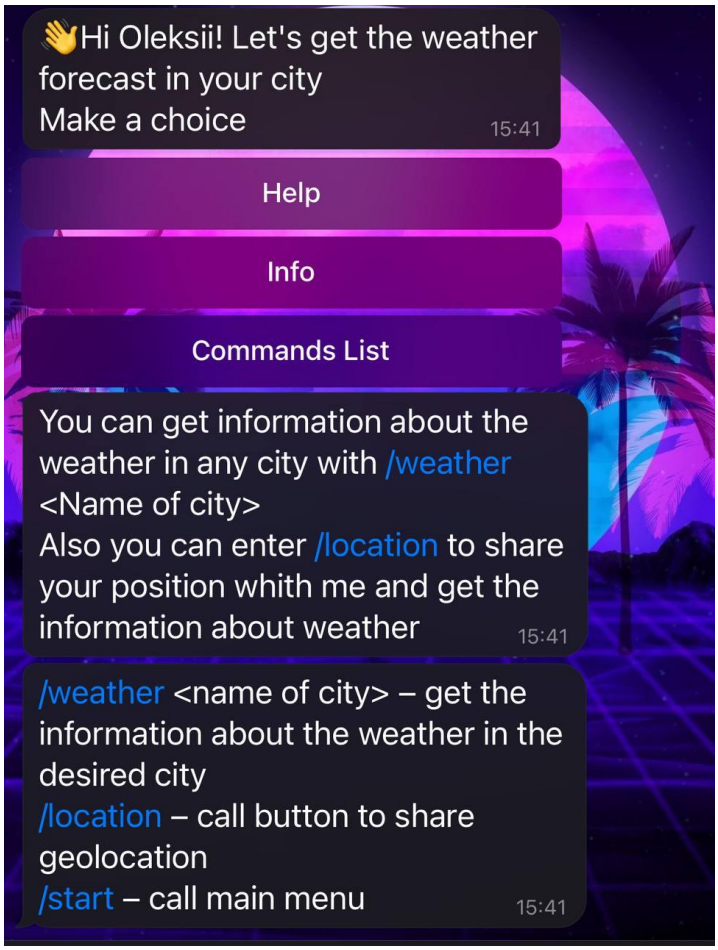


Рис 4.2 – Результат виконання кнопок Help та Commands List

На Рис 4.2 показано два повідомлення від бота. Перше – є результатом вибору користувачем кнопки Help, це повідомлення підказує як і для чого використовувати команди бота. Друге – є результатом вибору кнопки Commands List, воно виводить користувачеві список команд, які розуміє бот. Обравши кнопку Info, користувач отримає повідомлення з інформацією про цей проєкт та про його автора (Рис 4.3).

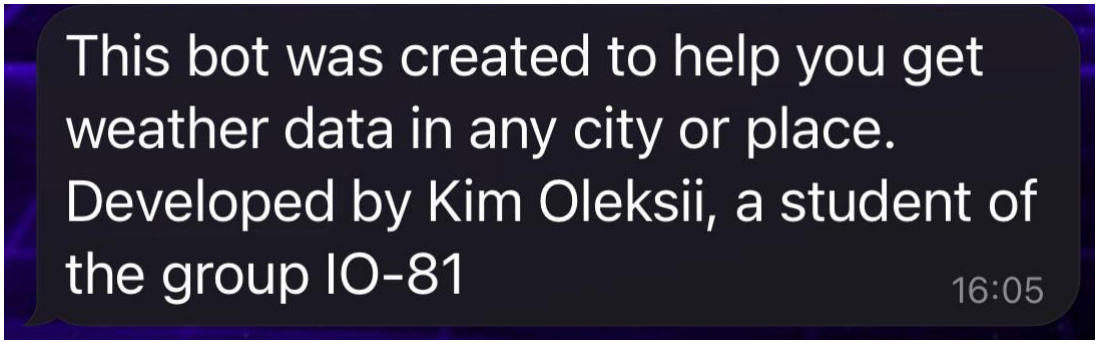


Рис 4.3 – Результат вибору кнопки Info

Отримавши список доступних команд та коротеньку довідку про те як саме ці команди використовувати, користувач може переходити до безпосередньої роботи з ботом.

Після використання команди «/weather» з вказаним містом, наприклад «/weather dnipro», користувач отримує повідомлення з інформацією про погодні умови у місті Дніпро (Рис 4.4). Для того щоб інформація сприймалася легше, бот використовує іконки Телеграму.

Рис 4.4 – Отримання погодних даних у місті

Отримавши назву міста, бот генерує посилання та вставляє туди це місто. Після чого відправляє запит за цим посиланням та отримує відповідь, з якої обирає та виводить потрібні користувачеві дані. Всі методи, які відповідають за цей алгоритм описані у розділі 3.

Також у боті реалізована валідація помилок вводу користувача. Якщо користувач введе команду «/weather» без аргументу, тобто без назви міста, то бот повідомить його про те що потрібно використовувати цю команду обов’язково з назвою міста (Рис 4.5).

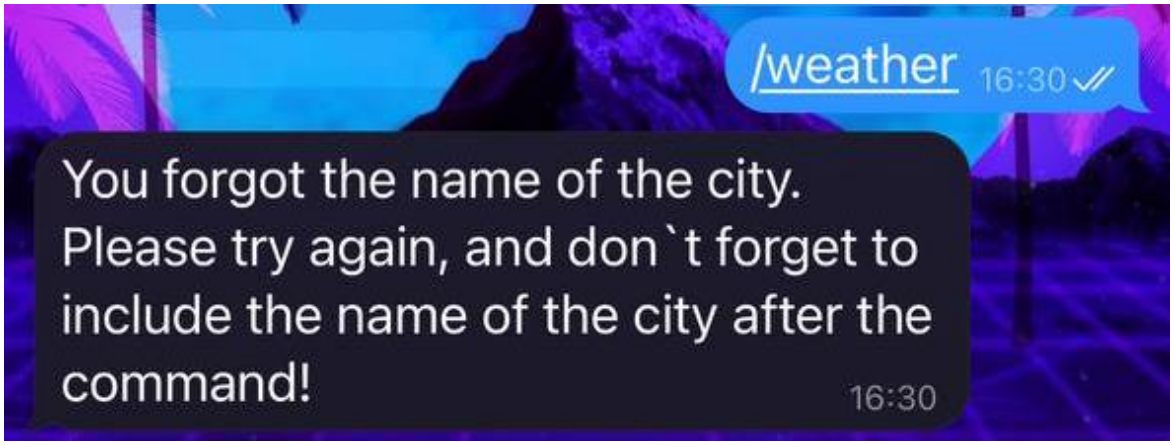


Рис 4.5 – Помилка використання команди «/weather»

Якщо користувач знаходиться десь за межами міста або з якихось причин не знає чи не хоче користуватися командою «/weather», він може скористуватись командою «/location» та поділитись своєю геолокацією з ботом (Рис 4.6). Після вводу команди буде викликано кнопку «Share location», натиснувши на яку та підтвердивши свою дію, користувач відправить свої координати. У відповідь отримає повідомлення з інформацією про погоду у місці свого розташування. Не обов’язково кожен раз вводити «/location», кнопка буде завжди доступна у іншому меню, яке можна відкрити однією кнопкою, яка доступна у правому нижньому кутку інтерфейсу.

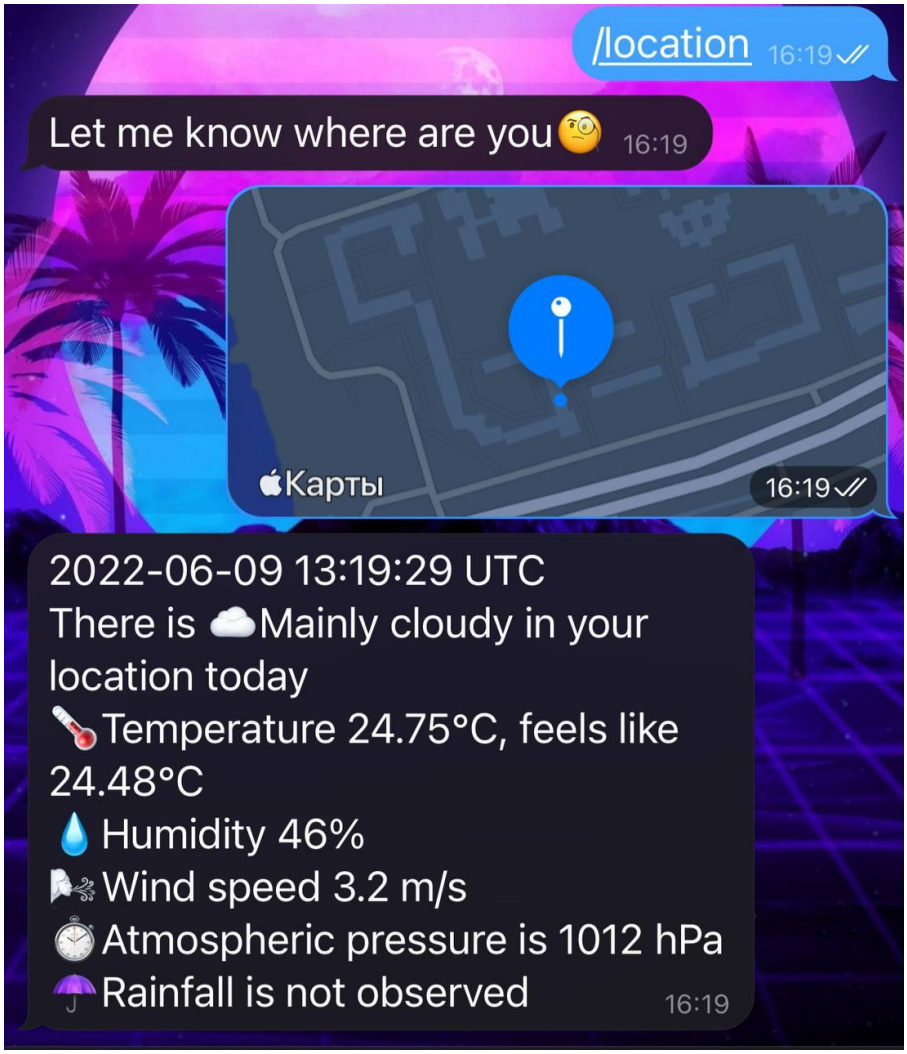


Рис 4.6 – Отримання погодних даних за координатами

У відправленій користувачем геолокації містяться географічні координати широти та довготи. За аналогією з отриманням погодних даних по назві міста, бот бере широту та довготу з цього повідомлення, та вставляє у посилання, за яким відправляє запит та отримує відповідь, далі все так само: з відповіді дістає потрібні параметри та відправляє їх користувачеві. Методи, відповідальні за цей алгоритм також описані у третьому розділі.

Якщо ж користувач введе будь-який інший текст, то бот у відповідь поверне повідомлення, у якому сповістить користувача про те що не

розуміє таких команд і підкаже команду «/start» для виклику меню, з якого можна буде отримати підказку (Рис 4.7).

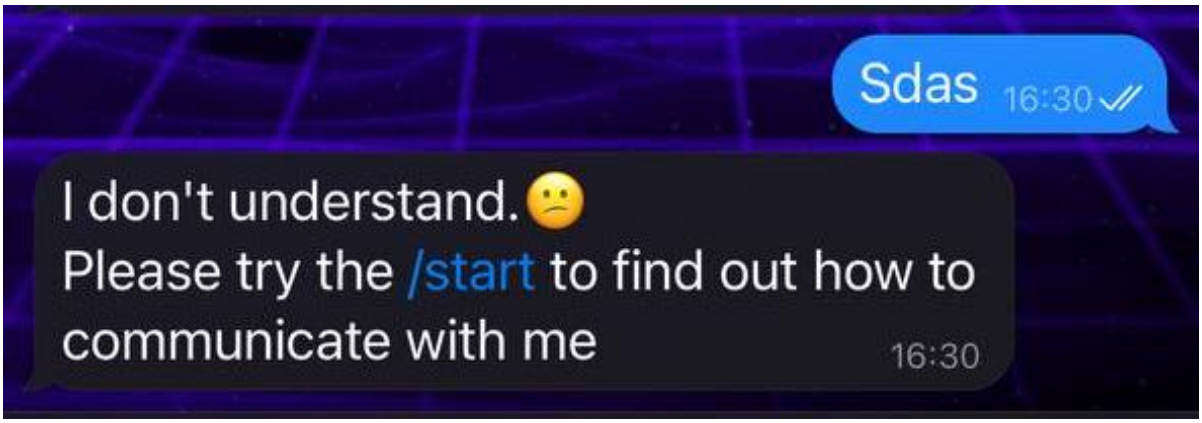


Рис 4.7 – Помилка вводу

Висновок до розділу 4

У ході роботи над четвертим розділом було оглянуто розроблений дипломний проєкт та продемонстровано його коректну роботу. Отриманий у результаті розробки телеграм-бот має досить простий та зручний інтерфейс. Бот повністю справляється з покладеним на нього функціоналом та повністю відповідає вимогам, описаним у першому розділі.

ВИСНОВКИ

Цей дипломний проєкт був націлений на створення чат-боту для інформування про погоду.

У ході роботи над дипломним проєктом було поставлено за мету створення чат-боту на платформі мессенджеру Telegram, який зможе за запитом користувача надавати йому дані про погодні умови у будь-якому місті або у місці розташування самого користувача.

Архітектура програми побудована так, що можна легко додати функцію показу інших видів інформації про погодні умови.

У першому розділі було проаналізовано предметну область. Було розглянуто популярні месенджери, на яких можна було б розмістити чат-бот. Після зробленого на користь Telegram вибору, було розглянуто вже існуючі на цій платформі погодні боти. Також було сформовано вимоги до програми.

У другому розділі було розглянуто та описано технології та інструменти для вирішення поставленого завдання.

Третій розділ повністю присвячено опису усіх етапів розробки та опису роботи чат-боту для месенджеру Telegram.

У четвертому розділі було проведено огляд та тестування розробленого чат-боту.

Результатом розробки став повністю функціонуючий чат-бот, який інформує користувача про стан погоди. Усі поставлені перед розробкою цілі були досягнуті.

ВИКОРИСТАНІ ДЖЕРЕЛА:

[1] Most popular global mobile messenger apps [Електронний ресурс] – Режим доступу до ресурсу: https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/

[2] Які мобільні додатки є найбільш популярними [Електронний ресурс] – Режим доступу до ресурсу: https://www.kiis.com.ua/?lang=ukr&cat=reports&id=1072&page=1

[3] MTProto [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/MTProto

[4] ELIZA [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/ELIZA

[5] Как я сделал телеграм-бота, который сообщает, что надеть по погоде [Електронний ресурс] – Режим доступу до ресурсу: https://vc.ru/life/100009-kak-ya-sdelal-telegram-bota-kotoryy-soobshchaet-chto-nadet-po-pogode

[6] Ruby/Базовые типы данных [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikibooks.org/wiki/Ruby/Базовые_типы_данных

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

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

[9] Go [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Go_(мова_програмування)

[10] PEP 20 – The Zen of Python [Електронний ресурс] – Режим доступу до ресурсу: https://peps.python.org/pep-0020/

[11] Bots: An introduction for developers [Електронний ресурс] – Режим доступу до ресурсу: https://core.telegram.org/bots

[12] Инструкция по работе с BotFather ботом [Електронний ресурс] – Режим доступу до ресурсу: https://botcreators.ru/blog/botfather-instrukciya/

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

[14] RubyMine: среда разработки для Ruby on Rails от JetBrains [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/ru-ru/ruby/

[15] Visual Studio Code [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Visual_Studio_Code

[16] Visual Studio Code - Code Editing. Redefined [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/docs

[17] OpenWeather [Електронний ресурс] – Режим доступу до ресурсу: https://openweathermap.org

[18] OpenWeatherMap [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/OpenWeatherMap

[19] Документація гему telegram-bot-ruby [Електронний ресурс] – Режим доступу до ресурсу: https://www.rubydoc.info/gems/telegram-bot-ruby/0.19.2

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

[21] Документація мови Ruby [Електронний ресурс] – Режим доступу до ресурсу: https://ruby-doc.org

[22] Справочник по Bot API [Електронний ресурс] – Режим доступу доресурсу: https://tlgrm.ru/docs/bots/api

[23] Sandi Metz. Practical object-oriented design. First Edition – 2012

[24] David A. Black, Joseph Leo III. The Well-Grounded Rubyist. Third Edition – 2019.

[25] От ICQ до WhatsApp: эволюция мобильных мессенджеров [Електронний ресурс] – Режим доступу доресурсу: https://uip.me/2015/01/ot-aski-do-whatsapp-evolyuciya-mobilnyx-messendzherov/

[26] Прогноз_погоди [Електронний ресурс] – Режим доступу доресурсу: https://uk.wikipedia.org/wiki/Прогноз_погоди

[27] Миттєві повідомлення [Електронний ресурс] – Режим доступу доресурсу: https://uk.wikipedia.org/wiki/Миттєві_повідомлення

[28] История мессенджеров: первая волна [Електронний ресурс] – Режим доступу доресурсу: https://www.astrosoft.ru/articles/unified-communications/istoriya-messendzherov-pervaya-volna/

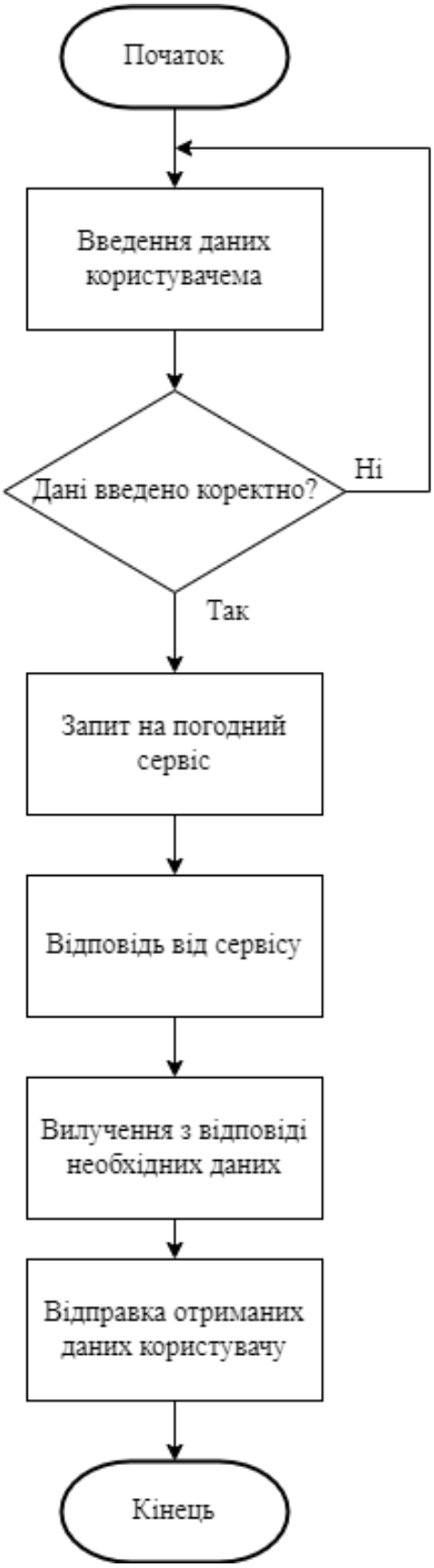
[29] Чат-бот [Електронний ресурс] – Режим доступу доресурсу: https://uk.wikipedia.org/wiki/Чат-бот

[30] METAR [Електронний ресурс] – Режим доступу доресурсу: https://ru.wikipedia.org/wiki/METAR

[31] Heroku - что это за облачная платформа [Електронний ресурс] – Режим доступу доресурсу: https://blog.skillfactory.ru/glossary/heroku/

Додаток 1
До дипломного проєкту
на тему: «Чат-бот для визначення прогнозу погоди»

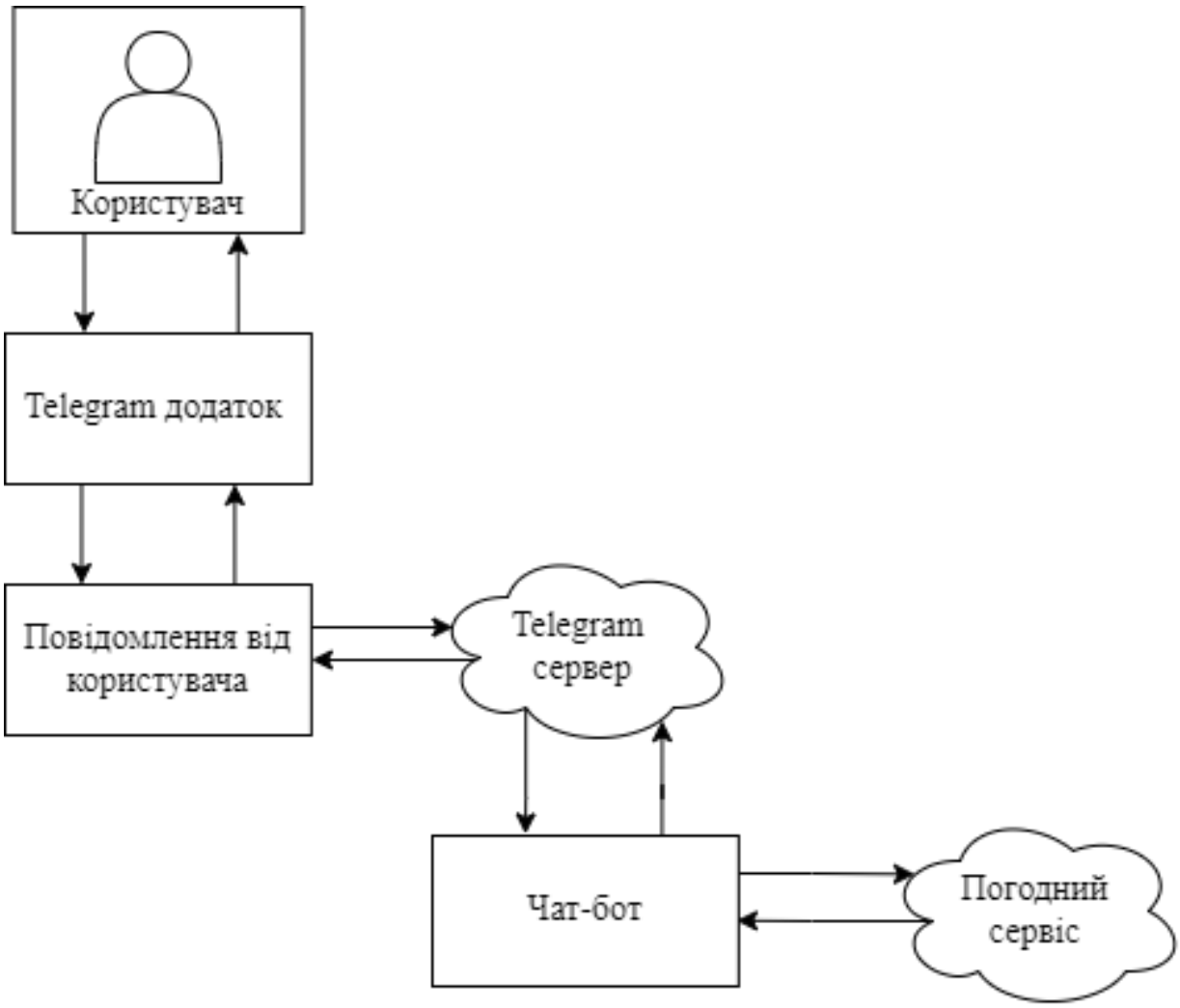
Київ – 2022



					ІАЛЦ 467200.004 Д1		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Кім О. В.			Чат-бот для визначення прогнозу погоди Принципова схема	Лит.	Аркуш 1 / Аркушів 1
Перевірив		Роковий О. П.					
Т. Контр.						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-81	
Н. Контр.		Сімоненко В. П					
Затвердив							

Додаток 2
До дипломного проєкту
на тему: «Чат-бот для визначення прогнозу погоди»

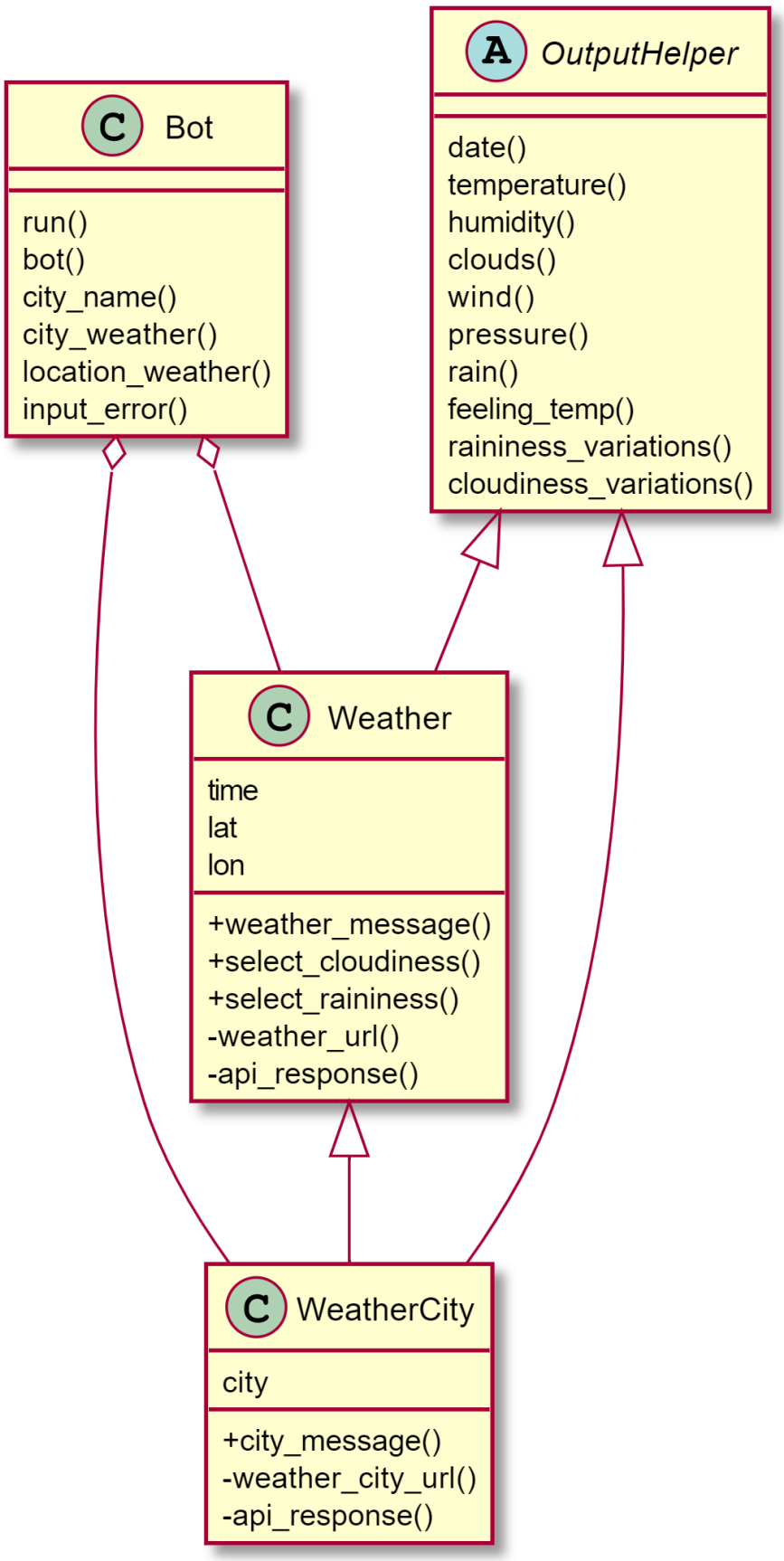
Київ – 2022



					ІАЛЦ 467200.005 Д2		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробив		Кім О. В.			Чат-бот для визначення прогнозу погоди Структурна схема	Лит.	Аркуш 1 / Аркушів 1
Перевірив		Роковий О. П.					
Т. Контр.						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-81	
Н. Контр.		Сімоненко В. П					
Затвердив							

Додаток З
До дипломного проєкту
на тему: «Чат-бот для визначення прогнозу погоди»

Київ – 2022



					ІАЛЦ 467200.006 Д3	
Зм.	Арк.	№ докум.	Підпис	Дата		
Розробив		Кім О. В.			Чат-бот для визначення прогнозу погоди Діаграма класів	Лит. / Аркуш 1 / Аркушів 1
Перевірив		Роковий О. П.				
Т. Контр.						КПІ ім. Ігоря Сікорського, ФІОТ, ІО-81
Н. Контр.		Сімоненко В. П				
Затвердив						

Додаток 4
До дипломного проєкту
на тему: «Чат-бот для визначення прогнозу погоди»

Київ – 2022

telegram_bot.rb

```
class Bot
  TOKEN = '5138871774:AAFgTD8-oq4xVHaEWuMEv_SI4HzqRN10r84'

  def run
    bot.listen do |message|
      case message
      when Telegram::Bot::Types::CallbackQuery
        case message.data
        when 'help'
            bot.api.send_message(chat_id: message.from.id, text: "You can get
information about the weather in any city with /weather <Name of city>\nAlso
you can enter /location to share your position whith me and get the
information about weather")
        when 'info'
            bot.api.send_message(chat_id: message.from.id, text: "This bot
was created to help you get weather data in any city or place.\nDeveloped by
Kim Oleksii, a student of the group IO-81")
        when 'commands'
          bot.api.send_message(chat_id: message.from.id, text: "/weather
<name of city> - get the information about the weather in the desired city
\n/location - call button to share geolocation \n/start - call main menu")
        end
      when Telegram::Bot::Types::Message
        case message.text
        when '/start'
            kb = [
                Telegram::Bot::Types::InlineKeyboardButton.new(text: 'Help',
callback_data: 'help'),
                Telegram::Bot::Types::InlineKeyboardButton.new(text: 'Info',
callback_data: 'info'),
                Telegram::Bot::Types::InlineKeyboardButton.new(text:
'Commands List', callback_data: 'commands')
            ]
            markup =
Telegram::Bot::Types::InlineKeyboardMarkup.new(inline_keyboard: kb)
            bot.api.send_message(chat_id: message.chat.id, text: "👋Hi
#{message.from.first_name}! Let's get the weather forecast in your city
\nMake a choice", reply_markup: markup)
        when '/location'
            kb = [
                Telegram::Bot::Types::KeyboardButton.new(text: 'Share
location', request_location: true)
            ]
            markup = Telegram::Bot::Types::ReplyKeyboardMarkup.new(keyboard:
kb)
            bot.api.send_message(chat_id: message.chat.id, text: 'Let me know
where are you', reply_markup: markup)
        when '/weather'
          bot.api.send_message(chat_id: message.chat.id, text: "You forgot
the name of the city.\nPlease try again, and don`t forget to include the name
of the city after the command!")
```

```
        end
      end
    location_weather(message)
    city_weather(message)
    input_error(message)
    rescue StandardError => e
      puts e.message
    end
  end

  private

  def bot
    Telegram::Bot::Client.run(TOKEN) { |bot| return bot }
  end

  def city_name(text)
    text.gsub('/weather', '').strip.tr(' ', '+')
  end

  def city_weather(message)
    return unless message.text.include? '/weather'

    bot.api.send_message(chat_id: message.chat.id, text:
WeatherCity.new(city_name(message.text)).weather_city_message)
  end

  def location_weather(message)
    if message.location != nil
      lat = message.location.latitude
      lon = message.location.longitude
      bot.api.send_message(chat_id: message.chat.id, text: Weather.new(lat,
lon).weather_message)
    end
  end

  def input_error(message)
    return unless message.text != nil
      unless message.text.include?('/start') ||
message.text.include?('/location') || message.text.include?('/weather')
        bot.api.send_message(chat_id: message.chat.id, text: "I don't
understand.☺\nPlease try the /start to find out how to communicate with me")
      end
  end
end
```

weather.rb

```
require_relative 'helpers/output_helper'

class Weather
  include OutputHelper

  API_KEY = '3b68c45dd8fbdd360565e2e2f206b3c9'
  time = Time.new

  def initialize(lat, lon)
    @lat = lat
    @lon = lon
  end

  attr_accessor :lat, :lon, :time

  def weather_message
    "#{date} \nThere is #{select_cloudiness(clouds)} in your location today
\n🌡Temperature #{temperature}°C, feels like #{feeling_temp}°C \n💧Humidity
#{humidity}% \n🌬Wind speed #{wind} m/s \n□Atmospheric pressure is #{pressure}
hPa \n#{select_raininess(rain)}"
  end

  def select_cloudiness(clouds)
    cloudiness = cloudiness_variations.select { |ico| ico ===
clouds}.values.first
    cloudiness
  end

  def select_raininess(rain)
    raininess = raininess_variations.select { |ico| ico ===
rain}.values.first
    raininess
  end

  private

  def weather_url

"api.openweathermap.org/data/2.5/weather?lat=#{lat}&lon=#{lon}&dt=#{time}&APP
ID=#{API_KEY}&units=metric"
  end

  def api_response
    @response_body ||= RestClient.get(weather_url).body
    JSON(@response_body)
  end
end
```

weather_city.rb

```ruby
require_relative 'helpers/output_helper'

class WeatherCity < Weather
  include OutputHelper

  def initialize(city)
    @city = city
  end

  attr_accessor :city

  def weather_city_message
    "#{date} \nThere is #{select_cloudiness(clouds)} in #{city} today
\n🌡Temperature #{temperature}°C, feels like #{feeling_temp}°C \n💧Humidity
#{humidity}% \n🌬Wind speed #{wind} m/s \n🌐Atmospheric pressure is #{pressure}
hPa \n#{select_raininess(rain)}"
  end

  private

  def weather_city_url

"api.openweathermap.org/data/2.5/weather?q=#{city}&APPID=#{API_KEY}&units=met
ric"
  end

  def api_response
    @response_body ||= RestClient.get(weather_city_url).body
    JSON(@response_body)
  end
end
```

output_helper.rb

```ruby
module OutputHelper
  def date
    date = Time.new.utc
    date
  end

  def temperature
    api_response.dig('main', 'temp')
  end

  def humidity
    api_response.dig('main', 'humidity')
  end

  def clouds
    api_response.dig('clouds', 'all')
  end
```

```ruby
    def wind
      api_response.dig('wind', 'speed')
    end

    def pressure
      api_response.dig('main', 'pressure')
    end

    def rain
      api_response.dig('rain', 'rain.1h').to_i
    end

    def feeling_temp
      api_response.dig('main', 'feels_like')
    end

    def raininess_variations
      {
        0 => '☂Rainfall is not observed',
        1..2 => '🌦There is light rain',
        3..8 => '🌧There is moderate rain',
        9..50 => '⛈There is heavy rain'
      }
    end

    def cloudiness_variations
      {
        0..20 => '☀Clear sky',
        21..40 => '🌤Few clouds observed',
        41..79 => '⛅Scattered clouds observed',
        80..90 => '🌥Cloudy',
        91..100 => '☁Mainly cloudy'
      }
    end
  end
```