МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ВИПУСКНА РОБОТА

на тему:

«Телеграм бот для аналізу прогнозу погодних умов»

Завідувач випускаючої кафедри — **Довбиш А.С.**

Керівник роботи — **Кузіков Б.О**

Студента групи ІН – 61 — **Щербака М.Ю.**

СУМИ 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Кафедра комп'ютерних наук

Затверджую_____

Зав. кафедрою Довбиш А.С.

"_____" _____2020 р.

ЗАВДАННЯ
до випускної роботи

Студента четвертого курсу, групи ІН-61 спеціальності "Інформатика" денної форми навчання Щербака Михайла Юрійовича.

Тема: " Телеграм бот для аналізу прогнозу погодних умов "

Затверджена наказом по СумДУ

№_____ від _____2020 р.

Зміст пояснювальної записки: 1) інформаційний огляд; 2) постановка завдання; 3) визначення методів рішення завдання; 4) проектування роботи; 4) реалізація поставлених задач; 5) аналізи результатів роботи.

Дата видачі завдання "_____" _____2020 р.

Керівник випускної роботи _____ Кузіков Б.О.

Завдання приняв до виконання _____ Щербак М.Ю.

РЕФЕРАТ

Записка: 37 стор., 25 рис., 2 табл., 2 додатки, 19 джерел.

Об’єкт дослідження — телеграм бот.

Мета роботи — телеграм бот, що збирає дані про погоду з сервісу Sinoptik.ua.

Методи дослідження — метод розробки телеграм боту з використанням Telegram Bot API.

Результати — реалізований телеграм бот “What to Wear?” мовою Python з використанням СУБД MySQL для перегляду прогнозу погоди.

ТЕЛЕГРАМ БОТ, ПРОГНОЗ ПОГОДИ, TELEGRAM BOT API, TELEBOT, MYSQL, PYTHON

ЗМІСТ

Вступ

Погода завжди була впливовим аспектом для суспільства. Клімат, погода та природні явища можуть мати різну ступінь впливу. Починаючи з вибору одягу на завтрашній день або планування відпустки в найближчому майбутньому, закінчуючи кількістю жертв, викликаних стихійним лихом. Від погодних умов залежить самопочуття кожної окремої людини, адже їх зміни можуть впливати на підвищення чи зниження тиску, викликати алергію або впливати на психічний стан. Крім того прогноз погоди має прямий вплив на економічну складову. Якщо не застосовувати метеорологічну інформацію в енергетиці, будівництві, авіації, судноплавстві, промисловості і сільському господарстві, то можна зазнати значних втрат. Гідрометеорологічні лиха можуть завдати шкоди економіці. Якщо використовувати прогнози ефективно, то можна максимізувати вигоду і мінімізувати втрати. Різні фактори впливу прогнозу погоди на суб'єктами, які належать до тієї чи іншої галузі може кардинально відрізнятися.

Наприклад, авіація – безперечно залежить від фактичної погоди. Для авіації дуже важливий довгостроковий прогноз погоди на 5 днів, 10 днів, місяць. Жоден літак не злітає і не сідає без інформації про фактичну погоду і прогнозів погоди за маршрутами, пунктів посадки та запасних аеродромів.

Морський і річковий флот залежать від погоди не менше авіації. Ця залежність флоту від погоди проявляється завжди: коли судна знаходяться на маршрутах і в районах лову і коли відстоюються в порту. Штормовий вітер, обмерзання, туман і інші явища здатні привести судно до загибелі.

Таким чином надання прогнозу погоди досить масштабна і актуальна проблема, вирішення якої необхідно для суспільства.

Оскільки рішень знаходження прогнозу погоди багато, і вони задовольняють потреби більшості споживачів, зручність сервісів набуває більшого значення. Останнє десятиріччя популярність соціальних мереж не припиняла зростати, а рекордним є приріст популярності месенджеру – Telegram, який за останні 3 роки потрапив у топ 10 найпопулярніших

соціальних мереж України за даними опитування компанії Research & Branding Group [8]. На початку 2020 року найпопулярнішими соціальними мережами в Україні є Facebook (58% від усіх респондентів), YouTube (41%), Instagram (28%) і telegram (14%). Telegram отримав свою популярність не тільки за захищеність даних користувачів, а й за різноманітність функціоналу. Окрім спілкування у чатах з одним співрозмовником або конференціях до 100 000 осіб, Telegram надає можливість вести та переглядати канали, здійснювати дзвінки. Для користувачів месенджеру, які тісно пов'язані з IT галуззю Telegram надає безкоштовне API для роботи з ботами. Боти - це сторонні додатки, які працюють у Telegram. Вони виглядають як акаунти, якими замість людей керують програми. Основне завдання бота - автоматично відповідати після команди, введеної користувачем. Таким чином, працюючи безпосередньо через інтерфейс Telegram, програма імітує дії живого користувача, завдяки чому використання такого бота є зручним і зрозумілим.

Виходячи з актуальності проблеми мета роботи - створити телеграм бота для аналізу прогнозу погоди на основі даних з метеорологічного сервісу. Для досягнення мети дослідження сформульовані наступні задачі роботи:

- Проаналізувати відомі рішення за темою «Сервіс прогнозу погоди» та «Телеграм бот для аналізу прогнозу погоди»
- Спроектувати модель для телеграм бота
- Обрати оптимальний сервіс прогнозу погоди для використання його даних
- Обрати засоби та середовище розробки для реалізації бота
- Реалізувати телеграм бот
- Виконати модульне тестування додатку та тестування варіантів використання
- Зробити висновки щодо створеного програмного продукту

1. Інформаційний огляд

1.1. Огляд останніх рішень

Наразі існує велика кількість сервісів, які дозволяють дізнатися прогноз погоди. Сервіси відрізняються способом реалізації та інформацією, яка надається користувачам. Наприклад, до іноземних веб-ресурсів належать Weather.com - англомовний ресурс від оператора кабельного і супутникового телебачення з США, Accuweather.com (Рисунок 1.1), Darksky.net. Серед українських веб-сайтів можна зазначити Sinoptik.ua (Рисунок 1.2), Gismeteo.ua (Рисунок 1.3), Meteoprog.ua (Рисунок 1.4), meteo.ua та інші. Популярні додатки для Android та iOS: 1Weather, AccuWeather, Yahoo.Weather.

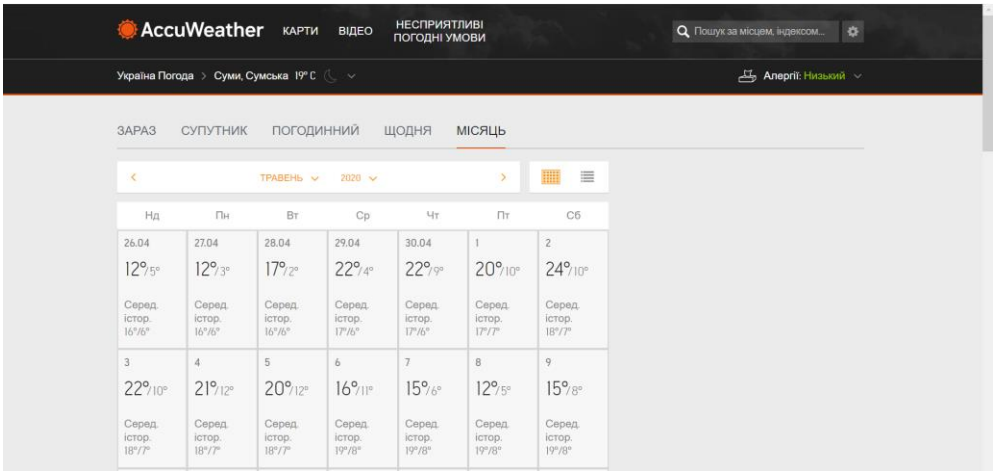

Рисунок 1.1 – Сайт Accuweather.com

Рисунок 1.2 – Сайт Sinoptik.ua

Рисунок 1.3 – Сайт Gismeteo.ua

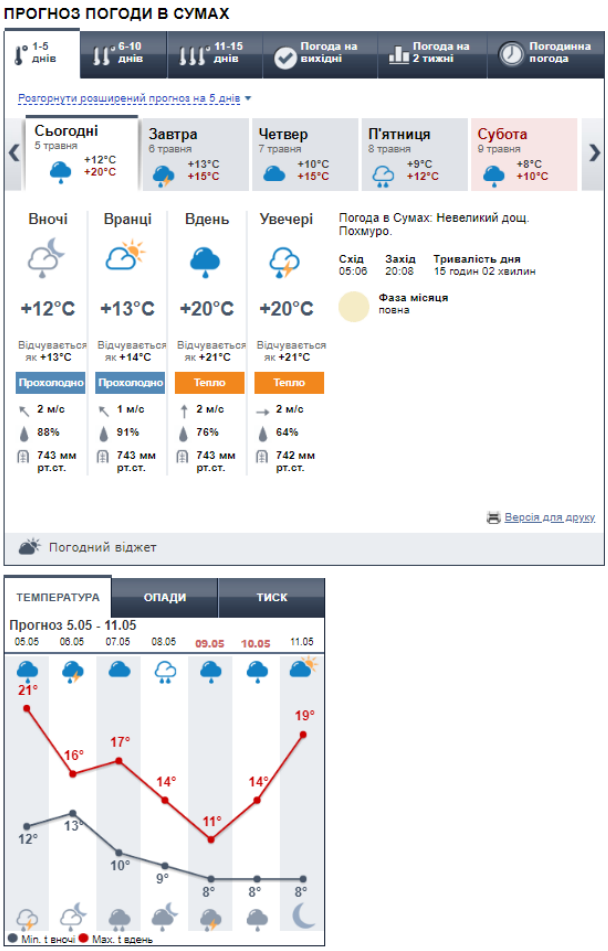

Рисунок 1.4 – Сайт Meteoprog.ua

Жоден з вище наведених сервісів не надає достовірну інформацію щодо прогнозу погоди. Похибки під час аналізу прогнозу – неминуче явище. Для того, щоб порівняти точність прогнозу різних сервісів використовувався ресурс rp5.ua [10], що представляє собою щоденник погоди (Рисунок 1.5). Під час дослідження дані про прогноз погоди збиралися з наступних джерел:

– Sinoptik.ua

– Gismeteo.ua

– Meteoprog.ua

– Accuweather.com

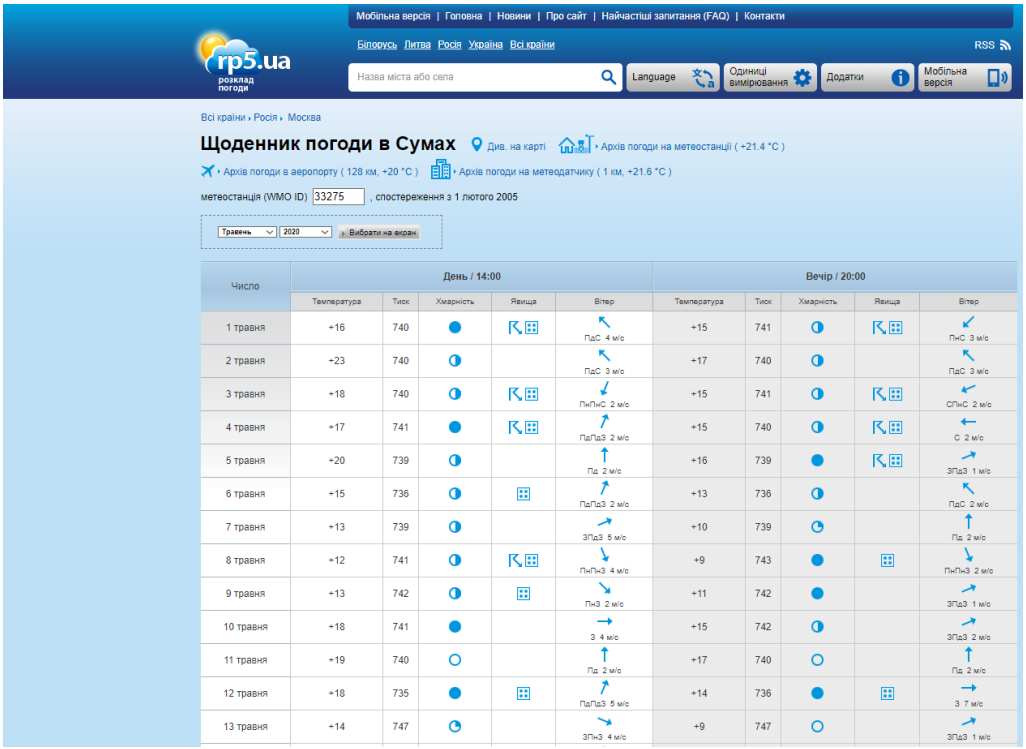

Рисунок 1.5 – Сайт rp5.ua

З кожного джерела отримувався прогноз погоди для міста Суми. Порівняння проводилося за наступними критеріями: виправданість прогнозу температури та прогнозу імовірності опадів. Спостереження проводилися у період з 1 по 15 травня. Імовірності виправданості розраховувались як залишок від відсоткової різниці між прогнозами метеорологічних сервісів та реальними показниками, збереженими вкінці спостережень. Результати випробувань наведені у таблиці 1.1 та таблиці 1.2.

Таблиця 1.1 – Виправданість прогнозу температури

Джерело прогнозу	Імовірність виправданості прогнозу у Сумах, %
Sinoptik	94.5
Gismeteo	89.36
Meteoprog	92.3
Accuweather	93.33

Таблиця 1.2 – Середнє значення прогнозу імовірності опадів у дні коли опади спостерігались та імовірність вдалого прогнозу опадів

Джерело прогнозу	Середнє значення прогнозу імовірності опадів, %	Імовірність вдалого прогнозу опадів, %
Sinoptik	-	95
Gismeteo	80	60
Meteoprog	79.6	59.7
Accuweather	80	60

За результатами випробувань можна зробити висновок, що серед досліджених сервісів прогнозу погоди найбільш точним виявився сайт Sinoptik.ua.

1.2. Огляд реалізацій телеграм ботів

Серед існуючих реалізацій можна виділити наступні додатки: «TheLair Wheather» (Рисунок 1.6) та Телеграмм бот «Погода» (Рисунок 1.7).

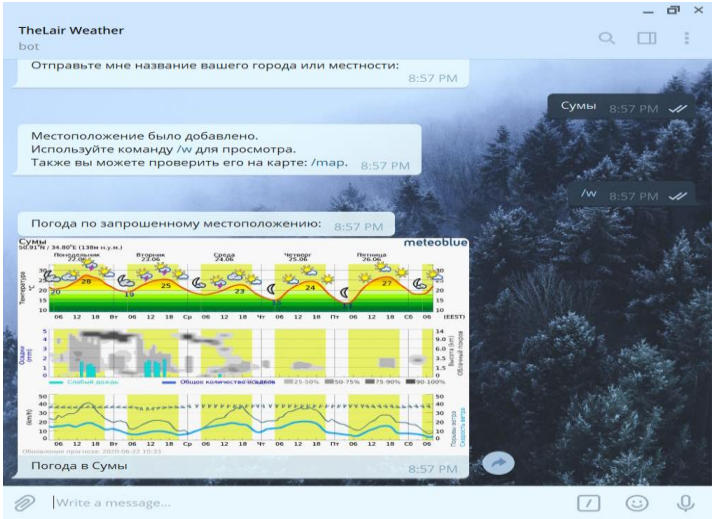

Рисунок 1.6 – Телеграм бот «TheLair Wheather»

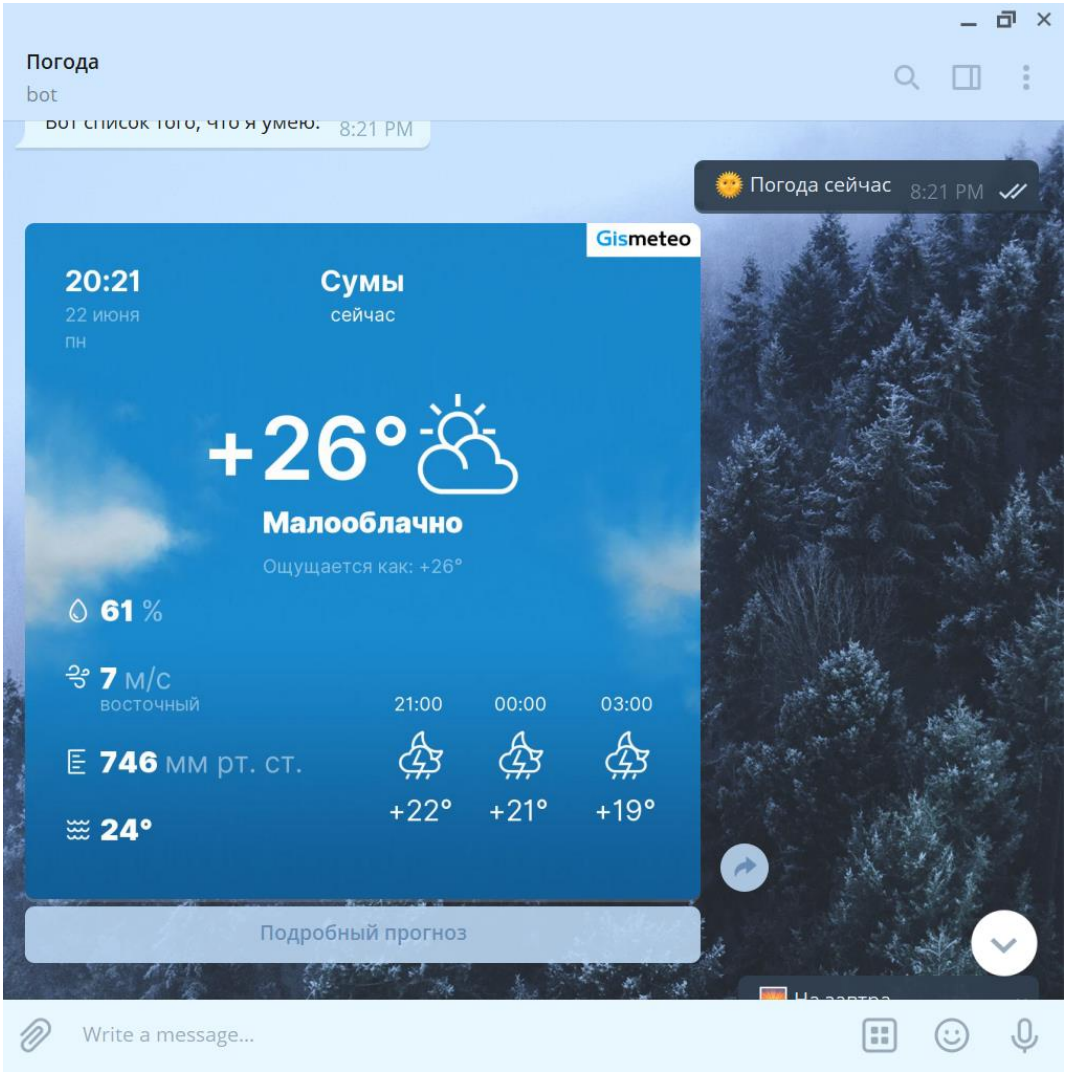

Рисунок 1.7 – Телеграмм бот «Погода»

Перечислені додатки схожі за реалізацією, мають інтерфейси керування у вигляді меню російською мовою. Додаток «TheLair Wheather» також підтримує зміну мови інтерфейсу на англійську. Серед переваг даних додатків можна підкреслити автоматичні сповіщення, які підтримує Телеграмм бот «Погода» (Рисунок 1.8), що є досить зручною функцією для користувача, та гнучкі налаштування додатку «TheLair Wheather», які дозволяють змінювати, окрім мови інтерфейсу, одиниці виміру швидкості вітру та температури (Рисунок 1.9).

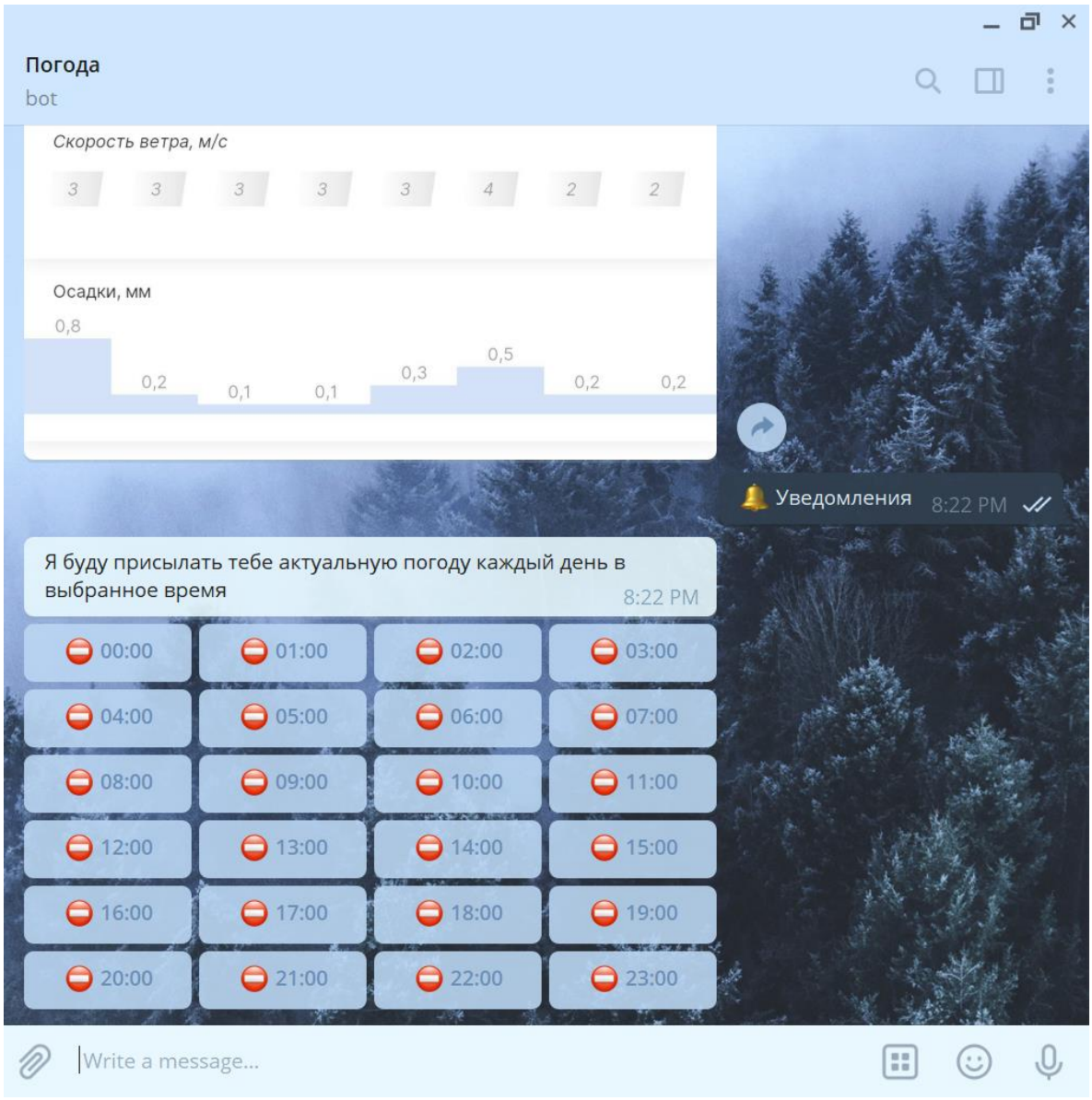

Рисунок 1.8 – Функція сповіщення Телеграмм бота «Погода»

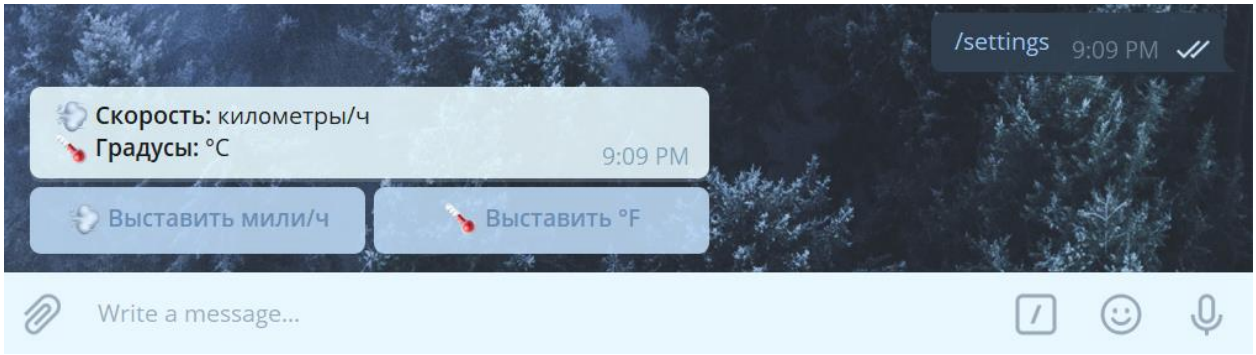

Рисунок 1.9 – Налаштування додатку «TheLair Wheather»

Серед переваг реалізації телеграм бота “What to Wear?” можна виділити наступні:

- Відсутня необхідність збереження міста або координат для пошуку прогнозу погоди. Досить ввести назву міста.
- Підтримується функція «Щоденника погоди», адже реалізація дозволяє робити запит на пошук у будь який день в минулому.
- Детальна інструкція користування додатком.
- Інтуїтивно зрозумілий інтерфейс.

1.3. Постановка задачі

Метою роботи є розробка телеграм боту, що збирає дані про погоду з сервісу Sinoptik.ua. Вимоги до функціоналу бота:

- Бот повинен мати можливість зберегти місто користувача для пошуку погоди у ньому надалі.
- Бот повинен підримувати функцію пошуку прогнозу погоди за назвою міста, яку вводить користувач або за назвою міста, яке користувач зберіг.
- Бот повинен мати можливість змінити збережене користувачем місто.
- Бот повинен мати можливість шукати прогноз погоди у вказаний користувачем день.
- Повідомлення з прогнозом погоди повинне містити в собі:
    1) фактичну температуру повітря
    2) температуру повітря “як відчувається”
    3) день, на який здійснюється прогноз
    4) посилання на адресу сторінки з прогнозом погоди
    5) коротке повідомлення з описом погоди протягом дня

Описати кроки відтворення кожного сценарію та зробити висновок про виконану роботу.

2. Технології розв’язку поставленої задачі

2.1. Структурно-функціонального моделювання процесів

Для функціонального моделювання процесів буде використовуватися методологія IDEF0. Основа методології IDEF0 – графічна мова опису системи. Спочатку проводиться опис системи в цілому і її взаємодії з навколишнім світом (контекстна діаграма), після чого проводиться функціональна декомпозиція - система розбивається на підсистеми і кожна підсистема описується окремо (діаграми декомпозиції). За допомогою блоків на даній діаграмі зображуються процеси, які з’єднуються стрілками – зв’язки між процесами. На графіку IDEF0 може бути 5 типів стрілок:

1) Вхід - об'єкти, що використовуються як вхідні дані для процесу. Допускається, що робота може не мати жодної стрілки входу. Стрілка входу направлена в лівий бік процесу.
2) Управління - інформація, керуюча процесами роботи. Керуючі стрілки несуть інформацію, яка вказує, що повинен виконувати процес і яким чином. Кожен процес повинен мати хоча б одну стрілку управління, яка направлена у верхню грань.
3) Вихід - об'єкти, в які перетворюються входи. Кожен процес повинен мати хоча б одну стрілку виходу, яка виходить з правої межі процесу.
4) Механізм - ресурси, які виконують роботу. Стрілка механізму малюється направленою в нижню межу процесу.
5) Виклик - спеціальна стрілка, що вказує на іншу модель роботи. Стрілка виклику направлена з нижньої частини процесу і використовується для вказівки того, що деяка робота виконується за межами модельованої системи. Не є обов’язковою.

Нижче наведені контекстна IDEF0 (Рисунок 2.1) та IDEF0 першого рівня діаграми (Рисунок 2.2)

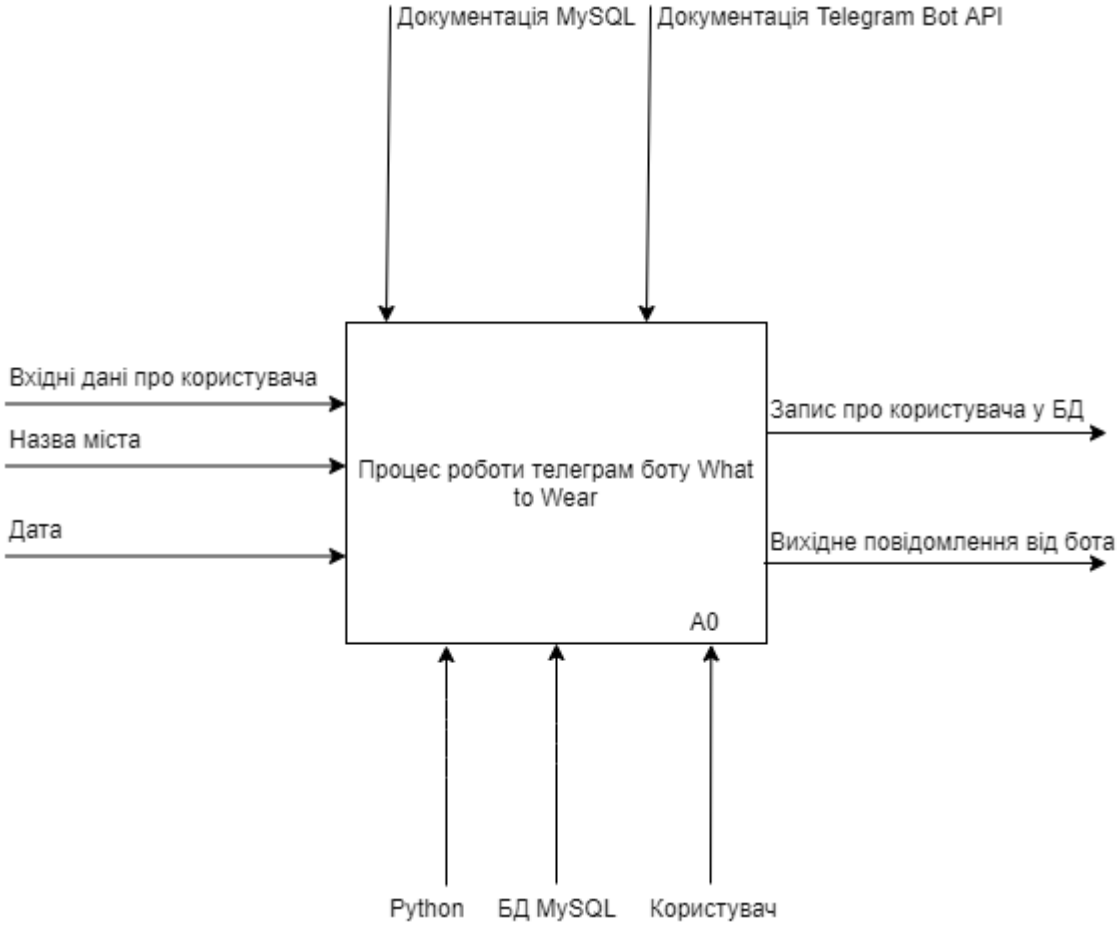

Рисунок 2.1 – Контекста IDEF0 діаграма

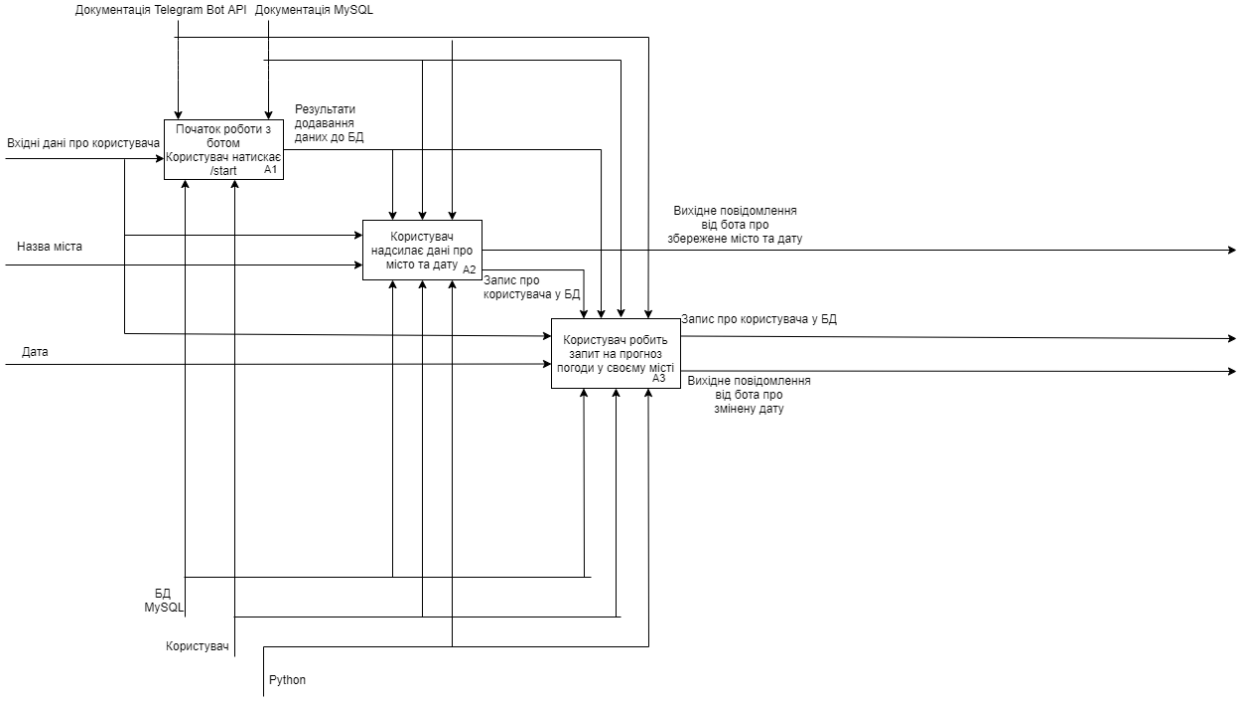

Рисунок 2.2 – IDEF0 діаграма першого рівня

2.2. Діаграма варіантів використання

Діаграми варіантів використання призначені для спрощення взаємодії з майбутніми користувачами системи, з клієнтами. Діаграма показує, що система повинна робити, не вказуючи самі застосовувані методи.

При роботі з варіантами використання важливо дотримуватися наступних правил:

- кожен варіант використання відноситься як мінімум до однієї дійової особи,
- кожен варіант використання має ініціатора,
- кожен варіант використання призводить до відповідного результату,

Варіанти використання також можуть взаємодіяти з іншими варіантами використання. Типи взаємодії між варіантами використання наведені нижче:

- Включення вказує, що варіант використання вбудовується в інший варіант використання.
- Додавання вказує, що в певних ситуаціях або в певній точці (точці розширення) варіант використання буде розширено іншим.
- Узагальнення вказує, що варіант використання успадковує характеристики «батьківського» варіанту використання і може перевизначити деякі з них або додати нові, подібно спадкоємства в класах.

Для реалізації діаграми варіантів використання для практичної роботи виділені наступні варіанти використання:

- Запит прогнозу погоди за введеним містом.
- Збереження введеного міста до бази даних.
- Зміна дати для прогнозу погоди.
- Запит прогнозу погоди у збереженому місті.
- Перегляд результатів виконання операцій.

Єдиним актором є користувач. Діаграма варіантів використання (Рисунок 2.3) наведена нижче:

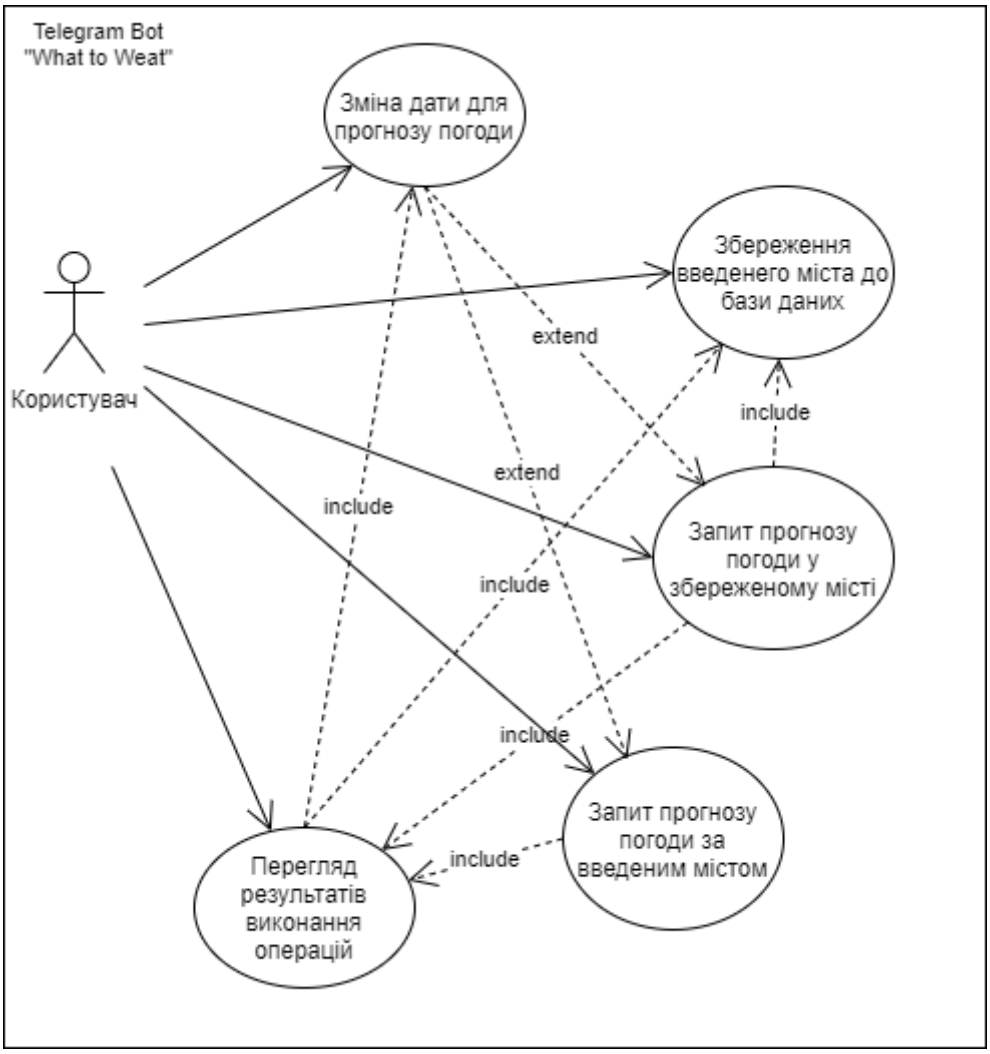

Рисунок 2.3 – Діаграма варіантів використання

З наведеної вище діаграми варіантів використання ми можем визначити наступні умови користування ботом:

1. Користувач мусить виконати будь-яку операцію для того, щоб переглянути результат.
2. Користувач повинен зберегти місто для того, щоб виконати запит прогнозу погоди у збереженому місті.
3. Користувач може змінити дату для запиту прогнозу погоди за введеним містом.
4. Користувач може змінити дату для запиту прогнозу погоди за збереженим містом.

2.3. ER – діаграма

ER-модель (Entity-relationship model або Entity-relationship diagram) - це семантична модель даних, яка призначена для спрощення процесу проектування бази даних. В основі ER-моделі лежать поняття «сутність», «зв'язок» і «атрибут». Дана діаграма дозволяє розглянути систему цілком і з'ясувати вимоги, необхідні для її розробки, що стосуються зберігання інформації.

Під час проектування ERD було з’ясовано, що для майбутньої реалізації бази даних досить однії таблиці. ER-діаграма (Рисунок 2.4) та опис атрибутів (Рисунок 2.5) наведені нижче:

Рисунок 2.4 – ER-діаграма

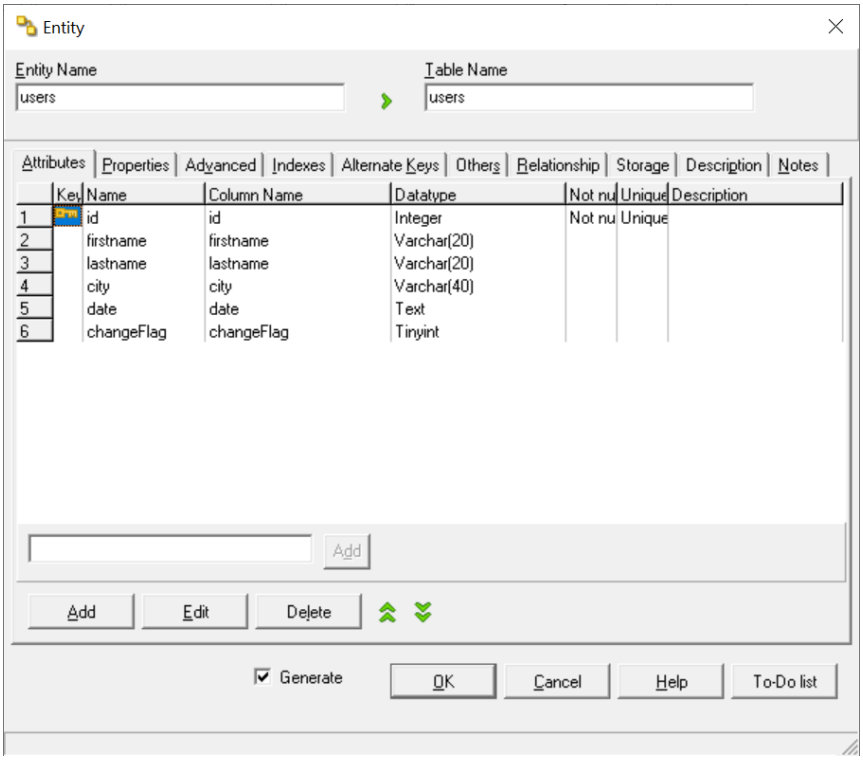

Рисунок 2.5 – Опис атрибутів сутності users

3. Програмна реалізація

3.1. Вибір засобів програмної реалізації

3.1.1. Мова програмування

Для реалізації телеграм боту “What to Wear” мовою програмування був обраний Python. Основна перевага мови Python полягає в легкості читання, ясності і більш високій якості, що відрізняють його від інших мов програмування. Програмний код на мові Python читається легше, а значить, багаторазове його використання і обслуговування виконується набагато простіше, ніж використання програмного коду на інших мовах програмування. Одноманітність оформлення програмного коду на мові Python полегшує його розуміння навіть для тих, хто не брав участі в його створенні. Крім того, Python підтримує сучасні механізми багаторазового використання програмного коду, яким є об'єктно-орієнтоване програмування (ООП).

Окрім зазначених переваг Python - одна з найкращих мов для досягнення високої швидкості розробки програмного забезпечення. У порівнянні з компілюючими або строго типізованими мовами, такими як C, C ++ і Java, Python у багато разів підвищує продуктивність праці розробника. Обсяг програмного коду на мові Python зазвичай становить 20-30% від еквівалентного коду на мові C ++ або Java. Це означає менший обсяг введення з клавіатури, менша кількість часу на налагодження і менший обсяг трудовитрат на підтримку. Крім того, програми на мові Python запускаються відразу ж, минаючи тривалі етапи компіляції.

Важливим фактором вибору цієї мови програмування є великий спектр корисних OpenSource пакетів, які досить зручно використовувати для реалізації проектів. Одним з таких пакетів є саме telebot – реалізація Telegram Bot API мовою Python.

Під час реалізації боту наступні методи з Telegram Bot API були використані:

1) sendMessage(chat_id, text) – метод для відправлення повідомлення ботом користувачу

2) messageHandler() – обробник, необхідний для вилову повідомлень від користувачів.

3.1.2. СУБД

У якості СУБД був обраний MySQL. Дана система характеризується великою швидкістю, стабільністю і легкістю у використанні. Частіше є рішенням для малих і середніх додатків. Нарівні з Oracle Database це одна з найшвидших СУБД на сьогоднішній день. Поширення СУБД MySQL на основі GPL і висока швидкість обробки запитів призвело до того, що ця база даних стала стандартом в послугах мережевого хостингу. Зазвичай MySQL використовується як сервер, до якого звертаються локальні або видалені клієнти, проте в дистрибутив входить бібліотека внутрішнього сервера, що дозволяє включати MySQL в автономні програми.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як надшвидкі таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і більш повільні, але надзвичайно стійкі таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Завдяки відкритій архітектурі і GPL ліцензуванню в СУБД MySQL постійно з'являються нові типи таблиць.

MySQL портована на велику кількість платформ: Windows, BSDi, Linux, Mac OS X, NetBSD і ін. Важливо відзначити, що на офіційному сайті СУБД для вільного завантаження надаються не тільки вихідні коди, але й компільовані і оптимізовані під конкретні операційні системи готові модулі СУБД MySQL.

Вхідні дані для бази даних відсутні, оскільки вони накопичуються у процесі роботи користувачів із ботом.

Для зручності адміністрування бази даних був використаний додаток PhpMyAdmin. Це Web-додаток з відкритим кодом, написаний на мові PHP і представляє собою Web-інтерфейс для адміністрування баз даних MySQL. За

допомогою програми phpMyAdmin можна створювати, видаляти і редагувати таблиці бази даних, виконувати окремі SQL-запити, створювати і видаляти користувачів, змінювати їх привілегії.

Програма phpMyAdmin також дозволяє через Web-браузер здійснювати адміністрування сервера MySQL, запускати команди SQL по роботі з вмістом таблиць баз даних, управляти СУБД MySQL без безпосереднього введення SQL команд, представляючи зручний для користувача інтерфейс.

3.1.3. Середовище розробки

Для розробки було використано популярне та високо оцінюване інтегроване середовище розробки – PyCharm. Це зручне IDE надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт-тестів та багато інших корисних дрібниць, таких як підказка по сигнатурі функції (Рисунок 3.1), знаходження всіх згадувань конструкції (Рисунок 3.2) та інші.

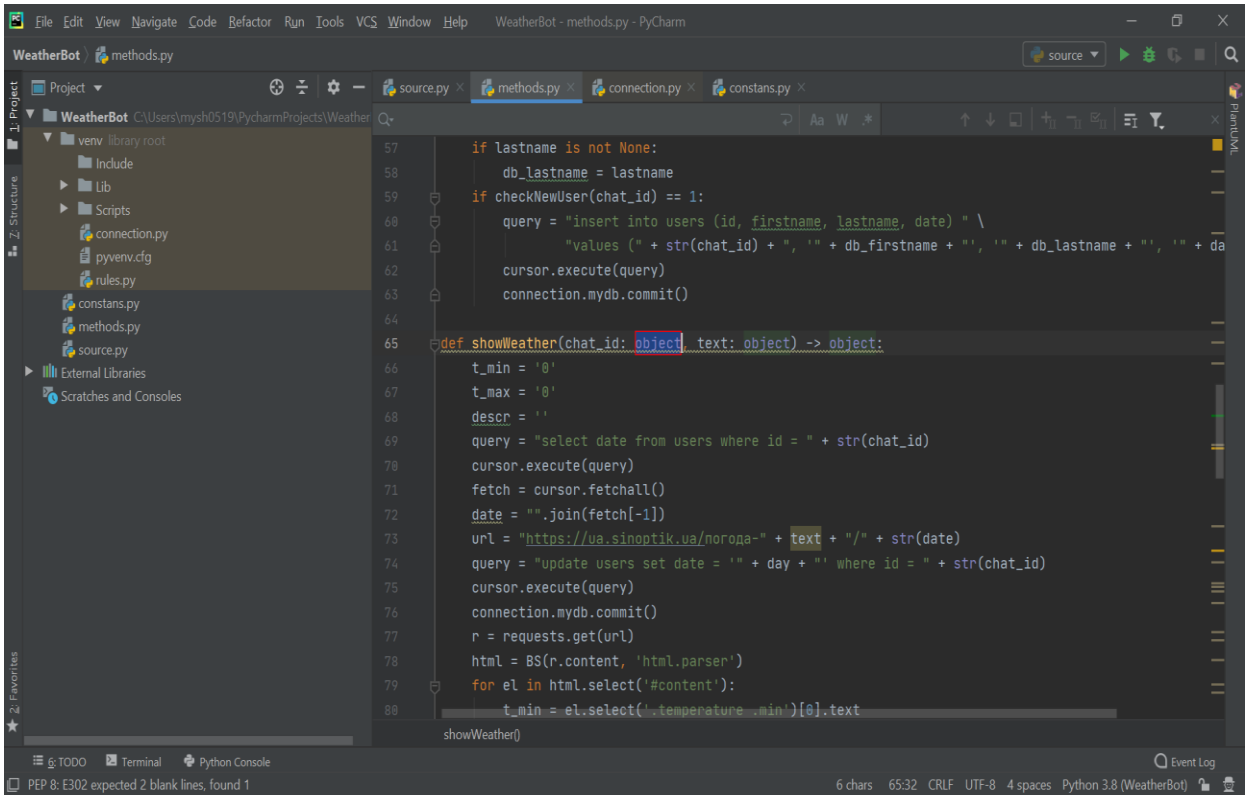

Рисунок 3.1 – Підказка по сигнатурі функції

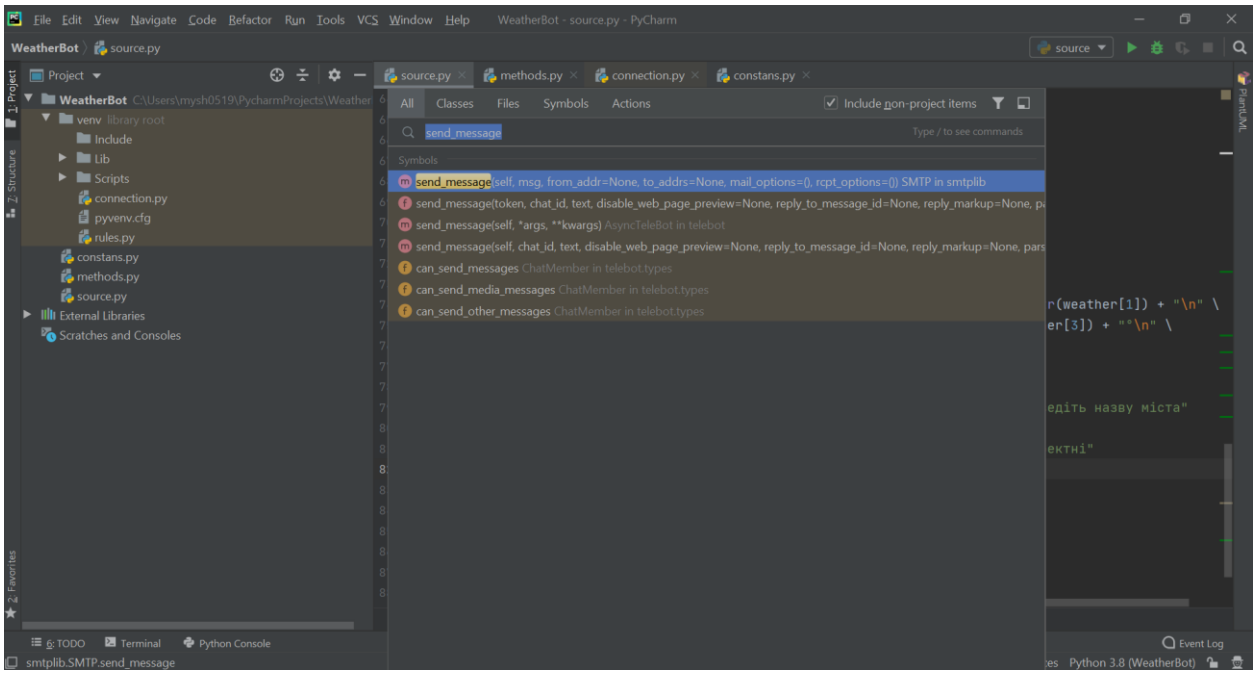

Рисунок 3.2 – Знаходження всіх згадувань конструкції

3.2. Опис розроблених методів

Нижче наведений список методів для роботи телеграм боту:

1) checkNewUser(chat_id) – метод приймає id чату з користувачем та повертає результат 1 або 0 в залежності від того, чи наявний запис про користувача у базі даних.
2) addUser(chat_id, firstname, lastname) – метод для занесення запису про користувача до бази даних.
3) showWeather(chat_id, text) – основний метод, який приймає id чату та назву міста як вхідні параметри. За допомогою парсингу метод знаходить та повертає необхідні дані із сторінки сайту Sinoptik.ua Парсинг сторінки реалізований за допомогою пакету Beautiful Soup. Метод select() запускає CSS selector для пошуку необхідних елементів сторінки та повертає їх масив. У разі виникнення проблем з дієздатністю сервісу Sinoptik.ua користувач побачить повідомлення про тимчасові несправності.
4) changeRequest(chat_id, text) – метод для зміни даних про користувача, збережених у базі даних. Даний метод змінює

збережене місто користувача або дату необхідну для прогнозу погоди.

5) setHometownFlag(chat_id) – метод, що фіксує намір користувача зберегти нове місто проживання.
6) setChangeDateFlag(chat_id) - метод, що фіксує намір користувача змінити дату для наступного прогнозу погоди.
7) findExtremums(t_sens) – метод для знаходження мінімальної та максимальної температури, що фактично відчувається у певний день. Метод викликається всередині методу showWeather(chat_id, text). Необхідний для більш детального аналізу температури.
8) getCity(chat_id) – метод, що повертає збережене користувачем місто. Фоновий метод необхідний для пошуку прогнозу погоди.

4. Результати роботи

Для демонстрації результатів роботи нижче наведені скріншоти з основними сценаріями:

1) До початку роботи з ботом база даних не має записів (Рисунок 4.1)

id	firstname	lastname	city	date	changeFlag

Рисунок 4.1 – Таблиця users до початку роботи з базою даних

2) Для початку роботи з ботом необхідно натиснути кнопку “Start” (Рисунок 4.2)

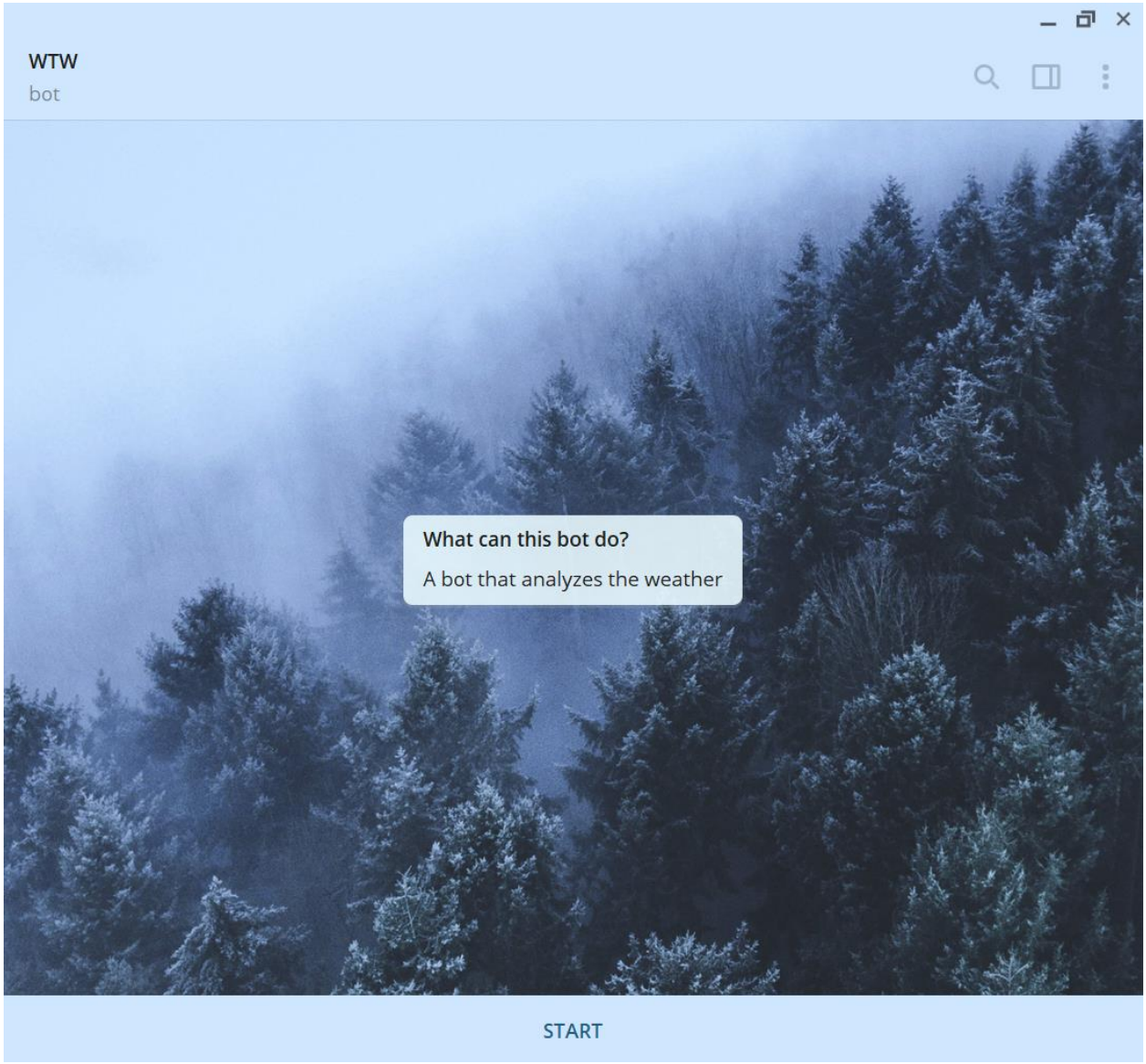

Рисунок 4.2 – Початок роботи з ботом

3) Після натискання кнопки “Start” з’явиться панель керування ботом та інструкція користування (Рисунок 4.3). Також додався запис про користувача до таблиці Users (Рисунок 4.4).

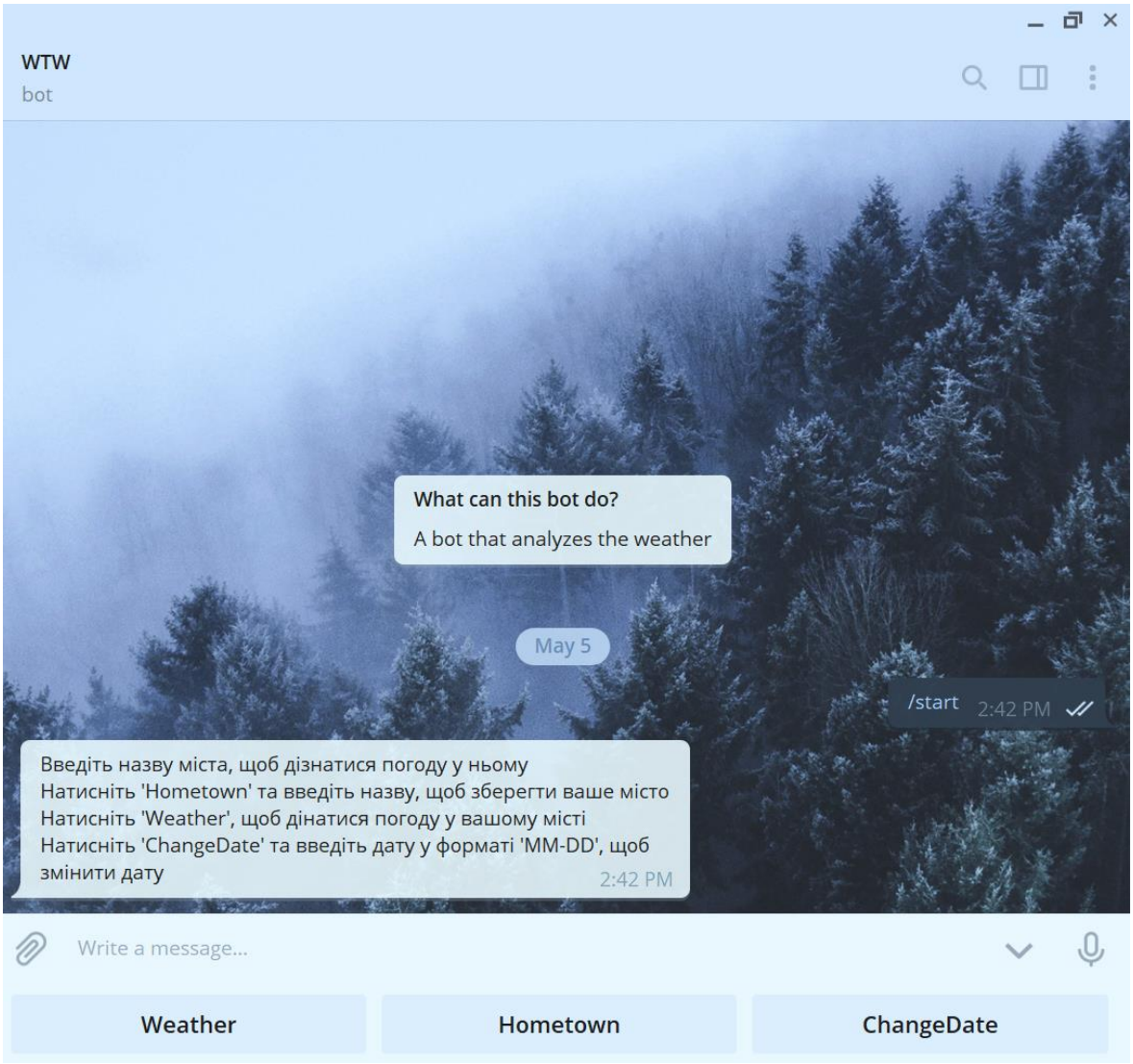

Рисунок 4.3 – Панель керування та інструкція

Рисунок 4.4 – Запис про користувача у таблиці Users

4) Для відображення прогнозу погоди у конкретному місті необхідно ввести його назву (Рисунок 4.5)

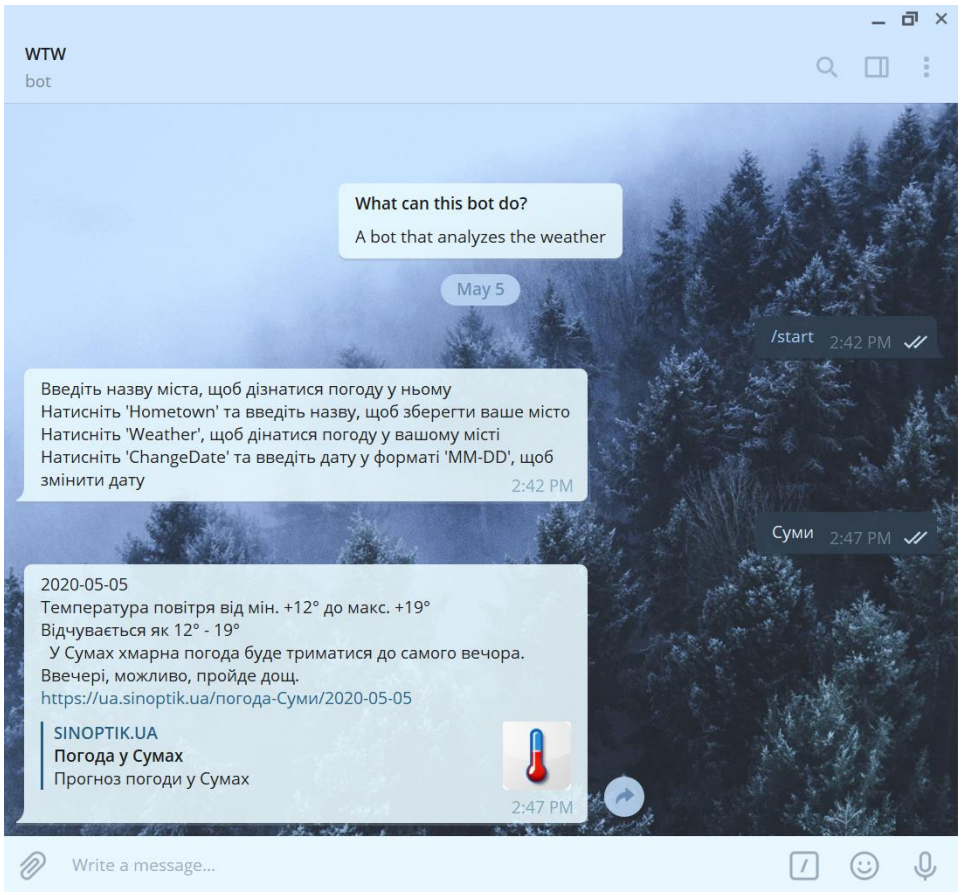

Рисунок 4.5 – Прогноз погоди у місті Суми

5) Нижче наведений скріншот роботи з ботом через панель керування (Рисунок 4.6) та збережене місто у базі даних (Рисунок 4.7)

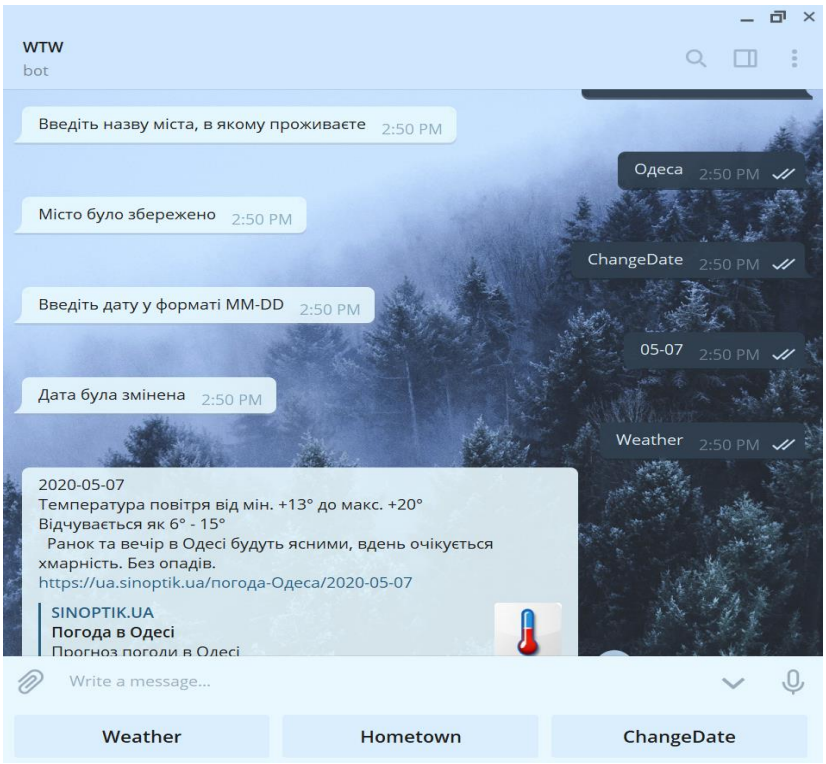

Рисунок 4.6 – Результат роботи через панель керування

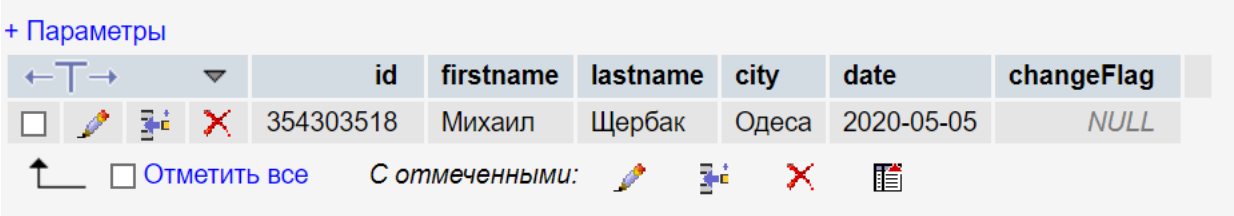

Рисунок 4.7 –Місто збережене у базі даних

6) Повідомлення у випадку некоректно заданих дати або міста для прогнозу погоди (Рисунок 4.8)

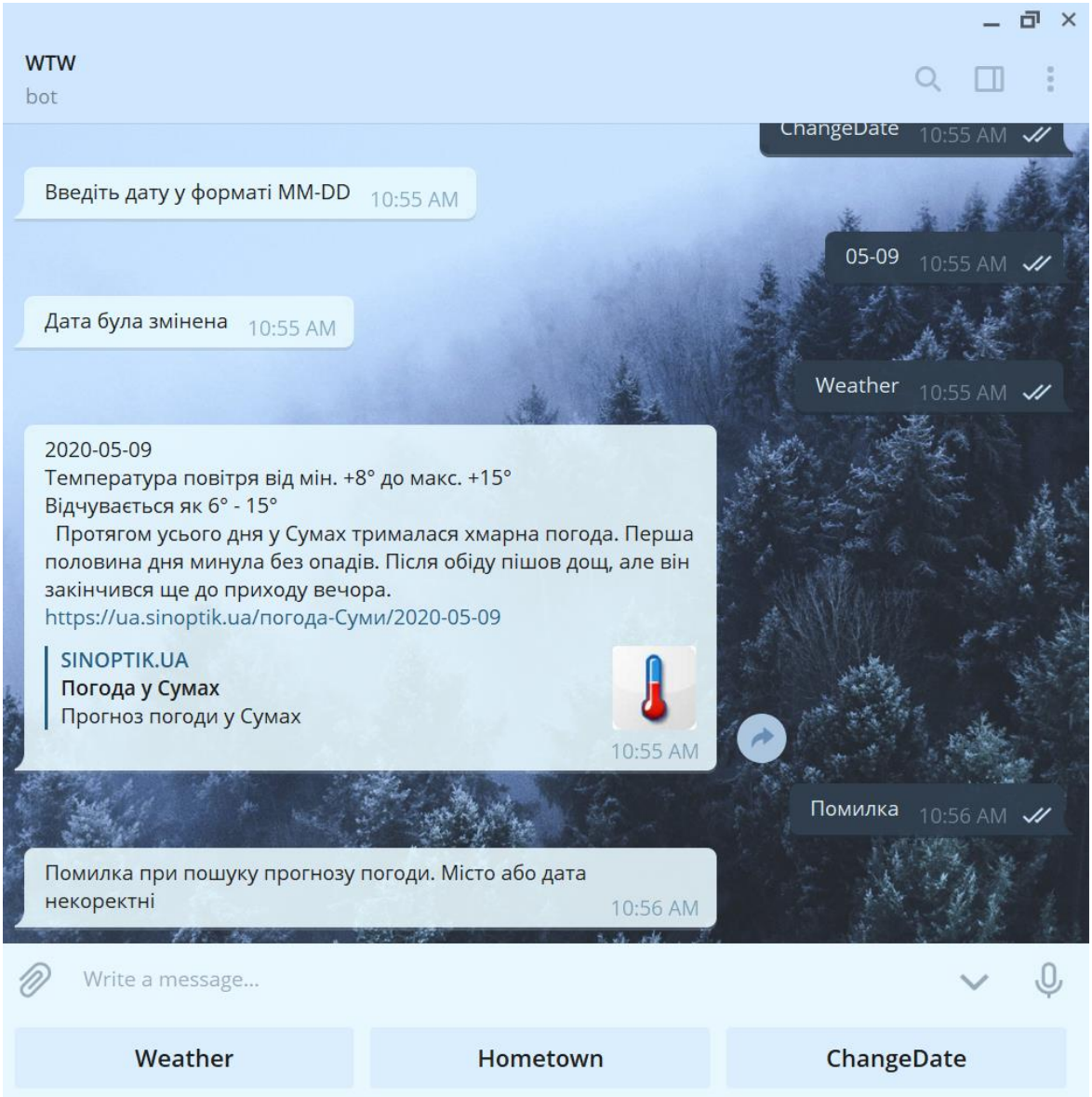

Рисунок 4.8 – Повідомлення про помилку вхідних даних

При натисканні на ім’я бота відкривається сторінка з профілем, що містить інформацію про бота (Рисунок 4.9)

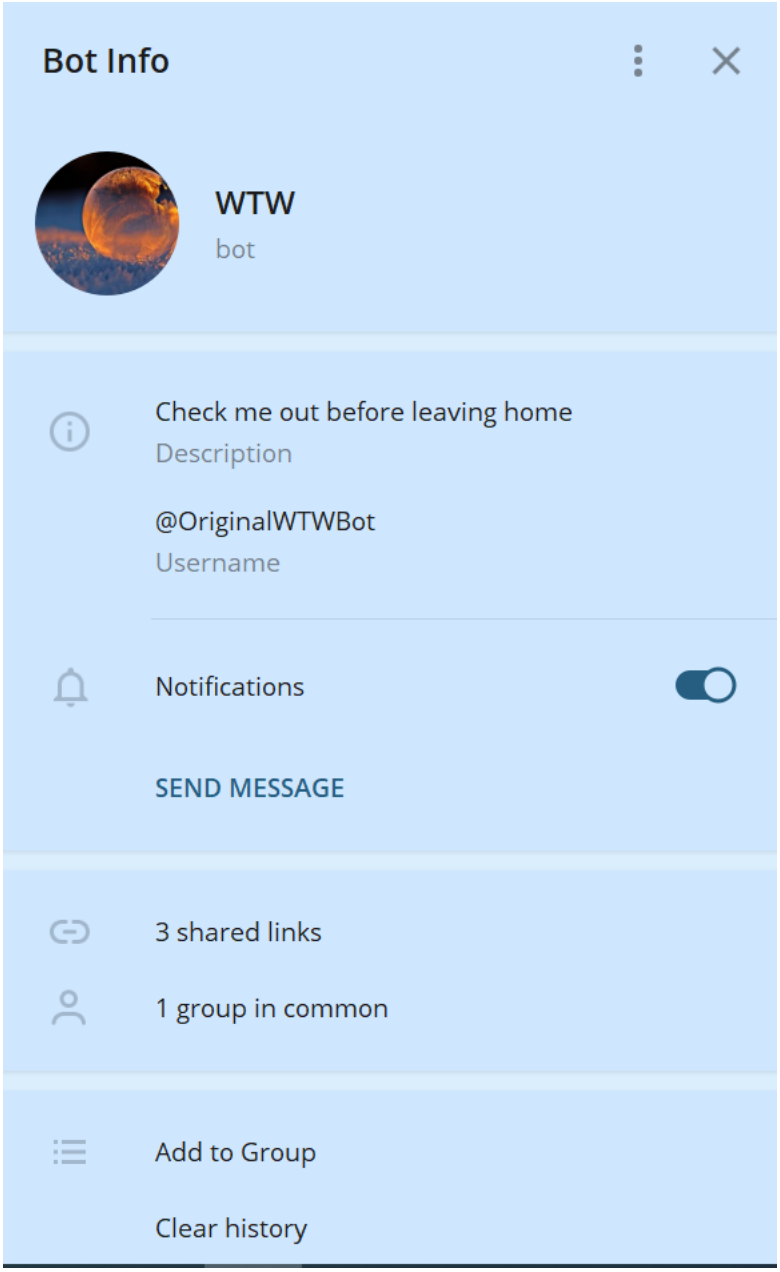

Рисунок 4.9 – Сторінка Bot Info

Висновок

Під час виконання роботи був реалізований телеграм бот “What to Wear?” мовою Python з використанням СУБД MySQL для перегляду прогнозу погоди. Всі вимоги дотримані, варіанти використання протестовані.

Відповідно до поставлених задач реалізовані можливості запиту прогнозу погоди за введеним або збереженим містом, збереження введеного міста до бази даних, зміни дати для прогнозу погоди та перегляду результатів виконання операцій.

З можливих покращень для бота у майбутньому можна виділити самостійний вибір ресурсу, з якого бот буде брати прогнози погоди, самостійне попередження ботом у разі виникнення погодних катастроф та прогноз погоди міста, що було визначене геолокацією користувача.

Серед технічних вдосконалень можна реалізувати кешування даних для підвищення швидкості роботи додатку. Ця функція є досить корисною через те, що в даній реалізації зустрічаються запити до бази даних, що виконуються досить часто за короткий проміжок часу.

За час виконня практики покращив навики у програмуванні мовою Python. Поглибив знання у сфері розробки ботів для телеграму.

Список літератури

1. Олексій Васильєв. Програмування мовою Python. 2019
2. Дмитро Златопольський. Основы программированя на языке Python. 2016
3. Real Python Course, Part 1 Real Python Team. Real Python. 2017
4. MySQL Connector/Python Developer Guide. [Електронний ресурс] — Режим доступу: https://dev.mysql.com/doc/connector-python/en/ (дата звернення 25.04.2020) – Назва з екрана.
5. Jesper Wisborg Krogh. MySQL Connector/Python Revealed: SQL and NoSQL Data Storage Using MySQL for Python Programmers 1st ed. Edition. 2018
6. David Stokes. MySQL and JSON: A Practical Programming Guide 1st Edition, Kindle Edition. 2018
7. Практика користування соцмережами в Україні [Електронний ресурс] — Режим доступу: http://rb.com.ua/blog/praktika-polzovanija-socsetjami-v-ukraine/?fbclid=IwAR3LAJVpOwYPgDWdQ7uqvkWXrUnX8S_rkXry9nhmtuxsNUsinpNnn96zIvs (дата звернення 02.05.2020) – Назва з екрана.
8. Telegram Bot API [Електронний ресурс] — Режим доступу: https://core.telegram.org/bots/api (дата звернення 03.05.2020) – Назва з екрана.
9. Щоденник погоди [Електронний ресурс] — Режим доступу: https://rp5.ua/ (дата звернення 05.05.2020) – Назва з екрана.
10. Beautiful Soup Documentation [Електронний ресурс] — Режим доступу: https://www.crummy.com/software/BeautifulSoup/bs4/doc/ (дата звернення 06.05.2020) – Назва з екрана.
11. Опади, що випадають з хмар та з повітря, їх види, вимірювання. Карта розподілу опадів [Електронний ресурс] — Режим доступу: https://vseosvita.ua/library/opadi-so-vipadaut-z-hmar-ta-z-povitra-ih-vidi-

vimiruvanna-karta-rozpodilu-opadiv-74220.html (дата звернення 07.05.2020) – Назва з екрана.

12. Python Pros and Cons: What are The Benefits and Downsides of the Programming Language [Електронний ресурс] — Режим доступу: https://www.netguru.com/blog/python-pros-and-cons (дата звернення 10.05.2020) – Назва з екрана.
13. Benefits of Python over Other Programming Languages [Електронний ресурс] — Режим доступу: https://www.invensis.net/blog/it/benefits-of-python-over-other-programming-languages/ (дата звернення 12.05.2020) – Назва з екрана.
14. Five Advantages & Disadvantages Of MySQ [Електронний ресурс] — Режим доступу: https://www.datarealm.com/blog/five-advantages-disadvantages-of-mysql/ (дата звернення 14.05.2020) – Назва з екрана.
15. Python 3.8.3 documentation [Електронний ресурс] — Режим доступу: https://docs.python.org/3/ (дата звернення 10.06.2020) – Назва з екрана.
16. The advantages and disadvantages of MySQL [Електронний ресурс] — Режим доступу: http://makble.com/the-advantages-and-disadvantages-of-mysql (дата звернення 11.06.2020) – Назва з екрана.
17. 2017 Bot Overview: Perspective on Top Bots and Top Channels for Bots [Електронний ресурс] — Режим доступу: https://chatbotsmagazine.com/2017-bot-overview-perspective-on-top-bots-and-top-channels-for-bots-d03310f9a864 (дата звернення 12.06.2020) – Назва з екрана.
18. What is PyCharm? Features, Advantages & Disadvantages [Електронний ресурс] — Режим доступу: https://hackr.io/blog/what-is-pycharm (дата звернення 13.06.2020) – Назва з екрана.
19. PyCharm Features [Електронний ресурс] — Режим доступу: https://www.jetbrains.com/pycharm/features/ (дата звернення 13.06.2020) – Назва з екрана.

Додаток А. SQL скрипт для створення таблиці

```
CREATE TABLE `users` (
  `id` int(10) NOT NULL,
  `firstname` varchar(20) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `lastname` varchar(20) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `city` varchar(40) COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `date` text COLLATE utf8mb4_unicode_ci DEFAULT NULL,
  `changeFlag` tinyint(1) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_unicode_ci;
```

Додаток Б. Програмний код

source.py

```
import telebot
import constans
import methods

bot = telebot.TeleBot(constans.token)

def log(message, answer):
    from datetime import datetime
    print(datetime.now())
    print("Message from {0} {1}. (id = {2}) \n Text -
{3}".format(message.from_user.first_name,

message.from_user.last_name,

str(message.from_user.id),

message.text))
    print(answer)


@bot.message_handler(commands=['start', 'help'])
def handle_start(message):
    answer = "Введіть назву міста, щоб дізнатися погоду у ньому\n" \
             "Натисніть 'Hometown' та введіть назву, щоб зберегти ваше
місто\n" \
             "Натисніть 'Weather', щоб дінатися погоду у вашому місті\n" \
             "Натисніть 'ChangeDate' та введіть дату у форматі 'MM-DD', щоб
змінити дату"
    user_markup = telebot.types.ReplyKeyboardMarkup(True, False)
    weather_button = telebot.types.KeyboardButton("Weather")
    hometown_button = telebot.types.KeyboardButton("Hometown")
    change_date_button = telebot.types.KeyboardButton("ChangeDate")
    log(message, answer)
    user_markup.add(weather_button, hometown_button, change_date_button)
    bot.send_message(message.chat.id, answer, reply_markup=user_markup)

    methods.addUser(message.chat.id, message.from_user.first_name,
message.from_user.last_name)


@bot.message_handler(content_types=['text'])
def handle_text(message):
    flags = None
    answer = ""

    response = methods.changeRequest(message.chat.id, message.text)
    if response == 1:
        bot.send_message(message.from_user.id, "Місто було збережено")
        flags = 1
    elif response == 0:
        bot.send_message(message.from_user.id, "Дата була змінена")
        flags = 0
    elif response is not None:
        bot.send_message(message.from_user.id, "Помилка при введенні дати
(формат mm-dd)")
        flags = "Error"

    if message.text == "Hometown":
        methods.setHometownFlag(message.chat.id)
```

```
        bot.send_message(message.from_user.id, "Введіть назву міста, в якому
проживаєте")
        flags = 1
    if message.text == "ChangeDate":
        methods.setChangeDateFlag(message.chat.id)
        bot.send_message(message.from_user.id, "Введіть дату у форматі MM-
DD")
        flags = 0

    if flags is None:
        if message.text == "Weather":
            city = methods.getCity(message.chat.id)
            if city != "Error":
                weather = methods.showWeather(message.chat.id, city)
            else:
                weather = "ErrorHometown"
        else:
            weather = methods.showWeather(message.chat.id, message.text)

        if weather != "Error" and weather != "ErrorHometown":
            answer = weather[5] + "\n"\
                     "Температура повітря від " + str(weather[0]) + " до " +
str(weather[1]) + "\n" \
                      "Відчувається як " + str(weather[2]) + "° - " +
str(weather[3]) + "°\n" \
                      "" + str(weather[4]) + "\n" \
                      "" + str(weather[6])
        elif weather == "ErrorHometown":
            answer = "Вы ще не зберегли своє місто. Натисніть 'Hometown' та
введіть назву міста"
        else:
            answer = "Помилка при пошуку прогнозу погоди. Місто або дата
некоректні"
        bot.send_message(message.from_user.id, answer)

    log(message, answer)


bot.polling(none_stop=True, interval=0)
```

methods.py

```
from datetime import date

import connection
import requests
from bs4 import BeautifulSoup as BS

day = date.today().strftime("%Y-%m-%d")
cursor = connection.mydb.cursor()

def checkNewUser(chat_id):

    query = "select id from users where id = " + str(chat_id)
    cursor.execute(query)
    fetch = cursor.fetchall()
    if len(fetch) != 0:
        return 0
    else:
        return 1

def changeRequest(chat_id, text):
    query = "select changeFlag from users where id = " + str(chat_id)
```

```
    cursor.execute(query)
    result = cursor.fetchall()
    if str(result[-1]) == "(1,)":
        query = "update users set city = '" + str(text) + "' where id = " +
str(chat_id) + ";"
        cursor.execute(query)
        query = "update users set changeFlag = null where id = " +
str(chat_id) + ";"
        cursor.execute(query)
        connection.mydb.commit()
        return 1
    elif str(result[-1]) == "(0,)":
        try:
            query = "update users set date = '2020-" + str(text) + "' where
id = " + str(chat_id) + ";"
            cursor.execute(query)
            query = "update users set changeFlag = null where id = " +
str(chat_id) + ";"
            cursor.execute(query)
            connection.mydb.commit()
            return 0
        except connection.mysql.connector.Error as err:
            return err

def setHometownFlag(chat_id):
    query = "update users set changeFlag = True where id = " + str(chat_id) +
";"
    cursor.execute(query)
    connection.mydb.commit()

def setChangeDateFlag(chat_id):
    query = "update users set changeFlag = False where id = " + str(chat_id)
+ ";"
    cursor.execute(query)
    connection.mydb.commit()

def addUser(chat_id, firstname, lastname):
    db_firstname = "null"
    if firstname is not None:
        db_firstname = firstname
    db_lastname = "null"
    if lastname is not None:
        db_lastname = lastname
    if checkNewUser(chat_id) == 1:
        query = "insert into users (id, firstname, lastname, date) " \
                "values (" + str(chat_id) + ", '" + db_firstname + "', '" +
db_lastname + "', '" + day + "')"
        cursor.execute(query)
        connection.mydb.commit()

def showWeather(chat_id, text):
    t_min = '0'
    t_max = '0'
    descr = ''
    query = "select date from users where id = " + str(chat_id)
    cursor.execute(query)
    fetch = cursor.fetchall()
    date = "".join(fetch[-1])
    url = "https://ua.sinoptik.ua/погода-" + text + "/" + str(date)
    query = "update users set date = '" + day + "' where id = " +
str(chat_id)
    cursor.execute(query)
    connection.mydb.commit()
```

```
    r = requests.get(url)
    html = BS(r.content, 'html.parser')
    for el in html.select('#content'):
        t_min = el.select('.temperature .min')[0].text
        t_max = el.select('.temperature .max')[0].text
        descr = el.select('.wDescription .description')[0].text
        t_sens1 = el.select('.temperatureSens .p1')[0].text
        t_sens2 = el.select('.temperatureSens .p2')[0].text
        t_sens3 = el.select('.temperatureSens .p3')[0].text
        t_sens4 = el.select('.temperatureSens .p4')[0].text
        try:
            t_sens5 = el.select('.temperatureSens .p5')[0].text
            t_sens6 = el.select('.temperatureSens .p6')[0].text
            t_sens7 = el.select('.temperatureSens .p7')[0].text
            t_sens8 = el.select('.temperatureSens .p8')[0].text
        except Exception as ex:
            t_sens5 = t_sens1
            t_sens6 = t_sens2
            t_sens7 = t_sens3
            t_sens8 = t_sens4
    try:
        t_sens = [t_sens1, t_sens2, t_sens3, t_sens4, t_sens5, t_sens6,
t_sens7, t_sens8]
        t_sens = findExtremums(t_sens)
        weather = (t_min, t_max, t_sens[0], t_sens[1], descr, date, url)
    except Exception:
        weather = "Error"
    return weather


def findExtremums(t_sens):
    min = 999
    max = -999
    result_t_sens = []
    for t in t_sens:
        if (t[0] == '-'):
            buf_str = t.replace('°', '').replace('-', '')
            result_t_sens.append(0 - int(buf_str))
        else:
            result_t_sens.append(int(t.replace('°', '')))
    for t in result_t_sens:
        if int(t) < min:
            min = t
        if int(t) > max:
            max = t
    result_t_sens.clear()
    result_t_sens = [min, max]
    return result_t_sens

def getCity(chat_id):
    try:
        query = "select city from users where id = " + str(chat_id)
        cursor.execute(query)
        fetch = cursor.fetchall()
        city = "".join(fetch[-1])
    except Exception as e:
        city = "Error"
    return city
```

connect.py

```
import mysql.connector
import constans

try:
    mydb = mysql.connector.connect(
        host=constans.host,
        user=constans.user,
        password=constans.password,
        database=constans.database
    )
except mysql.connector.Error as err:
  if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
    print("Something is wrong with your user name or password")
  elif err.errno == errorcode.ER_BAD_DB_ERROR:
    print("Database does not exist")
  else:
    print(err)
```