СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ
Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування

Пояснювальна записка

до магістерської дипломної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему Інформаційна система створення Telegram-ботів з апріорно визначним функціоналом

Виконав: студент _2_ курсу, групи_ІСТ-21дм_
126 «Інформаційні системи та технології»
(шифр і назва спеціальності)
Рудак М.Ф
(прізвище та ініціали)
Керівник Лифар В.О.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Сєвєродонецьк – 2022 року

СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Навчально-науковий інститут (факультет) інформаційних технологій та електроніки
Кафедра інформаційних технологій та програмування
Освітньо-кваліфікаційний рівень магістр
спеціальність 126 «Інформаційні системи та технології»
(шифр і назва спеціальності)

ЗАТВЕРДЖУЮ
Завідувач кафедри ПМ,
_____д.т.н., доц. Лифар В.О.
(підпис)
«___»_____2022 р.

ЗАВДАННЯ
на магістерську дипломну роботу студенту

_____Рудак Микиті Федоровичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система створення Telegram-ботів з апріорно визначеним функціоналом_____ керівник роботи доцент, д.т.н Лифар Володимир Олексійович_____,
(вчене звання, науковий ступінь, прізвище, ім'я, по батькові)

затверджений наказом університету від «__»_____2022 року №

2. Строк подання студентом роботи 20 листопада 2022 р.
3. Вихідні дані до роботи Матеріали науково-дослідної практики, науково-методична література; дані інтернет-мережі; ._____
_____

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   4.1 Вступ
   4.2 Аналітичний огляд питання (огляд публічних джерел інформації)
   4.3 Основна частина, в якої висвітлити методи, які будуть використовуватися для реалізації проекту.
   4.4 Практична частина – огляд технологій, які використовуються під час реалізації проекту.
   4.4 Висновки
   4.5 Перелік використаних джерел
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 8 жовтня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Одержання завдання на виконання роботи	8.10.2022	
2	Укладання і погодження з керівником плану і етапів виконання роботи	10.10.2022	
3	Узагальнення даних літературних джерел, укладання розділу «Аналіз предметної галузі»	14.10.2022	
4	Аналіз шляхів виконання завдання. Вибір і погодження з керівником оптимального шляху	18.10.2022	
5	Розробка технічного завдання	09.10. 2022	
6	Укладання програмного продукту	25.10. 2022	
7	Укладання, оформлення та погодження пояснювальної записки з керівником	05.11. 2022	
8	Здача готової пояснювальної записки на кафедру	20.11.2022	
9	Укладання доповіді і презентації	25.11.2022	

Студент_____ Рудак М.Ф__
(підпис) (прізвище та ініціали)

Керівник роботи_____ _____Лифар В.О.___
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текст – 53 с., рис. –19, табл. – 3, додатків – 1, літературних джерел – 15

Об’єкт дослідження – Інформаційна система для створення Telegram-ботів з апріорно визначеним функціоналом.

Мета розробки – підвищення зручності створення телеграм-ботів та спрощення розробки.

У ході виконання дипломної роботи були розглянуті актуальність розробки інформаційної системи, існуючі рішення та їх недоліки, висунуті вимоги до розробки системи та серверної частини. Була освітлена клієнт-серверна архітектура, спроектована база даних та проаналізовані технології реалізації серверу. Була розроблена база даних, та сформовані механізми взаємодії з клієнтом.

Кінцевим результатом є розроблений Telegram-бот. Всі поставлені задачі виконані та всі цілі досягнуті.

Ключові слова: Telegram -бот, бот, Python, SQLite, чат-бот, Telegram.

ABSTRACT

Text - 53 p., fig. –19, tab. – 3, appendices – 1, literary sources – 15

The object of research is an information system for creating Telegram bots with a priori defined functionality.

The goal of development is to increase the convenience of creating Telegram bots and simplify development.

In the course of the diploma work, the relevance of the development of the information system, existing solutions and their shortcomings, requirements for the development of the system and the server part were put forward. The client-server architecture was illuminated, the database was designed, and server implementation technologies were analyzed. A database was developed, and mechanisms of interaction with the client were formed.

The final result is a developed Telegram bot. All tasks have been completed and all goals have been achieved.

Keywords: Telegram bot, bot, Python, SQLite, chatbot, Telegram.

ПЕРЕЛІК ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

VS Code - Visual Studio Code

API - Application Programming Interface

CRM - Customer Relationship Management

IVR - Interactive Voice Response

NLP - Natural Language Processing

HTTPS - HyperText Transfer Protocol Secure

HTML5 - Hypertext Markup Language

URL - Uniform Resource Locator

JSON - JavaScript Object Notation

БД –База Даних

ЗМІСТ

ВСТУП

Актуальність досліджень.

В останні роки можна спостерігати тенденцію наскільки глибоко сучасні технології вкоренилися у житті суспільства. На цей час дуже складно знайти людину, яка б не користувалась гаджетами та інтернетом у повсякденному житті.

Наприклад, кожного дня майже всі володіють смартфоном, комп'ютером або безліччю інших пристроїв, які мають можливість підключення до інтернету. Найчастіше за допомогою цих пристроїв люди обмінюються інформацією, використовуючи для цього сервіси комунікації. Також неодмінною частиною нашого життя залишаються додатки та комп'ютерні програми, але від недавнього часу, стрімко набирають популярність чат-боти. Це пов'язано із простотою використання, швидкою розробкою та можливості інтегрувати бота у різні сервіси, наприклад месенджери.

Чат-бот – це віртуальний помічник, який в залежності від свого призначення, створений для комунікації із користувачем за допомогою повідомлень. Месенджери – це програми, які дозволяють передавати повідомлення в режимі реального часу через Інтернет. Еволюція месенджерів привела до того, що нині люди можуть відправляти не лише текстову інформацію, але аудіо і відео повідомлення та файли.

Більшість людей використовують у повсякденному житті Telegram, тому це стало однією з багатьох причин створення дипломного проекту на основі телеграм-бота.

Метою даної роботи: створення телеграм бота на платформі Telegram, який зможе надати доступ користувачам створення власного Telegram-бота зі своїми властивостями та налаштуваннями. Також одним з переваг розробки цього бота є те, що кожен користувач зможе самостійно і без будь-яких проблем створити бота, для різних завдань: для бізнесу, чат підтримка, отримання основної інформації про якийсь продукт і тд..

Відкривши документацію по Telegram Bot API можна дізнатися, що це HTTR інтерфейс, який надає можливість розробникам створювати різних ботів в Telegram. Телеграм-боти мають багато можливостей та функцій, які інколи обмежуються лише уявою розробника.

Навіщо і кому потрібні чат-боти?

Боти дозволяють мінімізувати витрати, пов'язані зі щоденною та однотипною взаємодією з великою кількістю користувачів. Як і в інших сферах бізнесу та виробництва, автоматизація робочого процесу доцільна в тому випадку, якщо завдання та цілі цього процесу можуть бути описані та конкретизовані. Однак було б невірним розглядати чат-ботів виключно у вузькому значенні – як роботів, які відповідають лише на обмежене коло питань. Сучасні алгоритми вміють взаємодіяти з користувачем і у складніших випадках.

Так, наприклад, вони можуть розуміти і відповідати на питання про те саме, але сформульовані по-різному: «де можна отримати товар?», «де найближчий пункт видачі?», «яка адреса магазину?» та ін. Таким чином, чат-боти корисні навіть там, де діалог з клієнтом може бути варіативним і нелінійним. У тих випадках, коли користувач формулює запит у неясній для робота формі, а також коли його алгоритми не дозволяють дати корисну та однозначну відповідь, програма має можливість переключити співрозмовника на реальну людину. Завдяки цьому вдається досягти зниження навантаження на операторів у 3 рази, відтинаючи типові питання, які становлять до 70%, в автоматичному режимі.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Telegram на даний момент має великий функціонал. Один з них — Bot API. Чат боти вже дуже широко поширені, як серед розробників-початківців, так і у бізнесі.

1.1. Переваги ботів телеграму

Телеграм-боти мають деякі особливості на відміну від звичайних користувачів:

- У ботів немає ніякого статусу , або , замість цього відображається мітка.
- Також боти надають користувачам обмежене по пам'яті хмарне зберігання – старіші повідомлення будуть видалені сервером після їх обробки.
- Боти не можуть почати чат з користувачем. Користувач повинен або додати їх до групи, або надіслати їм повідомлення. Також користувачі можуть скористатися t.me/ посиланням, або пошуковим іменем, щоб знайти свого бота.
- Ім’я кожного телеграм-бота завжди закінчується на «bot» . Наприклад: @Trivial_bot, @Github_bot та інші.
- Якщо додати бота до групи – він не буде отримувати усі повідомлення за замовчуванням.
- Боти працюють завжди і можуть відповідати користувачу у будь-якій годині дня і ночі. Телеграм-боти мають декілька незаперечних переваг над мобільними додатками, сайтами і групами в соц. Мережах.
- Персональна увага до кожного користувача. Кожен користувач будь-якого продукту обожнює персональний підхід до себе і готовий платити за це. В світі є мільйони безликих сайтів та додатків на кожному з яких приходиться орієнтуватися. Боти мають можливість працювати над конкретним питанням користувача окремо від інших й швидше переходити к його рішенню.
- Текстовий інтерфейс споживає менше трафіку. Не потрібно чекати завантаження сайту, переходити з однієї сторінки на іншу, а це значить більш

швидкий процес досягнення мети.

- Низька вартість Telegram-боти можуть працювати навіть на звичайному комп’ютері, їм не обов’язково орендувати сервера як сайтам. Боти не потребують певну версію Android або iPhone, їм потрібен лише Telegram на мобільному телефоні або комп’ютері, який є безкоштовним для всіх.
- Гнучкість и швидкість відповіді Телеграм-боти відповідають одразу на команди за запитання, якщо вони на це запрограмовані, легкість переробки та зміни бота дозволяю швидше повернути його до роботи.
- Захист від конкурентів завдяки тому, що бот веде чат с клієнтом один-на-один, а не на відкритому інтернет просторі — це дозволяє в деякій мірі захиститись від посяганні конкурентів на власного клієнта.
- Не потребує установки та авторизації Клієнту не потрібно регіструватися або проходити авторизацію, він починає працювати з ботом с першого повідомлення або команди.
- Зниження людського фактору. Телеграм-бот не забуде зв’язатися з користувачем, моментально відповість на запит та надасть всю потрібну інформацію користувачеві.
- Автоматизація виконання шаблонних справ. Телеграм-боти виконують усю рутинну працю набагато швидше та без помилок, наприклад: складання звітів, зведень, будь-яких актів, генерування документів та їх зберігання.

1.2. Види чат-ботів

Створення чат-боту обмежується тільки уявою та навичками програміста, але можна виділити деякі певні види чат-ботів:

- Чат-бот для сфери послуг
- Отримання інформації про робочий графік, вихідні, вільні місця, тощо
- Записи на прийом, співбесіди, тощо
- Отримання зворотного зв’язку
- Відправка повідомлень
- Прийом оплати
- Інтеграція з програмами лояльності
- Чат-бот для фінансових установ, банків
- Отриманні інформації о найближчим відділенні банкомату або банку
- Інформування о послугах
- Отримання заявок від користувача
- Отримання рішення на популярні запитання
- З’єднання з оператором у разі необхідності
- Погашення кредиту, заборгованості, інтернет банкінг
- Чат-бот для служби підтримки.
- База популярних запитань
- Отримання, обробка та фіксування заявки, звернення, тощо
- Додавання оператору в діалог у разі необхідності
- Відслідковування обробки та статусу заявки користувача
- Чат-бот для служби доставки
- Повне меню закладу, оформлення заказу
- Оплата заказу прямо в боті
- Відправка акцій і спец-пропозицій клієнтам

- Сегментація клієнтів
- Чат-бот для роботи з рекрутинговими агентствами, персоналом
- Первинне анкетування кадрів
- Роботизація HR-процесів
- Швидке надавання інформації персоналу або кадрам
- Повна інформація потрібна персоналу
- Чат-бот для конференцій і форумів
- Інформація про план, графіки початку та кінця заходів
- Інформація для про спікерів
- Чати для спілкування гостей
- Проведення опитування і тестів
- Розсилка інформації та потрібних файлів під час заходів
- Чат-бот для бронювання та купівлі квитків
- Список вільних місць, номерів, інше
- Оплата в боті
- Підключення оператора
- Отримання зворотного зв’язку
- Розповсюдження акцій та спец пропозиції
- Чат-бот для туристичних агентств
- Налаштування параметрів бота для отримання цікавих для клієнта турів
- Розповсюдження акцій
- Зв’язок під час відпочинку
- Отримання відгуків та оцінок про тур
- Індивідуальні чат-боти

У кожного є можливість розробити власного бота с будь-яким функціоналом адаптованим під завдання певної людини. Форматування, зберігання, розсилка, пошук, моніторинг, інше.

1.3. Класифікація чат-ботів

Нині чат-боти набирають великої популярності завдяки широкому використанню в багатьох сферах життєдіяльності людини. Ці віртуальні агенти можуть повідомити погоду, курс валют, записати клієнта на певну дату та навіть бути альтернативою онлайн бібліотеки.

Статистика ринку чат-ботів показує, що одна з причин, чому ця технологія стає все більш і більш популярною, полягає в тому, що чат-боти можуть відповісти на більшість запитань, які можуть поставити їм користувачі [10]. Для складніших питань все ще важливо мати кваліфікованих фахівців служби підтримки клієнтів. Але для щоденних питань служба чат-ботів зменшує витрати та пришвидшує час відгуку. Якщо подивитись на статистику, то можна побачити швидкий ріст популярності Telegram-ботів (рис.1.1)

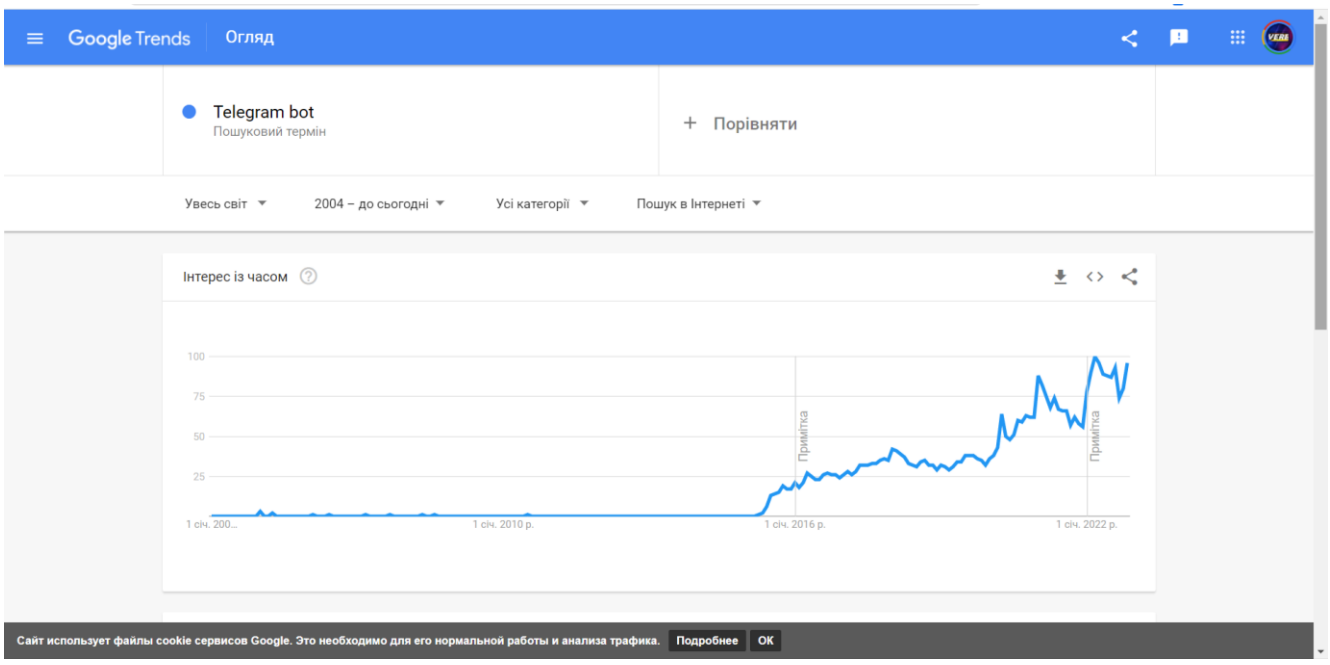

Рис.1.1. Статистика популярності “Telegram-bot” у світі

На цей момент існує велика класифікація чат-ботів: по платформі впровадження, технології розробки, способу спілкування з користувачами і функціональності [12].

Але найпоширенішими залишається поділ з точки зору реалізації:

- Бізнес класифікація яка полягає у сферах застосування чат-ботів
- Технічна класифікація полягає в основі створення чат-боту.

Бізнес класифікація:

- Продажі. Бот допомагає підібрати клієнту відповідний товар, відповідає на питання, розповідає про знижки та підводить клієнта до здійснення покупки.
- Чат-бот асистент. Інтеграція з системами-аналітики, CRM і бухгалтерією дає йому можливість складати звіти, аналізувати дані, заповнювати форми та ін.
- Обслуговування клієнтів. Замовити таксі, забронювати квитки, номер в готелі і столик в ресторані, оформити доставку продуктів - все це можна зробити за допомогою чат-бота.
- Медіа. З їх допомогою користувачі знаходяться в курсі останніх новин або вибирають і читають тільки цікаві для них розділи.
- Особистий помічник. Нагадуватиме про зустрічі, підкаже прогноз погоди та навіть знайде рецепт будь-якої страви.

За технічною класифікацією чат-боти поділяються на прості, розумні з підтримкою штучного інтелекту та гібридні. Простий бот – це бот, який відповідає на запитання на основі заздалегідь встановленого вибору інтегрованих відповідей.

Простих ботів також називаються ботами дерева прийняття рішень, як випливає з назви, вони використовують ряд визначених правил та нагадують текстову систему IVR (система попередньо записаних голосових повідомлень, що виконує функцію маршрутизації дзвінків всередині контакт-центру). Такі боти не

роблять жодних висновків з попередніх взаємодій та найкраще підходять для прямолінійних діалогів.

Простий бот зазвичай будується на базі кнопок, тому він ідеально підходить для опитування, підтримки продажів і практично для будь-якого простого завдання автоматизації процесів, де сценарії спілкування чітко визначені. Натомість розумні чат-боти з підтримкою штучного інтелекту створені для імітації майже людської взаємодії з клієнтами [11].

Для того, щоб вести вільні розмови і розуміти намір, мову та почуття розумні чат-боти використовують технології обробки природних мов (NLP, NLU тощо). Natural Language Processing (NLP) – це область штучного інтелекту, яка дозволяє комп'ютерам аналізувати та розуміти людську мову. Найбільша відмінність від простого чат-бота полягає у використанні моделей машинного навчання, що значно збільшує функціональність бота, оскільки він здатний ідентифікувати сотні різних запитань, написаних людиною. Бот запрограмований на самостійне навчання, оскільки він вводиться в нові діалоги та слова. Фактично, коли чат-бот отримує нові голосові або текстові повідомлення, кількість запитів, на які він може відповісти, і точність кожної відповіді, яку він дає, збільшується.

На відміну від простих чат-ботів, їм не вистачає 12 суворих дерев рішень, які б керували їх роботою. Схема роботи таких ботів зображена на рис. 1.2..

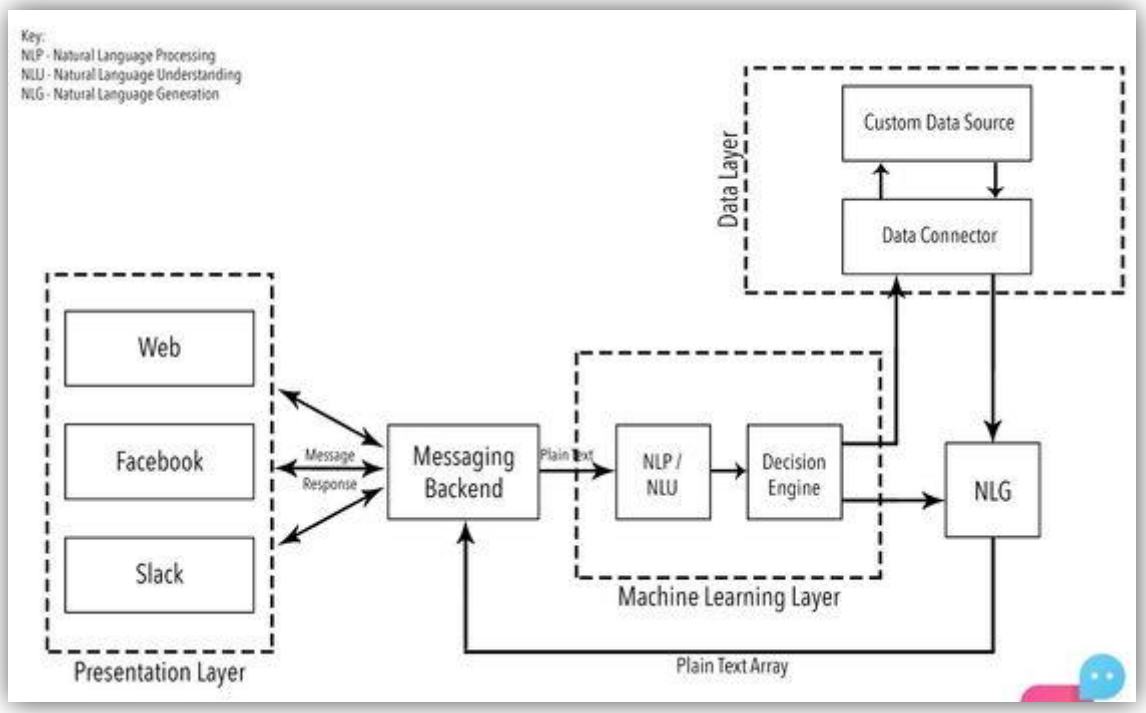

Рис.1.2. Схема роботи чат-бота з підтримкою штучного інтелекту

Взагалі типовий цикл роботи будь-якого чат-бота можна уявити ланцюжком наступних дій :

- Отримання запиту від клієнта
- Розбір запиту – розуміння висловлювання і визначення намірів клієнта в контексті його бізнес-кейсу
- Виконання дій згідно заздалегідь визначеним сценарієм
- Генерація відповіді на природній мові
- Надсилання відповіді клієнту

Найскладнішим етапом роботи є розбір клієнтського запиту. Чат-боти на базі Machine Learning використовують для цього методи NLU і NLP. Наприклад, для текстових чат-ботів процес розбору включає наступні етапи:

- Попередня обробка тексту – розбиття на слова, виправлення помилок, відкидання стоп-слів (артиклі, вигуки, сполучники і ін.), розширення запиту за допомогою словників синонімів

- Класифікація запиту на основі прикладів фраз і алгоритмів або формальних правил (шаблонів)

Останній вид за технічною класифікацією є гібридний чат-бот. Гібридний чат-бот – це поєднання боту побудованого на базі штучного інтелекту та простого. Зазвичай такі боти говорять: «Я бот, який може допомогти вам у розв'язанні таких питань; будь ласка, натисніть відповідну кнопку X, Y, Z або введіть своє запитання у поле нижче ». Особливість цих ботів полягає у тому, що вони використовують і дерева прийняття рішень і технології обробки природних мов

1.4. Чат-бот у контексті Telegram

Чат-бот у контексті Telegram – це сторонній додаток, який працює в екосистемі Telegram. Користувачі взаємодіють з ботами надсилаючи їм повідомлення, команди та inline-запити. Для керування Telegram-ботом існує спеціальне API, яке вимагає використання HTTPS-протоколу.

За допомогою Telegram-боту можливо:

1. Отримання кастомізованих сповіщень та новин.
2. Інтеграція зі сторонніми сервісами: Gmail, Wiki, YouTube, Github тощо.
3. Отримання платежів від користувачів.
4. Створення корисних утиліт: перекладачі з різних мов, для форматування тексту, для нагадування про події тощо.

5. Розробка ігор, що базуються на технології HTML5.

Telegram-бот з точки зору месенджеру – це спеціальний аккаунт, з яким користувачі можуть взаємодіяти двома способами:

1. Відкрити чат з ботом, або ж додати його до групи, після чого надсилати текст або команди через поле вводу повідомлень.

2. Відкрити будь-який чат та вписати "@ім'я бота" в поле вводу повідомлень. Таким чином працюють з inline-ботами.

Варто зазначити, що:

1. Ініціювати спілкування з ботом може тільки користувач, але не навпаки. Це попереджає можливість спаму від недоброчесних ботів.

2. Боти мають обмежене хмарне сховище для повідомлень. Потрібно заздалегідь створити локальне сховище для зберігання застарілих повідомлень, якщо у цьому є необхідність.

3. Ім'я ботів завжди мають закінчення "bot".

Розглянемо Telegram Bot API. Кожен бот потребує авторизації з боку серверів Telegram. При створенні бота генерується спеціальний унікальний токен (далі –) авторизації (приклад – "123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11"), який передається частиною URL при кожному запиті. Усі запити до Telegram Bot API повинні відбуватися з використанням HTTPS протоколу на наступну адресу: "https://api.telegram.org/bot/METHOD_NAME". Підтримуються GET та POST HTTP запити. Існують наступні методи передачі параметрів у запиті:

1. URL query string.

2. Application/x-www-form-urlencoded.

3. Application/json.

4. Multipart/form-data.

URL query string – частина URL, яка містить пари "ключ=значення", де ключ – певний параметр. Структура query string:

URL?param1=1&param2=2&param3=3, де URL – унікальний ідентифікатор ресурсу, ? – символ, що відділяє URL та безпосередньо query string, & –символ, що розділяє декілька параметрів.

Application/x-www-form-urlencoded – тип тіла HTTP запиту, при якому дані запиту передаються у форматі “ключ=значення”. Тип зазначається у заголовка Content-Type запиту.

Зазначені вище методи можуть використовувати спеціальне кодування символів – URL encoding або ж інакше percent encoding. Це кодування використовуються для того, щоб вирішити проблему використання заборонених символів (приклад для URL – пробіл). Механізм кодування – заміна 8-ми бітових заборонених символів іншими, дозволеними символами. Кодування символів представлене на рис. 1.3.

':'	'/'	'?'	'#'	'['	']'	'@'	'!'	'$'	'&'	"'"	'('	')'	'*'	'+'	','	';'	'='	'%'	' '
%3A	%2F	%3F	%23	%5B	%5D	%40	%21	%24	%26	%27	%28	%29	%2A	%2B	%2C	%3B	%3D	%25	%20 or +

рис. 1.3. Кодування символів

Символ пробілу кодується як “%20” при використанні в URL, та як “+” при використанні Application/x-www-form-urlencoded.

Application/json – тип тіла HTTP запиту, при якому дані запиту передаються у форматі JSON. Тип зазначається у заголовку Content-Type запиту.

JSON – текстовий формат обміну даними між комп’ютерами, призначений для опису об’єктів та структур даних. JSON нативно підтримується мовою JavaScript для серіалізації та десеріалізації об’єктів. Підтримується два варіанти структур:

1. Пари “ключ-значення”.

2. Впорядкований масив значень (масив, вектор, послідовність тощо).

Multipart/form-data – тип тіла HTTP запиту, який призначений для передачі файлів у бінарному вигляді. Тип зазначається у заголовку Content-Type запиту.

Детально передача файлів з використанням протоколу HTTP описана у документі стандарту RFC7578 [11].

При запиті до Telegram Bot API у відповідь Telegram надсилає об’єкт JSON, який завжди містить булеве поле “ok” та строку “description”. Якщо результат запиту успішний, то поле “ok” рівне “true”, інакше – “false”. При неуспішності запиту поле “description” містить опис помилки, а також присилається додаткове поле “error_code” з кодом помилки.

Варто зазначити, що усі методи Telegram Bot API не чутливі до регістру, а кодування даних має відбуватись у форматі UTF-8.

Розглянемо функціональні можливості та обмеження Telegram-ботів. При розгляді зазначатимемо короткий опис функціональної можливості, роботу з нею з точки зору Telegram Bot API (використовувані методи API), обмеження за їх наявності.

Першочергово потрібно зазначити, що робота з Telegram Bot API поділена на дві частини: отримання оновлень та виклик методів API. У першому випадку Telergam надсилає повідомлення про конкретні дії користувача з ботом, у другому випадку – бот надсилає дані користувачу.

Отримання оновлення відбувається з отримання об’єкту Update. Під оновленнями розуміються події, що користувачі генерують під час взаємодії з чат-ботом. Основні поля об’єкту Update показані в табл. 1.3.

Таблиця 1.1

Основні поля об’єкту Update

Поле	Тип	Опис
update_id	Цілочисельний	Унікальний ідентифікатор оновлення
message	Message	Необов’язкове. Нове повідомлення – текст, фото, відео або голосове повідомлення тощо.
edited_message	Message	Необов’язкове. Відредаговане

		повідомлення.
callback_query	CallbackQuery	Необов’язкове. Зворотній запит (результат натискання на кнопку клавіатури).

Листування. Основною функціональною можливістю будь-якого месенджеру є можливість листування між його користувачами. Листування відбувається за допомогою текстових, фото, відео або голосових повідомлень тощо. В API для цілей листування існує основний об’єкт – Message, основні поля якого показані в табл. 1.4.

Таблиця 1.2

Основні поля об’єкту Message

Поле	Тип	Опис
1	2	3
message_id	Цілочисельний	Унікальний ідентифікатор поточного чату.
date	Цілочисельний	Час в форматі Unix
chat	Chat	Чат, якому належить цей Message.
edit_date	Цілочисельний	Необов’язкове. Останній час редагування в форматі Unix
entities	Масив MessageEntity	Масив нікнеймів, URL та команд бота, що є у тексті повідомлення.
text	Рядок	Масив нікнеймів, URL та команд бота, що є у тексті повідомлення. повідомлення в форматі UTF-8, 0-4096 символів.
caption	Рядок	Необов’язкове. Текст прикріплений до файлового повідомлення в форматі UTF-8, 0- 1024 символів.
photo	Масив PhotoSize	Необов’язкове. Фото, масив розмірів

		фото
document	Document	Необов’язкове. Звичайний файл
reply_markup	InlineKeyboardMarkup	Необов’язкове. Inline-клавіатура.

Продовження табл. 1.2

1	2	3
Chat		
id	Цілочисельний	Унікальний ідентифікатор чату.
type	Рядок	Тип чату: приватний, група, супер група, канал.
username	Рядок	Необов’язкове. Нікнейм користувача.
first_name	Рядок	Необов’язкове. Ім’я користувача
last_name	Рядок	Необов’язкове. Прізвище користувача.

Клавіатури. Додатково до будь-якого повідомлення Telegram дає можливість прикріпити клавіатуру з довільними кнопками. Це значно розширює користувацький досвід та надає можливість створювати у боті меню, відкривати посилання на сторонні веб-ресурси, проводити опитування тощо. Хорошою практикою вважається оновлення повідомлень при використанні клавіатур замість надсилання нових.

Бот має змогу прикріпити клавіатуру до об’єкту Message зазначивши поле reply_markup. Основні поля InlineKeyboardMarkup показані в табл. 1.5.

Таблиця 1.3

Поле	Тип	Опис
1	2	3
inline_keyboard	Масив з масиву InlineKeyboardButton	Масив рядів кнопок.
InlineKeyboardButton		

text	Рядок	Текст, що відображається на кнопці.
url	Рядок	Необов’язкове. Посилання.

Продовження табл. 1.3

1	2	3
callback_data	Рядок	Дані, що будуть надіслані разом з зворотнім запитом (CallbackQuery).
CallbackQuery		
id	Рядок	Унікальний ідентифікатор запиту.
from	User	Чат, з якого прийшов цей запит.
message	Message	Оригінальний Message, що пов'язаний з цим запитом.
data	Рядок	Дані, що були асоційовані з кнопкою.

При введені символу “/” користувачу буде запропонований перелік доступних команд боту. Будь-яка надіслана команда підсвічується та може бути повторно надіслана за допомогою натискання на неї. Стандартні команди, що має підтримувати будь-який бот: “/start”, “/help”, “/settings”. При цьому команда “/start” є обов’язковою, оскільки вона використовується при ініціюванні діалогу користувачем з ботом. Команди та відповідні дії на них (API) формує розробник.

Висновки до розділу

Отже, провівши аналіз даних першого розділу, виявлено, що чат-боти мають певну класифікацію в залежності від цілей використання. Оскільки ці програмні додатки можна інтегрувати у будь-який вебсервіс, розглянуто найпопулярніші месенджери. Адже на основі статистичних даних виявлено, що частка мобільного трафіку значно зросла, у порівнянні з іншими технологіями, а найчастіше суспільство використовує месенджери та соціальні мережі. В Україні ж перше місце займає Telegram завдяки великій кількості переваг. Узявши до уваги період популярності месенджера та ріст популярності чат-ботів, виявилось що це відбулось в один проміжок часу. Це сталось завдяки тому що Телеграм відкрив платформу для створення ботів які відгукуються на команди користувачів і дають можливість взаємодії із зовнішніми сервісами. Тому на основі аналізу отриманих результатів було вирішено створити чат-бота саме на базі месенджеру Телеграм.

РОЗДІЛ 2. ВИБІР ЗАСОБІВ РЕАЛІЗАЦІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

На початку розробки будь-якої системи або програмного забезпечення кожен розробник аналізуючи висунуті вимоги до проекту має обрати стек технологій та засобів для реалізації продукту.

На цей вибір можуть впливати багато факторів, але з них можна виділити деякі основні, а саме:

– насамперед це цілі та вимоги до самої системи

– операційні системи, які повинні підтримувати програму та бути з нею сумісними, швидкість роботи, надійність, масштабованість, тощо. Розробник аналізує їх та виділяє технології, за допомогою яких можливо досягти потрібного результату;

– не менш важливий фактор це вміння, навички та досвід роботи розробника з технологіями виділеними з попереднього пункту. Зрозуміло, що при рівних умовах завжди надають перевагу технологіями в яких є більший досвід, тому що це впливає на швидкість та якість створення програмного забезпечення;

– також до уваги потрібно брати актуальність використання технології в момент розробки продукту та враховувати її перспективи в майбутньому, тому що використання застарілих технології може привести до нестабільної роботи системи через декілька років, або до важкої підтримки та розробки нових функцій продукту.

В залежності від складності та розмірів системи, вона може поєднувати в собі декілька технологій з однієї категорії. Наприклад серверна частина може бути створена у вигляді мікросервісної архітектури, тобто мати декілька серверів, які виконують різні функції та написані різними мовами програмування з використанням різних технології зв’язані в одну велику систему, але при цьому можуть бути навіть незалежними один від одного. Цей підхід до розробки стає доволі популярним, оскільки дозволяє різним розробникам одночасно працювати

над окремими модулями не заважаючи один одному, що значно прискорює розробку.

2.1. Аналіз вимог до системи

Перед створенням та проектуванням системи потрібно проаналізувати та визначити основні вимоги до розроблюваної системи. Зазвичай ці вимоги класифікують як функціональні та нефункціональні.

Функціональні вимоги визначають та описують те, що система повинна робити та які функції надавати користувачам. Проаналізувавши предметну область обраної теми, були визначені наступні функціональні вимоги, описані у наступних підрозділах.

2.1.1. Реєстрація

Реєстрація у будь-якій системі це важлива та необхідна вимога. Вона повинна відбуватись зручно та швидко для всіх користувачів. В боті реєстрація проходить
коли користувач натискає кнопку старт. Після натискання система записує користувача до бази даних для подальшої індефікації.

2.1.2. Доступні команди

Команди для взаємодії ботом повинні відповідати функціоналу та бути доступними для ознайомлення. Приклад доступних команд:

- /start - початкове привітання при приєднанні до чат-боту.
- /addbot - створення нового бота.
- /newpost - надсилає нове повідомлення всім підписникам.
- /setdescription - змінює опис бота.
- /deletebot - видаляє бота.

- /help - викликає додаткову інформацію, для вирішення будь-яких питань.
- /cancel - закриває поточну операцію.

2.1.3. Розсилка повідомлень

Система повинна спрощувати взаємодію та спілкування між адміністратором та користувачем.

Для покращення комунікації сторін, бот повинен надавати змогу розсилки, після чого одночасно отримають повідомлення в чат-боті.

2.1.4. Нефункціональні вимоги

Для повного опису системи недостатньо лише функціональних вимог, тому потрібно брати до уваги нефункціональні вимоги, які глобально поділяються на наступні категорії:

– практичність. Відповідає за те, щоб програмне забезпечення було простим та легким у використанні для будь-якого користувача, не завдавало зайвих труднощів та було інтуїтивно зрозумілим;

– надійність. Відповідає за роботу програмного забезпечення, та описує як система повинна себе поводити у разі помилок чи збоїв в роботі та зберегти важливі дані навіть у цьому випадку;

– продуктивність. Визначає характеристики системи під час взаємодії з користувачами, тобто наступні характеристики - відповідає за оптимальний час на відповідь запита користувача, кількість користувачів, яку одночасно може прийняти система та визначає які системні ресурси їй на це знадобляться;

– можливість обслуговування. Означає можливість легко модифікувати, змінювати програмне забезпечення для введення нового функціоналу або з метою виправлення помилок. Виходячи з цього можна визначити необхідні нефункціональні вимоги до системи:

– система повинна бути надійною та доступною для використання користувачами в будь-який момент часу;

– в разі збоїв в роботі програмне забезпечення повинно адекватно реагувати на це та намагатись самостійно відновити роботу;

– в разі помилок перш за все важливе збереження всієї інформації в базі даних;

– система має бути побудована таким чином, щоб в майбутньому для неї було легко створювати нові модулі та функції або виправляти код з метою покращення;

– система повинна бути кросплатформеною, тобто працювати на різних операційних системах.

2.2. Огляд засобів створення чат-бота

Оскільки чат-боти набрали вже досить великої популярності, це призвело до
того, що з’явилось безліч рішень для їх створення. Створити чат-бота можна двома
способами:

- за допомогою мови програмування;
- за допомогою конструктора на сервісі

Загалом чат-бот, написаний певною мовою програмування, являється серверним додатком, в якому функції чату працюють через власний API. Для того,
щоб створити такого бота необхідна певна інфраструктура: хостинг, сервер (фізичний або хмарний) та база даних. Можливості таких чат-ботів обмежуються лише можливостями платформи, на яку вони інтегруються.

Натомість чат-бот, який створюється власноруч, за допомогою конструктора,

обмежується особливостям сервісу на якому він створюється. Перед тим як створити чат-бота необхідно зрозуміти як саме відбувається
взаємодія між користувачем та сервером. Повідомлення, команди і запити, надіслані користувачами, передаються на
програмне забезпечення, запущене на серверах розробників. Сервер Telegram, який являється посередницьким та анонімним, обробляє
шифрування і здійснює зворотний зв'язок між користувачем і утилітою. Взаємодія між користувачем і ботом виглядає наступним чином:

1. Користувач надсилає боту команду
2. Бот передає команду на сервер
3. Програма на сервері оброблює отриманий від бота запит
4. Сервер віддає відповідь боту
5. Бот виводить відповідь користувачеві на вікні керування.

І цей цикл повторюється раз за разом під час взаємодії з будь-яким Telegram-ботом. Спілкування відбувається за допомогою простого HTTPS-інтерфейсу, який є спрощеною версією API Telegram. Інакше цей інтерфейс можна назвати програмним каталогом або бот-алгоритмом.

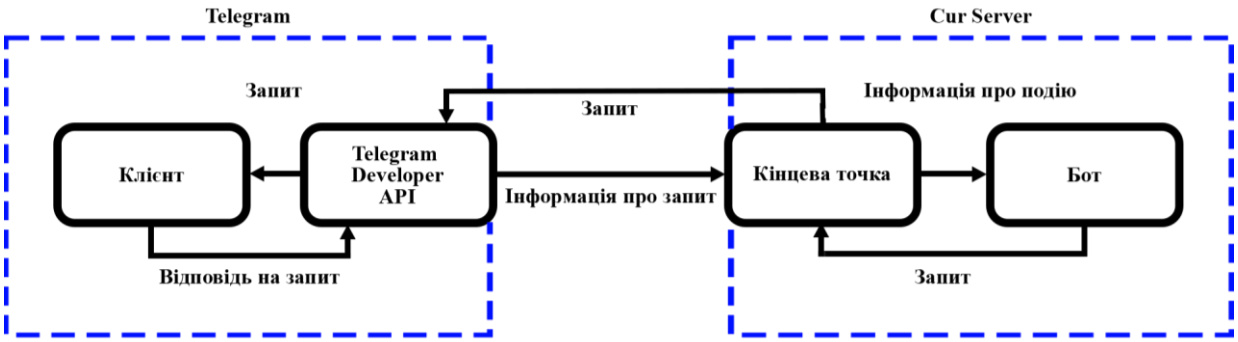

Рис. 2.1 Схема обміну даними між користувачем і Telegram-ботом

2.3. Telegram Bot API

Для месенджера Telegram був створений протокол MTProto, що передбачає використання декількох протоколів шифрування [8]. Протокол підрозділяється на три практично незалежні компоненти:

- Мова API запитів – це компонент високого рівня. З його використанням визначається метод для перетворення запитів та відповідей у двійкові повідомлення.
- Криптографічний рівень. З його використанням визначається метод для шифрування повідомлень перед передачею через транспортний протокол.
- Транспортний рівень. З його використанням визначається спосіб передачі між клієнтом та сервером повідомлень, за допомогою наявних мережевих протоколів (HTTP, HTTPS, WS, WSS, TCP, UDP). Bot API - це інтерфейс на основі HTTP для створення ботів в Telegram [10]. Згідно з документацією Telegram Bot API існує два способи отримання оновлень, що відрізняються один від одного:
- За допомогою запитів
- За допомогою вебхуків

Під оновленням мається на увазі дія, що була виконана над ботом – наприклад, отримання запиту від користувача. Вхідні оновлення зберігаються до того моменту, поки сервер не обробить їх, але не більш ніж 24 години.

Спочатку веб-додатки розроблялись навколо моделі клієнт / сервер, де вебклієнт завжди є ініціатором транзакцій, вимагаючи дані від сервера. Таким чином, не було механізму, щоб сервер самостійно надсилав дані клієнту, не отримуючи до цього запиту. Щоб подолати цей недолік, розробники веб-програм можуть застосувати техніку, яка називається HTTP Long Pooling, коли клієнт опитує сервер, що вимагає нової інформації. Сервер тримає запит відкритим, поки не з’являться нові дані.

Отримавши доступ, сервер відповідає та надсилає нову інформацію. Коли клієнт отримує нову інформацію, він негайно надсилає інший запит, і операція повторюється. Вебхуки працюють зовсім інакше. Якщо в чат приходить повідомлення, то месенджер Telegram сам говорить про це. Саме в цьому і

полягає робота вебхука.

У зв’язку з цим, відпадає необхідність періодично опитувати, тим самим, зникає причина помилок пошукових ботів. Однак для використання такої можливості, необхідно платити для установки повноцінного вебсервера. Telegram Bot API надає декілька доступних методів для роботи:

- getUpdates – метод використовується для отримування оновлення за допомогою Long Pooling;
- setWebhook – метод, що необхідний для задання URL вебхука, на який бот буде відправляти оновлення;
- getWebhookInfo – метод збирає інформацію щодо того чи іншого вебхука;
- sendMessage – метод використовується для відправлення текстових повідомлень.

2.4. Шляхи оновлення бота: Webhook та getUpdates

На даний момент в Телеграм є два способи оновлення телеграм-боту[17]:

- Long Polling.
- setWebhook.

Коли розробник пише telegram бота і запускає його без вебхуков, то наш бот (який крутиться на нашому сервері / комп'ютері) постійно звертається до серверів телеграма(рис. 2.2) і питає чи є для нього інформація(Long Polling). Іноді сервера телеграма не відповідають і наш бот перестає працювати. Звичайно при такому розкладі буде періодично простій бота в 2-15 сек і наш бот буде постійно звертатися до серверів телеграма, створюючи на них навантаження.

Для збільшення швидкості роботи бота, відмовостійкості і прибирання створеного навантаження використовують вебхук. При роботі бота на вебхуках сервер телеграма сам звертається до нашого боту коли є нові дані. Власне і на наш

сервер навантаження буде менше, адже бот частіше буде простоювати, що розглянемо як безсумнівний плюс(рис. 2.3).

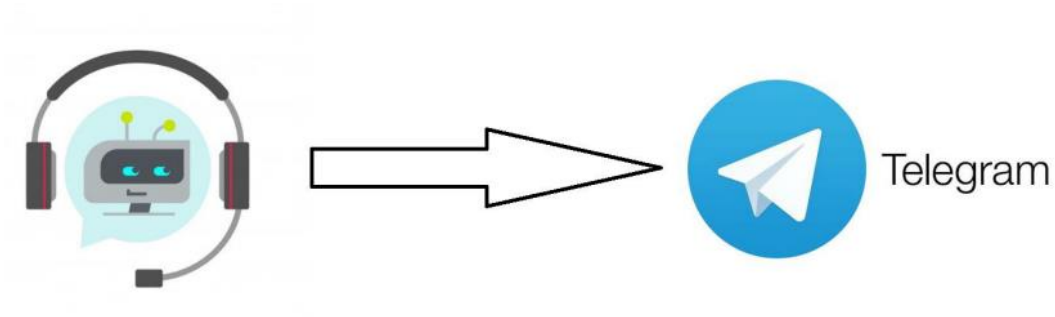

рис. 2.2 Long Polling

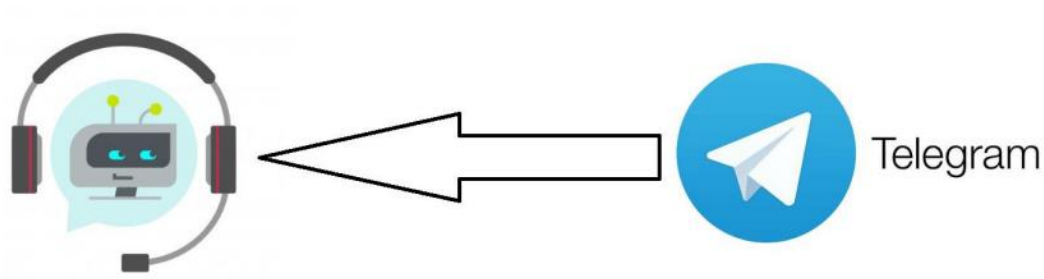

рис. 2.3 setWebhook

Обидва способи одночасно (вебхук і поллінг) використовуватися не можуть, якщо встановлений вебхук, то відповідати на запити сервери телеграм НЕ будуть.

Клієнту при використанні поллінгу замість вебхуків не потрібена зворотня адреса (хостинг і доменне ім'я), тобто можна хоч на домашньому комп'ютері Telegram-бота підняти. Є і недоліки, але додаткові складності, - апдейти тепер будуть прилітати не по одному, а масивами, відповідно, спочатку потрібно буде розібрати отримані дані на окремі апдейти, а потім вже обробляти їх аналогічно тому, як це робиться з вебхуками. API telegram для роботи через поллінг.

Отже, для отримання масиву апдейтів потрібно скористатися методом getUpdates і відправити на сервера телеграм запит наступного виду: https://api.telegram.org/botYOUR_BOT_TOKEN/getUpdates[?options], де

YOUR_BOT_TOKEN - токен вашого бота, options - список додаткових опцій. Опції можуть бути наступними:

- offset - ідентифікатор першого запитуваного апдейта. З цією функцією будуть отримані всі апдейти, починаючи з апдейта, який з зазначеним в поле offset ідентифікатором. Тобто, для того, щоб отримувати всі апдейти – розробник повинен кожен раз в запиті вказувати ідентифікатор на одиницю більше, ніж ідентифікатор (update_id) останнього отриманого апдейта. Всі апдейти з ідентифікаторами менше отриманого значення offset «забуваються»;
- limit - максимальна кількість апдейтів, яке буде отримано за один раз. Може приймати значення від 1 до 100 (за замовчуванням 100);
- timeout - час очікування для «довгого» запиту. За замовчуванням = 0 (тобто за замовчуванням обирається звичайне опитування);
- allowed_updates - тип повідомлень, які розробники хотіли б отримувати (наприклад, типу message). Ця опція додана недавно, тому інколи не вдається домогтися її заявленої роботи (вибрані типи повідомлень приходять не від усіх користувачів).

2.5. Мова програмування Python

В залежності від необхідної функціональності, створення чат-бота в Telegram досить просто виконується за допомогою використання однієї з серверних мов програмувань: Ruby, Python, PHP, Node.JS.

Серед цих мов, що дозволяють швидко і безпроблемно виконати створення бота, Python є одним з найбільш популярних рішень [11]. В основному це пов'язано з широкими можливостями, доступними як при використанні стандартних бібліотек, так і з застосуванням вже готових варіантів, таких як TelegramBotAPI, розрахованих на роботу безпосередньо з Telegram. Python - це мова програмування яка допомагає підвищити продуктивність розробника, а не коду, який він пише. Шляхом простоти коду, подальший супровід програм,

написаних на Python, стає легше і приємніше в порівнянні з Java або C++. Ця мова програмування являється універсальною завдяки багатій 24 стандартній бібліотеці, тому широко використовується у різних областях: веброзробці, машинному навчанні, розробці ігор, комп'ютерній безпеці тощо. Інтерпретатор Python практично був реалізований на всіх платформах і операційних системах, що є безперечною перевагою. Першою такою мовою була C, однак типи даних цієї мови на різних машинах займають різну кількість пам'яті, а це представляє деяку перешкоду при написанні великих програм.

Також, велике значення надається розширюваності мови. Як казав сам розробник Python, кожен програміст може вдосконалювати цю мову й ділитися своїми напрацюваннями з іншими. Інтерпретатор був розроблений мовою програмування C і вихідний код доступний для будь-яких змін. Його з легкістю можна додати у свою програму і використовувати як вбудовану оболонку. Наступна перевага – наявність великого числа модулів, що забезпечують додаткові можливості мови. Ці модулі пишуться на Python, або мовою C і зазвичай створюються більш досвідченими програмістами. Широковживані вважаються такі модулі:

- Numerical Python – цей модуль надає розширені математичні можливості, такі як маніпуляції з цілими векторами і матрицями;
- Tkinter – модуль, за допомогою якого будуються додатки із використанням графічного інтерфейсу користувача (GUI);
- OpenGL – модуль, завдяки якому отримується доступ до використання великої бібліотеки графічного моделювання двовимірних і тривимірних об'єктів Open Graphics Library;
- Pygame — набір крос-платформових модулів для Python, призначених для створення відеоігор. Містить у собі бібліотеки комп'ютерної графіки і звуку.

Одним з недоліків мови є невелика швидкість виконування програм. Але це сповна компенсується плюсами мови, які були описані раніше.

- PyCharm – найпопулярніше середовище розробки, яке використовується спеціально для мови Python та являється кросплатформним. Підтримується на різних операційних системах таких як: MacOS, Linux, Windows. Використання PyCharm допомагає провести аналіз написаного коду, виділити синтаксис та 25 отримати повідомлення про помилки інтерпретації. В цьому середовищі розробки спрощена система навігації по проєктах та файлах, що забезпечує швидкий перегляд та перехід між методами, класами та місцями, де вони використовуються або викликаються.

2.6. Вибір СУБД (SQLite)

Назва SQLite говорить сама за себе – це «легковажна» реалізація бази даних SQL. На відміну від PostgreSQL, MySQL та багатьох комерційних баз даних, таких як Oracle або MS SQL, база даних SQLite не є самостійним сервером, це просто бібліотека, що реалізує інтерфейс доступу до дискових файлів бази даних. Як і інші «легковажні» реалізації типово складних служб.

Перевага SQLite:

1) простота в налаштуванні та використанні;

2) низьке навантаження на сервер.

Недоліки SQLite:

1) недостатня функціональність;

2) невисока продуктивність під час роботи з великими обсягами даних.

Внаслідок цього SQLite чудово підходить для швидкого запуску процесу розробки або для маленьких сайтів, де немає необхідності встановлювати повноцінний веб-сервер баз даних. Однак після етапу початкового навчання, а також для більш серйозних випадків розгортання необхідно буде придбати щось суттєвіше.

Щоб мати змогу взаємодіяти з базою даних SQLite, потрібна бібліотека для мови Python. Використовуючи Python 3.11, у нашому розпорядженні є вбудований модуль sqlite3.

На відміну від серверів баз даних SQLite не вимагає явного створення бази даних або механізму керування користувачами. Натомість просто вибирається каталог у файловій системі для бази даних (один з тих, що доступний веб-серверу для читання та запису) та його ім'я записується у вигляді параметра DATABASES у файлі settings.py

Висновок до розділу

В розділі були розглянуті та описані технології, які були обрані для реалізації системи. Мовою програмування було обрано Python, як найбільш зручна та гнучка мова програмування. Для бази даних було вирішено використовувати sqlite через її гнучкість та надійність. Вибір обраних технологій був аргументований перевагами кожної із них.

РОЗДІЛ 3. ОПИС РОЗРОБКИ ЧАТ-БОТА

При створенні даного дипломного проекту була використана певна низка бібліотек Python, головними з яких є:

- Aiogram
- pyTelegramBot
- telebot
- telegram

2.1 Створення телеграм-боту. Опис використаних функцій

Перед тим, як почати роботу з телеграм ботом – треба його створити. Створювати бота в телеграм можна лише у іншого бота, а саме у @BotFather(рис. 3.1).

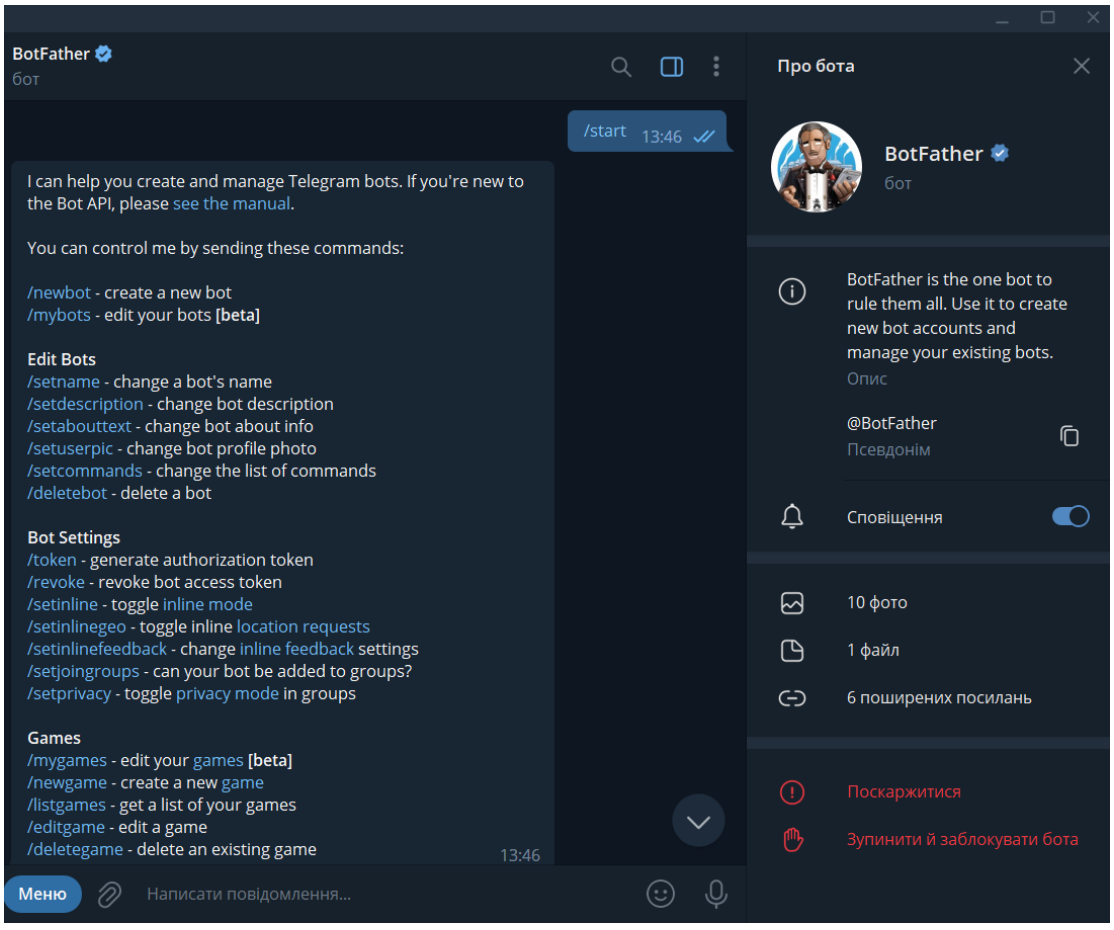

рис. 3.1 Бот для створення ботів – @BotFather

Вводимо команду /newbot і за допомогою зручного меню створюємо телеграм-бота, отримуючи найважливіше – його токен(рис 3.2).

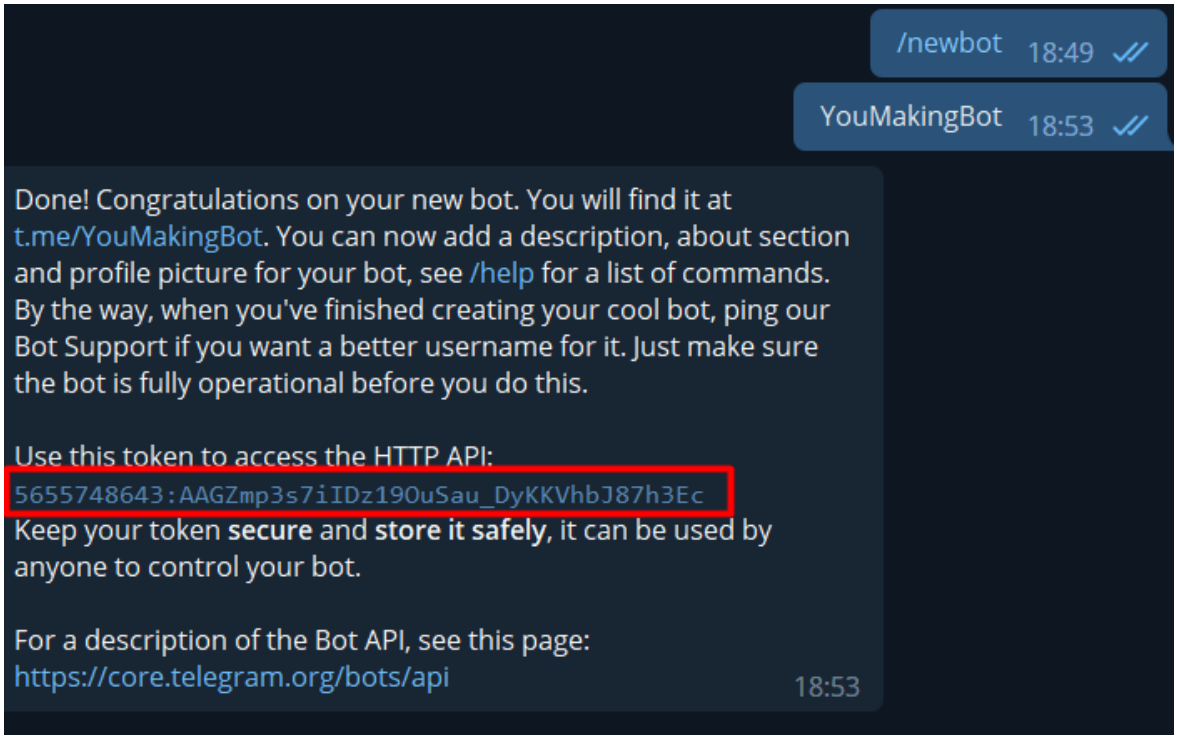

рис. 3.2 Отримуємо токен від @BotFather

Токен являє собою адресу, по якій можна програмувати телеграм-бота.

Токен – це приватна інформація, так як кожний, хто має токен стороннього боту – може його змінювати. Але у разі його загублення завжди можна спитати @BotFather токен одного з ваших ботів. Також у налаштуваннях @BotFather розробник може змінити:

- Ім’я боту
- Його тег
- Його картинку на заставці
- Описання боту та його інфо
- Команди боту
- А також отримати новий токен, або знайти загублений

Елементи зовнішнього оформлення бота виступають:

- Опис, що демонструється користувачеві при першому знайомстві з ботом
- Перше повідомлення, яке користувач отримує автоматично на початку роботи з ботом
- Зображення, яке використовується для спрощення пошуку бота серед листування.
- Ім'я бота

В описі вказані основні можливості, назва та принцип роботи робота.

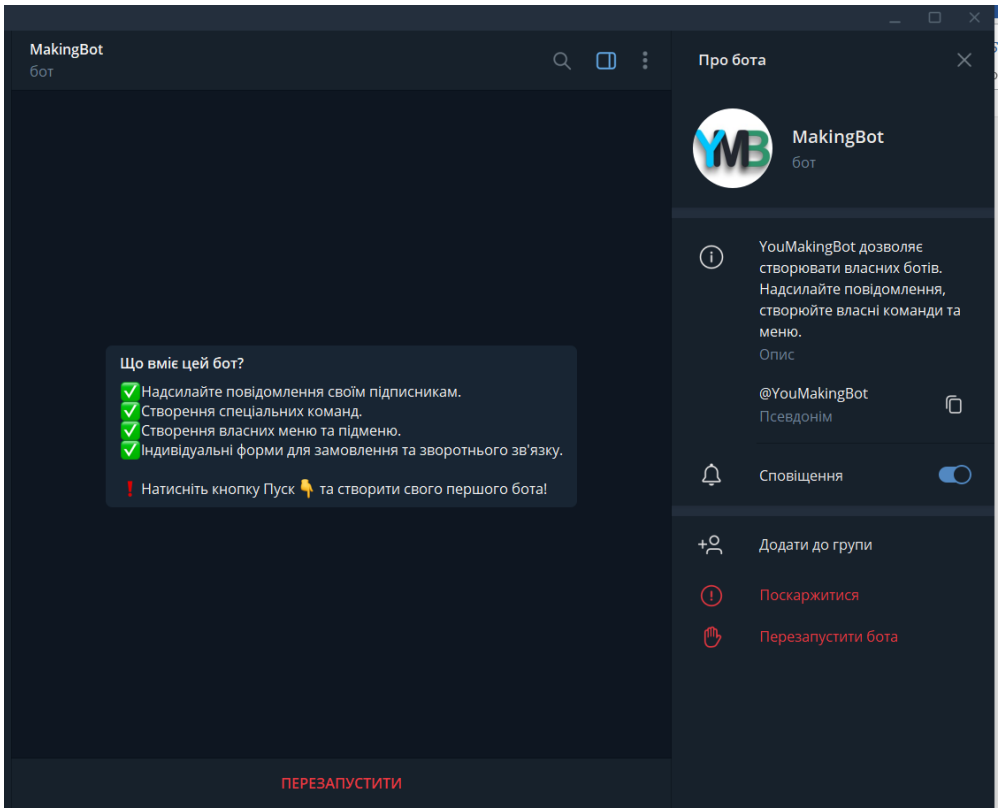

рис. 3.3 Відформатований зовнішній вигляд Telegram-бота

2.2 Розробка застосунку

Отримавши токен бота розробник вже може починати його програмування та налаштування на будь-які цілі. Перше, що треба налаштувати в боті – це команду /start (рис. 4.3)..

```
import telebot
bot = telebot.TeleBot('5655748643:AAGZmp3s7iIDz19OuSau_DyKKVhbJ87h3Ec')


@bot.message_handler(commands=['start'])
def start(message):
    mess = f'Привет, {message.from_user.first_name}!\n \nYouMakingBot допоможе вам створити свій бот.
    bot.send_message (message.chat.id, mess, parse_mode='html')
```

рис. 3.4 Налаштування команди /start

Для початку треба розробити стандартне меню для користувача інтерфейсу. За допомогою параметру reply_markup можна виводити меню кнопок для користувача (рис. 3.4):

```
@bot.message_handler(commands=['start'])
def start(message):

    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    start1 = types.KeyboardButton('Добавити бота')
    start2 = types.KeyboardButton('Допомога')
    start3 = types.KeyboardButton('Інструкція')
    markup.add(start1)
    markup.add(start2, start3)

    mess = f'Привет, {message.from_user.first_name}!\n \nYouMakingBot допоможе вам ст
    bot.send_message (message.chat.id, mess, parse_mode='html', reply_markup=markup)
```

рис. 3.5 Налаштування команди /start

Кожна змінна startX – це створена кнопка, їх потрібно три. За допомогою параметра resize_keyboard=True робимо розмір кнопок трохи менший . Тобто меню буде головним та завжди можна буде користувачу з нього почати. Об’єкт KeyboardButton у даній реалізації використовувався для створення основних функціональних кнопок.

За допомогою методу .add(,) у телеграм-боті формується стовпчики та строки таблиці кнопок(рис. 3.6):

```
markup.add(start1)
markup.add(start2, start3)
```

рис. 3.6 Формування таблиці з кнопок в телеграм-боті

Виводити цю таблицю кнопок допомогає команда bot.send_message().

Використовуючи хендлер @bot.message_handler(commands=[‘start’]) задаємо боту реакцію на команду /start. Таким же чином можна задати реакцію на будь-яку команду.

Запустивши виконання коду, запускаємо телеграм-бота. Кожен Telegram-бот завжди в початку роботи з користувачем пропонує натиснути START – це і буде команда /start та висвітить меню телеграм-бота з текстом привітання(рис.3.7):

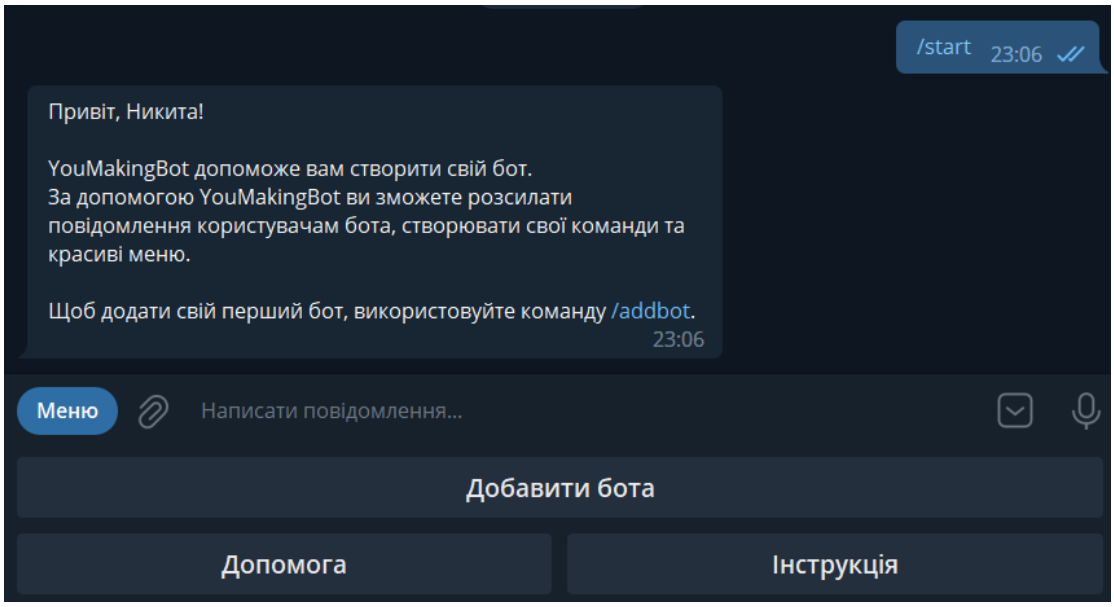

рис. 3.7 Отримання меню кнопок від телеграм-боту

На рис. 3.7 можна побачити функціональні кнопки при воді команди /start. Після цього користувач бачить доступні кнопки. Це три функціональні кнопки, одна з яких пропонує користувачеві створити бота. Інші кнопки призначені для допомоги та ознайомлення з ботом.

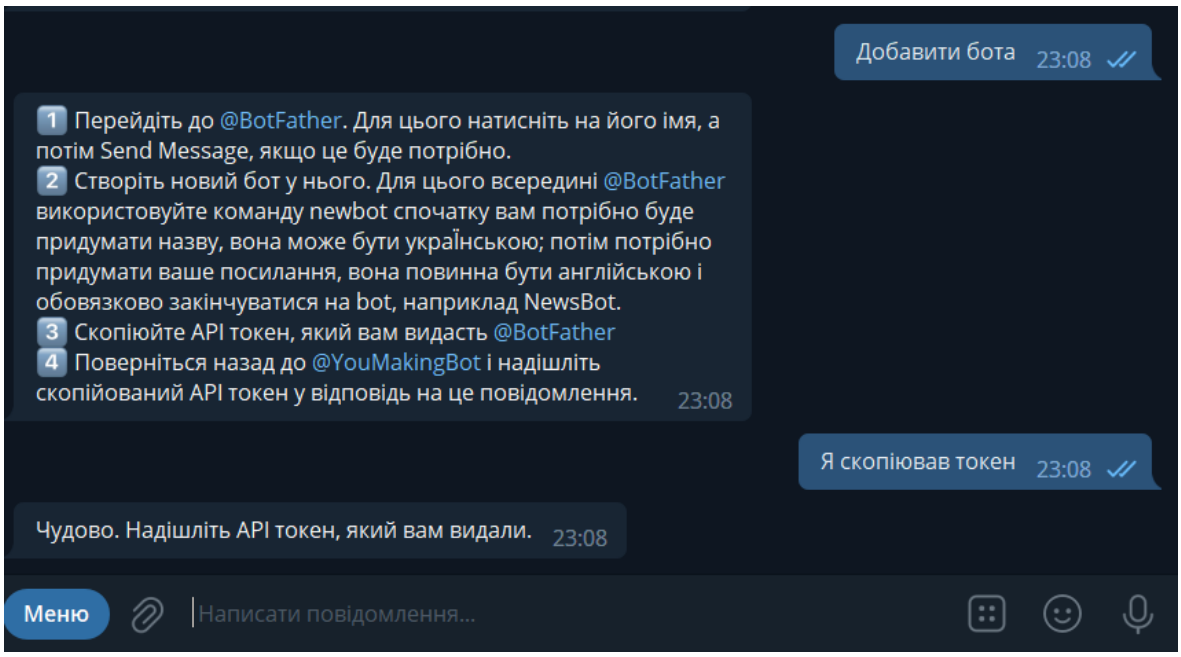

рис. 3.7 Отримання токену від користувача

Для обробки повідомлень використовується функція message handler. Після прийому повідомлення від Telegram його необхідно опрацювати в залежності від того, що це за повідомлення: текст, файл, посилання, тощо. Це можливо реалізувати якщо використовувати безліч конструкцій if-then-else, але такий підхід складний і дозволяє швидко заплутатися.

Для вирішення цієї проблеми автор бібліотеки pyTelegramBotAPI реалізував механізм handler, який використовує 41 декоратори. Тобто у handler описується, в якому випадку необхідно виконувати ту чи іншу функцію.

```python
@dp.message_handler(state="add_bot", content_types="text", regexp="^[^/].+")
async def bot_added(message: types.Message, state: FSMContext):
    """
    Користувач додає бота і ми чекаємо від нього токен
    """
    token = re.findall(token_pattern, message.text)

    async def on_invalid_token():
        await message.answer(dedent(_("""
       Це не токен бота.

        Токен виглядає ось так: 123456789:AAAA-abc123_AbcdEFghijKLMnopqrstu12
        """)))

    async def on_dummy_token():
        await message.answer(dedent(_("""
        Не вдалося запустити цього бота: неправильний токен
        """)))

    async def on_unknown_error():
        await message.answer(dedent(_("""
        Не вдалося запустити цього бота: непередбачена помилка
        """)))

    async def on_duplication_bot():
        await message.answer(dedent(_("""
        Такий бот вже є у базі даних
        """)))

    if not token:
        return await on_invalid_token()
```

рис. 3.8 Обробка невірного токена

На (рис. 3.8) надано приклад коду, якщо користувач введе неправильний токен, і повідомить про неполадку при додаванні

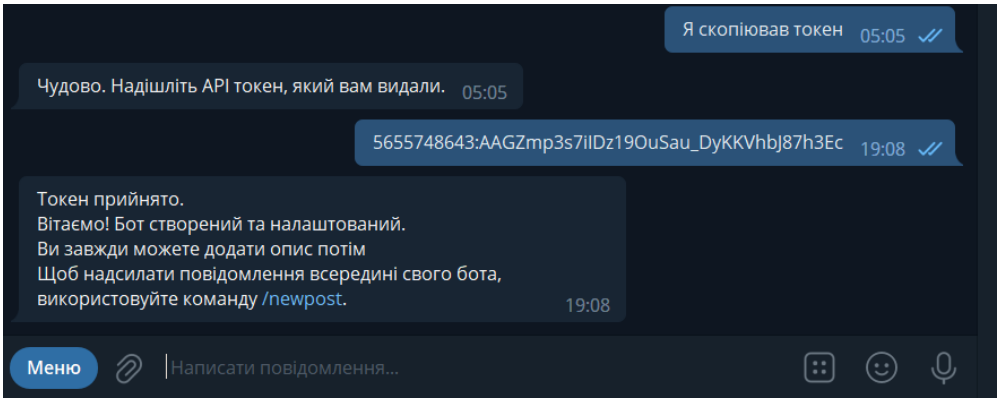

рис. 3.9 Повідомлення чату при додаванні токена

Після додавання токена, бот створює налаштування користувача(рис3.8), та пропонує перейти в бота для надсилання повідомлення у своєму боті. Після цього повідомлення налаштування завершено, і користувач може приступити до додавання команд, створення меню, додавання опитування, також він зможе переглянути кількість його підписників.

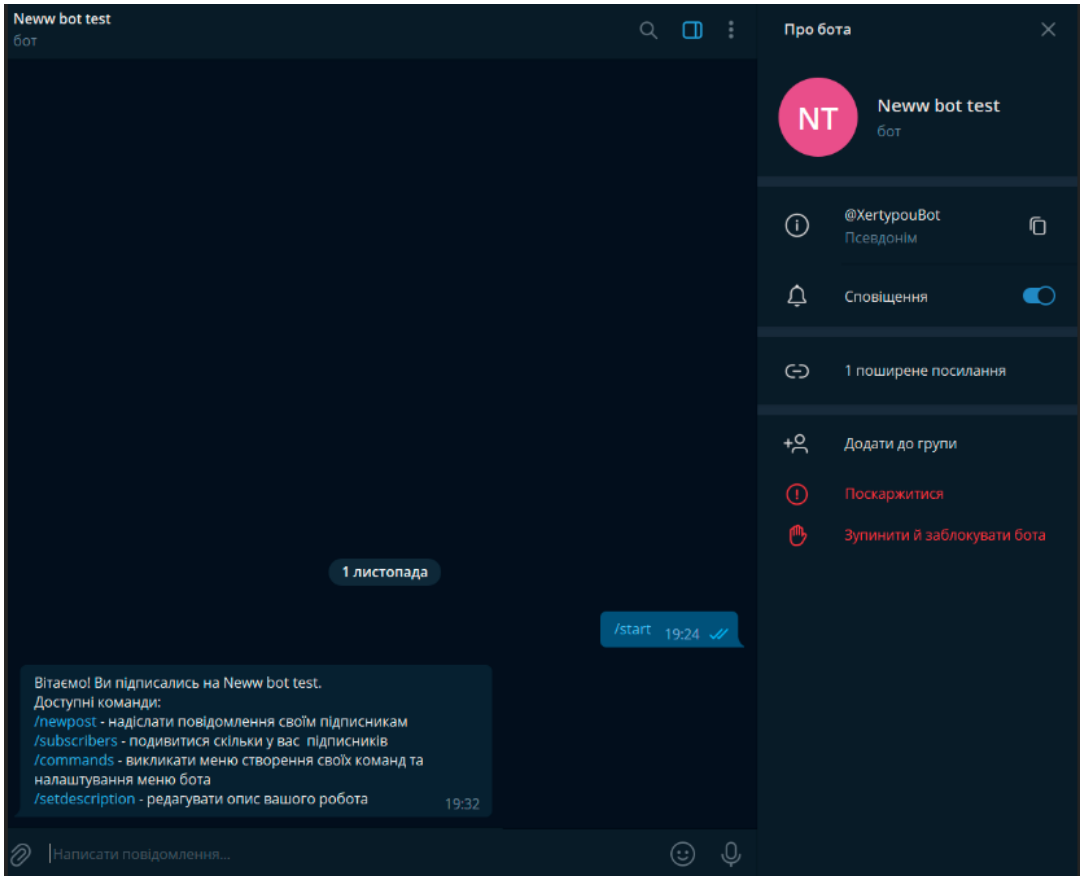

рис. 3.9 Приклад створного бота

Після натискання кнопки /start, вже в новому боті, ми бачимо повідомлення з доступними командами, після яких ми можемо налаштувати функціонал нашого бота для подальшої роботи.

Таким чином була створена інформаційна система створення Telegram-ботів з апріорно визначеним функціоналом.

2.3 СУБД (SQLite)

```python
import sqlite3

class BotDB:

    def __init__(self, db_file):
        self.conn = sqlite3.connect(db_file)
        self.cursor = self.conn.cursor()

    def user_exists(self, user_id):
        """Перевіряємо, чи є користувач у базі"""
        result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user_id` = ?", (user_id,))
        return bool(len(result.fetchall()))

    def get_user_id(self, user_id):
        """Дістаємо id користувача в основі за його user_id"""
        result = self.cursor.execute("SELECT `id` FROM `users` WHERE `user_id` = ?", (user_id,))
        return result.fetchone()[0]

    def add_user(self, user_id):
        """Додаємо користувача в базу"""
        self.cursor.execute("INSERT INTO `users` (`user_id`) VALUES (?)", (user_id,))
        return self.conn.commit()

    def add_record(self, user_id, operation, value):
        """Створюємо запис про додавання бота"""
        self.cursor.execute("INSERT INTO `records` (`users_id`, `operation`, `value`) VALUES (?, ?, ?)",
            (self.get_user_id(user_id),
            operation == "+",
            value))
        return self.conn.commit()
```

рис. 3.9 Реалізація запису у БД

У цій базі даних були створені запити для додавання даних про користувача, який приєднався до бота. Ці дані зібрані для того, щоб індефікувати користувача надалі.

	Name	Data type	Primary Key	Foreign Key	Unique	Check	Not NULL	Collate	Generated	Default value
1	id	INTEGER								NULL
2	user_id	INTEGER								NULL
3	join_date									(DATETIME('now'))

рис. 3.9 Структура БД

ВИСНОВКИ

В ході виконання дипломної роботи, було виявлено, що створена інформаційна система створення telegram-ботів актуальною темою, і тому являється чудовою альтернативою різним платформам та додаткам, зі схожою функціональністю.

Відбулося дослідження технологій для побудови чат-бота. Виявлено, що обмін повідомлень у чат-боті відбувається шляхом клієнт-серверної архітектури. Всі необхідні дані передаються по HTTP протоколу за допомогою формату даних.

Для реалізації обрано наступну мову програмування та технології: Python, модуль Telebot, Telegram API та база даних SQLlite.

Результатом проведеного дослідження стало створення чат-бота Telegram-ботів з апріорно визначеним функціоналом. Система демонструє всі можливі шляхи реалізації організації на базі месенджера Telegram.

Створений чат-бот розглядається як сучасна система, з можливістю розширення функціоналу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Python Developers Guide [Електронний ресурс]: Режим доступу: https://devguide.python.org/
2. Стаття «Створення Telegram Bot» [Електронний ресурс]: Режим доступу: https://codeguida.com/post/410
3. Стаття «Telegram Bots» [Електронний ресурс]: Режим доступу: https://core.telegram.org/bots
4. Документація Telegram Bot API [Електронний ресурс]: Режим доступу: https://core.telegram.org/bots/api#available-methods
5. Стаття «SQL vs ORM» [Електронний ресурс]: Режим доступу: https://habr.com/ru/company/pgdayrussia/blog/328690/
6. Python Secrets Module [Електронний ресурс]: Режим доступу: https://pynative.com/python-secrets-module/
7. Python 3.11.0 [Електронний ресурс]: Режим доступу: https://www.python.org/downloads/release/python-3110/
8. Робота з бібліотекою pyTelegramBotAPI бібліотека [Електронний ресурс]: Режим доступу https://pypi.org/project/pyTelegramBotAPI/
9. Documentation for Visual Studio Code [Електронний ресурс]: Режим доступу https://code.visualstudio.com/docs
10. Що таке чат-боти та які вони бувають. [Електронний ресурс]: Режим доступу https://www.carrotquest.io/blog/chatbottypes/
11. Five Different Types of Chatbot [Електронний ресурс]: Режим доступу: https://medium.com/voiceui/five-different-types-of-chatbot17bb255b23b4
12. Types of Chatbots. Rule-Based Chatbots vs AI Chatbots. [Електронний ресурс]: Режим доступу: https://mindtitan.com/resources/guides/chatbot/types-of-chatbots/
13. Telegram [Електронний ресурс]: Режим доступу: https://ru.wikipedia.org/wiki/Telegram

14. Bot Code Examples — Telegram APIs [Електронний ресурс]: Режим доступу: https://core.Telegram.org/bots/samples
15. Стаття «Створення Telegram Bot» [Електронний ресурс]: Режим доступу: https://codeguida.com/post/410

ДОДАТОК А

Вміст файлу bots.py

```
"
from aiogram import types, Bot as AioBot
from aiogram.dispatcher import FSMContext
from aiogram.utils.exceptions import Unauthorized, TelegramAPIError
from tortoise.exceptions import IntegrityError
import re
from textwrap import dedent

from makinbot.models.models import Bot, User
from makinbot.settings import makinbotSettings, BotSettings
from makinbot.commands.menu import send_bots_menu
from server.server import register_token
from locales.locale import _

from makinbot.router import dp

import logging

logger = logging.getLogger(__name__)


token_pattern = r'[0-9]{8,10}:[a-zA-Z0-9_-]{35}'


@dp.message_handler(commands=["mybots"], state="*")
async def my_bots(message: types.Message, state: FSMContext):
    """
    Команда /mybots (список ботів)
    """
    return await send_bots_menu(message.chat.id, message.from_user.id)


@dp.message_handler(commands=["addbot"], state="*")
async def add_bot(message: types.Message, state: FSMContext):
    """
    Команда /addbot (додати бота)
    """
    user = await User.get_or_none(telegram_id=message.from_user.id)
    max_bot_count = makinbotSettings.max_bots_per_user()
    if user and await user.is_promo():
        max_bot_count = makinbotSettings.max_bots_per_user_promo()
    bot_count = await Bot.filter(owner__telegram_id=message.from_user.id).count()
    if bot_count >= max_bot_count:
        await message.answer(_("У вас вже надто багато роботів. Видаліть якийсь свій
бот"
                               "(/mybots ->(Вибрати бота) -> Видалити бот)"))
```

```
        return

    await message.answer(dedent(_("""
    1⃣ Перейдіть до @BotFather. Для цього натисніть на його імя, а потім Send
Message, якщо це буде потрібно.
    2⃣ Створіть новий бот у нього. Для цього всередині @BotFather використовуйте
команду newbot спочатку вам потрібно буде придумати назву, вона може бути
українською; потім потрібно придумати ваше посилання, вона повинна бути англійською і
обовязково закінчуватися на bot, наприклад NewsBot.
    3⃣ Скопіюйте API токен, який вам видасть @BotFather
    4⃣ Поверніться назад до @YouMakingBot і надішліть скопійований API токен у
відповідь на це повідомлення.
    """)))
    await state.set_state("add_bot")


@dp.message_handler(state="add_bot", content_types="text", regexp="^[^/].+")  # Not
command
async def bot_added(message: types.Message, state: FSMContext):
    """
    Користувач додає бота і ми чекаємо від нього токен
    """
    token = re.findall(token_pattern, message.text)

    async def on_invalid_token():
        await message.answer(dedent(_("""
       Це не токен бота.

        Токен виглядає ось так: 123456789:AAAA-abc123_AbcdEFghijKLMnopqrstu12
        """)))

    async def on_dummy_token():
        await message.answer(dedent(_("""
        Не вдалося запустити цього бота: неправильний токен
        """)))

    async def on_unknown_error():
        await message.answer(dedent(_("""
        Не вдалося запустити цього бота: непередбачена помилка
        """)))

    async def on_duplication_bot():
        await message.answer(dedent(_("""
        Такий бот вже є у базі даних
        """)))

    if not token:
        return await on_invalid_token()
```

```
    token = token[0]

    try:
        test_bot = AioBot(token)
        test_bot_info = await test_bot.get_me()
        await test_bot.session.close()
    except ValueError:
        return await on_invalid_token()
    except Unauthorized:
        return await on_dummy_token()
    except TelegramAPIError:
        return await on_unknown_error()

    if token == BotSettings.token():
        return await on_duplication_bot()

    user, created = await User.get_or_create(telegram_id=message.from_user.id)
    bot = Bot(token=Bot.encrypted_token(token), owner=user,
name=test_bot_info.username,
              super_chat_id=message.from_user.id)
    try:
        await bot.save()
    except IntegrityError:
        return await on_duplication_bot()

    if not await register_token(bot):
        await bot.delete()
        return await on_unknown_error()

    await message.answer(_("Токен прийнято.\nВітаємо! Бот створений та
налаштований.\nВи завжди можете додати опис потім\nЩоб надсилати повідомлення
всередині свого бота, використовуйте команду /newpost."))
    await state.reset_state()
```