**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 проект

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою “Комп’ютерні системи та мережі”**

**спеціальності 123 “Комп’ютерна інженерія”**

на тему: Система рекомендацій гардеробу на основі прогнозу погоди

Виконав: студент 4 курсу, групи ІО-82

Білич Максим Валерійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник проф., д.ф-м.н., с.н.с. Гордієнко Ю.Г.
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

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

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

Київ - 2022 р.

## ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

| № з/п | Формат | Позначення | Найменування | Кількість листів | Примітка |
|---|---|---|---|---|---|
| 1 | А4 | | Завдання на дипломний проект | 2 | |
| 2 | А4 | ІАЛЦ.467200.001 ВП | Відомість дипломного проекту | 1 | |
| 3 | А4 | ІАЛЦ.467200.002 ТЗ | Технічне завдання | 3 | |
| 4 | А4 | ІАЛЦ.467200.003 ПЗ | Пояснювальна записка | 91 | |
| 5 | А4 | ІАЛЦ.467200.004 Д1 | Структурна схема проектованого програмного продукту типу чат-бот для Telegram | 1 | |
| 6 | А4 | ІАЛЦ.467200.004 Д2 | Функціональна схема: Діаграма діяльності процесу нечіткого виводу | 1 | |
| 7 | А4 | ІАЛЦ.467200.004 Д1 | Принципова схема: Блок-схема алгоритму чат-бота, що надає інформація про погодні умови | 1 | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| Змн. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.467200.001 ВП | | |
|---|---|---|---|---|---|---|---|
| Розроб. | | Білич М.В.. | | | Система рекомендацій гардеробу на основі прогнозу погоди | Літ. | Арк. 2 / Акрушів 1 |
| Перевір. | | Гордієнко Ю.Г. | | | | | |
| Н. Контр. | | Сімоненко В.П. | | | | КПІ ім. Ігоря Сікорського ФІОТ Група ІО-82 | |
| Затверд. | | Стіренко С.Г. | | | | | |

**Національний технічний університет України**
**«Київський політехнічний інститут»**

Інститут (факультет) інформатики та обчислювальної техніки
(повна назва)

Кафедра обчислювальної техніки
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»
(повна назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____Сергій, СТІРЕНКО
" " 2022 року

**ЗАВДАННЯ**
**на дипломний проект студента**

_____Білич Максима Валерійовича_____
(ім'я, прізвище)

1. Тема проекту *«Система рекомендацій гардеробу на основі прогнозу погоди»*

керівник проекту _____*Юрій ГОРДІЄНКО, професор, д.ф.-м.н., с.н.с* _____,
(ім'я, прізвище, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "11" _____травня_____ 2022 року N 1139-с

2. Термін подання студентом проекту _____

3. Вихідні дані до проекту технічна документація, теоретичні дані, інтернет-публікації за темою роботи _____

4. Зміст пояснювальної записки

*− Аналіз предметної галузі;*
*− Проектування системи;*
*− Програмна реалізація продукту;*
*− Тестування створеного програмного продукту;*

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)
*− Структурна схема проектованого програмного продукту типу чат-бот для Telegram;*
*− Функціональна схема: Діаграма діяльності процесу нечіткого виводу;*
*− Принципова схема: Блок-схема алгоритму чат-бота, що надає інформація про погодні умови;*

6. Консультанти розділів проекту

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| 1 | *професор, д.ф.-м.н., с.н.с Гордієнко Ю.Г.* | *10.03.2022* | *11.06.2022* |
| 2 | *професор, д.ф.-м.н., с.н.с Гордієнко Ю.Г..* | *10.03.2022* | *11.06.2022* |
| 3 | *професор, д.ф.-м.н., с.н.с Гордієнко Ю.Г.* | *10.03.2022* | *11.06.2022* |
| 4 | *професор, д.ф.-м.н., с.н.с Гордієнко Ю.Г.* | *10.03.2022* | *11.06.2022* |

7. Дата видачі завдання *30.08.2021*

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проекту | Строк виконання етапів проекту | Примітка |
|---|---|---|---|
| *1* | *Затвердження теми роботи* | *18.12.2021* | |
| *2* | *Вивчення та аналіз завдання* | *18.12.2021-20.03.2022* | |
| *3* | *Проектування системи* | *20.03.2022-29.03.2022* | |
| *4* | *Програмна реалізація продукту* | *29.03.2022-16.04.2022* | |
| *5* | *Оформлення пояснювальної записки* | *16.04.2022-10.06.2022* | |
| *6* | *Захист програмного продукту* | *24.04.2022* | |
| *7* | *Передзахист* | *11.06.2022* | |
| *8* | *Захист* | *20.06.2022* | |

**Студент** _____ Білич М.В.
(підпис) (прізвище та ініціали)

**Керівник проекту (роботи)** _____ Гордієнко Ю.Г.
(підпис) (прізвище та ініціали)

## АНОТАЦІЯ

Метою роботи є створення бота на широко відомій платформі Telegram, який буде генерувати автоматичні рекомендації, щодо процесу вибору одягу відповідно до погоди. Бот повинен використовуватися на основі метеорологічних даних.

Проаналізувавши суть задачі вироблення рекомендацій на основі метеорологічних даних, описавши існуючі продукти-аналоги та зробивши висновки про доцільність проведення власної розробки, було створено технічне завдання на розробку та відповідно до нього проведено ряд проектних рішень про характеристики системи надання рекомендацій на основі метеорологічних даних.

На основі розроблених рішень та дослідивши застосування дерева ухвалення рішень для системи рекомендацій гардеробу на основі прогнозу погоди, було проведено програмну реалізацію, виконано її тестування, оцінено ефективність, розроблено супутню документацію.

Практичне значення роботи полягає у створенні працездатного програмного продукту, за допомогою якого можна ефективно проводити вибір одягу, причому робити це за допомогою чат-боту на основі актуальних даних про погоду у найближчий час у даній місцевості.

## ANNOTATION

The aim of the work is to create a well known Telegram platform based bot, which will generate automatic recommendations on the process of choosing clothes according to the weather. The bot should be used based on meteorological data.

After analyzing the essence of the task of making recommendations based on meteorological data, describing existing products-analogues and drawing conclusions about the feasibility of their own development, a technical task was created for development and a number of design decisions was made on the meteorological data based recommendation giving system characteristics.

Based on the developed solutions and researching the application of the decision tree for the system of wardrobe recommendations based on the weather forecast, a software implementation was conducted, it's testing performed and efficiency evaluated. Supporting documentation has also been developed.

The practical significance of the work is to create a workable software product that can be used to effectively choose clothes, and do it with a chatbot based on current weather data in the near future in the area.

**Технічне завдання до дипломного проекту**
**ЗМІСТ**

## 1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання розповсюджується на розробку системи рекомендацій гардеробу на основі прогнозу погоди.

Область застосування: соціальна мережа Telegram.

## 2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки служить завдання на розробку системи рекомендацій гардеробу на основі прогнозу погоди, затвердженою кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

## 3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою даного проекту роботи є створення бота на широко відомій платформі Telegram, який буде генерувати автоматичні рекомендації щодо процесу вибору одягу відповідно до погоди.

## 4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами для розробки служать науково-технічна література з комп’ютерних технологій, публікації в періодичних виданнях, відео з відеохостингу Youtube, публікації в Інтернеті за даним питанням.

## 5. ТЕХНІЧНІ ВИМОГИ

### 5.1. Вимоги до розроблюваного продукту

1) проаналізувати суть задачі вироблення рекомендацій на основі метеорологічних даних, описати існуючі продукти-аналоги та зробити висновки про доцільність проведення власної розробки;

2) створити технічне завдання на розробку та відповідно до нього провести ряд проектних рішень про характеристики системи надання рекомендацій на основі метеорологічних даних;

3) на основі розроблених рішень провести програмну реалізацію, виконати її тестування, оцінити ефективність, розробити супутню документацію.

**5.2. Вимоги до програмного забезпечення:**

- Telegram API
- BotFather
- JavaScript
- Sublime Text Editor

# Пояснювальна записка
## до дипломного проекту
## на тему: «Система рекомендацій гардеробу на основі прогнозу погоди»

Київ – 2022 року

**ЗМІСТ**

# ВСТУП

Розвиток алгоритмів штучного інтелекту та відповідної апаратної бази, що може реалізувати ці алгоритми, зумовлює до все більшого запровадження інтелектуальних систем у різноманітні сфери життя суспільства. Втім «інтелектуальними» іноді іменують не тільки системи, що працюють за принципами ШІ, а й такі, що мають певні інформаційно-комунікаційні можливості, зокрема, підключення до мережі Інтернет. Сюди можна віднести численні інтелектуальні пристрої (як, наприклад, інтелектуальні сенсори), які не використовують методи та засоби ШІ, але вміють з'єднуватися через комп'ютерні мережі (Інтернет, або локальні) із певним центром управління та передавати йому поточну телеметрію, а також – отримувати у відповідь певні прості команди. Крім того, слід відмітити, що й самі засоби штучного інтелекту бувають надзвичайно різними за принципами свого функціонування та сильно відрізняються за складністю реалізації. З іншого боку, вказані інтелектуальні системи можуть виконувати надзвичайно різну й обширну функціональність, починаючи від простих рекомендацій фільмів для перегляду і до оцінки ефективності крупних економічних угод. Одним із таких прикладів, порівняно простих за предметною галуззю, але не дуже простих алгоритмічно, може бути система рекомендацій одягу людини на основі даних про прогноз погоди. Використання такої системи дозволяє економити час на читання прогнозу погоди, а також вирішення, який саме одяг максимально підходить під ту, чи іншу температуру, вологість, передбачені атмосферні явища і т.п. Крім того, зайнятим людям іноді принципово важко весь час займатися питанням, що слід одягати при певній поточній погоді (яскравими прикладами людей, які мали труднощі у виборі одягу були Альберт Ейнштейн, Стів Джобс, Марк Цукерберг, Барак Обома, тощо). Таким чином, задача розробки системи надання рекомендацій, заснованих на метеорологічних даних (тобто на прогнозі погоди), зокрема рекомендацій найбільш

підходящого одягу, є **актуальною** для фахівців в галузі сучасних інформаційних технологій.

**Метою** роботи є створення бота на широко відомій платформі Telegram, який буде генерувати автоматичні рекомендації щодо процесу вибору одягу відповідно до погоди. Бот повинен використовуватися на основі метеорологічних даних.

**Задачі** роботи наступні:

1) проаналізувати суть задачі вироблення рекомендацій на основі метеорологічних даних, описати існуючі продукти-аналоги та зробити висновки про доцільність проведення власної розробки;

2) створити технічне завдання на розробку та відповідно до нього провести ряд проектних рішень про характеристики системи надання рекомендацій на основі метеорологічних даних;

3) на основі розроблених рішень провести програмну реалізацію, виконати її тестування, оцінити ефективність, розробити супутню документацію.

**Об'єктом дослідження** є методи генерації автоматичних рекомендацій щодо вибору одягу згідно з метеорологічними даними.

**Предмет дослідження** — розробка чат-боту для генерації автоматичних рекомендацій гардеробу на основі прогнозу погоди.

**Методи дослідження**: методи галузі штучного інтелекту, а також методи та технології прикладного програмування.

**Практичне значення** роботи полягає у створенні працездатного програмного продукту, за допомогою якого можна ефективно проводити вибір одягу при виході із дому, причому робити це за допомогою чат-боту на основі актуальних даних про погоду у найближчий час у даній місцевості.

У **перспективі** дана робота може розширюватися в частині залучення більшої кількості альтернатив по елементам одягу, що підходять для певної погоди. Також можуть удосконалюватися алгоритми вибору, що застосовуються у розробленому програмному забезпеченні.

# РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

## 1.1 Аналіз галузей застосування, видів та основних особливостей сучасних інтелектуальних систем

Як було зазначено у вступі, інтелектуальні системи можуть застосовуватися для вирішення величезної кількості різноманітних завдань. У найбільшій мірі вони все ж таки орієнтуються на економічну галузь, хоча зустріти їх можна практично повсюдно. Наприклад, системи підтримки прийняття рішень у економічній сфері або іншими словами системи бізнес-аналітики безпосереднім чином надають рекомендації фахівцям, які продукти (наприклад, на ринках цінних паперів) слід покупати, які продавати, а з якими краще не проводити жодних операцій. Також існують системи рекомендацій по торгівлі валютами, матеріальними активами, конкретними видами товарів і т.п.

Значна частка роботи інтелектуальних систем присвячена галузі медицини. Тут у більшій мірі подібні системи називають експертними і вони допомагають виставляти правильний діагноз, вести профілактичні заходи та проводити елементи первинних оглядів, а в окремих випадках – і призначати лікування пацієнтам.[1]

Ще одна досить велика частка інтелектуальних систем займається управлінням технічними системами. Часто робота технологічного пристрою є досить складною та потребує зведення потужної математичної моделі, що не завжди можливо. При цьому інтелектуальні методи управління (засновані на досвіді експертів у відповідній галузі) надають досить гарні результати без використання точних (детермінованих) математичних моделей.

Слід сказати, що певна кількість інтелектуальних систем призначається для виконання невеликих задач автоматизації, що лежать на грані із глибоко побутовою або навіть розважальною галуззю. Наприклад, отримали досить широкого поширення системи рекомендацій кінофільмів для онлайн

перегляду, пісень, текстових творів та ін. Сюди ж можна віднести і ті продукти, що забезпечують функціональність, яка описується в даній роботі, а саме – формування рекомендацій по одягу, який добре підходить до погоди, прогнозованої у короткочасній перспективі.

Слід сказати, що інтелектуальні системи можуть засновуватися на різних варіантах математичного апарату ШІ та машинного навчання. Найбільш популярним на сьогодні підходом є використання нейронних мереж, що можуть мати різну структуру та принципи роботи, але в цілому аналогічні біологічним підсистемам, які є основою нервової системи усіх тварин і людини зокрема. Також, досить широкі можливості мають підходи на основі нечіткої логіки, еволюційних алгоритмів, деревом ухвалення рішень, тощо. Вибір конкретного методу, який буде покладено в основу проектованої системи, буде проводитися пізніше у наступному розділі.

Загальною рисою усіх дійсно інтелектуальних систем (а не таких, що називаються інтелектуальними через наявність можливості підключення до мережі Інтернет) є те, що вони засновані на використанні досвіду людини у вирішенні відповідних задач. Для нейронних мереж такою інформацією є дата-сет для навчання, для систем нечіткої логіки – це сукупність правил нечітких продукцій, для дерев ухвалення рішень – це сукупності значень аргументу, при яких спостерігаються стрибки функції та конкретні її значення з обох боків кожного стрибка, тощо. Без використання знань людини побудувати інтелектуальну систему, що надаватиме певні рекомендації, не уявляється можливим.

Також слід відмітити, що з технічної точки зору інтелектуальні системи можуть будуватися досить різними способами. Часто це може бути повноцінний настільний додаток, або загальнодоступний (мається на увазі з різних куточків світу, але в рамках політики безпеки та розмежування доступу) веб-додаток.[2] Останнім часом такі системи реалізовуються як мобільні додатки, або взагалі чат-боти різноманітних месенджерів. Саме такі

програмні продукти, що допомагають обирати одяг на основі погоди, розглянемо докладніше.

### 1.2 Пошук існуючих систем рекомендацій одягу на основі даних про погоду та суміжних рішень

Перед стартом будь-якої розробки слід розглянути існуючі на ринку або у вільному доступі програмні продукти, що виконують схожі функції, тобто програмні продукти-аналоги, призначені для відстежування метеорологічних умов та надання рекомендацій одягу на основі них.

В першу чергу все ж таки можна говорити про сервіси, за допомогою яких можна дізнаватися хоча би про очікувану погоду, а уже потім власноруч робити висновки про доцільний при цій погоді одяг. Прогноз погоди можна дізнаватися за допомогою відповідних сайтів. Виявити найбільш популярні із них можна за допомогою звичайного пошукового запиту в Google – рис. 1.1.

Очевидно, що для користування будь-якими сайтами потрібний доступ в Інтернет, але для користування месенджерами мобільні оператори у багатьох тарифних планах надають повністю вільний необмежений доступ. Крім того, інтерфейс сайтів може здаватися перевантаженим, причому не тільки зайвими опціями, а й, головне, рекламою, чого немає у Telegram, Також використання боту саме у часто вживаному месенджері дозволяє спростити процес отримання потрібної інформації про погоду для пересічного користувача (який не має високої кваліфікації у сфері ІТ).[3] Отже, розглянемо також інший варіант існуючих аналогічних програмних продуктів – чат-ботів для визначення погоди та надання рекомендацій одягу на цій основі.

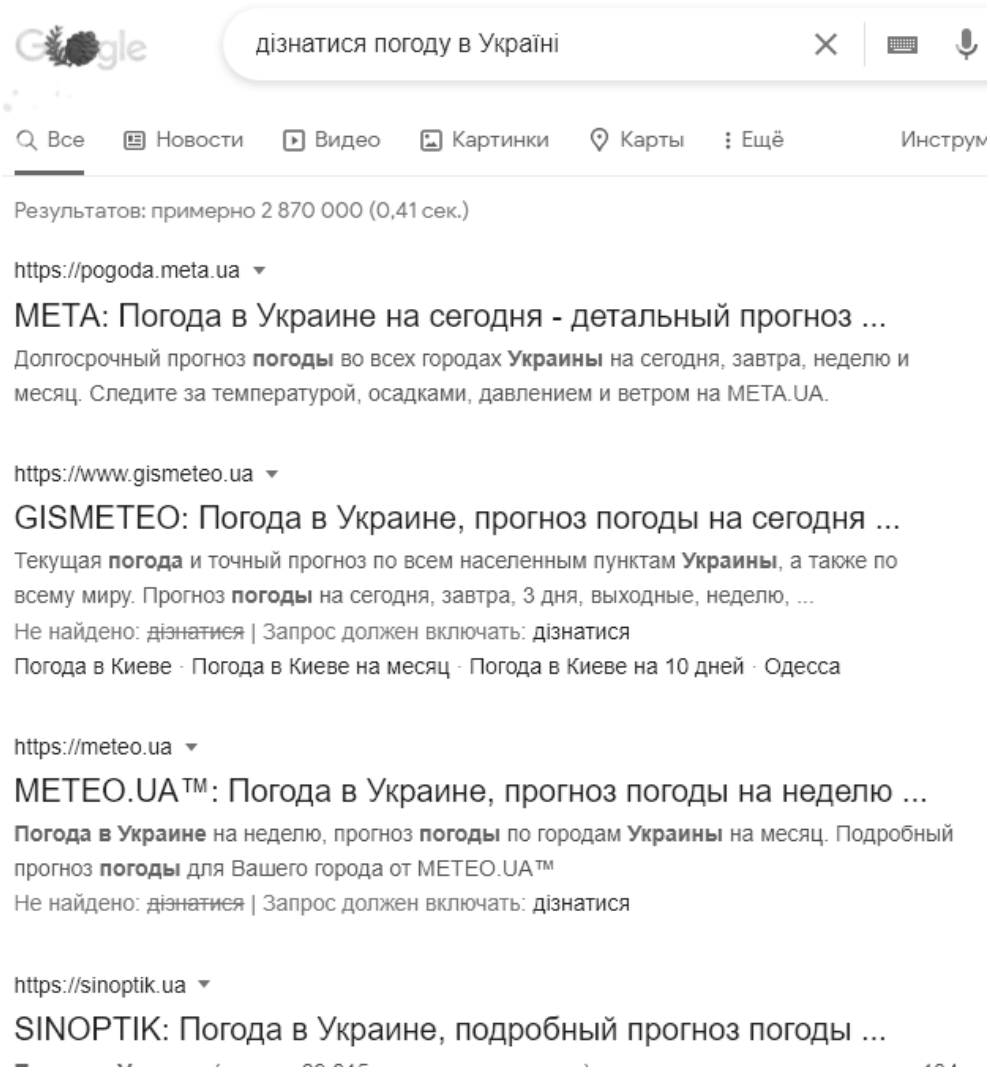



Рис. 1.1 Найбільш популярні сайти для відстежування погоди в Україні

В першу чергу, можна розглянути чат-бот «Погодник». Явною його перевагою є можливість вибору мови спілкування (в т.ч. української). Особливістю цього продукту є досить докладна інформація, що видається користувачеві боту про погоду – рис. 1.2.

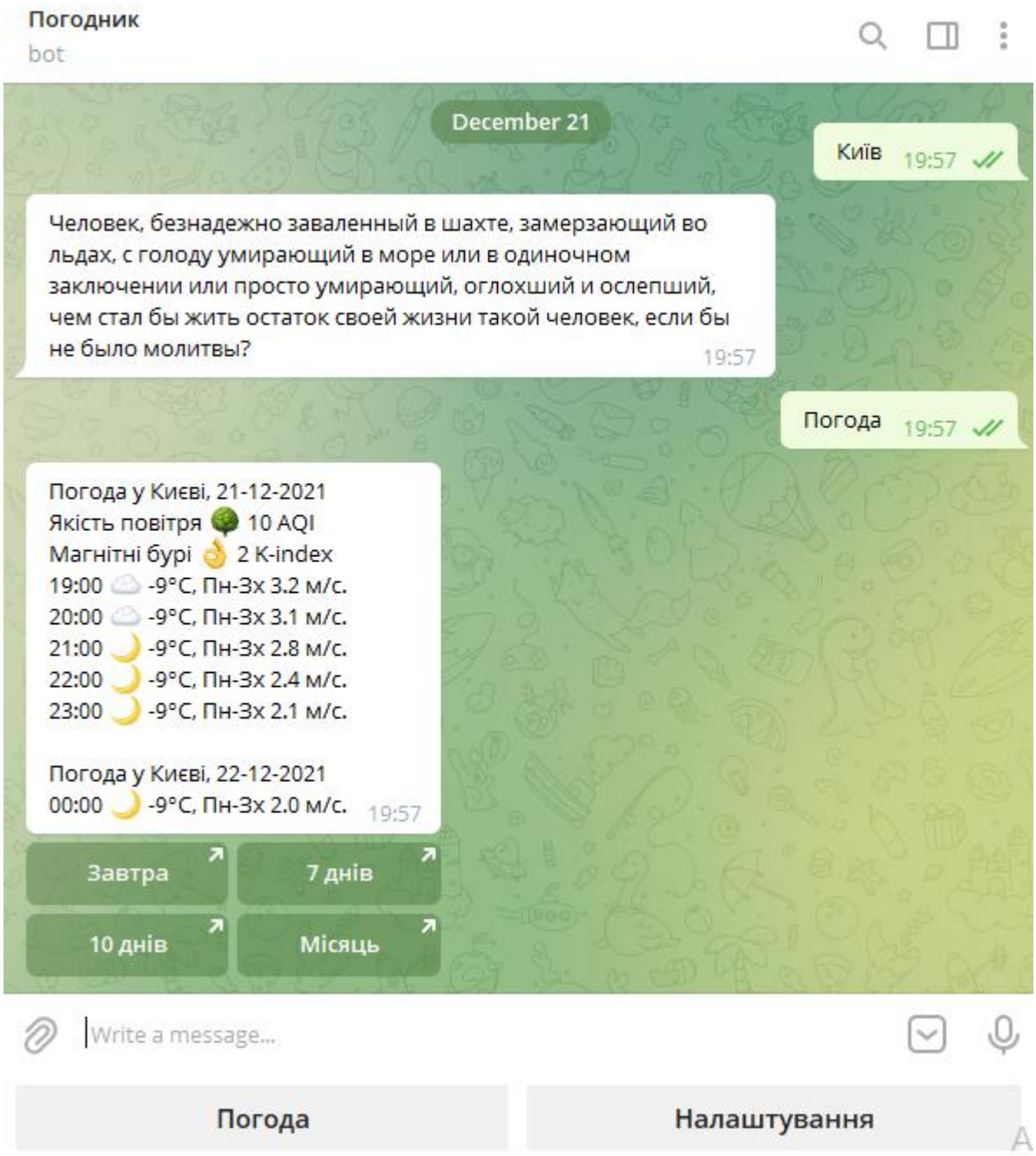



Рис. 1.2 Інтерфейс чат-боту «Погодник» для Telegram

Втім, логіка роботи боту є дещо надмірною, адже у відповідь на перший запит із назвою міста «Київ» бот видає досить велике філософське повідомлення, яке збиває з пантелику користувача і вимагає значних зусиль для того, щоби зрозуміти, що він все ж правильно потрапив саме у бот для відстежування погодних умов і, крім того, що робити далі. Насправді, цілком очевидно, що при вході у погодний бот і надсиланні назви відомого міста

користувач хоче дізнатися погоду у даному місті, а не отримувати обширні філософські тексти.

Ще одним схожим рішенням є бот MeteoBot – рис. 1.3.

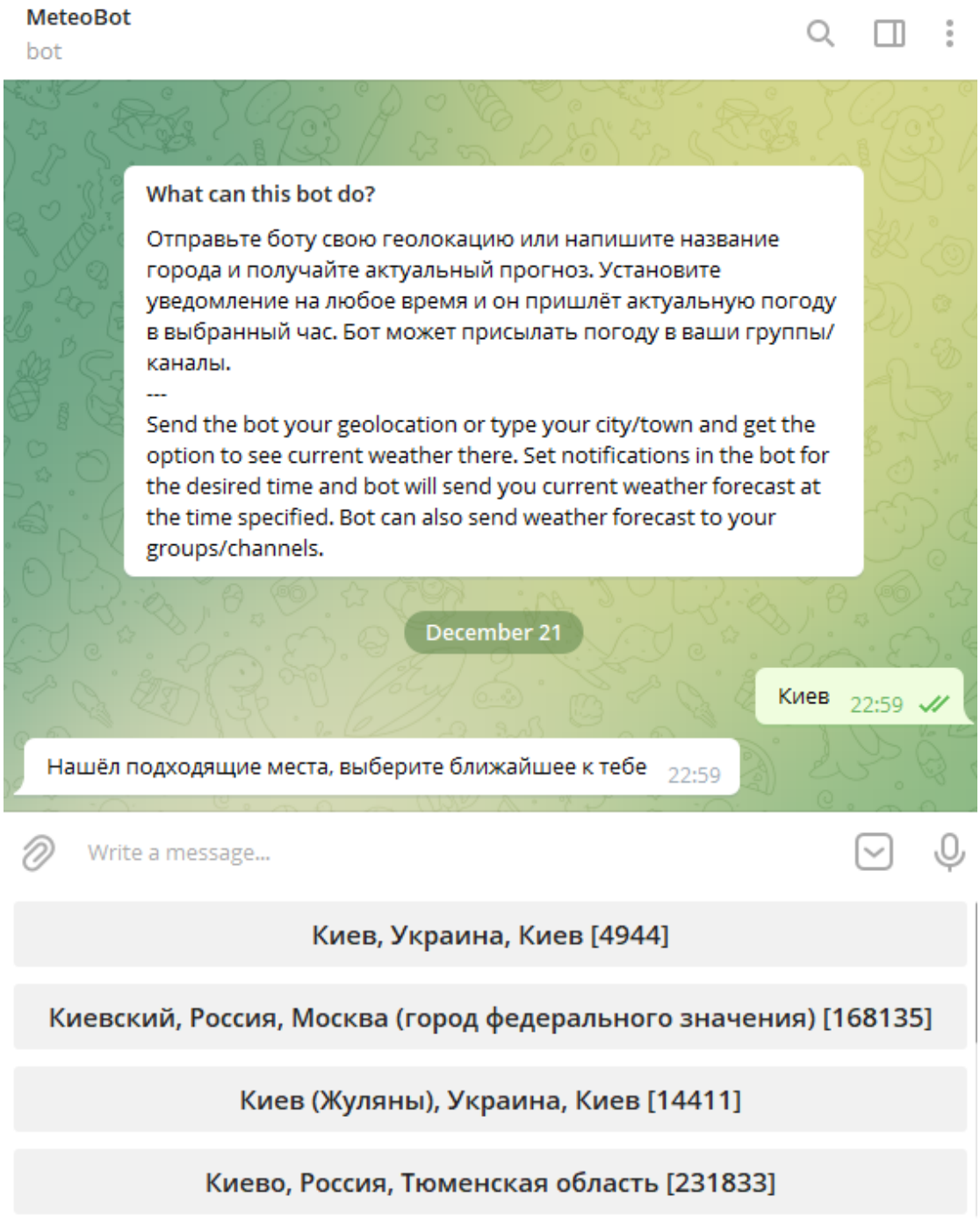



Рис. 1.3 Інтерфейс чат-боту «Meteobot» для Telegram

Meteobot має наступні недоліки:

- не має можливості вільного вибору мов, зокрема, української;

- має недоробки у тексті власних повідомлень, наприклад, на першому ж екрані боту, що показаний на рис. 1.3, написано «выберите ближайшее к тебе»;

- не ефективно працює з географічними місцями, зокрема, на конкретний ввід відомого міста «Київ» пропонує обирати одну з численних (до того ж не зовсім зрозумілих – рис. 1.3) альтернатив, хоча будь-яка людина при такому запиті однозначно зрозуміла би, про яке місто йде мова.

Розглянуті рішення по визначенню погоди у Telegram не є єдиними, однак мають типові недоліки, що притаманні і іншим рішенням даного класу.

Нарешті, найбільш повно відповідає тематиці даної роботи чат-бот «Прогноз одежды», який на основі визначення метеорологічних умов одразу рекомендує, який одяг доцільно одягти у заданому місці – рис. 1.4.[4]

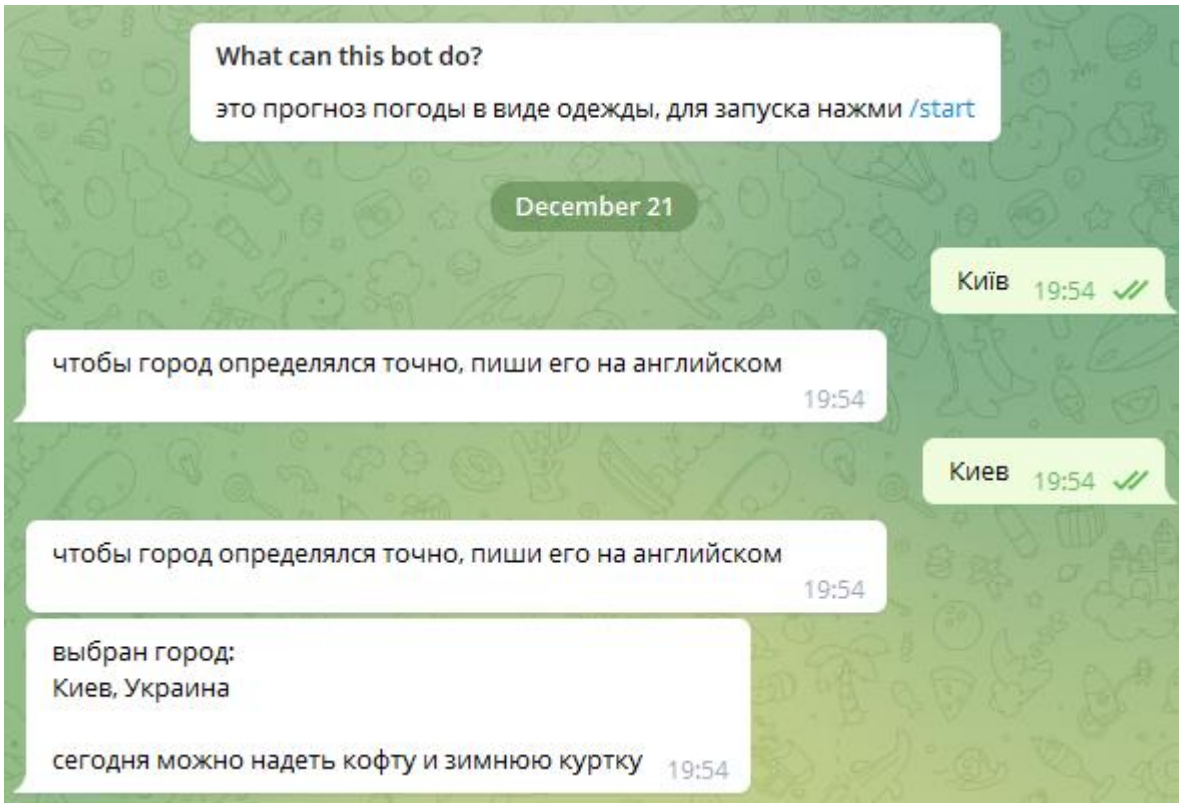



Рис. 1.4 Інтерфейс чат-боту «Прогноз одежды» для Telegram

Дане рішення має ряд недоліків:

- неможливість визначення самих метеорологічних умов, а лише відповідного до них одягу (все ж більш ефективно надавати користувачеві

обидві інформації: в першу чергу – рекомендацію одягу, але і по-друге - дані про погоду, на яких заснована надана рекомендація);

- сама рекомендація одягу є занадто скудною, зокрема немає ніякої інформації про взуття, головний убір, рукавички, тощо (бот пропонує всього лише дві речі, чого явно замало для формування повноцінного наряду для виходу на вулицю);

- відсутність вибору мови.

Також слід відмітити, що деякі продукти, які на перший погляд видаються підходящими аналогами рішення з рекомендацій одягу на основі погодних даних, на ділі не надають потрібної функціональності – рис. 1.5.

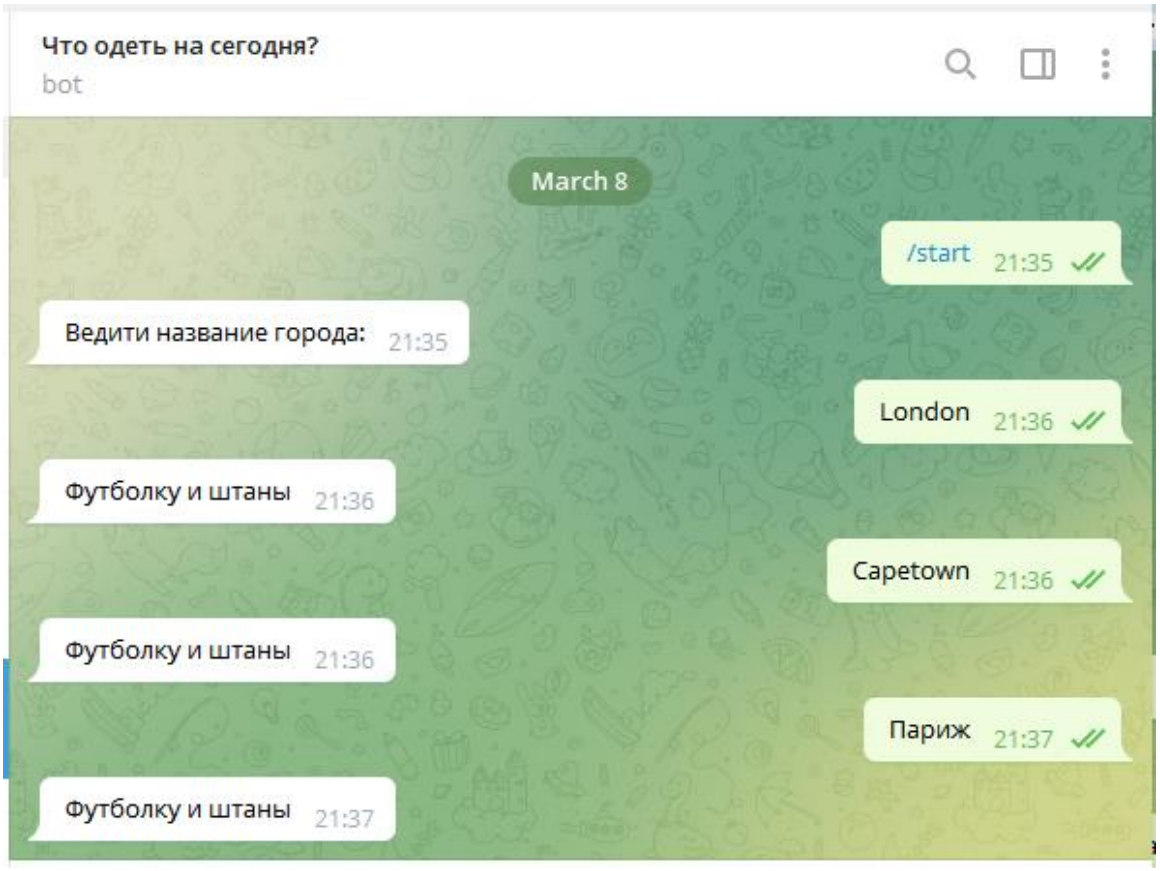



Рис. 1.5 Приклад чат-боту, що номінально відповідає потрібній функціональності, але в реальності не забезпечує її

## 1.3 Обґрунтування необхідності проведення власної розробки та деталізація вимог до неї

У попередньому підрозділі вказано, що існує певний набір програмних продуктів, у яких заявлена функціональність в цілому відповідає тематиці даної роботи, однак, реально вони її не забезпечують. Можливо такий стан справ пов'язаний із тим, що ці продукти мають тимчасові технічні труднощі, можливо є учбовими (незавершеними) рішеннями, тощо. При цьому визначити, наскільки ефективно працює програмний продукт, до моменту початку його експлуатації не можливо. Проведений же аналіз численних рішень для рекомендацій відповідного погоді одягу (зокрема, доступних у Телеграм), показав, що згаданий вище «Прогноз одежды» є найкращим аналогом (недоліки якого є суттєвими і розглянуті вище), тому доцільно провести власну реалізацію відповідного програмного продукту. Орієнтуватися слід на використання мобільних пристроїв типу смартфонів, як найбільш частих супутників сучасної прогресивної людини.[5]

Багато різноманітних компаній, котрі розвертають бізнес через Інтернет, користуються можливостями ботів з кількох причин:

- *бот позволяє використовувати наступний канал комунікації з цільовою аудиторією;*
- *бот швидко реалізовує монотонну роботу, що дозволяє розвантажити співробітників та заощаджувати гроші компанії;*
- *бот ніколи не спить і не потребує відпочинку та відпусток;*
- *бот ніколи не збивається, якщо алгоритм його роботи розроблений з урахуванням усіх нюансів компанії.*

Для створення ботів для Telegram використовуються різні мови програмування. Найпоширенішими є наступні:

- *Python;*
- *PHP;*

- *JS (javascript + node.js).*

Інструменти Python допускають швидко розробити будь-якого бота, але для запуску вам необхідно підібрати хостинг, що підтримує Python.

Розробка в php відбувається не так швидко, як на python, але швидкість самого бота вища. Оскільки майже всі хостингі мають підтримку php, отже для установки підійде зовсім будь-який.

Розробка на js трохи складніша, ніж на php, проте швидкість до 100 разів вище. Але установка вимагає VDS, що не завжди можливо для невеликих проектів.

Розглядаючи це, ми робимо висновок, що мова Python більш за все підходить для рішення цієї проблеми.

Взаємодія між користувачем і ботом розвертається в наступному:

- *користувач бота дає йому команду;*
- *бот надсилає команду своєму серверу;*
- *програма на сервері обробляє запит, якій отриманий від бота;*
- *сервер відповідає боту;*
- *бот відображає відповідь у вікні керування користувача.*

І цей цикл повторяється щоразу, коли користувач тисне кнопки взаємодії в телеграм-боті.

Перед створенням бота в телеграмі, необхідно сформулювати мету яку виконуватиме програма та вигадати назву для бота.

Після цього слід почати роботу з ботом @BotFather, який призначений для створення нових ботів - написати команду «/start».

Щоб створити новий бот, необхідно натиснути на команду (або написати в командному рядку) «/ newbot».

Після цього BotFather запропонує написати ім'я бота - це може бути будь-яке неунікальне ім'я (кирилиця, латиниця і цифри).

Потім вам потрібно буде ввести унікальне ім’я бота (цифри, латиниця, символи «_»), і відразу бот вивірить, чи ім’я вільне. І якщо так, створюється новий бот. Ця назва має закінчуватися на «бот».

У відповідь BotFather відправить власнику бота унікальний ключ – токен, котрий необхідно зберегти і нікому його не показувати.

Після виконання цих дій буде створено новий бот із зазначеним ім'ям. Ви можете встановити для нього аватар і додати опис.

Потім необхідно навчити його виконувати ці функції. Для цього напишіть код на мові програмування, розмістить його на сервері і зв’яжіть файл на сервері з телеграм-ботом, щоб запити, які надіслано в месенджері користувачем бота, оброблялися розробленою програмою.

Таким чином, розробці підлягає програмний продукт для надання рекомендацій щодо підходящого одягу, який відповідає актуальній метеорологічній інформації.

Вид продукту – чат-бот для месенджера Telegram.

Тип архітектури продукту – клієнт-серверна архітектура.

Цільова платформа сервера – рішення Node.JS на базі веб-інтерпретатора мови програмування JavaScript.

Цільова платформа клієнта – будь-яка актуальна версія месенджеру Telegram (настільна або мобільна).

Мови програмування та засоби розробки – сучасні, достатньо популярні, рішення на базі яких можуть в подальшому підтримуватися широким колом програмістів (без необхідності залучення «вузьких» спеціалістів). Передбачається використання мови JavaScript.

Протокол взаємодії – HTTP.

Методи вироблення кінцевого рішення (рекомендації) – на основі одного з підходів галузі штучного інтелекту чи машинного навчання.

Керуючись даними вимогами, можна переходити до наступного етапу процесу створення (проектування) програмного продукту.

## ВИСНОВОК ДО РОЗДІЛУ 1

У даному розділі було проведено аналіз галузей застосування, видів та основних особливостей сучасних інтелектуальних систем та пошук існуючих систем рекомендацій одягу на основі даних про погоду та суміжних рішень. Після чого було обґрунтовано необхідність проведення власної розробки та деталізація вимог до неї.

# РОЗДІЛ 2 ПРОЕКТУВАННЯ СИСТЕМИ

## 2.1 Вибір архітектури системи, що проектується

Метою даної роботи є створення умов для ефективної рекомендації одягу для користувача програмного продукту відповідно до погодних умов у місцевості, де він знаходиться. Звичайно, можуть існувати різні варіанти типів додатків, що виконуватимуть подібні функції. У найпростішому (елементарному) варіанті можливим є створення статичного веб-сайту із прогнозом погоди та підходящим одягом, який оновлюватиметься вручну (наприклад, кожного ранку) уповноваженою людиною оператором. Очевидно, такий підхід є вкрай неефективним, оскільки потребує величезної кількості ручної праці і майже зовсім не використовує переваг сучасних розвинутих інформаційних технологій. Більш ефективним тут є створення веб-сайту, який би мав власну базу даних та комунікацію із одним або кількома професійними серверами організацій, що публікують прогнози погоди (метеорологічних служб). Такий програмний продукт підпадає під категорію веб-додатків і, взагалі кажучи, є досить ефективним, але потребує від користувача активних дій – в першу чергу, відкриття даного сайту у браузері. Було б значно зручніше, якби необхідна інформація уже знаходилася би у пристрої користувача автоматично, а користувач міг би миттєво отримати її без вчинення додаткових дій при виникненні такої необхідності. [6]

Вказану можливість забезпечує, наприклад, мобільний додаток, однак суттєвим недоліком такого рішення є те, що його слід спеціальним чином встановлювати у смартфон, він споживатиме ресурси смартфону, постійно знаходячись у пам’яті (а якщо він не буде працювати резидентно, то, відповідно, матиме ті ж недоліки, що і веб-додаток).

Ще однією можливістю є запровадження чат-боту Telegram, який має наступні переваги:

- інформація надходить до пристрою користувача в автоматичному режимі, без прикладання додаткових зусиль з боку користувача, і миттєво надається йому при виникненні такої потреби;

- не потребує встановлення додаткового програмного забезпечення, що споживає апаратні ресурси клієнтського пристрою;

- використовує функціональність надзвичайно поширеного програмного продукту – месенджеру Telegram.

Суттєві недоліки у такого рішення практично відсутні, через що його і було прийнято у якості основного у даній роботі.

Отже, згідно обґрунтованого вище вибору платформи Telegram (а також згідно завданню на розробку), у даній роботі створюється програмний продукт типу чат-бот для цього месенджера. Архітектура таких програмних продуктів є фіксованою і на логічному рівні відповідає концепції «клієнт-сервер», оскільки команди, які вводить користувач у власному клієнті Telegram передаються на серверну частину (бек-енд), яка має знаходитися на комп'ютері-сервері, що має постійне підключення до мережі Інтернет. Там ці команди обробляються і на клієнта пересилається відповідь, що відображується користувачеві. Число клієнтів, з якими спілкується сервер (тобто, власне програма чат-бот), обмежується лише апаратною продуктивністю комп'ютера-сервера і може складати тисячі користувачів одночасно і більше. Таким чином, на логічному рівні маємо повну відповідність архітектурі «клієнт-сервер».[7]

Слід сказати, що фізично усі повідомлення, що передаються у системі Telegram від кожного абонента (людини чи бота), спочатку надсилаються на анонімні сервери самої програми Telegram, які забезпечують процеси шифрування повідомлень і взагалі керують усім процесом спілкування клієнтів месенджеру між собою (в тому числі – і користувачів з ботами). Проходження повідомлень від бота до клієнта і назад через сервери Telegram відбуваються у повністю прозорому режимі, тобто відображувати цей процес якимось чином у програмному коді бота не потрібно.

Слід відмітити, що повнофункціональний чат-бот запам’ятовує користувачів, які хоча б раз до нього звернулися, для чого слід записувати ідентифікатор відповідного чату з ними до бази даних. Відповідно серверний код, який обробляє запити клієнтської частини, часто працює у зв’язці з сервером баз даних.

Отже, архітектура проектованого чат-боту буде такою, як показано на рис. 2.1.

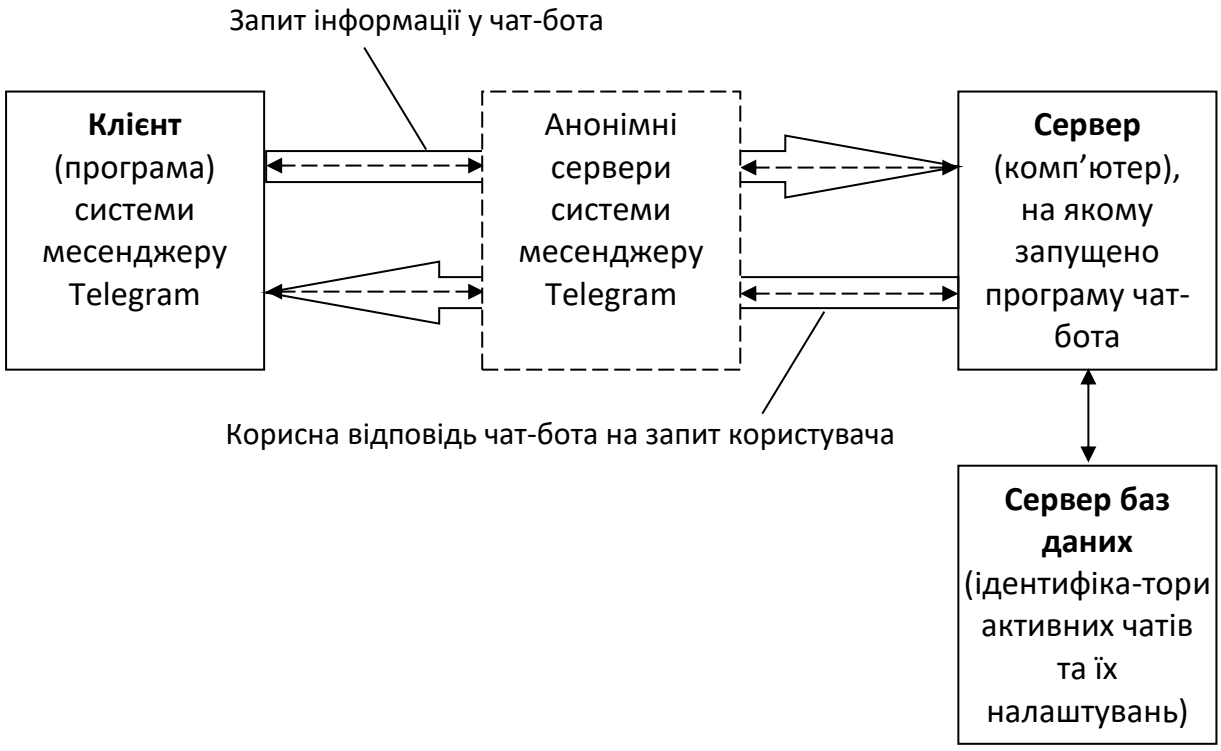



Рис. 2.1 Архітектура проектованого програмного продукту типу чат-бот для Telegram

Таким чином, архітектура проектованого програмного продукту є визначеною, тому можна переходити до вибору засобів розробки та подальшої програмної реалізації.

## 2.2 Аналіз існуючих методів інтелектуального аналізу даних та вибір базового з них для даної роботи

В першу чергу, на сьогоднішній день значна частка методів інтелектуального аналізу даних спирається на технологію Big Data, особливості якої розглянемо докладніше (інша ж частина методів ІАД використовує досвід експертів, що принципово кажучи, також є аналогом Big Data, але пропущеними через інтелект людини). Отже, поняття Big Data початково характеризувалося трьома основними рисами (3 “V”):

- Volume – великі загальні обсяги даних у байтах та похідних одиницях (наприклад, диски середнього ПК призначені для 1 Тб, тому, наприклад, величини порядку кількох терабайт підпадають від категорію Big Data);

- Velocity – велика швидкість надходження даних (так, швидкість запису інформації на середній SSD носій складає не менше 0,5 Гб/с, тому, наприклад, при швидкості генерації нової інформації в обсязі 10 Гб/с, вони точно відносяться до Big Data;

- Variety – різнорідність даних, їх багатство різними випадками, відсутність повторюваності (наприклад, просте повторення однієї і тієї ж бітової комбінації трильйон разів не є великими даними).[8]

Остання властивість взагалі кажучи є очевидною, оскільки в протилежному випадку сучасні засоби кодування (архівування, оптимізації) стиснуть такий потік із одними і тими ж повторюваними даними до малих розмірів і, відповідно, малої швидкості передачі (генерації або обробки).

В дещо іншій мірі властивість різнорідності проявляється якраз при використанні великих даних для Data Mining. Так називають процеси видобутку нових даних про властивості об’єктів та процесів навколишнього світу виключно на основі аналізу великих масивів даних, що їх описують з того, чи іншого боку, тобто на різних рівнях абстракції (одним із частинних випадків є отримання інформації про поведінку людини в мережі Інтернет). Щоби отримати якісь нові залежності досліджувати слід відомості, створені

значною кількістю схожих об’єктів. Наприклад, описи покупок в Інтернет-магазині не можна оцінювати за статистикою дій у ньому одного користувача, навіть якщо він там знаходиться цілодобово і робить дуже багато покупок.

При Data Mining явище, об’єкт чи система описуються феноменологічно, тобто без використання відомостей про їх внутрішню будову та суть процесів, які там протікають. Однією із значних переваг Data Mining якраз і є те, що не вдаючись глибоко у суть предмета, на основі великих описових даних про нього можна скласти певні залежності між його параметрами (величинами, що його характеризують).

Для аналізу Big Data використовують методи аналізу даних, описані у попередньому підрозділі (статистичні, з галузі штучного інтелекту та ін.).

Самостійним класом таких систем, які, в принципі, можуть застосовуватися для задач аналізу даних в цілому та Big Data зокрема, є продукти на базі нейронних мереж, яким присвячена значна кількість наукових робіт останніх десятиліть, наприклад [3]. Така ситуація у великій мірі обумовлена певним «бумом», яка спостерігається в галузі штучного інтелекту та суміжних напрямках, навколо поняття нейронної мережі. Цей об’єкт масово застосовується для задач розпізнавання, класифікації, апроксимації та прогнозування.[9]

Штучною нейронною мережею (далі – ШНМ), як загальновідомо, називають систему, котра, приймаючи на вхід вектор вхідних величин, за складними (нелінійними) внутрішніми законами робить вектор вихідних величин, що є більш-менш задовільним розв’язком поставленої задачі.

Елементарною складовою ШНМ є один нейрон – рис. 2.2. Це об’єкт, який має певну кількість входів (на рисунку – *n* входів), що називають дендритами та один вихід, який називається аксоном.

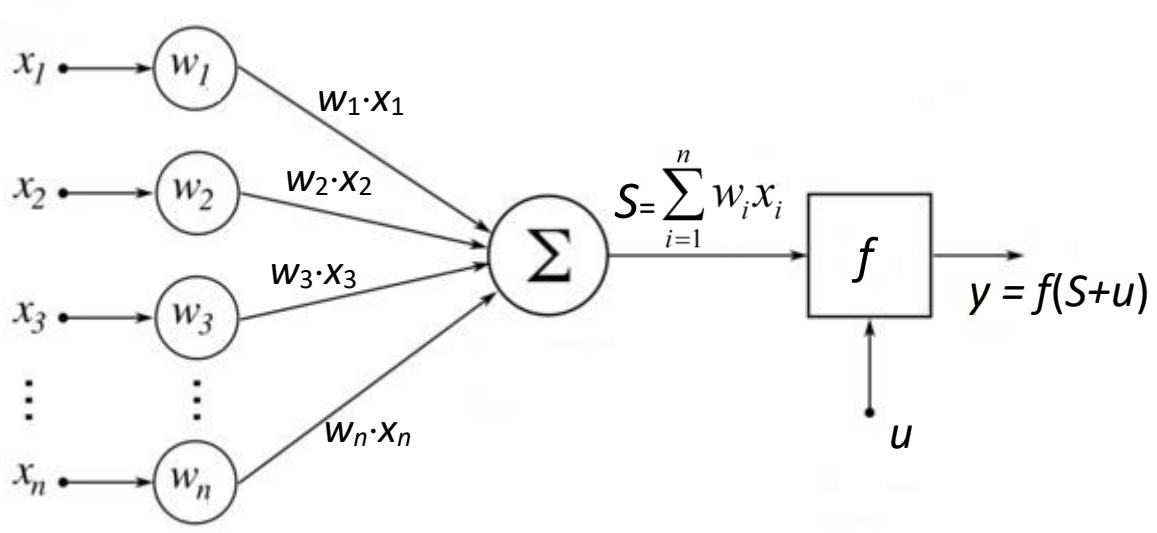



Рис. 2.2 Структура штучного нейрону

На входи нейрону поступають вхідні сигнали $x_1, x_2, \ldots, x_n$, що є виходами попередніх нейронів, розташованих раніше по ходу проходження сигналу. Кожен вхідний сигнал $x_i$ помножується на певне число $w_i$, що іменують вагою (або синоптичною вагою) відповідного входу. Усі таким чином зважені вхідні сигнали подаються на суматор, який виробляє зважену суму $S$ виду:

$$S = \sum_{i=1}^{n} w_i x_i \, .$$

(2.1)

До цієї зваженої суми вхідних сигналів може додаватися зсув або зміщення $u$ (у поширеному частинному випадку зміщення відсутнє, тобто $u = 0$) і від цієї суми виробляється певна функція $f$, що іменується активаційною; так і формується вихідний сигнал нейрону $y$, що переходить далі по нейронній мережі:

$$y = f(S + u). \qquad (2.2)$$

Відповідно комбінуючи (2.1) та (2.2) маємо функцію, що виконує штучний нейрон[10]:

$$y = f\left(\sum_{i=1}^{n} w_i x_i + u\right). \qquad (2.3)$$

При розгляді такого способу завдання функції, який здійснює нейрон, бачимо, що важливу роль для кожного нейрона виконають три речі (особливо перші дві):

- набір конкретних значень ваг $w_i$, які описують кожен вхід нейрона;

- вид активаційної функції $f$ даного нейрона;

- присутність зсуву або порогу активації $u$ (котрий може бути як додатним, так і від'ємним).

Зважаючи на те, що проста ШНМ складається як мінімум із кількох нейронів (а частіше – з десятків, чи навіть сотень нейронів), а в кожного з них є багато входів, то задача вибору вагових коефіцієнтів $w_i$ стає дуже серйозною за своєю складністю. Особисте, тривалий процес завдання конкретних «правильних» значень цих вагових коефіцієнтів іменують процесом навчання нейронної мережі. Бувають різні алгоритми навчання нейронних мереж, які укрупнено можна поділити на дві групи: «з учителем» та «без учителя». В цілому, навчання нейронних мереж є настільки складним процесом, що має досліджуватися в окремому великому блоці інформації і не може бути розглянуте в рамках даної роботи.[10,11]

Вид активаційної функції $f$, очевидно, ще є дуже важливим моментом для функціонування нейронів, хоча й у значно меншій мірі, ніж вибір синаптичних вагових коефіцієнтів. На рис. 2.3 наведено найбільш розповсюджені види функцій активації, що найчастіше використовуються на практиці при роботі з нейронними мережами. З точки зору обчислювальної складності більш кращим є вживання порогових функцій, що фактично являють собою константну залежність (бо хоча б лінійних, проте не сигмоїдних).

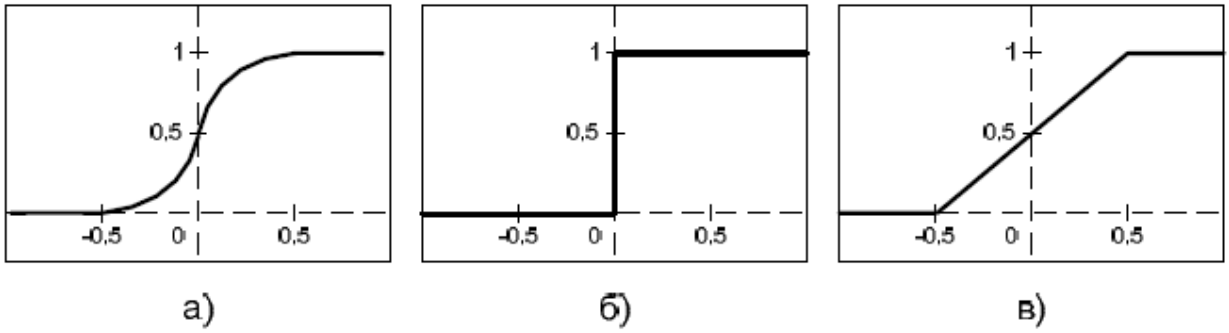



Рис. 2.3 Найбільш розповсюджені типи функцій активації:
*а* – сигмоїдна, *б* – порогова, *в* - лінійна

При побудові цілої нейронної мережі, крім характеристик окремих нейронів, важним стає також спосіб їх з'єднання (так само, як і їх загальна кількість, організація нейронів у групи, шари, і т.д.).

Найпростішим і в той же час розповсюдженим способом з'єднання нейронів у конкретну ШНМ є персептрон (перцептрон), що зумовлено великим спектром задач, котрі можна розв'язувати за допомогою ШНМ з такою внутрішньою структурою.[12]

Слід відзначити, що нейронні мережі хоча і є найбільш поширеним інструментом для вирішення задач з галузі штучного інтелекту, найкраще всього застосовувати їх для задач апроксимації та прогнозування, а не вироблення певних висновків. Для таких цілей краще використовувати інші методи, наприклад, теорії нечітких множин.

Через велику популярність доцільно розглянути також і такий метод аналізу даних, як апарат теорії нечітких множин. Таким чином, одним із дуже поширених методів отримання кінцевого рішення в задачі вибору є вживання апарату нечіткої логіки (fuzzy logic), що є ще одним багатонадійним механізмом, котрий широко застосовується в галузі інтелектуального аналізу даних [4]. Перевагою таких систем є перспектива роботи із нечисловими, розмитими, або загалом якісними, вираженими словами, лінгвістичними даними. Процедура фазифікації вхідної інформації, де використовуються

лінгвістичні змінні допускає поставити у відповідність елементам множини параметрів наступного стану об’єкта їх текстові описи, та опісля застосувати простий та зрозумілий набір правил нечітких продукцій для вироблення кінцевого рішення у нечіткій, розмитій формі. Після застосування процедури дефазифікації одержуємо конкретне значення привабності того чи іншого постачальника, на основі якого і виконуємо кінцевий вибір. Основною перевагою при такому підході є брак необхідності зведення складної математичної моделі (можливо на основі методів багатовимірного регресійного аналізу, інших статистичних методів), а уживання простої бази правил, яку може створити на основі аналізу логіки роботи системи (або ходу протікання процесу) будь-яка тверезо мисляча людина.

Зважаючи на велику поширеність систем нечіткого виводу, дослідимо докладніше весь процес їх вживання (зокрема, для задач вибору, як у даній роботі). Він полягає у циклічному виконуванні процедури нечіткого виводу (див. нижче) за допомогою системи нечіткого виводу. Єдиним кроком, що не здійснюється під час робочого часу системи, а виконується лише один раз – при розробці системи – є створення системи нечітких правил продукцій (бази правил нечітких продукцій). [13]

Під системою нечітких правил продукцій мається на увазі узгоджена множина окремих правил нечітких продукцій, побудованих на основі виразів виду:

ПРАВИЛО <1>: ЯКЩО «$\beta_1$ є $\alpha_{11}$» ТО «$\beta_2$ є $\alpha_{21}$», (2.4)

де «є» означає «являється», або англійською «IS».

Тут нечіткі вирази антецеденту *A* та консеквентну *B* розписані у формі «$\beta_i$ є $\alpha_{ij}$» через лінгвістичні змінні $\beta_i$, деякі з яких є вхідними (це змінні, що беруть участь у антецедентах), деякі – вихідними (змінні, які входять до консеквентів). $\alpha_{ij}$ – який-небудь конкретний терм із терм-множини лінгвістичної змінної $\beta_i$.

Отже, початковим кроком процедури нечіткого виводу, який до того ж здійснюється лише один раз, є створення бази правил нечітких продукцій. Це

задача, яку повинен виконувати експерт (експерти) у цій предметній галузі, оскільки від якості цієї системи (її повноти та адекватності кожного правила) залежить вдала робота системи підтримки прийняття рішень у майбутньому. Помилки при проєктуванні бази правил загалом кажучи можуть приводити до катастрофічних наслідків, притому найбільша частка складних проблем, пов’язаних з fuzzy системами, має за причину саме недоліки системи нечітких правил продукцій.[13,14]

Визначною властивістю бази правил має бути її узгодженість, тобто правила не зобов'язані суперечити одне одному (так завжди і є, коли розробку виконує один експерт, котрий має закінчений, усталений і ефективний погляд на рішення задач заданого класу). При розробленні бази правил командою експертів, перевірка на узгодженість має реалізуватися окремим підетапом.

Після зведення бази правил нечітких продукцій та її виконання у певному електронному апаратному забезпеченні починається процес нормального функціонування системи вибору, що є ітеративним, циклічним і кожна ітерація складається з наступних складових (рис. 2.4):

- фазифікація вхідних величин, котрі входять до умов правил нечітких продукцій;

- агрегування підумов для правил, що мають складені антецеденти;

- активізація підвисновків;

- акумулювання висновків (для тих вихідних змінних, різні терми яких входять до різних активних правил нечітких продукцій, а зазвичай для кожної вихідної змінної так і є);

- дефазифікація вихідних змінних.[15]

В загалому, можна відзначати, що нечітка логіка є дійовим інструментом для вирішення слабко формалізованих задач, зокрема у галузі інтелектуального аналізу даних вона є технічно більш простим засобом, проте не менш результативним з точки зору достовірності результатів, бо отримуються з використанням апарату нейронних мереж.

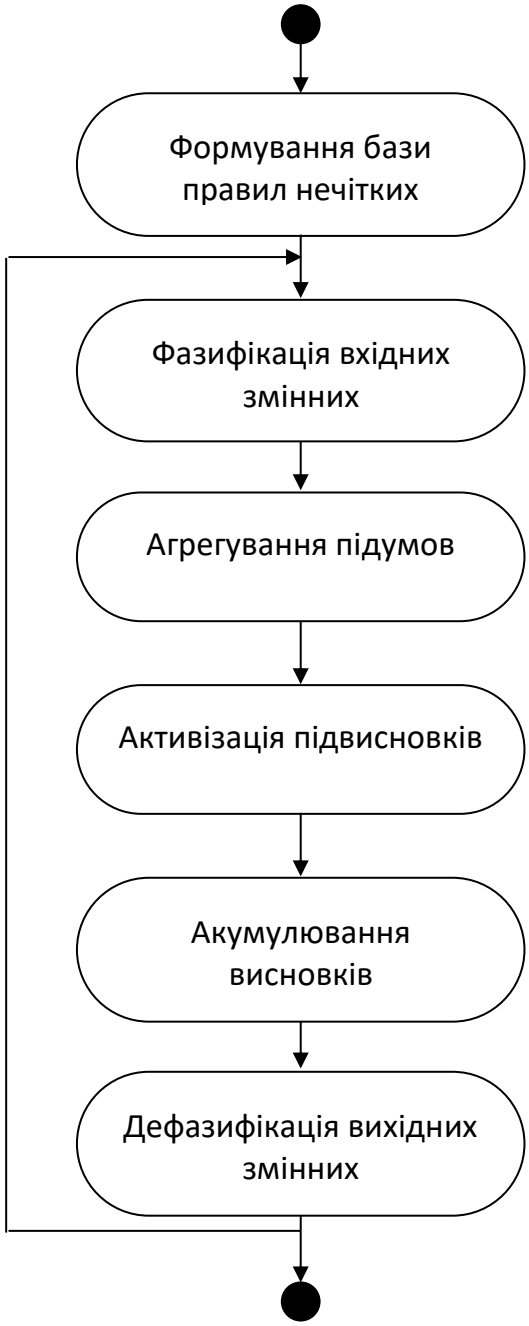



Рис. 2.4 Діаграма діяльності процесу нечіткого виводу (у формі діаграми діяльності мови UML).

Слід відмітити, що загальновизнаним недоліком методу нечітких множин є свобода у виборі функцій належності окремих термів лінгвістичних змінних, і беручи різні функції, можна отримувати досить далекі один від одного вихідні результати. Також слід відмітити, що інтелектуальна складова у даній роботі не є досить складною (як, наприклад, при управління технологічними системами із багатьма степенями свободи та численними вхідними змінними), тому замість нечіткої логіки можна застосувати у деякій мірі близькій підхід, а саме – алгоритми на основі дерев ухвалення рішень.

Серед інших методів галузі штучного інтелекту, які можна застосовувати для задач аналізу даних є:

- побудова різних класифікаторів: *k*-найближчих сусідів, байєсівського, опорних векторів, дерев ухвалення рішень, логістичної регресії;

- методи зменшення розмірності, пошуку правил;

- генетичні алгоритми;

- Q-навчання, побудова глибоких Q-мереж.

У разі Q-підходу формується якась функція корисності, причому на основі дій особи приймає рішення або автоматичної системи управління. При цьому стратегія поведінки суб'єкта формується на підставі сукупності попередніх кроків, а кожен крок генерує деяку складову корисності, тому, перебираючи різні набори кроків, поступово приходимо до оптимальної послідовності дій (переходів в просторі станів системи).

У генетичному алгоритмі також є якийсь аналог функції корисності, яку прийнято називати функцією пристосованості. Саме через наявність цих функцій, даний клас методів називається навчанням з підкріпленням. Відзначимо, що такі методи застосовуються для більш вузькоспеціалізованих завдань, аніж нейронні мережі, тому зважаючи на практиці віддають перевагу останнім перед навчанням з підкріпленням.[16,17]

Нарешті, до ансамблевих методів належать бустінг, стекінг і бегінг. Це різні варіанти організації єдиної системи на основі декількох однорідних або різнорідних моделей, які прийнято називати слабкими учнями.

## 2.3 Опис алгоритмів роботи програмного продукту, що проектується

Як уже зазначалося вище, алгоритми роботи з проектованим програмним забезпеченням не є досить складними, а навпаки досить просто виражаються схемою виду рис. 2.5.


Рис. 2.5 Блок-схема алгоритму роботи чат-бота, що надає інформацію про погодні умови

Для розробки даної інформаційної системи використано саме метод дерева рішень.

Метод дерева ухвалення рішень - це один з методів автоматичного аналізу величезних масивів даних. Перші ідеї створення "дерев ухвалення рішень" починаються з робіт П.Ховленда і Е.Ханта кінця 50-х років XX століття [12,17]. Але основоположною роботою, що дала імпульс для розвитку

цього напряму, стала книга Е.Ханта, Дж.Мерина і П.Стоуна "Experiments in Induction", яку було видано в 1966 р.

Область вживання методу "дерева ухвалення рішень" можна об'єднати в три класи:

- *опис даних: застосування " дерева ухвалення рішень " допускає зберігати інформацію про вибірку даних в компактній і зручній для обробки формі, що включає в собі точні описи об'єктів;*
- *класифікація: застосування " дерева ухвалення рішень " допускає справитися із задачами класифікації, тобто відношення об'єктів до одного з описаних класів;*
- *регресія: якщо змінна має недостовірні значення, тоді застосування "дерева рішень" допускає установити залежність цієї цільової змінної від незалежних (вхідних) змінних.*

Для прийняття рішення за допомогою " дерева ухвалення рішень " необхідно виконати такі крокі:

1) описати стан ринку вектором чинників X = (x1, x2, x3, x4) (цей крок виконується користувачем системи);

2) обчислити клас зрісту прибули шляхом руху вектора X = (x1, x2, x3, x4) по дереву рішень з верхніх рівнів до нижніх (цей крок виконується системою).

У методиці використовується схема ієрархічної структури. Для її побудови прийняті належні позначення елементів (подій) і логічних операцій.

В основу методу " дерева ухвалення рішень " покладено підпорядкованість, розгортаємось і ранжування цілей. Дерева ухвалення рішень з кількісними показниками, що використовуються в якості одного із засобів при прийнятті рішень, і носить назву "дерева рішень".

Головна перевага " дерева ухвалення рішень " перед іншими методами - можливість пов'язати ставлення цілі з діями, що підлягають реалізації в сьогоденні.[17,18] При побудові багаторівневого " дерева ухвалення рішень",

мета досягнення кожного рівня моделі забезпечується набором показників попереднього рівня. Відповідно до причинно-наслідкових зв'язків кожен рівень " дерева ухвалення рішень " повинен займати певну позицію в ієрархічній послідовності.

Правильно побудоване дерево має властивості узагальнення, тобто якщо виникає нова ситуація (новий клієнт), ймовірно, що ця ситуація вже відбулася і клієнт поводиться майже так само, як клієнти зі схожими характеристиками.

Переваги дерева ухвалення рішень:

- *Швидкий процес навчання*
- *Створення правил у сферах, де експерту важко формалізувати свої знання*
- *Правила формуються природною мовою*
- *Інтуїтивно зрозуміла модель*
- *Висока точність передбачення*
- *Дерева ухвалення рішень - це адаптовані моделі з мінімальним втручанням людини*

Головною ідеєю в такому підході є поєднання можливостей дерев рішень та нечіткої логіки. Відмінною рисою дерев рішень є те, що кожен приклад напевно належить конкретному вузлу. У нечіткому разі це не так. Для кожного атрибута необхідно виділити кілька його лінгвістичних значень та визначити ступеня належності прикладів до них.

Замість кількості прикладів конкретного вузла нечітке дерево рішень групує їх рівень власності. Коефіцієнт – це співвідношення прикладів вузла N для цільового значення i, що обчислюється як: де μN(Dj) – ступінь належності прикладу Dj до вузла N, μi(Dj) – ступінь належності прикладу щодо цільового значення i, S N – безліч усіх прикладів вузла N Потім знаходимо коефіцієнт, що позначає загальні характеристики прикладів вузла N.

У стандартному алгоритмі дерева рішень визначається відношення числа прикладів, що належать конкретному атрибуту, до загального числа прикладів. Для нечітких дерев використовується ставлення, до розрахунку якого враховується ступінь власності.

Вираз дає оцінку середньої кількості інформації для визначення класу об'єкта з множини P N. На наступному кроці побудови нечіткого дерева рішень алгоритм обчислює ентропію для розбиття за атрибутом A зі значеннями aj: , де вузол N|j – дочірній для вузла N. Алгоритм вибирає атрибут з максимальним приростом інформації.

Вузол N розбивається кілька підвузлів N|j. Ступінь приналежності прикладу Dk вузла N|j обчислюється покроково з вузла N як де μi(Dk,aj) показує ступінь приналежності Dk до атрибуту aj. Підвузел N|j видаляється, якщо всі приклади в ньому мають ступінь належності, що дорівнює нулю. Алгоритм повторюється до того часу, доки всі приклади вузла нічого очікувати класифіковані чи доки використовуватимуться для розбиття все атрибути.

Приналежність до цільового класу для нового запису знаходиться за формулою, де коефіцієнт співвідношення прикладів листа дерева l для значення цільового класу k, μl(Dj) – ступінь приналежності прикладу до вузла l, χk – належність значення цільового класу k до позитивного значення результату класифікації.

Висока якість результату досягається за рахунок визначення значущих факторів для вибору відповіді.[18,20]

Отриманий результат є статистично обґрунтованим.

Завдання, які необхідно вирішити, можна розділии на три класи:

- *Опис даних. Дерева ухвалення рішень дозволяють зберігати інформацію про дані в компактному вигляді та точно описувати об'єкти.*
- *Класифікація. Дерева ухвалення рішень добре справляються із завданнями класифікації – приписуючи об'єкти одному з відомих класів. Цільова змінна повинна мати дискретні значення.*
- *Регресія. Якщо цільова змінна має безперервні значення, дерева рішень дозволяють встановити залежність цільової змінної від незалежних (вхідних) змінних. Наприклад, цей клас включає задачі чисельного передбаченняя (прогнозування значень цільової змінної).*

Керуючись інформацією, наведеною у даному підрозділі, а також у вищенаведеному викладі, можна переходити до процесу проведення програмної реалізації чат-боту для месенджера Telegram, що і виконується у наступному розділі.

### 2.4 Проектування бази даних системи

При проектуванні бази даних слід визначитися, яка саме інформація буде і ній зберігатися та як її організувати. В даній роботі мова іде про одяг, який програма може рекомендувати на основі поточної або прогнозованої погоди. Відповідно, слід виділити види погоди, від яких залежить кінцеве рішення програми. Деякі з таких показників виражаються у числовій величині, деякі – у бальній, або у якісній формі (по дихотомічній системі «так/ні») і т.п. Отже, конкретні показники погоди, що будуть ураховуватися в даній роботі:

а) температура – головний показник, виражається у числовій формі;

б) опади у вигляді дощу – важливий показник, що виражається у бальній формі (1 – слабкий, 2 – середній, 3 – злива);

в) опади у вигляді снігу – показник, що виражається по бінарній формі (так/ні);

г) опади у вигляді граду (так/ні);

д) вітер – важливий показник, що виражається у числовій формі (м/с);

е) інсоляція або хмарність – другорядній показник, що описується градаціями (1 - ясно, 2 - змінна хмарність, 3 - хмарно);

є) явище туману – другорядний показник по системі (так/ні);

ж) відносна вологість – другорядний показник, що визначається у числовій формі (0-100%), який не є важливим через те, що пересічній людині важко визначити дану величину «на око».

Слід відмітити, що кожен тип одягу може характеризуватися наступними величинами:

- інтервалами числових величин (температури, вітру, вологості – *а*, *д*, *ж*) які підходять для даного типу одягу;

- бінарними характеристиками погоди (*в*, *г*, *є*);

- бальними характеристиками (*б*, *е*).

## 2.5 Обґрунтування вибору засобів розробки

### 2.5.1 Вибір мови програмування

Для програмування чат-ботів можна використовувати будь-яку серверну мову програмування. Аналіз публікацій навчального характеру з даного напрямку свідчить про те, що часто для даних цілей використовується мова Python та JavaScript (мається на увазі – на базі серверної платформи Node.JS). Якщо не прив'язуватися до досить вузької теми створення саме чат-ботів, то взагалі кажучи, Python більше використовується для створення додатків із потужною інтелектуальною складовою, наприклад нейронних мереж (які потребують використання досить складних алгоритмів типу зворотного поширення помилки та ін.). Функціонування даного чат-боту спирається на відносно просту логіку типу «запит-аналіз-відповідь», де аналіз ведеться на основі дерева ухвалення рішень (які можна легко реалізувати власноруч), тому використання Python із усіма його потужними бібліотеками та засобами розробки не є цілком обґрунтованим. Таким чином, у даній роботі будемо використовувати мову програмування JavaScript, за допомогою якої створена платформа Node.JS, можливості яких розглянемо докладніше.[19,20]

Важливою особливістю мови JavaScript є те, що її вихідні програмні коди інтерпретуються, це досить гнучке, але повільне рішення. Оператори у цій, в цілому С-подібній мові, розділяються крапкою з комою. Мова чутлива до регістру, що часто випускають з виду початківці, ймовірно тому, що HTML, застосовуваний зазвичай з JavaScript спільно, не залежить від регістру (імена тегів і атрибутів HTML можна писати як малими, так і великими літерами).

Однорядковий коментар виділяється символом //, а багаторядковий - парою символів / * і * /, в чому JavaScript знову повторює С.

До даних застосовується слабкий (динамічний) контроль типів. В операторах з різнотипними даними останні автоматично приводяться до необхідного типу. Типи даних можуть бути примітивними і складеними. Примітивні типи містять прості однорідні значення, такі дані можна передавати функціям як параметри за значенням, а не за посиланням. Складені типи містять різнорідні дані (в тому числі і складені), їх передають у функції тільки за посиланням.[21]

Мова JavaScript є об'єктно-орієнтована, проте вона заснована на прототипах, а не на класах. Є чотири типи об'єктів: вбудовані, об'єкти браузера, об'єкти документа і об'єкти користувача (програміста).

Введення-виведення в основному обмежене взаємодією з документами і користувачами. За умовчанням очікується, що доступ до локальної файлової системи недозволений. Однак браузери можуть надавати спеціальні об'єкти, за допомогою яких забезпечується робота з файловою системою користувача, хоча і з видачею попереджень про небезпеку здійснення файлових операцій.

Сценарії JavaScript активно взаємодіють з об'єктами, вбудованими в Web-сторінку. Для цього вони, власне, і створюються. Але перш, ніж ця взаємодія стане можливою, необхідно впровадити код сценарію в текст HTML-документа. Існує декілька способів поєднати HTML-документ з конкретним сценарієм (скриптом), але зазвичай їх просто розташовують всередині контейнерного тега <SCRIPT>, тобто між дескрипторами <script> і </script>.

Контейнер <SCRIPT> в цьому випадку буде знаходитися безпосередньо в HTML-документі, причому у довільному його місці. Програмний код пишуть саме в HTML-документі або в особливих текстових файлах, котрі можна викликати з головного HTML-документа. Для початку вивчимо перший варіант. Перш за все браузер знаходить тег <script> в тілі веб-документа, і весь подальший текст намагається обробити як скриптовий код. І так до тих пір,

поки не зустрінеться закриваючий тег </script>. Після цього всі подальші символи будуть сприйматися HTML-текстом. Всякий HTML-документ може розміщати довільне число «скриптових включень», проте кожне має відкриватися і завершуватися належним тегом. Від їх розташування у тілі HTML-документа подекуди може залежати функціонування всієї Web-сторінки, але про це буде сказано згодом.

Контейнерний тег <script> може розміщувати атрибут SRC, який показує ім'я або URL-адресу текстового файлу, що містить код сценарію. Цей атрибут потрібний в тому випадку, якщо сценарій розташований не безпосередньо в HTML-документі, а в окремому файлі. Розширення файлу зі сценарієм може бути яким завгодно, але зазвичай користуються js:

<SCRIPT SRC = «myscripts.js»> </ SCRIPT>.

Якщо сценарій розташовується в окремому файлі, то в ньому, зрозуміло, теги <SCRIPT> і </ SCRIPT> не пишуть. Сценарій, якій завантажений з зовнішнього файлу, можна уявити собі просто як його вставку в HTML-документ.

У браузерах, ще потенційно підтримують сценарії, дану функцію користувач може вимкнути (зокрема, із міркувань підвищеної безпеки). Крім того, є браузери, які принципово не приймають сценарії. У даній ситуації бажано хоча б відобразити повідомлення про те, що на сторінці був сценарій, але в цьому визначеному випадку він не здійсняється. З цією метою в HTML-документі користуються контейнерним тегом <noscript>, в якому розміщають текст, який появиться користувачеві за умови, що сценарій з тієї, чи іншої причини не здійснятиметься. Усі браузери, які підтримують сценарії, проігнорують вміст тегів <noscript> крім тих випадків, коли підтримка сценаріїв відключена. Браузери, які принципово не підтримують сценарії, навпаки, зміст тега <script> скинуть (особливо, якщо він вкладений у теги <!-- -->, які є HTML-коментарем), а тега <noscript> - відобразять.

Зразок наведено нижче:

<HTML> <HEAD>

<TITLE> Приклад застосування тега NOSCRIPT </ TITLE>

</ HEAD>

<SCRIPT TYPE = "text / JavaScript»>

<! -

alert ( «Підтримка JavaScript включена»); // ->

</ SCRIPT>

<NOSCRIPT>

<H2> Ваш браузер не підтримує JavaScript або його підтримка відключена </ H2>

</ NOSCRIPT>

</ HTML>

В цьому зразку було використано функція alert (повідомлення) для виведення повідомлень в маленькому діалоговому вікні.

Як уже зазначалося, в окремих файлах зазвичай розташовують бібліотеки функцій (визначення функцій), а також сценарії, котрі використовуються в кількох HTML-документах одного або декількох сайтів. Сценарій дозволено також писати у вигляді рядка операторів, між якими використовується крапка з комою. Такий рядок, взятий в лапки, служить в якості ролі атрибута-події, наприклад:

<IMG SRC = "mypicture.gif" ONCLICK = "alert ('Привіт!')".

Виклики функцій і їх визначення можуть слідувати в довільному порядку, проте тільки якщо вони розташовані в одному і тому ж контейнері <script>. Якщо функції розміщуються у відмінних контейнерах <script>, тоді необхідно, щоб визначення функції передувало її виклику. Аналогічно, якщо визначення функцій знаходяться в окремому файлі, то його треба завантажити в HTML-документ раніш викликів цих функцій.[22]

Браузер інтерпретує теги HTML послідовно. Так, зустрінувши вираз виклику функції myfunc() в одному контейнері <script>, браузер ще не має в пам'яті визначення цього об'єкта (функції), розташованого в іншому

контейнері <script>. Коли визначення функції і її виклик знаходяться в одному і тому ж контейнері <script>, то його вміст спочатку завантажується в пам'ять, а потім аналізується. Якщо ж інтерпретатор зустрічає виклик функції, тоді він шукає її визначення в пам'яті і в разі успіху реалізовує код цієї функції.

Якщо треба, щоб сценарій проявився в браузері перш, ніж буде загружено елементи HTML-документа, тоді його слід розмістити у верхній частині HTML-коду, а ще краще в контейнері <head> в заголовку документа.

Вперше мова JavaScript була виключно браузерною, яка була призначена для створення динамічних веб-сторінок, але у 2009 р. з’явилася перша версія движка системи Node.JS, що перетворив JavaScript на мову загального призначення. Втім, настільні додатки на Node.JS майже відсутні, а даний інструмент використовується для організації бек-енд складових сучасних веб-додатків. Отже, завдяки цій платформі, мова JavaScript стала досить розповсюдженою мовою серверного програмування.

### 2.5.2 Вибір інтегрованого середовища розробки

Якщо зафіксовано мову програмування, слід вибрати середовище розробки (IDE – Integrated Development Environment) і, у випадку з JavaScript існує досить багато варіантів таких продуктів. Деякі розробники цілком задовольняються можливостями консольного інтерфейсу Node.JS та пишуть код у текстових редакторах типу Sublime Text Editor – рис. 2.6.

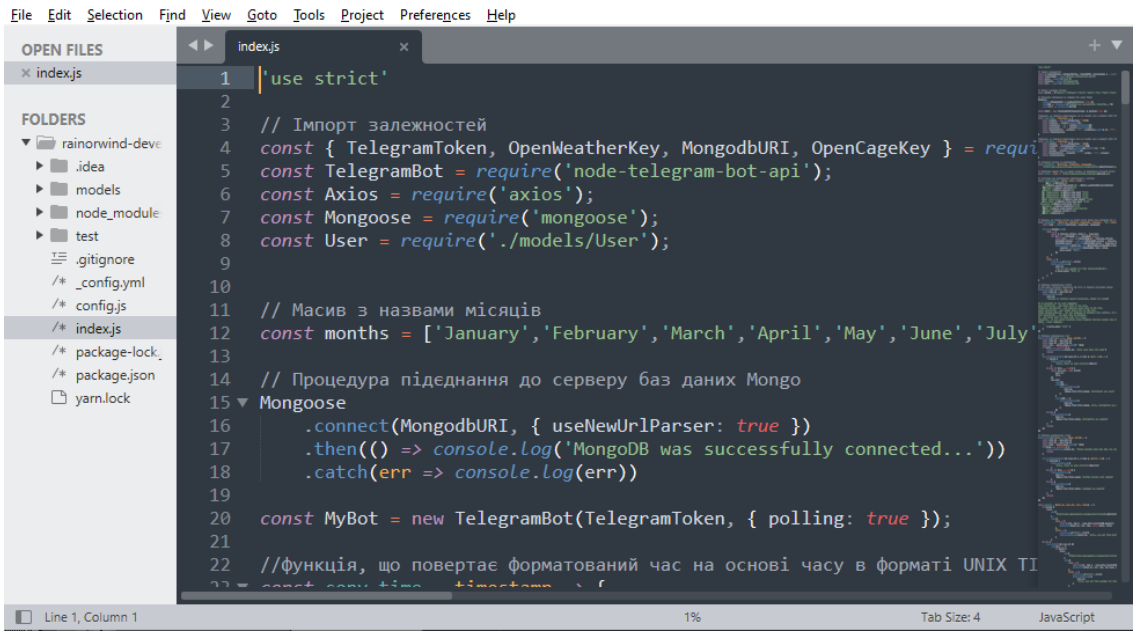



Рис. 2.6 Інтерфейс текстового редактору Sublime Text Editor

Sublime має підсвічування синтаксису та зручні автодоповнення коду. У лівій частині вікна може відображуватися дерево файлів певного каталогу (папки проекту), що доступно, якщо у головному меню «File» вибрати пункт «Open folder». Важливою рисою даного продукту є його малі системні вимоги, тому він дозволяє комфортно вести розробку навіть на дуже простих ПК.[23]

Трохи більш просунутим варіантом розробки на JavaScript є використання «легкого» середовища Visual Studio Code, яке є досить популярним для даної мови програмування. Слід відмітити, що даний продукт має вищі системні вимоги та у базовому варіанті є ненаповненим: усі можливості слід до встановлювати, що з одного боку не є проблемою, а з іншого – може бути неприємним фактором при необхідності активно попрацювати певний час без доступу до Інтернет.

Нарешті, можливим є використання одного з потужних середовищ розробки, наприклад, побудованих на базі движка JetBrains або Microsoft Visual Studio. Слід відмітити, що компанія-розробник JetBrains створила чимало варіацій такого середовища, що підходять для використання з різними мовами програмування. Зокрема, для розробки на JavaScript із використанням платформи Node.JS зручно використати основний продукт IntelliJ IDEA, яке

чудово підтримує розробку на JavaScript Node.JS. Дане інтегроване середовище розробки має величезну кількість плюсів, але великим його недоліком є дуже високі системні вимоги.

Зважаючи на те, що проектується не дуже великий продукт типу чат-бот, найпотужніші засоби розробки тут можна не використовувати, а створення програмного забезпечення будемо проводити з використанням найлегшого Sublime Text Editor.

### 2.5.3 Вибір додаткового програмного забезпечення для розробки

Для створення такої програми, як чат-бот у Telegram, не обов'язково використовувати ще додаткове програмне забезпечення, однак, продукт, що створюється саме у даній роботі призначений для роботи із погодними умовами. Очевидно, що їх потрібно зчитувати з одного із якихось зовнішніх джерел, оскільки створювати свій метеорологічний центр в даному випадку ані доцільно, ані можливо взагалі. Дані про погоду можна зчитувати безкоштовно за допомогою сервісу OpenWeatherMap від компанії OpenWeather (м. Лондон). Даний проект використовує у якості основи відомості з сайтів метеорологічних служб розвинутих країн (які публікують ці дані в Інтернет), тобто має функцію веб-агрегування. Однак, даний ресурс не є агрегатором у чистому вигляді, оскільки у великій мірі спирається на дані приватних метеостанцій, кількість яких зростає у всьому світі досить високими темпами. Таким чином, OpenWeatherMap має свої внутрішні алгоритми, які об'єднують інформацію з різних джерел і видають єдиний консолідований прогноз для вказаної місцевості.

При використанні сервісу OpenWeatherMap виникає два технічних моменти, що вимагають вирішення. По-перше, на ньому слід зареєструватися (сервіс допускає безкоштовний тарифний план із дещо зменшеною функціональністю, якої вистачає для цілей даної роботи), в результаті чого

сервіс надає ключ, що буде потрібний для використання у боті, що проектується – рис. 2.7.

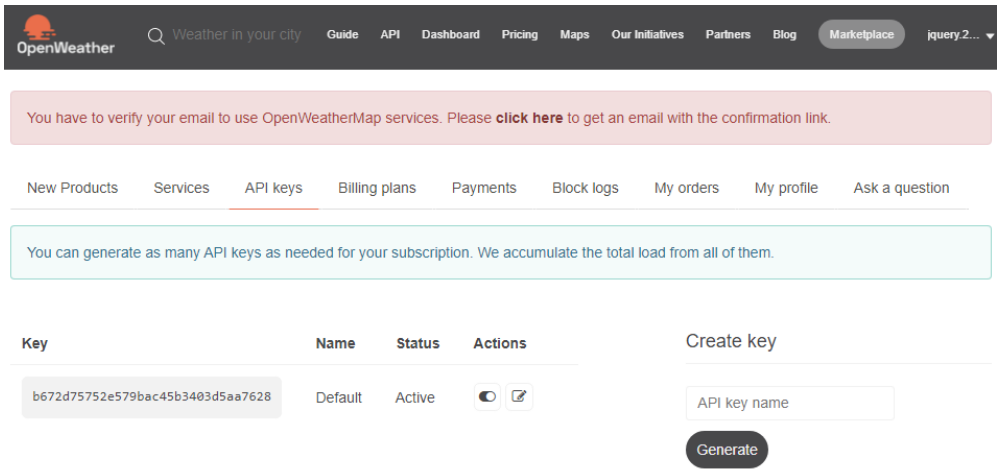



Рис. 2.7 Персональна сторінка безкоштовного тарифного плану на сервісі OpenWeatherMap

По-друге, для використання сервісу OpenWeatherMap, йому у HTTP запиті слід передавати широту і довготу місцевості, у якій треба визначити погоду. Очевидно, що користувач чат-боту, який хоче дізнатися відомості про потрібний гардероб навіть у своєму рідному місті навряд чи знає його географічні координати. Відповідно, виникає проблема переведення імені місцевості (зазвичай, це просто ім’я населеного пункту) у точні координати – широту та довготу, Для цього можна використати ще один сторонній програмний продукт, що допускає вільне використання – ресурс OpenCage.[23,24]

OpenCage є геокодувальною системою, однією із функцій якої якраз і є переведення текстової назви місцевості у географічні координати. Звичайно, для використання даного сервісу також слід зареєструватися, в результаті чого також надається відповідний ключ (Geocoding API Key) – рис. 2.8.

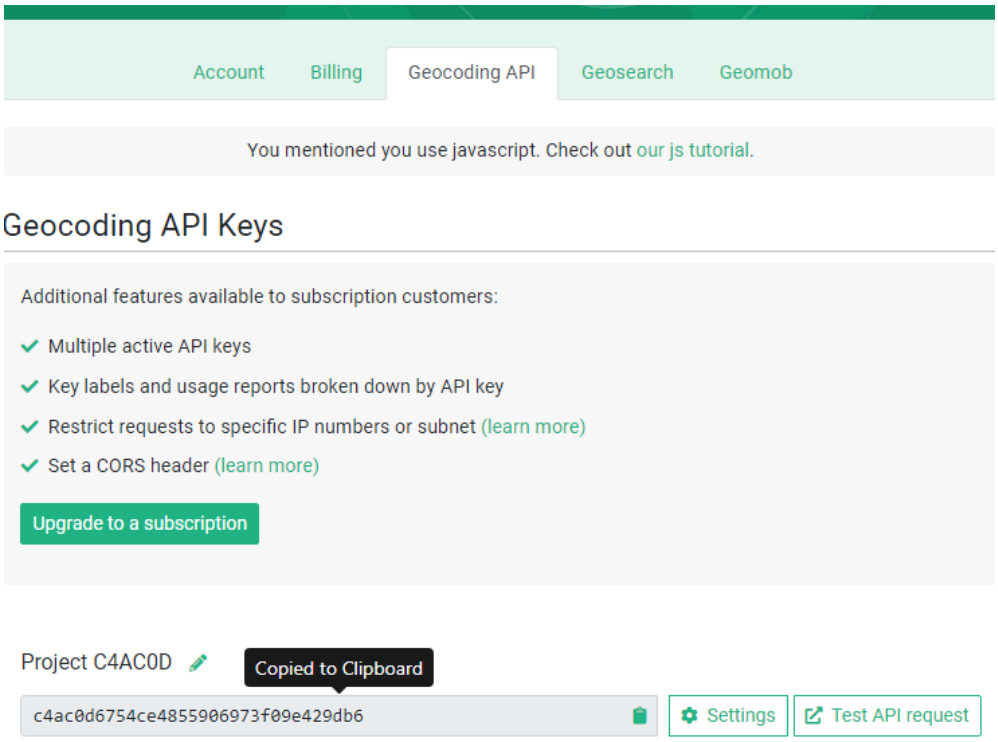



Рис. 2.8 Персональний кабінет пробного тарифного плану на платформі OpenCage

Також, як було встановлено вище, основою для розробки у даній роботі буде платформа Node.JS, яку також слід завантажити (з офіційного сайту https://nodejs.org) та встановити на цільовому комп'ютері, що гратиме роль сервера.

Також для роботи з Node.JS проектами потрібний менеджер пакетів командного рядка. Один із них уже вбудований в систему Node.JS: npm розшифровується як Node Package Manager. За бажанням за допомогою npm можна встановити один із інших менеджерів пакетів, наприклад, yarn для того, щоби проводити встановлення та оновлення необхідних додаткових модулів системи Node.JS.

Отже, за допомогою менеджеру пакетів можна встановлювати нові пакети (бібліотеки, розширення), потрібні для конкретного програмного продукту, що розробляється. В першу чергу, слід створити окрему папку, де буде писатися весь проект і зайти у неї за допомогою командного рядку,

використовуючи традиційні команди типу cd для зміни поточного каталогу. Далі слід виконати команду створення нового порожнього проекту Node.JS:

npm init

Для того, щоби вручну не відповідати на усі питання, а дозволити відповіді за умовчанням, команда має містити ключ yes:

npm init --yes

Після встановлення необхідних елементів нового проекту (яке може бути досить тривалим процесом, залежно від потужності цільового комп'ютера), слід вручну довстановлювати ті компоненти, які будуть специфічними для проекту чат-боту.

Для надсилання запитів HTTP, за якими саме і працює чат-бот Telegram необхідно встановити бібліотеку Axios:

npm install axios

Також при розробці чат-боту Telegram доцільно завантажити пакет node-telegram-bot-api, що можна зробити, наприклад, командою:

npm install node-telegram-bot-api

На цьому підготовче налаштування програмного середовища можна вважати завершеним і переходити безпосередньо до реалізації чат-боту для відстежування метеорологічних умов за допомогою мови програмування Node.js.

Інше додаткове програмне забезпечення, у даній роботі не є обов'язковим, а роботу по програмуванню логіки самого додатку, що розробляється, будемо виконувати безпосередньо за допомогою базового синтаксису обраної мови JavaScript.[24]

HTTP-запити

Мова розмітки гіпертексту (HTTP) є одним із найкращих протоколів TCP/IP (Протокол керування передачею/протокол Інтернету). Початково він

був розроблений для розміщення та отримання сторінок HTML (мова розмітки гіпертексту), а зараз його використовують для розподілених інформаційних систем. HTTP часто використовується в Інтернеті, а саме для передачі інформації і є найпоширеним використовуваним прикладним протоколом.

HTTP – це протокол, котрий визначається типом запиту/відповіді. Коли клієнт або веб-браузер надсилає запит на сервер, протокол HTTP автоматично визначає типи повідомлень. Ці повідомлення надсилаються на сервер, які клієнт використовує для запиту веб-сторінки, а також типи повідомлень, які сервер використовує для відповіді. Три найпопулярніші типи повідомлень: GET, POST і PUT. На рисунку 2.9. додано приклад запиту GET.

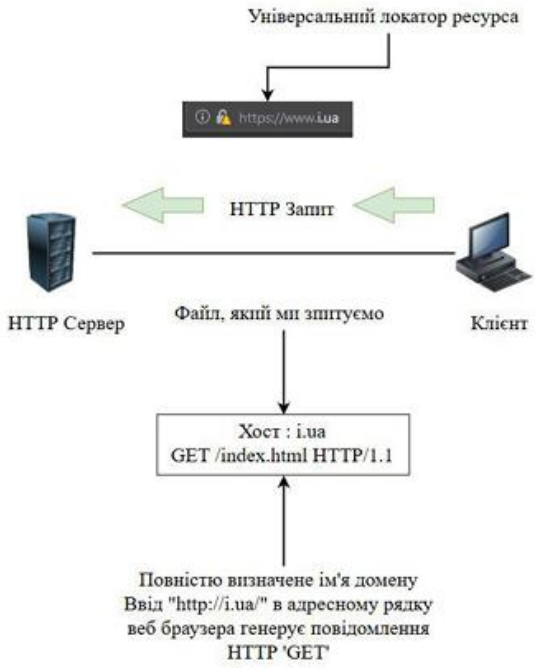



Рис. 2.9 Протокол HTTP за допомогою GET

POST і PUT використовуються для надсилання повідомлень, що завантажують дані на веб-сервер. Наприклад, коли користувач вводить дані до форми, яка знаходиться на веб-сторінці, POST додає ці дані до повідомлення та надсилається на сервер. PUT завантажує ці ресурси або інший вміст на веб-сервер.

Протокол HTTP надзвичайно гнучкий, але це не робить його безпечним протоколом. Повідомлення POST надсилають дані на сервер у вигляді простого тексту, який можна легко перехопити. Аналогічно, відповіді від сервера зазвичай не шифруються.

Захищений протокол HTTP (HTTPS) використовується для створення безпечного зв'язку через Інтернет, він дозволяє отримати доступ або публікувати інформацію на веб-сервері. HTTPS дозволяє використовувати аутентифікацію та шифрування для захисту даних від перехоплення під час їх переміщення між клієнтом і сервером.

HTTPS визначає деякі свої правила для передачі даних між прикладним і транспортним рівнями.

Формат JSON

JSON (JavaScript Object Notation) — це формат для обміну даними, зручний для читання, запису та зберігання, як людьми, так і комп'ютерами. Він був заснований на підмножині відомої мови програмування JavaScript і визначений в ECMA-262 3rd Edition - грудень 1999. JSON - це майже простий текстовий формат, повністю незалежний від мови реалізації, одначе він використовує деякі відомі правила на С-програмістів, таких як C, C ++, C #, Java, JavaScript, Perl, Python та багато інших. Саме ці властивості роблять JSON ідеальною мовою для спілкування.

Спочатку JSON базувався на двох структурах даних:

а) набір пар ключ/значення. Ця концепція різними мовами може бути реалізована як об'єкт, запис, структура, словник, хеш, іменований список або асоціативний масив;

б) упорядкований список цінностей. Більшість мов програмування реалізують це як масив, вектор, список або послідовність. Це універсальні структури даних. Майже всі сучасні мови програмування можуть приймати їх у будь-якій формі. Легко допустити, що формат даних, що зберігається в

JSON, не залежить від мови програмування і має базуватися на перерахованих вище структурах.

У нотації JSON це виглядає так:

а) об'єкт - це неупорядкований набір пар ключ/значення. Об’єкт починається з відкритої фігурної дужки і закінчується закритою фігурною дужкою. Кожне ім’я в об’єкті супроводжується двокрапкою, а пари ключ/значення розділяються комою (рисунок 2.10);

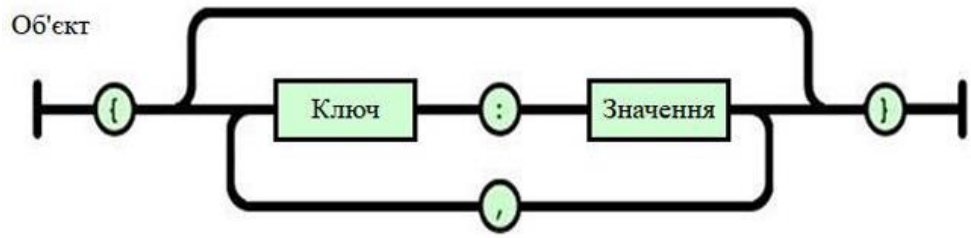



Рис. 2.10 Схематичне зображення об'єкта JSON

б) масив — це впорядкована колекція, яка може містити різні значення. Масив починається і закінчується квадратними дужками. Значення в масиві відокремлюються комами (рисунок 2.11);


Рис. 2.11 Схематичне зображення об'єкта JSON

в) значення - може бути числом, рядком у подвійних лапках, true, false, null, об'єктом або масивом. Ці структури можуть бути вкладеними (рисунок 5.5.3.3);

d) рядок — це так набір символів у кодуванні Unicode, узятий у подвійні лапки, або символ, який використовує зворотну косу риску як символ екранування. Символ представлений у вигляді одного символьного рядка. Подібний синтаксис використовується в мовах програмування C і Java (рисунок 2.12);


Рис. 2.12 Схематичне зображення об'єкта JSON

Число в JSON передається як у C або Java, за винятком того, що використовується тільки десяткова система числення (рис. 2.13).


Рис. 2.13 Схематичне зображення рядка в JS

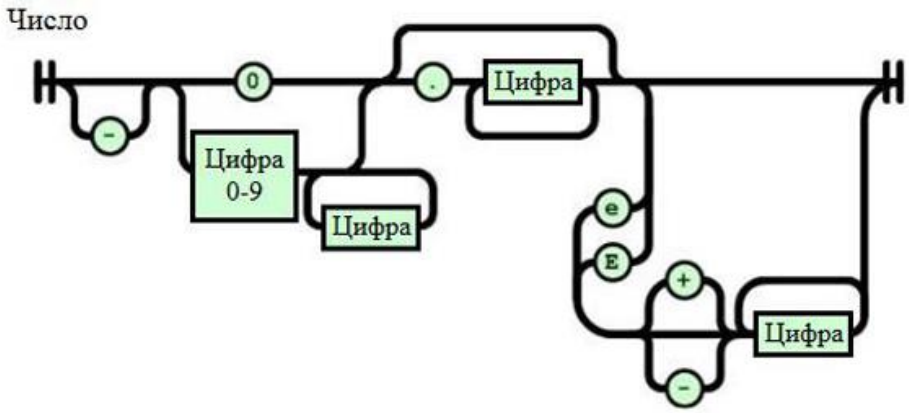



Рис. 2.14 Схематичне зображення числа в JSON

### 2.5.4 Вибір системи управління базами даних

Слід відмітити, що програмний продукт, який розробляється у даній роботі, буде зіставляти погодні умови із особливостями одягу, що

пропонуватиметься людині. Відмітимо, що фізично переважна більшість вказаної інформації являється текстовими рядками, які, можуть зберігатися як в в одному, так і окремих файлах. Найбільш логічним для таких цілей є використання текстових форматів (доцільніше всього – простого текстового формату *.TXT) або файлів електронних таблиць (у цьому випадку найбільш доцільно використовувати також досить прості формати типу *.CSV – Comma Separated Values, тобто значення, розділені комами), а також систем управління базами даних (реляційних БД та документо-орієнтованих).

Окремого розгляду потребує питання доцільності використання для окресленої задачі програм для роботи з реляційними базами даних. Цей дуже потужний засіб підлягає використанню у наступних випадках:

- дані, що використовуються, мають досить складну структуру із багатьма взаємозв’язками;

- дані, що підлягають збереженню, прості, але їх дуже багато (мається на увазі самих записів, кожен з яких є сам по собі досить простим);

- для забезпечення цілісності даних (оскільки практично будь-яка система управління базами даних – СУБД – не дозволяє видаляти записи, на які посилаються інші таблиці бази, а за рахунок нормалізації таблиць БД виключаються ситуації опису одного і того ж об’єкту реального світу кількома різними однотипними записами у базі даних – за всім цим слідкує процесор баз даних);

- для забезпечення швидкості доступу до даних при виконанні інтенсивних пошуків записів, що відповідають заданій умові (дуже поширений вид прикладної задачі), а також при виконанні сортувань різних типів (менш поширена операція, але така, що також досить часто зустрічається на практиці).

Проаналізуємо ці умови, при яких доцільно використовувати збереження даних у СУБД, стосовно до задачі, що розглядається. Усі відомості відносяться до переліку видів одягу, які підлягають рекомендуванню. Взагалі у світі кількість найменувань одягу може сягати тисяч і більше, а це число

записів є досить великим і обумовлює використання СУБД, що дозволяє швидко проводити складні пошуки та сортування. В той же час проектується універсальна програма, що повинна надавати універсальні рекомендації, без прив'язки до національного одягу та інших особливостей локальної місцевості, де живе користувач. Відповідно, число видів одягу звужується до величини порядку десятків, а в ця величина є значно меншою і використання реляційних баз даних тут не є доцільним.

Структура даних, що підлягають збереженню, задається завданням на дану роботу, і її не можна назвати досить складною, із багатьма взаємозв'язками. Простота структури даних та практично повна незалежність окремих компонентів інформації (відсутність зв'язків між ними) у системі свідчить про недоцільність використання СУБД у даній роботі.

Отже, всі ці факти свідчать про те, що у даному конкретному випадку застосування реляційних СУБД, наповнених обширними табличними даними, не є обґрунтованим, і дозволяє прийняти наступне рішення: цілком достатньо для даної розробки застосувати так звану текстову базу даних. Текстовою базою даних називають звичайний текстовий файл (часто просто формату *.TXT), у який інформація пишеться за певними правилами, у форматі, визначеному розробниками даної конкретної прикладної програми. Розробка формату збереження даних частково визначається вимогами завдання на розробку (наприклад, що окремі компоненти інформації про даний вид одягу мають відділятися один від одного символами зірочки «*»), а частково будуть визначені автором роботи (наприклад, порядок слідування окремих компонентів інформації) нижче, у відповідному підрозділі.

Сам файл текстової бази даних будемо розміщувати у тому ж каталозі, що й прикладна програма типу *.EXE. Доступ до файлу, відповідно до його суті, можна здійснювати як засобами прикладної програми, що розробляється, так і за допомогою звичайних текстових редакторів типу Блокнот.

Слід відмітити, що ще однією технологією доступу до даних, яку можна застосувати у даному проекті, є документо-орієнтована СУБД Mongo. Одним

із тривіальних варіантів використання саме MongoDB є дуже простий випадок, коли потрібно десь зберігати налаштування програми (які неефективно розміщувати в таблицях, оскільки практично всі вони є унікальними і не повторюються). В загальному випадку ця система використовується, коли у сховищі даних слід зберігати обширні дані, що не є структурованими, наприклад, являють собою окремі документи (і тому сама СУБД називається документоорієнтованою).

СУБД MongoDB слід завантажити з офіційного серверу mongodb.com, обираючи безкоштовну версію MongoDB Community Server – рис. 2.9.

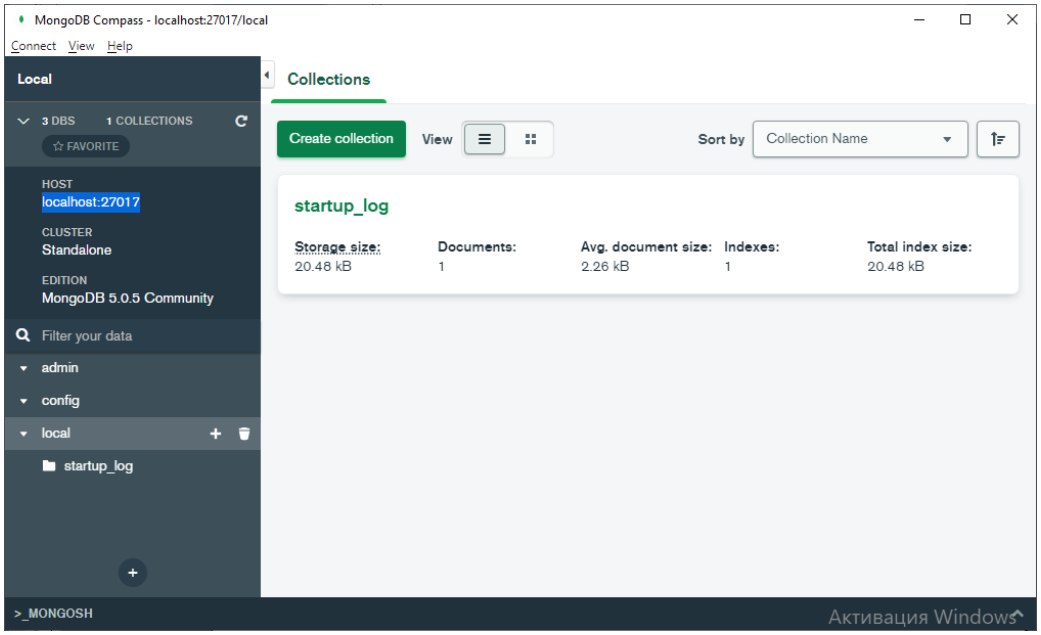



Рис. 2.9 Вікно засобу Compass для управління СУБД MongoDB

Для роботи з базою даних MongoDB зручно використати спеціальну бібліотеку для мови JavaScript, яка називається mongoose. Mongoose зображає собою спеціальну ODM-бібліотеку (Object Data Modelling) для роботи з MongoDB, вона дозволяє зіставляти об’єкти класів та документи колекцій із бази даних. Дана бібліотека встановлюється у командному рядку в каталозі проекту рядком виду:

npm install mongoose

Таким чином, у даному розділі обрано основні проектні рішення по розробці програми чат-боту, реалізація якої буде описана нижче.

## ВИСНОВОК ДО РОЗДІЛУ 2

У даному розділі було обрано архітектуру системи, що проектується та проведено аналіз існуючих методів інтелектуального аналізу даних та вибір базового з них для даної роботи. Після було описано алгоритми роботи програмного продукту, що проектується та спроектована база даних системи. Мовою програмування було обрано JavaScript, далі були визначені інтегроване середовище розробки, додаткове програмне забезпечення та система управління базами даних.

# РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ПРОДУКТУ

## 3.1 Реалізація бази даних системи

При реалізації бази даних рядок із її налаштуванням вбудовуємо у файл config.js:

```
const MongodbURI = "mongodb://localhost:27017"
```

Далі, інформація, що є налаштуваннями чат-боту, як уже було сказано вище, має зберігатися у базі даних MongoDB. У такому випадку створюється схема даних сховища, яка вміщується у каталог models (як відомо, відповідно до шаблону Model-View-Controller дані, що обробляються програмою, відносяться до сутності «модель») та являє собою файл User.js:

```
const userSchema = mongoose.Schema({
  user_id: {
    type: Number,
    required: true,
    unique: true
  },
  city: {
    type: String,
    required: true
  },
  lang: {
    type: String,
    default: 'en'
  }
});
```

В базі зберігаються ідентифікатор користувача в системі месенджера Telegram, обране ним місто та мова (за умовчанням – англійська). При

додаванні нового користувача у БД створюється новий запис наступного виду – рис. 3.1.

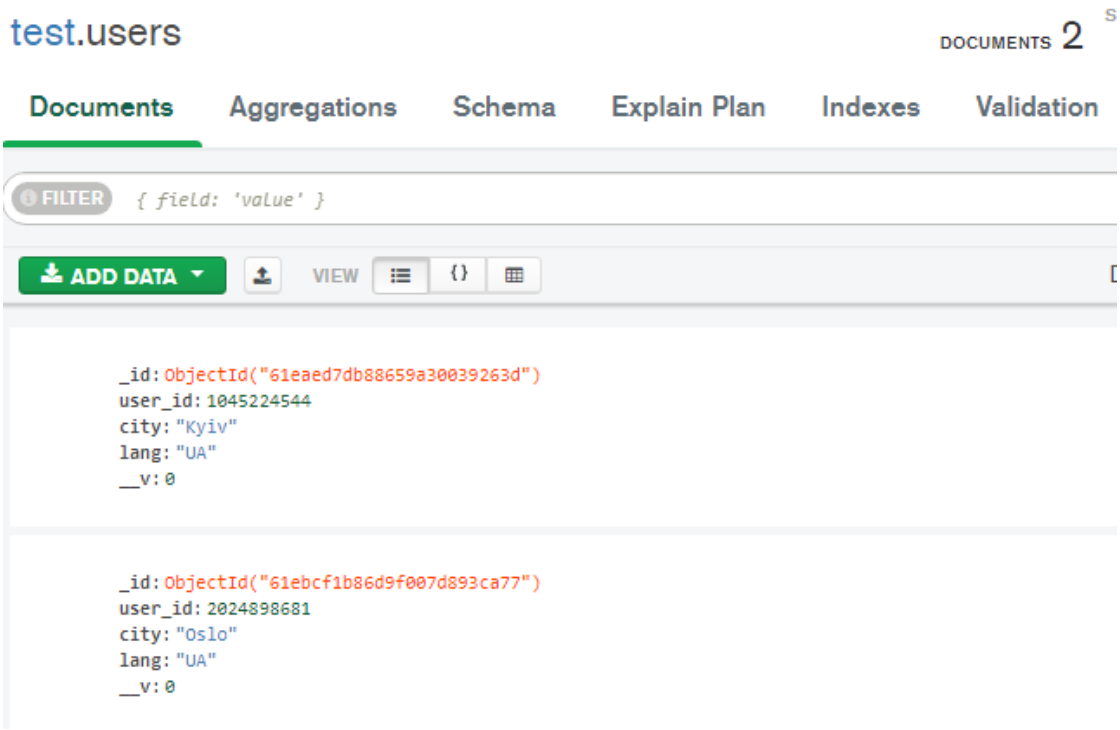



Рис. 3.2 Дані, що зберігаються у базі MongoDB

Таким чином, структура даних, що зберігаються у системі розроблено і можна переходити до наступного кроку.

### 3.2 Розробка інтерфейсу проектованого програмного забезпечення

Система обміну повідомленнями, яка виглядає як чат-бот, повинна шукати різні повідомлення в Telegram Messenger. Бот повинен мати зручний інтерфейс без зайвих доповнень. Користувачі повинні відразу зрозуміти, як виконати пошук. Оснащена найновішими сервісами Google, система допускає розробникам виробляти зручні, якісні, мобільні та актуальні чат-боти. З аналізу завдання, огляду методики та програмного забезпечення можна визначити вимоги до інформаційної системи.

Система повинна:

- *Використовувати всі версії Telegram Messenger;*
- *Зробити голосові пошукові запити;*
- *Створення текстових пошукових запитів;*
- *Підтримувати діалог із системою;*
- *Підтримує пошукові запити 2 мовами.*

Боти часто використовуються в робочих групах для автоматизації завдань, які зазвичай потребують участі людини. Але абсолютна відмова від взаємодії може обмежити творчість і продуктивність людей. Тому боти в основному використовуються для полегшення контактів і координації роботи. Наприклад, боти можуть інформувати про хід виконання завдання, інформувати всю групу, зв'язувати експертів з новачками і служити для формування довіри та командного духу.

У цьому випадку низька продуктивність або надмірне використання бота може призвести до перевантаження інформації або надмірних збоїв. В ідеалі взаємодія користувача з роботом має бути комфортним і безперебійним. Цього можна досягти шляхом ретельного планування сценаріїв розмов, особливо для спамерів. Зокрема, робот може повторяти введені команди, для того щоб користувач знав, що його «почули» і зрозуміли безпомилково. Бот ще повинен вміти розпізнавати, коли розмова зупинилася, і надавати підказки щодо того, чи може взаємодія продовжуватися. Роботи можуть мати елементи графічного інтерфейсу, які прискорюють взаємодію. Забезпечуючи глобальну перевірку за допомогою ключових слів, пов'язаних із навігацією («довідка», «повернення», «скасувати», «почати спочатку», «вихід» тощо), ви можете уникнути створення занадто «дурних» ботів.

Такі інструменти, як BotMock, допоможуть вам імітувати взаємодію користувача з ботами. Крім того, на багатьох платформах є готові протоколи спілкування, яким повинні дотримуватися боти, наприклад, у Facebook є набір простих прийомів для організації розмов. Чат-боти відрізняються залежно від завдань, які вони повинні вирішити. Для комерційних чат-ботів необхідно

працювати в межах певного простору розмови, теми та сфери відповідальності чат-бота, без використання методів машинного навчання.

Для цього типу чат-ботів використовуйте основні методи обробки мови:

- вибір іменованих сутностей або ключових слів;

- сегментація (це поділ на речення);

- стемінг / лемматизація.

А також технології створення регулярних граматик за типом вхідного патера є шаблоном належної реакції, приклад з AIML:

- масив (змінні) для вивчення або запам'ятовування контексту;

- регулярні вирази;

- рекурсія;

- цикли та інше.

Якщо у вас багато статистики, то краще створити більш складний чат-бот на технології машинного навчання, а саме метод глибоких нейронних мереж.

Інтерфейс програмного забезпечення, що розробляється, визначається можливостями месенджера Telegram. Традиційно робота чат-боту головним чином управляється текстовими командами виду /command. Конкретно у проектованому випадку передбачаємо використання наступних команд:

а) /start, що виводить перше повідомлення та видає перелік доступних команд;

б) /set, що дозволяє задати базове місто даного користувача;

в) /lang, що дозволяє встановити мову, якою буде видаватися пояснювальний текст про одяг;

г) /now, що видає погоду та рекомендований одяг на поточний момент у певному місті. Дану команду проектуємо у двох варіантах: в чистому вигляді (/now) та з одним аргументом (/now London) – назвою міста. Якщо команда задається без аргументу, то видається поточна інформація по місту, яке є базовим для даного користувача, тобто раніше було задано командою /set;

д) /tomorrow, що видає погоду та рекомендований одяг на завтра у певному місті. Дану команду також проектуємо у двох варіантах: в чистому вигляді (/tomorrow) та з одним аргументом (/tomorrow London) – назвою міста. Якщо команда задається без аргументу, то видається інформація на завтра по місту, яке є базовим для даного користувача, тобто раніше було задано ним командою /set;

е) /help, що видає допоміжну інформацію про чат-бот (аналогічно команді /start).

Для отримання токена, необхідно написати спеціальний бот @BotFather.

Приклади доступних методів для API:

а) отримати оновлення -це метод,який використовується для отримання оновлень щодо технології тривалого опитування;

б) setWebhook - цей метод прив'язується до URL-адреси бота домену, де знаходиться поточний бот;

в) sendMessage - метод, який відправляє текстове повідомлення в клієнтську частину Telegram;

г) sendLocation - метод, який надсилає повідомлення з координатами розташування пристрою;

е) getFile - метод, який повертається вкладений файли на їх ім'я.

Запити POST і GET дозволені.

Існує 4 способи передачі параметрів до API бота:

а) URL-запит;

б) додаток / x-www-form-urlencoded;

в) додаток / json;

г) багатокомпонентні / форма-дані (підходять для завантаження файлів).

Для роботи з Telegram Bot API ми вивчили документацію, в якій описані всі методи і параметри, виявилося, що всі відповіді приходять в JSON-форматі.

Під час написання чат-бота були протестовані та використані такі методи та типи:

Метод getUpdates використовується для отримання довгих оновлень опитування. Відповідь повертається у вигляді масиву об’єктів Update. Параметри методу наведені в таблиці 3.1.

Таблиця 3.1 – Параметри GetUpdates

| Параметри | Тип | обов'язковим | Опис |
|---|---|---|---|
| Зміщення | Ціле число | Ні | Зміщення стартового масиву зміщення дозволяє бачити новий повідомлення, ігноруючи старі. |
| Ліміт | Ціле число | Ні | Межі номер елементів масив Оновлення. Приймає значення від 1 до 100. |
| Час вийшов | Ціле число | Ні | Час вийшов в секунд для довго та опитування. За замовчуванням 0, тобто так викликав короткий запит. |

### 3.3 Опис реалізації окремих аспектів розробленого програмного забезпечення

Для всіх програм існує кілька незалежних віртуальних процесів, які називаються dynos. Вони розміщені в спеціальній віртуальній мережі, що складається з кількох серверів. Однією з особливостей Heroku є наявність власної системи контролю версій.

Черга повідомлень (MQ) — це архітектура для обміну повідомленнями між різними компонентами програмних систем в асинхронному режимі. Оце допускає одночасно надсилати повідомлення одному компоненту системи, що надсилає, і отримувати й обробляти інший компонент, що приймає, зовсім іншим. Системи цього типу складаються з споживача (одержувача) та виробника (відправника), котрі взаємодіють один з одним через провайдера (брокера), який інакше можна визначити як сервер MQ.

Черга — це структура даних з обмеженим доступом до елементів, які можна описати як «першим прийшов, першим обслужений». Обмеження полягає в тому, що елемент можна додати тільки в кінці черги, а вибрати елемент тільки з початку, при видаленні елемента з черги він знищується.

Особливості використання MQ-архітектури:

а) слабке зв'язування - створюються неявні інтерфейси, які обмінюються даними, що дозволяють процесам бути незалежними один від одного і зберігати потрібний формат повідомлення;

б) надлишковість - дає можливість уникнути нераціонального використання ресурсів системи або мережі, у свою чергу, зберігаючи ще необроблену інформацію;

в) масштабованість - за рахунок розподілу процесів обробки інформації дозволяє підвищити продуктивність MQ-сервера;

г) еластичність і здатність витримувати пікові навантаження - при високому навантаженні тому, черги повідомлень можуть служити так званим буфером для накопичення інформації, це дозволяє змінювати швидкість

обробки інформації і, таким чином, скорочувати загальне навантаження на систему або мережа;

д) стабільність - дозволяє відокремити один процес від іншого і одержувати повідомлення, таким чином, у разі відмови процесора є можливість поновити робочий стан системи, відклавши обробку повідомлень;

д) гарантована доставка - повідомлення в усякому випадку будуть оброблені та доставлені, самостійно від робочого стану системи-відправника та системи-одержувача повідомлень. Це добувається використанням асинхронного зв'язку та можливості зберігання повідомлення на сервері MQ до його обробки;

є) гарантована та одноразова процедура доставки - значна частина MQ-серверів дозволяє доставити дані в тому порядку, в якому вони були надіслані, забезпечуючи при цьому, що після прочитання повідомлення воно буде видалено з черги;

з) буферизація - це гарантія доставки повідомлень з максимальною ефективністю, яка досягається за рахунок незалежності швидкості обробки повідомлень від швидкості їх відправлення, це в свою чергу пов'язано з використанням наведеної вище структури черги - свого роду буфер між системою-обробником і системою. відправник.

API бота Telegram

Bot API — це HTTP-інтерфейс для функціонування з ботами в Telegram. Кожен бот - це спеціально створений обліковий запис для автоматичної обробки та відправки повідомлень.

Документація Telegram Bot API передбачає два щонайбільше протилежних способи отримання оновлень:

а) періодичні запити;

б) встановлення веб-гачків.

Вхідні оновлення зберігаються, доки сервер їх не обробить, проте не більше 24 годин. У відповідь ми отримуємо об’єкт Update, він серіалізований у JSON, незалежно від того, як ви отримуєте оновлення.

Перший і найпростіший варіант — періодично опитувати сервери Telegram на предмет нової інформації. З’єднання відкривається на короткий час, та всі оновлення відразу відправляють боту через з’єднання «Довге опитування». Цей спосіб простий, все-таки не дуже надійний.

Веб-хуки працюють трохи інакше. Якщо в чат поступає повідомлення, про це говорить сам Telegram, це робота вебхука. Отепер немає необхідності періодично опитувати сервери, усунувши, отже, мотив помилок пошукових систем. Проте за цю функцію необхідно заплатити за установку повноцінного веб-сервера на пристрій, на якому ви намічаєте запускати пошукових ботів.

Так само для належної роботи вам потрібен особистий сертифікат-SSL (Secure Sockets Layer), адже веб-хуки в Telegram служать тільки на HTTPS (рисунок 3.3).

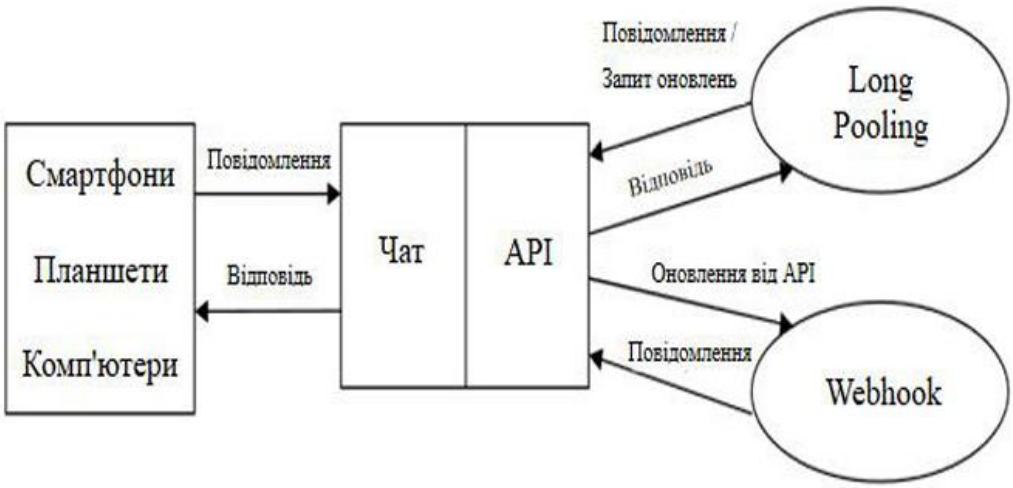



Рис. 3.3. Принцип роботи чат-бота на платформі Telegram

Реалізації підлягає програмний продукт типу чат-бот для месенджера Telegram, основна функціональність якого пов’язана із наданням

користувачеві інформації про погоду та рекомендований для неї одяг. Перед тим, як проводити програмування такого додатку, потрібно провести певні організаційні дії, пов’язані із його налаштуваннями. В першу чергу, для створення довільного чат-бота у Telegram потрібно отримати його ідентифікатор, або за термінологією Telegram – токен (англ. Token).[15]

За допомогою даного ідентифікатору здійснюється уся взаємодія в системі «чат-бот – користувач» і без нього створити чат-бота будь-яким способом не можливо. Для отримання токена слід у Telegram звернутися у чаті до головного чат-бота BotFather і провести декілька дуже простих команд:

а) при натисненні кнопки START – рис. 3.4 – висвітлюється досить обширний перелік усіх команд по налаштуванню чат-ботів;

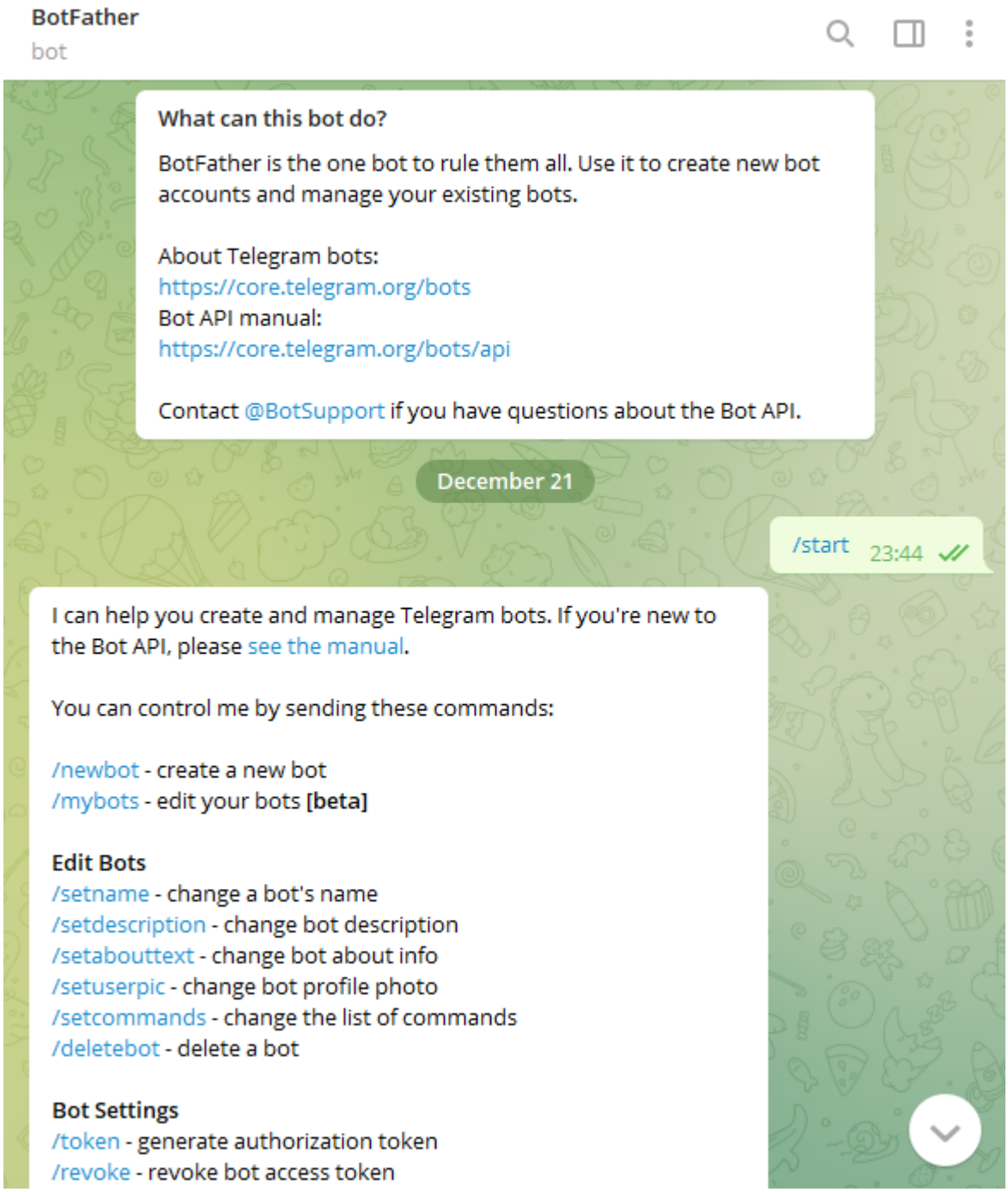



Рис. 3.4 Стартовий екран засобу BotFather, який дозволяє створювати нові чат-боти

б) найпершою командою є /newbot, що дозволяє перейти до процесу створення нового бота;

в) в першу чергу BotFather пропонує задати ім’я для нового бота – рис. 3.5, яке відповідає імені користувача при спілкуванні в Telegram, тому це

поле може бути не унікальним (тобто може співпадати з іменами існуючих користувачів та іменами інших ботів);

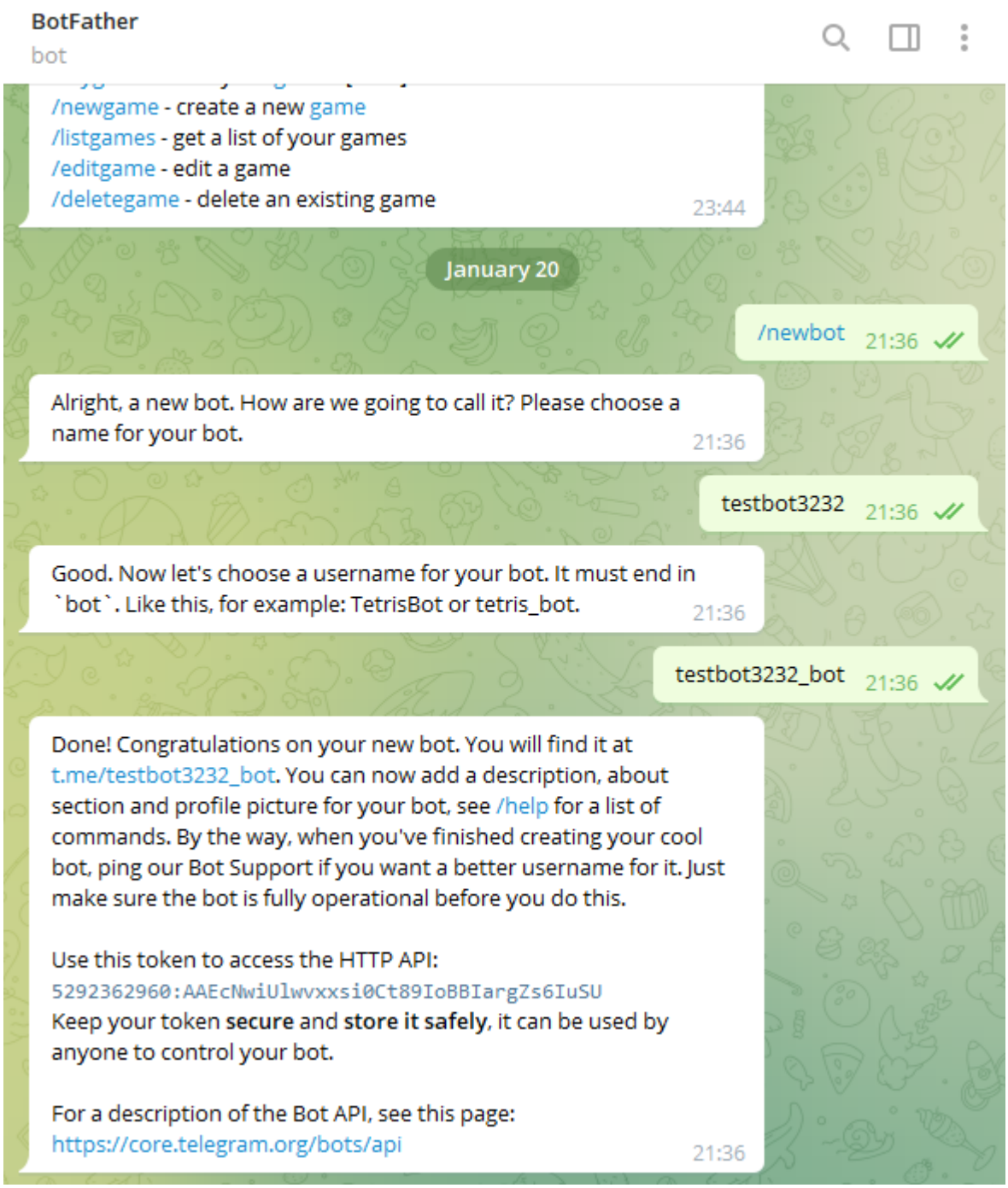



Рис. 3.5 Завдання імені та ніка для нового бота, а також отримання його токену

г) після введення імені BotFather запитує унікальне ім’я бота (або нік за термінологією Telegram), що не може співпадати з іменами інших ботів та повинно закінчуватися на літери «bot»;

д) у відповідь BotFather надсилає велике повідомлення, у якому найголовнішою інформацією є токен виду 270485614: AAHfiqksKZ8WmR2zSjiQ7_v4TMAKdiHm9T0. Цей текст треба зберегти та не поширювати, оскільки він надає повний контроль над ботом.

Для створення такої програми, як чат-бот у Telegram обов'язкових налаштувань більше не потрібно (хоча з додатковими налаштуваннями можна ознайомитися безпосередньо на рис. 3.3), тому можна переходити до опису програмної реалізації додатку у початкових кодах.

При створенні програмного коду, по-перше, у у файл config.js слід дописати ключі (ідентифікатори), що були отримані від компаній, що надають відповідні послуги, під час виконання дій, описаних у попередньому розділі:

```
const TelegramToken = "5292362960:AAEcNwiUlwvxxsi0Ct89IoBBIargZs6IuSU"
const OpenWeatherKey = "b672d75752e579bac45b3403d5aa7628"
const OpenCageKey = "c4ac0d6754ce4855906973f09e429db6"
module.exports = { TelegramToken, OpenWeatherKey, MongodbURI, OpenCageKey }
```

Далі у тілі програми, яка по суті і являє собою чат-бота (файл index.js) запроваджено 7 окремих функцій та 9 методів для об'єкта MyBot, який створюється рядком:

```
const MyBot = new TelegramBot(TelegramToken, { polling: true });
```

Функції, розроблені у програмі, є наступними (слід відмітити, що у коді програми об'ява функцій виконана у стрілковому стилі, який є не дуже зручним для опису в тексті документа, тому тут вони наводяться у традиційній формі «назва-список аргументів»):

а) conv_time(timestamp) – функція, що перетворює час, заданий у формі TIMESTAMP UNIX (тобто це кількість секунд, що пройшли з 01.01.1970) у гарний рядок;

б) conv_date(timestamp) – функція, аналогічна до попередньої, але яка формує не час, а дату у вигляді рядка;

в) wEndpoint(latitude, longitude, language) – функція, що отримує погоду від сервісу OpenWeatherMap за допомогою координат;

г) wIcon(icon) – функція, що повертає картинку, яка відповідає погоді;

д) wHTMLTemplate(data, date, city) – функція, яка формує великий рядок (у форматі HTML), що містить усю необхідну інформацію про погоду, дату, місцевість та головне – про рекомендований одяг;

е) getWeather(chat_id, latitude, longitude, language = 'en', index, city) – сервісна функція, яка є обгорткою для функції wEndpoint і виконує додаткові дії;

є) getInfo(chat_id, user_id, city, index) – функція ще більш високого рівня абстракції, аніж getWeather, і включає попередні запити до бази даних, з метою визначення базового міста користувача та встановленої ним мови, а також визначення координат у сервісі OpenCage за назвою міста.

Серед функцій, що є обробниками повідомлень користувача в чаті, можна назвати наступні:

- обробник запиту /start, що виводить перше повідомлення та видає перелік доступних команд;

- обробник запиту /set, що дозволяє задати базове місто даного користувача;

- обробник запиту /lang, що дозволяє встановити мову, якою буде видаватися пояснювальний текст про погоду;

- обробник запиту /now, що видає погоду зараз у заданому місті;

- обробник запиту /tomorrow, що видає погоду на завтра у заданому місті;

- обробник запиту /help, що видає допоміжну інформацію про чат-бот (аналогічно команді /start).

Інші особливості програмного коду можна дізнатися із початкового тексту, наведеного у Додатку А.

### 3.4 Керівництво користувача програмного продукту

Використання продукту типу «чат-бот» є надзвичайно простим та інтуїтивним, оскільки фактично вимагає тільки найпростіших умінь читати та набирати текст на клавіатурі. Однак, невеликий опис із можливими командами все ж буде необхідним.

Отже, розроблений продукт можна використовувати по наступній схемі:

а) завантажити месенджер Telegram;

б) знайти чат-бота у пошуку Telegram за іменем testbot3232_bot – рис. 3.6;

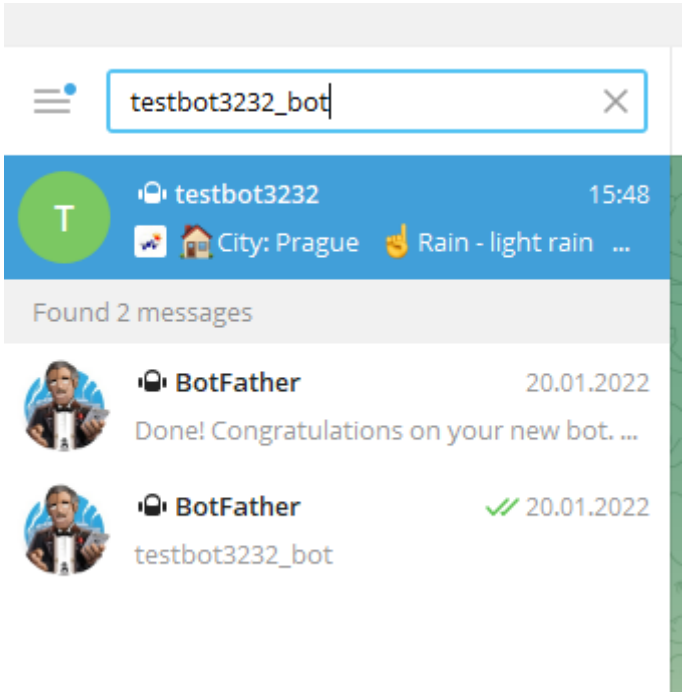



Рис. 3.6 Вікно пошуку чат-бота

в) натиснути кнопку START або написати команду /start у стандартному полі для введення текстових повідомлень;

г) прочитати вітальне повідомлення від бота, впевнившись, що він працює коректно;

д) написати і відправити “/set London” щоб задати базове місто, яке буде використовуватися ботом за умовчанням (замість слова London підставити назву свого міста англійською мовою; лапки писати не потрібно);

е) написати і відправити “/lang EN” щоб задати мову, якою буде видаватися інформація ботом за умовчанням (доступно дві опції: EN та UA; лапки писати не потрібно);

є) написати і відправити “/now” щоб дізнатися коротку інформацію про поточну погоду та рекомендований одяг у зарані заданому базовому місті (лапки писати не потрібно);

ж) написати і відправити “/tomorrow” щоб дізнатися коротку інформацію про погоду на завтра та рекомендований одяг у зарані заданому базовому місті (лапки писати не потрібно);

з) написати і відправити “/help”, щоби прочитати довідкове повідомлення про роботу даного боту.

## ВИСНОВКИ ДО РОЗДІЛУ 3

Таким чином, у даному розділі описано процес реалізації системи рекомендацій одягу на основі метеорологічних даних у вигляді чат-боту Telegram. Описано використану базу даних, спосіб взаємодії чат-боту з користувачем, реалізацію обраних елементів алгоритму роботи системи. Також у розділі наводиться коротке керівництво користувача продукту.

# РОЗДІЛ 4 ТЕСТУВАННЯ СТВОРЕНОГО ПРОГРАМНОГО ПРОДУКТУ

## 4.1 Тестування створеної інформаційної системи

Будь-який програмний продукт після проведення реалізації має бути підданий тестуванню. У даній роботі будемо проводити функціональне розширене позитивне ручне системне бета-тестування за схемою білої скриньки. Сама процедура тестування є наступною. У каталозі проекту у командному рядку задається команда завантаження проекту:

npm start

Після цієї команди проект завантажується на виконання і не закінчується, а блокує консоль і знаходиться у режимі виконання – рис. 4.1.

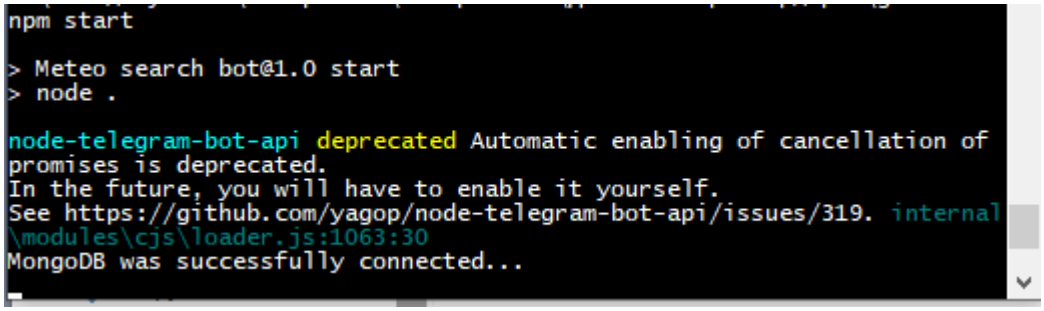



Рис. 4.1 Блокування консолі завантаженим проектом протягом всього часу його роботи

Для припинення роботи чат-боту потрібно натиснути Ctrl+C та підтвердити вихід із додатку натисканням кнопки «y» («yes»), або будь-яким способом закрити вікно консолі. Слід відмітити, що будь-яке спілкування з ботом у Telegram можливе тільки тоді, коли на якомусь комп'ютері, підключеному до мережі Інтернет, запущено консоль із відповідним (розробленим у даній роботі) додатком на платформі Node.JS. Ця умова виключає розміщення програмного забезпечення чат-боту на домашньому комп'ютері, який періодично вимикається (адже під час таких вимкнень бот у Telegram не буде працювати). Тому, для розміщення програми чат-боту у

професійному випадку слід використовувати хостинг мережі Інтернет, який дозволяє розміщувати Node.JS+MongoDB проекти.

Нажаль, отримати безкоштовний доступ до сервера саме Node.JS не є простою задачею (на відміну від повністю вільних PHP хостингів, яких чимало): часто доступні сервери змінюють власну політику, а через їх малу кількість зміна навіть одного сервера є досить важким ударом по даній галузі. Таким чином, більш надійним рішенням є використання платного хостингу, або безкоштовних варіантів із веденням постійного контролю, чи не потрібно переходити на інший безкоштовний варіант.

Отже, тестування відбувалося у повністю локальному режимі, тобто і Node.JS сервер і MongoDB сервер запускалися на звичайному локальному комп'ютері, що має доступ до мережі Інтернет.

В першу чергу, слід сказати про стартовий екран – рис. 4.2.

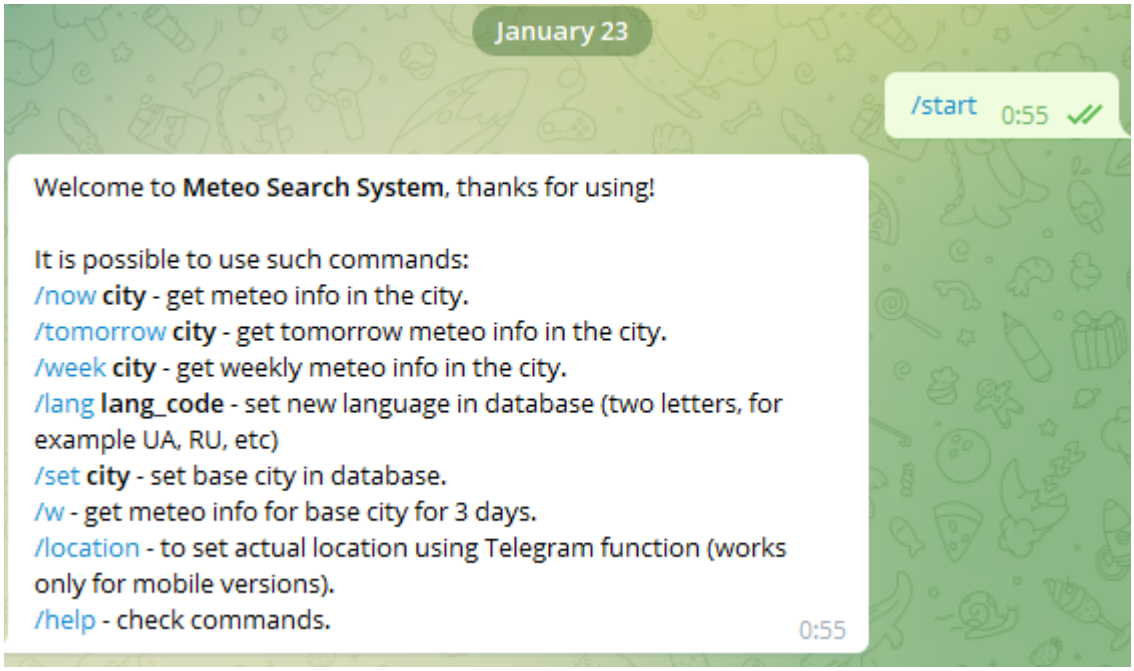



Рис. 4.2 Зовнішній вигляд стартового екрану створеного чат-боту

Екран, аналогічний до рис. 4.2 з'являється також при виборі команди /help.

При завданні команди /now із вказанням назви якогось міста у відповідь чат-бот видає правильну поточну погоду та рекомендований одяг – рис. 3.8. Тут слід звернути увагу, що по-перше, видається картинка, що відповідає погоді, наприклад, на рис.3.8 абсолютно у відповідності до тексту “clear sky” відображується картинка із ясним сонечком – це означає, що функція wIcon працює вірно. Під картинкою видається гарно відформатоване повідомлення, яке у точності відповідає HTML-шаблону, що формується функцією wHTMLTemplate.

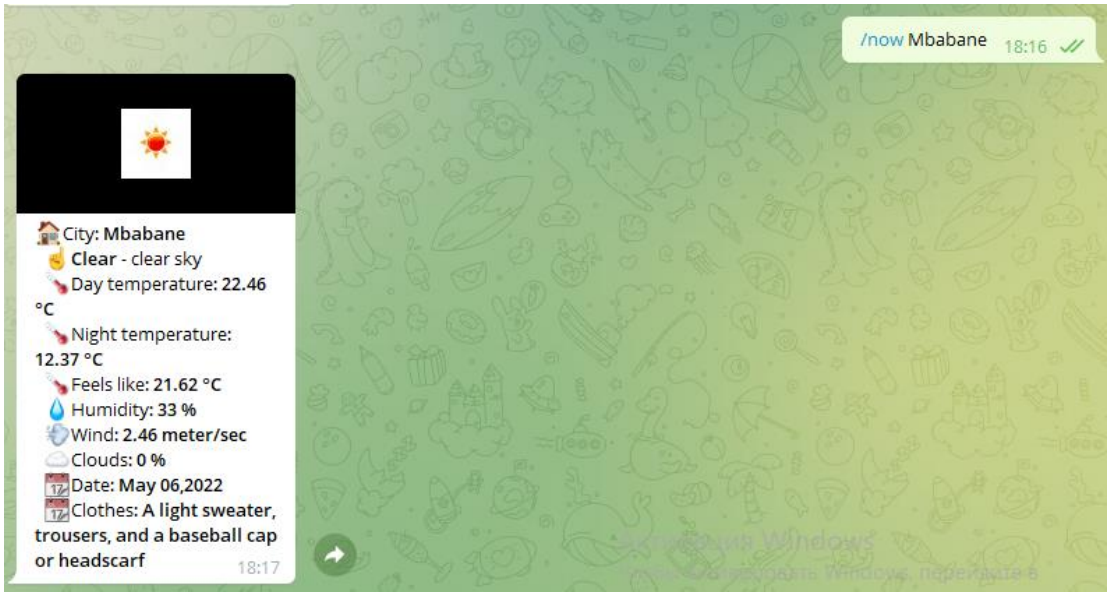



Рис. 4.3 Коректне відображення поточної погоди та рекомендованого одягу для заданого міста

Аналогічно до команди /now працює і інша команда /tomorrow. Двом розглянутим командам можна задавати назву міста, де користувач хоче дізнатися погоду та рекомендований одяг, або можна вказувати їх без аргументів, в результаті чого видається інформація для базового міста, яке задається командою /set.

Вказані команди для визначення погоди /now та /tomorrow можуть використовувати назву міста, яке користувач задав раніше командою /set.

Наприклад, на рис. 3.9, спочатку задано місто London, а потім команда /now, що працює без вказування міста, видає інформацію про погоду саму у Лондоні.

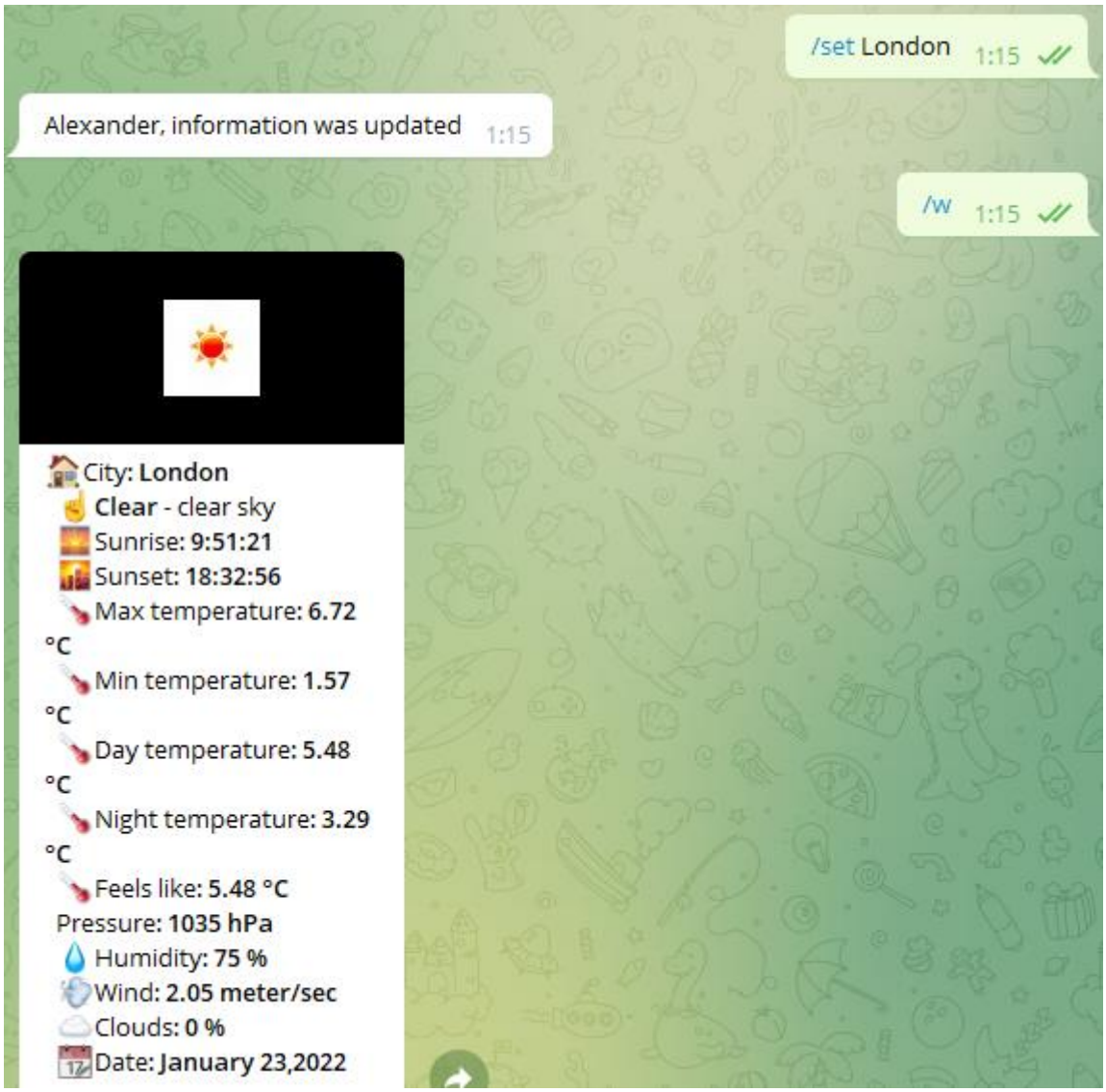



Рис. 4.4 Коректна робота команди /set для завдання міста у базі даних та /now для видачі погоди на три дні у цьому (базовому) місті

Нарешті, тестування команди /lang показало, що вона дійсно коректно працює (на рис. 4.5 спочатку задається українська мова, а потім видно, що видача використовує український текст, який можна порівняти з рис. 3.9), але частина всієї інформації, що підлягає переведенню є досить малою, тому особливої доцільності у використанні даної команди немає.

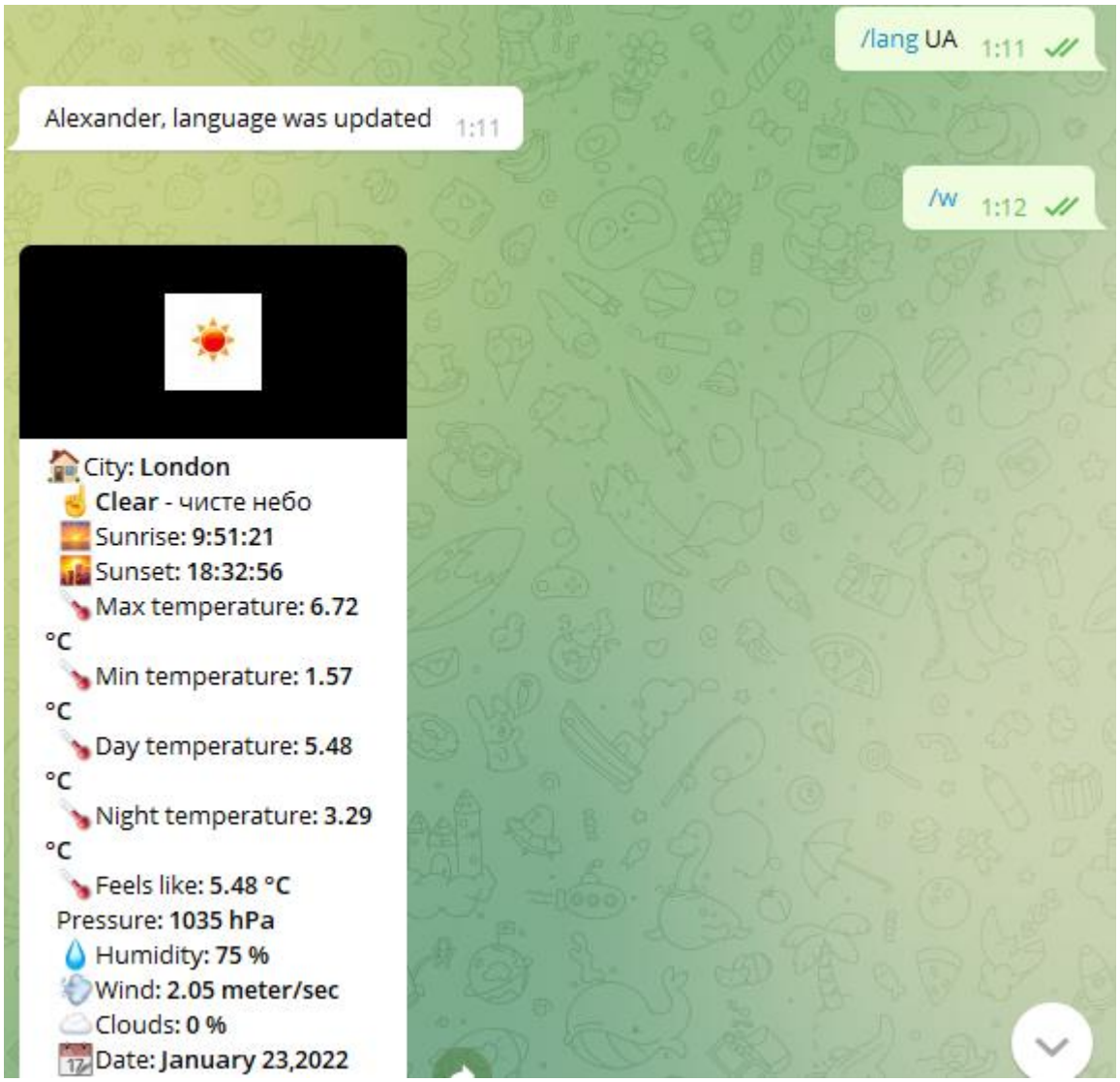



Рис. 4.5 Коректна робота функції завдання мови для видачі прогнозу погоди

Таким чином, проведене тестування показало, що чат-бот коректно виконує власну функціональність, що полягає у видачі користувачеві метеоумов та рекомендованого одягу для поточного моменту, а також прогнозних значень на найближчий час, причому на різних мовах та у географічно різних місцях (які може задавати користувач).

## ВИСНОВКИ ДО РОЗДІЛУ 4

Створено система рекомендацій гардеробу на основі прогнозу погоди і проведено тестування її можливостей. Результати продемонстрували можливість підбору гардеробу на основі прогнозу погоди. Вибір одягу був наданий чітко в рамках заданих температурних обмежень. І була надана інформація про одяг, в якому найкомфортніше, саме зараз.

# ЗАГАЛЬНІ ВИСНОВКИ

Таким чином, у даній роботі проведено розробку системи рекомендацій одягу, що працює на основі метеорологічних даних та реалізована у вигляді чат-боту для месенджеру Telegram.

В першу чергу, визначено які функції притаманні продуктам типу чат-бот та проаналізовано, як найбільш ефективно можна отримувати відомості про погоду та рекомендований до неї одяг. Встановлено, що використання чат-боту є одним із найбільш ефективних рішень і має чимало плюсів, порівняно з іншими способами.

Після фіксації технічних вимог до програмного продукту, було обрано його архітектуру (трирівнева, традиційна для чат-ботів), а також інструментальні засоби для розробки.

У якості алгоритму вибору вибрано метод дерева ухвалення рішень. Обраною мовою програмування є одна з найбільш популярних на сьогоднішній день мов JavaScript, а для того, щоби завантажувати серверні додатки (яким є обробник команд чат-боту), використано перспективну платформу Node.JS, поширення якої у веб весь час зростає.

Було розроблено необхідне алгоритмічне підґрунтя, спроектовано спосіб взаємодії бота з користувачами, описано дані, що зберігатимуться у БД продукту. Усі ці аспекти проекту враховано при проведенні його програмної реалізації, що описано у третьому розділі. Тестування розробленого чат-боту показало, що він у повній мірі виконує поставлену перед ним задачу, працює стабільно та без виникнення помилок системного чи логічного характеру.

Таким чином, розроблений програмний продукт може бути рекомендований до впровадження та постійного використання на базі месенджера Telegram (для чого потрібно лише розмістити проект на хостингу у мережі Інтернет, що підтримує платформу Node.JS).

У перспективі можливим є розширення переліку команд, що їх розпізнає чат-бот, а також дослідження можливостей по вибору інших алгоритмів вибору, окрім методу дерево ухвалення рішень, застосованого у даній роботі.

## ПЕРЕЛІК ПОСИЛАНЬ


1. Тидвелл Дж. Разработка пользовательских интерфейсов.— Питер, 2007.

2. Dan Goodin. Хакеры взломали SSL шифрование, используемое миллионами сайтов [Электронный ресурс]. – Хабр, 2011. – Режим доступа: https://habr.com/ru/post/128833

3. Хокинс С. Администрирование Web-сервера Apache и руководство по электронной коммерции.: Пер. с англ. М.: Издательский дом «Вильямс», 2001. 336 с.: ил.

4. Росс В. С. Создание сайтов: HTML, CSS, PHP, MySQL. Учебное пособие, М.:МГДД(Ю)Т, 2010. Ч.1. 107 с.

5. Никсон Р. Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript и CSS. 2-е изд //СПб.: Питер. – 2013. – 201 с.

6. Выразительный Javascript / Под ред. Marijn Haverbeke - No Starch Press, 2015. С. 37-41.

7. JavaSpript. Подробное руководство / Под ред. Дэвид Флэнаган. СПб, СимволПлюс, 2008. – 923 с.

8. JavaScript. Шаблоны / Под ред. Стоян Стефанов - СПб, Символ-Плюс, 2011. - 123 с.

9. Дэвид, Хэррон Node.js. Разработка серверных веб-приложений на JavaScript / Хэррон Дэвид. - М.: ДМК Пресс, 2016. - 405 с.

10. Изучаем Node.js. - М.: Питер, 2013. - 400 с.

11. Кантелон, М. Node.js в действии / М. Кантелон. - М.: Питер, 2015. - 950 с.

12. Кантелон, Майк Node.js в действии / Майк Кантелон. - М.: Питер, 2014. - 564 с.

13. Каскиаро, Марио Шаблоны проектирования Node.js / Марио Каскиаро. - М.: ДМК Пресс, 2016. - 900 с.

14. Крис, Аквино Front-end. Клиентская разработка для профессионалов. Node.js, ES6, REST / Аквино Крис. - М.: Питер, 2017. - 825 с.

15. Сухов, К. К. Node.js. Путеводитель по технологии / К.К. Сухов. - М.: ДМК Пресс, 2015. - 485 с.

16. Django, 2019. Как работают веб-приложения. [В Интернете] Available at: https://habr.com/ru/post/450282/ [Дата обращения: 12 март 2021].

17. Martin, S., 2019. JS-фреймворки для фронтенд-разработки. [В Интернете] Available at: https://habr.com/ru/company/ruvds/blog/476286/ [Дата обращения: 10 марта 2021].

18. Techrocks, p., 2019. Самые популярные фреймворки и языки программирования. [В Интернете] Available at: https://techrocks.ru/2019/07/20/most-popular-frameworks-and-languages-in-startups/ [Дата обращения: 12 март 2021].

19. Бибо Бер, Кац Иегуда jQuery. Подробное руководство по продвинутому JavaScript; Символ-плюс - М., 2017. - 624 с.

20. Дронов Владимир JavaScript и AJAX в Web-дизайне; БХВ-Петербург - М., 2015. - 736 с.

21. Дунаев Вадим HTML, скрипты и стили; БХВ-Петербург - М., 2015. - 816 с.

22. Никсон Робин Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5; Питер - М., 2016. - 768 с.

23. Резиг Джон , Бибо Беэр Секреты JavaScript ниндзя; Вильямс - М., 2015. - 416 с.

24. Роббинс Дженнифер HTML5, CSS3 и JavaScript. Исчерпывающее руководство (+ DVD-ROM); Эксмо - М., 2017. - 528 с.

# ДОДАТОК А. Код програми.

User.js

```
'use strict';

const mongoose = require('mongoose');

// User Schema
const userSchema = mongoose.Schema({
  user_id: {
    type: Number,
    required: true,
    unique: true
  },
  city: {
    type: String,
    required: true
  },
  lang: {
    type: String,
    default: 'en'
  }
});

const User = mongoose.model('User', userSchema);

module.exports = User;

'use strict';

const axios = require('axios');
require('dotenv').config();
const ApiKey = process.env.API_KEY;
const Token = process.env.TELEGRAM_TOKEN;
const ChatId = process.env.CHAT_ID;
const fetchDataCity = (city, variant) => axios
  .get(`http://api.openweathermap.org/data/2.5/${variant}?q=${city}&units=metric&&appid
=${ApiKey}`)
  .then(response => response.data)
  .catch(err => console.log(err));

const fetchDataCoords = (lat, lon) => axios
  .get(`http://api.openweathermap.org/data/2.5/weather?lat=${lat}&lon=${lon}&units=metri
c&&appid=${ApiKey}`)
  .then(response => response.data)
  .catch(err => console.log(err));
```

```
const fetchAPITelegram = (method, data) => axios
  .get(`https://api.telegram.org/bot${Token}/${method}?chat_id=${ChatId}&${data}`)
  .then(response => response.data)
  .catch(err => console.log(err));

// Months
const months = ['January', 'February', 'March', 'April', 'May', 'June',
  'July', 'August', 'September', 'October', 'November', 'December'];
// Convert time from timstamp to string
const convertTime = timestamp => {
  const date = new Date(timestamp * 1000);
  const hours = date.getHours();
  const minutes = '0' + date.getMinutes();
  const seconds = '0' + date.getSeconds();
  const formattedTime = hours + ':' + minutes.substr(-2) + ':' + seconds.substr(-2);
  return formattedTime;
};

const convertDate = timestamp => {
  const date = new Date(timestamp * 1000);
  const month = months[date.getMonth()];
  const day = String(date.getDate()).padStart(2, '0');
  const year = date.getFullYear();
  const output = month + ' ' + day + ',' + year;
  return output;
};


module.exports = { fetchDataCity, fetchDataCoords, fetchAPITelegram, convertTime,
convertDate };
'use strict';

const data = require('./data.json');
const { fetchDataCity, fetchDataCoords, fetchAPITelegram, convertTime, convertDate } =
require('./util');

describe.each(data.current_weather)('Getting weather info by city name:', (city, expected,
coords) => {
  test('testing by country (now)', () => {
    fetchDataCity(city, 'weather').then(data => {
      expect(data.sys.country).toEqual(expected);
    });
  });

  test('testing by country (tommorow)', () => {
    fetchDataCity(city, 'forecast').then(data => {
      expect(data.city.country).toEqual(expected);
```

```
    });
  });

  test('testing by coords', () => {
    fetchDataCoords(coords.lat, coords.lon).then(data => {
      expect(data.sys.country).toEqual(expected);
    });
  });
});

describe.each(data.convert_time)('Converting time from timestamp:', (timestamp, expected)
=> {
  test('from timestamp to string', () => {
    expect(convertTime(timestamp)).toBe(expected);
  });
});

describe.each(data.convert_date)('Converting date from timestamp:', (timestamp, expected)
=> {
  test('from timestamp to string', () => {
    expect(convertDate(timestamp)).toBe(expected);
  });
});

describe.each(data.api_telegram)('Testing Telegram API:', (name, method, data, expected)
=> {
  test(name, () => {
    fetchAPITelegram(method, data).then(data => {
      expect(data.ok).toEqual(expected);
    });
  });
});
```

Index.js

```
'use strict'

// Імпорт залежностей
const { TelegramToken, OpenWeatherKey, MongodbURI, OpenCageKey } =
require('./config');
const TelegramBot = require('node-telegram-bot-api');
const Axios = require('axios');
const Mongoose = require('mongoose');
const User = require('./models/User');



// Масив з назвами місяців
```

```
    const months =
['January','February','March','April','May','June','July','August','September','October','November','
December'];

    // Процедура підєднання до серверу баз даних Mongo
    Mongoose
        .connect(MongodbURI, { useNewUrlParser: true })
        .then(() => console.log('MongoDB was successfully connected...'))
        .catch(err => console.log(err))

    const MyBot = new TelegramBot(TelegramToken, { polling: true });

    //функція, що повертає форматований час на основі часу в форматі UNIX TIMESTAMP
    const conv_time = timestamp => {
        const curdate = new Date(timestamp * 1000);
        const curhours = curdate.getHours();
        const curminutes = '0' + curdate.getMinutes();
        const curseconds = '0' + curdate.getSeconds();
        const TimeFormatted = curhours + ':' + curminutes.substr(-2) + ':' + curseconds.substr(-
2);
        return TimeFormatted;
    }

    //функція, що повертає форматовану дату на основі часу в форматі UNIX TIMESTAMP
    const conv_date = timestamp => {
        const curdate = new Date(timestamp * 1000);
        const curmonth = months[curdate.getMonth()];
        const curday = String(curdate.getDate()).padStart(2, '0');
        const curyear = curdate.getFullYear();
        const DateFormatted = curmonth + ' ' + curday + ',' + curyear;
        return DateFormatted;
    }

    // отримання погоди за координатами
    const wEndpoint = (latitude, longitude, language) =>
        `https://api.openweathermap.org/data/2.5/onecall?lat=${latitude}&lon=${longitude}&ex
clude=hourly,minutely&appid=${OpenWeatherKey}&units=metric&lang=${language}`

    // Допоміжна адреса URL, що надає іконку, що відповідає поточній погоді
    const wIcon = icon => `http://openweathermap.org/img/w/${icon}.png`

    // заготовка для відображення повідомлення з погодою
    const wHTMLTemplate = (data, date, city) =>
        `🏠City: <b>${city}</b>
        ☝ <b>${data.weather[0].main}</b> - ${data.weather[0].description}
        👲Sunrise: <b>${data.sunrise}</b>
        🌇Sunset: <b>${data.sunset}</b>
```

```
🌡Max temperature: <b>${data.temp.max} °C</b>
🌡Min temperature: <b>${data.temp.min} °C</b>
🌡Day temperature: <b>${data.temp.day} °C</b>
🌡Night temperature: <b>${data.temp.night} °C</b>
🌡Feels like: <b>${data.feels_like.day} °C</b>
Pressure: <b>${data.pressure} hPa</b>
💧Humidity: <b>${data.humidity} %</b>
🌬Wind: <b>${data.wind_speed} meter/sec</b>
☁Clouds: <b>${data.clouds} %</b>
🗓Date: <b>${date}</b>
`


    // функція, що отримує погоду за назвою міста (якщо вона вказана) або за
координатами
    const getWeather = (chat_id, latitude, longitude, language = 'en', index, city) => {
        const endp = wEndpoint(latitude, longitude, language)

        Axios.get(endp).then(
                resp => {
                        const { timezone_offset, daily } = resp.data
                        for (let i = index[0]; i <= index[1]; i++) {
                                const date = conv_date(daily[i].dt + timezone_offset)
                                daily[i].sunrise = conv_time(daily[i].sunrise + timezone_offset)
                                daily[i].sunset = conv_time(daily[i].sunset + timezone_offset)
                                let temp = daily[i].feels_like.day
                                let wind = daily[i].wind_speed
                                let rain = false;

        if(daily[i].weather[0].description.toLowerCase().indexOf('rain')>=0)
                                        rain = true

        if(daily[i].weather[0].description.toLowerCase().indexOf('дощ')>=0)
                                        rain = true
                                let snow =
daily[i].weather[0].description.toLowerCase().indexOf('snow')>=0 ? true : false
                                let clothes = ''
                                if(temp<=-20){
                                        clothes='Екстремально тепла зимова куртка чи шуба.
Теплі ватні штани. Валянки. Рукавиці. Шапка вушанка. '
                                }
                                if(temp>-20 && temp<=0){
                                        clothes='Зимова куртка чи тепле пальто. Брюки чи тепла
спідниця. Зимові чоботи. Рукавички. Елегантний капелюх. '
                                }
                                if(temp>0 && temp<=18){
```

```
                                clothes='Демісезонна куртка чи теплий світер. Брюки чи
спідниця. Осінні туфлі. '
                        }
                        if(temp>18 && temp<=35){
                                clothes='Легкий светр. Легкі брюки чи шорти. Легкі
туфлі чи сандалі. '
                        }
                        if(temp>35){
                                clothes='Пляжний одяг і до водички! '
                        }
                        if(rain){
                                clothes+='Не забудьте взяти з собою парасольку!'
                        }
                        MyBot.sendPhoto(chat_id, wIcon(daily[i].weather[0].icon), {
                                caption: wHTMLTemplate(daily[i], date, city)+clothes,
                                parse_mode: 'HTML',
                        })
                }
        },
        error => {
                console.log('error', error)
                MyBot.sendMessage(
                        chat_id,
                        "Can not get weather for this <b>location!</b>",
                        { parse_mode: 'HTML' }
                )
        }
    )
}

// Обробник повідомлення /start
// Він видає загальну інформацію про бота та перелік доступних команд
MyBot.onText(/\/start/, msg => {
    const chat_id = msg.chat.id
    MyBot.sendMessage(
            chat_id,
            `Welcome to <b>Meteo Search System</b>, thanks for using!

It is possible to use such commands:
/now <b>city</b> - get meteo info in the city.
/tomorrow <b>city</b> - get tomorrow meteo info in the city.
/week <b>city</b> - get weekly meteo info in the city.
/lang <b>lang_code</b> - set new language in database (two letters, for example UA, RU,
etc)
/set <b>city</b> - set base city in database.
/w - get meteo info for base city for 3 days.
/location - to set actual location using Telegram function (works only for mobile versions).
```

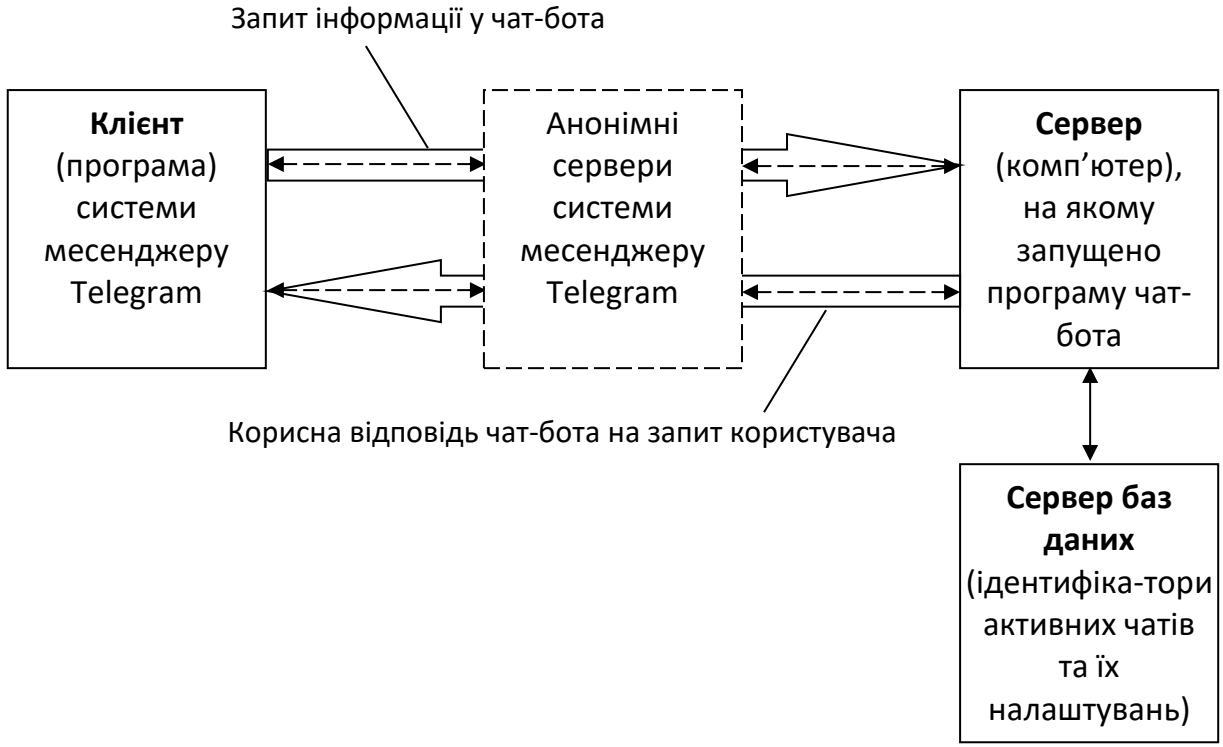
```
    /help - check commands.
      `,
            { parse_mode: 'HTML' }
      )
    })

    // Обробник повідомлення /set
    MyBot.onText(/\/set (.+)/, (msg, match) => {
      const chat_id = msg.chat.id
      const user_id = msg.from.id
      const city = match.input.split(' ')[1]
      if (city === undefined) {
            MyBot.sendMessage(chat_id, `Enter your base city name`)
            return
      }
      User.findOneAndUpdate({ user_id }, { city }, (err, res) => {
            if (err) {
                  MyBot.sendMessage(
                        `Sorry, there is some error:\n\r${err}`
                  )
            } else if (res === null) {
                  const new_user = new User({
                        user_id,
                        city,
                  })
                  new_user
                        .save()
                        .then(() =>
                              MyBot.sendMessage(
                                    chat_id,
                                    `${msg.from.first_name}, information was saved`
                              )
                        )
                        .catch(() => {
                              MyBot.sendMessage(
                                    chat_id,
                                    `${msg.from.first_name}, sorry, information was
not saved`
                              )
                        })
            } else {
                  MyBot.sendMessage(
                        chat_id,
                        `${msg.from.first_name}, information was updated`
                  )
            }
            return
```

```
        })
    })

    // Обробник повідомлення /lang
    MyBot.onText(/\/lang (.+)/, (msg, match) => {
        const chat_id = msg.chat.id
        const user_id = msg.from.id
        const lang = match.input.split(' ')[1]
        if (lang === undefined) {
            MyBot.sendMessage(chat_id, `Please provide lang code (UA, RU, EN, ...)`)
            return
        }

        User.findOneAndUpdate({ user_id }, { lang }, (error, res) => {
            if (error) {
                MyBot.sendMessage(
                    `Sorry, there is some error:\n\r${error}`
                )
            } else if (res === null) {
                MyBot.sendMessage(
                    chat_id,
                    `${msg.from.first_name}, firstly provide /set command`
                )
            } else {
                MyBot.sendMessage(
                    chat_id,
                    `${msg.from.first_name}, language was updated`
                )
            }
            return
        })
    })

    const getInfo = (chat_id, user_id, city, index) => {
        if (city) {
            Axios
                .get(

        `https://api.opencagedata.com/geocode/v1/json?q=${city}&key=${OpenCageKey}&pret
ty=1`
                )
                .then(
                    resp => {
                        const { lat, lng } = resp.data.results[0].geometry
                        getWeather(chat_id, lat, lng, null, index, city)
                    },
                    error => {
```

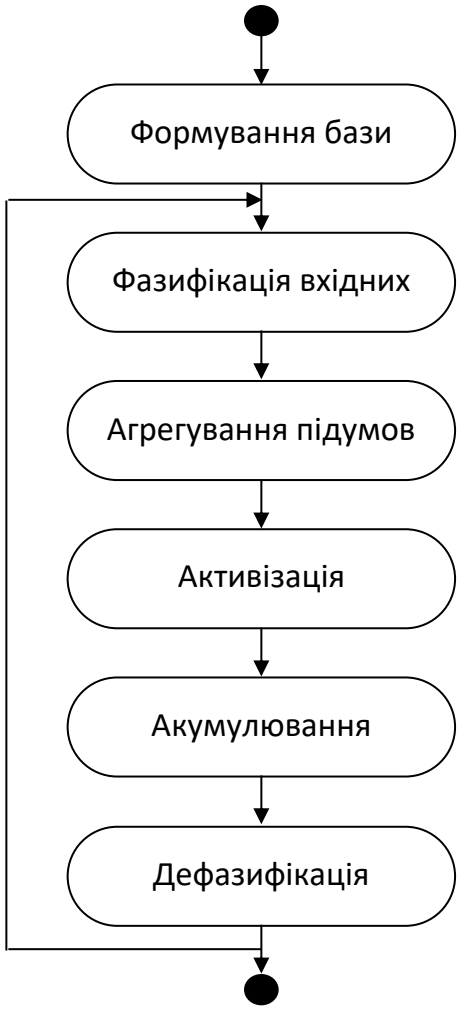
```
                                        console.log('error', error)
                                        MyBot.sendMessage(chat_id, `Sorry, can not find latitude
and longitude.`+error)
                                }
                        )
        } else {
                User.findOne({ user_id })
                        .then(doc => {
                                if (doc) {
                                        Axios
                                                .get(

        `https://api.opencagedata.com/geocode/v1/json?q=${doc.city}&key=${OpenCageKey}&
pretty=1`
                                                )
                                                .then(
                                                        resp => {
                                                                const { lat, lng } =
resp.data.results[0].geometry
                                                                getWeather(chat_id, lat, lng, doc.lang,
index, doc.city)
                                                        },
                                                        error => {
                                                                console.log('error', error)
                                                                MyBot.sendMessage(
                                                                        chat_id,
                                                                        `Sorry, can not find weather
for this query.`
                                                                )
                                                        }
                                                )
                                } else {
                                        MyBot.sendMessage(
                                                chat_id,
                                                `Can not check this info.
            \n\rPlease, firstly provide /set [city] command.`
                                        )
                                }
                        })
                        .catch(error => {
                                MyBot.sendMessage(
                                        chat_id,
                                        `Sorry, there is some error:\n\r${error}`
                                )
                        })
        }
    }
```

```
// Обробник повідомлення /week
MyBot.onText(/\/week (.+)/, (msg, match) => {
    const chat_id = msg.chat.id
    const city = match.input.split(' ')[1]
    getInfo(chat_id, null, city, [0, 7])
})

// Обробник повідомлення /now
MyBot.onText(/\/now (.+)/, (msg, match) => {
    const chat_id = msg.chat.id
    const city = match.input.split(' ')[1]
    getInfo(chat_id, null, city, [0, 0])
})

// Обробник повідомлення /now
MyBot.onText(/\/tomorrow (.+)/, (msg, match) => {
    const chat_id = msg.chat.id
    const city = match.input.split(' ')[1]
    getInfo(chat_id, null, city, [1, 1])
})

// Обробник повідомлення /w
MyBot.onText(/\/w/, msg => {
    const chat_id = msg.chat.id
    const user_id = msg.from.id
    getInfo(chat_id, user_id, null, [0, 2])
})

// Обробник повідомлення /location
MyBot.onText(/\/location/, msg => {
    const options = {
        reply_markup: JSON.stringify({
            keyboard: [[{ text: 'Location', request_location: true }]],
            resize_keyboard: true,
            one_time_keyboard: true,
        }),
    }
    MyBot.sendMessage(msg.chat.id, `Send me location using the button`, options)
})

// Обробник для завдання поточного положення location
MyBot.on('location', msg => {
    const chat_id = msg.chat.id
    const { latitude, longitude } = msg.location
    getWeather(chat_id, latitude, longitude, 'en', [0, 0], 'by coordinates')
})
```

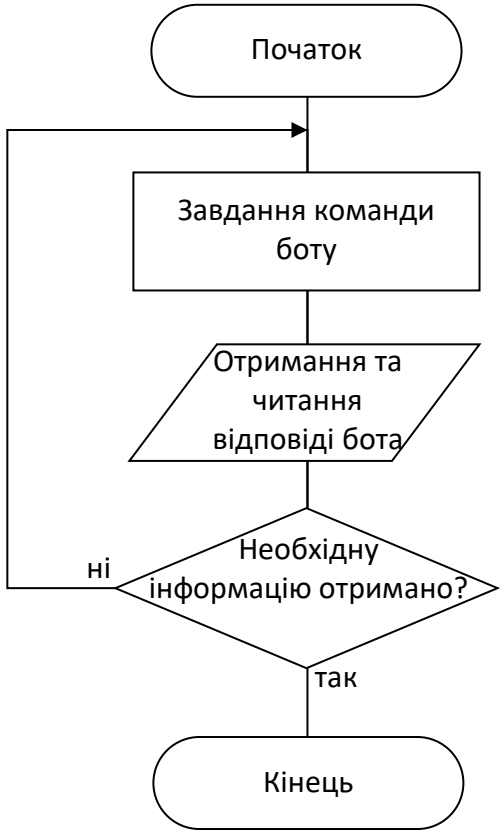
```
// Обробник повідомлення  /help
MyBot.onText(/\/help/, msg => {
    const chat_id = msg.chat.id
    const answer = `
Hi!
Here you can see commands that you can type for this bot:
/now <b>city</b> - get meteo info in the city.
/tomorrow <b>city</b> - get tomorrow meteo info in the city.
/week <b>city</b> - get weekly meteo info in the city.
/lang <b>lang_code</b> - set new language in database (two letters, for example UA, RU, etc)
/set <b>city</b> - set base city in database.
/w - get meteo info for base city for 3 days.
/location - to set actual location using Telegram function (works only for mobile versions).
/help - check commands.
    `
    MyBot.sendMessage(chat_id, answer, { parse_mode: 'HTML' })
})

// Обробник довільного повідомлення
MyBot.on('text', msg => {
    const chat_id = msg.chat.id
    if (!msg.location) {
        if (msg.text.toLowerCase() === 'hi' || msg.text.toLowerCase() === 'hello') {
            const str = `Hello, ${msg.from.first_name}. I'm bot for showing weather information by using [OpenWeatherMap](https://openweathermap.org/) API.\nMy creator is @sylvenis. Also my code is [here](https://github.com/mezgoodle/weather-bot).\nGood luck!😉`
            MyBot.sendMessage(chat_id, str, { parse_mode: 'Markdown' })
        }
        if (msg.text.toLowerCase() === 'bye') {
            MyBot.sendMessage(chat_id, 'Have a nice day, ' + msg.from.first_name)
        }
    }
})

// Оброник помилок
MyBot.on('polling_error', err => console.log(err))
```




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | *ІАЛЦ.467200.004 Д1* | | |
| *Зм.* | *Арк* | *№ докум.* | *Підпис* | *Дата* | | | |
| | | Білич М. | | | *Структурна схема проектованого програмного продукту типу чат-бот для Telegram* | *Літ.* | *Аркуш* | *Аркушів* |
| *Перевірив.* | | *Гордієнко Ю.Г.* | | | | | 1 | 1 |
| *.* | | | | | | *НТУУ "КПІ ім. Ігоря Сікорського" ФІОТ гр. ІО-82* | | |
| *Н. Контр.* | | *Сімоненко В.П.* | | | | | | |
| *Затвердив.* | | | | | | | | |




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | *ІАЛЦ.467200.004 Д2* | | |
| *Зм.* | *Арк* | *№ докум.* | *Підпис* | *Дата* | | | |
| | | Білич М. | | | *Функціональна схема: Діаграма діяльності процесу нечіткого виводу* | *Літ.* | *Аркуш* 1 / *Аркушів* 1 |
| *Перевірив.* | | *Гордієнко Ю.Г.* | | | | | |
| . | | | | | | *НТУУ "КПІ ім. Ігоря Сікорського" ФІОТ гр. ІО-82* | |
| *Н. Контр.* | | *Сімоненко В.П.* | | | | | |
| *Затвердив.* | | | | | | | |




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | *ІАЛЦ.467200.004 Д2* | | |
| *Зм.* | *Арк* | *№ докум.* | *Підпис* | *Дата* | | | |
| | | Білич М. | | | *Функціональна схема: Діаграма діяльності процесу нечіткого виводу* | *Літ.* | *Аркуш* 1 / *Аркушів* 1 |
| *Перевірив.* | | *Гордієнко Ю.Г.* | | | | | |
| . | | | | | | *НТУУ "КПІ ім. Ігоря Сікорського" ФІОТ гр. ІО-82* | |
| *Н. Контр.* | | *Сімоненко В.П.* | | | | | |
| *Затвердив.* | | | | | | | |