ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ АВТОМАТИЗОВАНИХ СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «РОЗРОБКА ЧАТ-БОТУ ДЛЯ АВТОМАТИЗАЦІЇ ЮРИДИЧНИХ ПРОЦЕСІВ В КОМПАНІЇ»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Максим СМЕТАНА
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав:
здобувач вищої освіти Максим СМЕТАНА
група ІСДМ-63

Керівник: Андрій АРОНОВ
науковий ступінь, вчене звання к.т.н., доцент

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

Ступінь вищої освіти Магістр

Спеціальність Інформаційні системи та технології

Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІПЗАС

_____ Каміла СТОРЧАК
«_____» _____2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Сметані Максиму Юрійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка чат-боту для автоматизації юридичних процесів в компанії

керівник кваліфікаційної роботи Андрій АРОНОВ к.т.н., доцент
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологійвід «19» 10.2023р. №145

2. Строк подання кваліфікаційної роботи «29» грудня 2023р.

3. Вихідні дані до кваліфікаційної роботи: Принципи створення чат-ботів, набір інструментів для розробки проекту, науково-технічна література з питань, пов'язаних з розробкою чат-ботів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз та визначення головних принципів роботи чат-ботів, а також основні методики щодо їх створення.

Аналіз загальних принципів та засобів побудови чат-боту для месенджера Telegram та програмні засоби для його розроблення.

Створення та тестування чат-боту для автоматизації юридичних процесів

5. Перелік графічного матеріалу: *презентація*
   1. Блок-схема основних задач чат-боту.
   2. Результати процедури тестування чат-боту.
   3. Презентація проекту.

6. Дата видачі завдання 30.09.2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури.	30.09.2023-10.10.2023	виконав
2	Аналіз існуючих рішень	10.10.2023-15.10.2023	виконав
3	Розроблення структури додатку.	15.10.2023-20.10.2023	виконав
4	Програмна реалізація додатку.	21.10.2023-30.10.2023	виконав
5	Тестування додатку.	31.10.2023-30.11.2023	виконав
6	Підготовка матеріалів текстової та графічної частини проекту.	1.12.2023-15.12.2023	виконав
7	Оформлення технічної документації проекту та підготовка презентації	15.12.2023-20.12.2023	виконав

Здобувач вищої освіти _____ Максим СМЕТАНА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Андрій АРОНОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 80стор., 6 табл., 20 рис., 2 додатки, 25 джерел.

Мета роботи – є дослідження існуючих чат-ботів, спрощення та автоматизація взаємодії з користувачем через чат-боти, а також розробка та налаштування персоналізованого бота. Результатом є створений бот, спрямований на задоволення потреб юридичного відділу компанії та опрацювання питань користувачів. Завдання роботи включають у себе визначення концепції чат-бота та реалізацію власного екземпляра.

Об'єкт дослідження – чат-бот для автоматизації юридичних процесів в компанії, спеціально адаптований для месенджера Telegram.

Предмет дослідження – технологій машинного навчання для вирішення задач створення хмарних сервісів

Короткий зміст роботи: у роботі проведено дослідження предметної області, включаючи визначення, класифікацію та основні принципи роботи чат-ботів. Також досліджено методи та інструменти для створення чат-ботів у месенджері Telegram, з метою кращого розуміння їх функціональності та взаємодії з користувачем. Надано детальний опис процесу розробки, створення та тестування чат-бота.

КЛЮЧОВІ СЛОВА: ЧАТ-БОТ, TELEGRAM, API, PYTHON.

ABSTRACT

The text part of the qualification work: 80 pages, 6 tables, 20 figures, 2 appendices, 25 sources.

The purpose of the work is to research existing chatbots, simplify and automate interaction with the user through chatbots, as well as develop and configure a personalized bot. The result is a created bot aimed at meeting the needs of the company's legal department and processing user issues. Work tasks include defining the concept of a chatbot and implementing its own instance.

The object of the research – is a chatbot for automating legal processes in the company, specially adapted for the Telegram messenger.

The subject of research – is machine learning technologies for solving the problems of creating cloud services

Summary of the work: the research of the subject area was carried out in the work, including the definition, classification and basic principles of chatbots. Methods and tools for creating chatbots in the Telegram messenger were also investigated, in order to better understand their functionality and interaction with the user. A detailed description of the process of developing, creating and testing a chatbot is provided.

KEYWORDS: CHAT-BOT, TELEGRAM, API, PYTHON.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШІ – штучний інтелект

NLU - Natural Language Understanding

API - Application Programming Interface

HTTP - HyperText Markup Language

JSON - JavaScript Object Notation

URL - Uniform Resource Locator

ООП - об'єктно-орієнтоване програмування

NLTK- NaturalLanguageToolkit

NDA - Non-disclosure agreement

ID - identity document

ВСТУП

У сучасному швидкому цифровому світі йдуться постійні пошуки способів покращення комунікації між працівниками великих компаній і обслуговування клієнтів та способів підвищення загального користувацького досвіду. Одним з інструментів, який здобув значну популярність в останні роки, є використання чат-ботів. Ці віртуальні помічники, революціонізували обслуговування людей, забезпечуючи безперешкодну та ефективну взаємодію з клієнтами та працівниками.

Актуальність теми дослідження в сучасному світі набуває все більшого значення в контексті стрімкого технологічного розвитку та необхідності оптимізації юридичних процесів у підприємствах. Спостереження за сучасним станом наукового розвитку свідчить про необхідність ефективного впровадження інновацій, зокрема застосування чат-ботів, для автоматизації юридичних процесів у компаніях. Ця технологія може виявитися ключовою у полегшенні рутинних операцій, підвищенні швидкості та точності вирішення правових завдань.

Наукова актуальність теми визначається не тільки загальною тенденцією до впровадження цифрових технологій у різноманітні галузі, але й специфічними викликами, що стоять перед сучасними юридичними підрозділами підприємств. Україна, як частина глобального ринку, не може залишатися осторонь цих тенденцій, і необхідність впровадження чат-ботів у юридичну практику стає надзвичайно важливою для підтримки конкурентоспроможності та підвищення ефективності підприємств.

Попередні дослідження у цій області свідчать про потребу в подальшому удосконаленні технічних засобів, які використовуються для автоматизації юридичних процесів. Однак, деякі аспекти цієї проблеми залишаються невирішеними. Ця робота спрямована на заповнення такої прогалини та на розробку чат-боту, який враховує специфіку юридичного середовища та вирішує конкретні завдання.

Деякі питання щодо використання чат-ботів у юридичних практиках залишаються відкритими, і ця робота прагне вирішити деякі такі проблеми.

Таким чином, дана кваліфікаційна робота є важливим кроком у напрямку подальшого розвитку і впровадження чат-ботів для автоматизації юридичних процесів у компаніях, зокрема в умовах сучасної бізнес-середи, яка вимагає швидкості та точності прийняття юридичних та бізнес рішень.

Мета кваліфікаційної роботи:

Розробка чат-боту для автоматизації юридичних процесів в компанії з метою підвищення ефективності та оптимізації рутинних завдань юридичного департаменту компанії.

Завдання дослідження:

1. Вивчення сучасних підходів та визначення рутинних операцій, які можуть бути автоматизовані за допомогою чат-боту.
2. Вивчення існуючих рішень та технологій в області чат-ботів. Історія та аналіз існуючих платформ та програмних засобів для створення чат-ботів з метою визначення найбільш ефективних рішень для реалізації цієї роботи у контексті юридичної автоматизації.
3. Розробка концепції чат-боту для юридичних процесів. Формулювання вимог до чат-боту, враховуючи особливості та потреби певного юридичного середовища компанії.
4. Реалізація прототипу чат-боту. Створення функціонального прототипу на основі визначених вимог, з урахуванням можливостей обраної технології.
5. Тестування та вдосконалення чат-боту. Проведення тестів для оцінки ефективності та коректності роботи чат-боту, а також внесення необхідних корекцій для підвищення його функціональності.
6. Аналіз результатів тестування та оцінка впливу чат-боту на ефективність юридичних процесів в організації.

В процесі дослідження вирішувалися ці завдання для досягнення мети розроблення чат-боту, спрямованого на автоматизацію та підвищення ефективності юридичних процесів у компанії.

Об'єктом дослідження є процеси та задачі, пов'язані з розробкою та впровадженням чат-боту для оптимізації та полегшення роботи юридичного відділу/департаменту компанії.

Предметом дослідження є розробка та впровадження чат-боту для автоматизації рутинних завдань та полегшення роботи юридичного відділу компанії. Предмет дослідження фокусується на конкретних викликах та завданнях, які стоять перед юридичним відділом, і які можуть бути вирішені за допомогою чат-боту.

Наукова новизна цієї дипломної роботи базується на кількох ключових аспектах, що роблять внесок у розвиток сфери автоматизації юридичних процесів:

1. Новий підхід до вирішення завдань юридичного відділу. Ця робота пропонує інноваційний метод застосування чат-боту для автоматизації рутинних та повсякденних завдань, що передбачаються юридичним відділом. Розроблений чат-бот спеціально адаптований до потреб юридичного середовища та має здатність ефективно взаємодіяти з користувачами.

2. Новий рівень ефективності використання чат-бот технологій. Робота вдосконалює та оптимізує технічний аспект використання чат-ботів, адаптуючи їх до конкретного сценарію використання в юридичному відділі компанії. Розроблений чат-бот надає практичні рішення для широкого спектру завдань, що раніше можливо було вирішити лише традиційними методами.

3. Інтеграція з існуючими юридичними процесами. Здатність чат-боту інтегруватися з існуючими юридичними процесами компанії, надаючи плавну та згодовану роботу між людським та автоматизованим виконанням завдань.

4. Специфічні адаптації до вимог юридичного середовища в Україні. Зважаючи на особливості законодавства та вимоги юридичного відділу в Україні, розроблений чат-бот враховує конкретні обставини, нормативні акти та процедури, що робить його практично корисним та відповідним специфіці даної країни.

Ці аспекти визначають новаторський характер даної роботи та її значення для подальшого розвитку сфери автоматизації юридичних процесів в компаніях.

Отримані результати дозволять ефективніше використовувати чат-бот технології для вирішення конкретних завдань юридичного відділу, що в свою чергу призведе до підвищення продуктивності та забезпечення якості правової підтримки в організації.

Однією з основних переваг ботів є їхня здатність легко інтегруватися з різними платформами та каналами. Розгортання чат-ботів можливе на веб-сайтах, в додатках соціальних мереж, таких як Facebook Messenger та Instagram, а також через текстові повідомлення. Зустрічаючи клієнтів там, де їм найзручніше, бізнеси можуть надавати різноманітні варіанти обслуговування клієнтів. Цей омніканальний підхід забезпечує можливість взаємодії клієнтів з бізнесом на їхніх улюблених платформах, покращуючи загальний досвід.

Серед різних платформ для обміну повідомленнями Telegram є однією з найпопулярніших платформ для інтеграції чат-ботів.

Чат-боти у Telegram можуть взаємодіяти з користувачами, надавати допомогу та пропонувати унікальні функції. Вони надають різноманітні послуги, від відповіді на запитання користувачів до виконання конкретних завдань та надання допомоги в реальному часі. Ці боти призначені для імітації розмови людини та надання персоналізованого та інтерактивного досвіду для користувачів.

Тому було поставлено актуальну задачу по розробці чат-боту для автоматизації юридичних процесів компанії в месенджері Telegram. Розробка чат-бота у Telegram для юридичного відділу компанії має поліпшити роботу відділу, а також взаємодію з клієнтами та співробітниками, зокрема:

1. Покращити внутрішню комунікацію - чат-бот може служити інструментом для спрощення внутрішньої комунікації в юридичному відділі. Він може надавати зручний канал для обміну інформацією та завданнями між колегами.

2. Швидка відповідь на потреби користувачів - відділ може швидше та ефективніше реагувати на запитання та потреби співробітників чи клієнтів, що є критичним у юридичних питаннях.

3. Зниження навантаження на персонал - автоматизація рутинних завдань дозволяє співробітникам юридичного відділу зосереджуватися на більш складних завданнях та стратегічному плануванні.

Практична значущість цієї кваліфікаційної магістерської роботи проявляється у кількох аспектах, які можуть бути використані на конкретному підприємстві:

1. Ефективність роботи юридичного відділу: розроблений чат-бот може значно полегшити рутинні завдання юридичного відділу, такі як підготовка документів, надання консультацій та взаємодія зі співробітниками. Це призведе до підвищення ефективності та швидкості вирішення правових питань.

2. Економія ресурсів: автоматизація через чат-бот дозволить оптимізувати робочий час юридичного відділу, звільняючи його від рутинних операцій. Це в свою чергу може призвести до економії ресурсів та зниження витрат на виконання повсякденних завдань.

3. Підвищення точності та уніфікації рішень: чат-бот забезпечить юридичний відділ засобами для консистентного та уніфікованого рішення правових питань, що забезпечить високий стандарт якості юридичних послуг в компанії.

4. Зменшення часу на навчання нових співробітників: використання чат-бота спростить процес навчання нових співробітників компанії, оскільки він може служити джерелом інформації та допомагати у засвоєнні процесів та процедур.

5. Підвищення рівня обслуговування клієнтів: якщо чат-бот інтегрувати з системою обслуговування клієнтів, він може забезпечити швидку та ефективну відповідь на юридичні запитання з боку клієнтів компанії, підвищуючи загальний рівень обслуговування.

Усі ці аспекти свідчать про те, що розробка чат-боту для автоматизації юридичних процесів може мати конкретний та невідомий раніше позитивний вплив на функціонування юридичного відділу та загальну продуктивність підприємства.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття чат-боту та історія виникнення

Чат-бот (chatbots) - це програма, що взаємодіє з користувачем чи клієнтом за допомогою текстових або голосових повідомлень. Це різновид програм штучного інтелекту (ШІ), створений для автоматизації комунікації та виконання завдань через чатові інтерфейси.

Основна ідея чат-ботів полягає в тому, щоб надавати користувачам можливість отримувати інформацію, задавати питання, отримувати рекомендації чи виконувати конкретні завдання, використовуючи природний мовний інтерфейс. Це може бути використано в різних сферах, таких як обслуговування клієнтів, освіта, розваги, медицина, фінанси та інші галузі.

У 1966 році був створений перший чат-бот під назвою ELIZA. ELIZA моделювала роботу психотерапевта, перетворюючи висловлення користувача на запитальні речення. Цей бот виник від ідеї Алана Тьюрінга про створення програми для спілкування з групою людей, які не могли б усвідомлювати, що їхнім співрозмовником є штучний інтелект. Ця концепція відома як Тест Тьюрінга, і вона вважається ключовою для розвитку чат-ботів. ELIZA використовувала шаблони і вибір відповідей на основі цих шаблонів, але її основним недоліком було обмежене знання, через що вона могла обговорювати лише конкретні теми.

У 1972 році з'явився чат-бот PARRY, який імітував хворого на шизофренію. PARRY вважався більш вдосконаленим порівняно з ELIZA через наявність "особистості" та більш розробленої структури контролю. Він формулював свої відповіді на основі системи припущень та "емоційних реакцій", активованих зміною ваги у висловлюваннях користувача. Однак він також мав низьку швидкість реагування і не здатний був вчитися з розмов.

З 1988 по 2001 рік було декілька спроб зробити чат-боти з використанням штучного інтелекту. Наприклад, у 1988 році – чат-бот Jabberwacky, Chatterbot у 1991 році, ALICE – 1995 рік.

У 2001 році був створений чат-бот SmarterChild, що визначив важливий етап у розвитку технологій чат-ботів. Його можна було використовувати у месенджерах, таких як America Online (AOL) і Microsoft (MSN), і він мав здатність отримувати інформацію з різних баз даних, включаючи час показу фільмів, новини, спортивні результати, ціни на акції, погоду тощо. Ця можливість визначила значний розвиток як у сфері штучного інтелекту, так і в напрямках взаємодії людини з комп'ютером, дозволяючи отримувати доступ до інформаційних систем через спілкування з чат-ботом.

Наступним етапом у розвитку чат-ботів із штучним інтелектом стало створення віртуальних персональних помічників, найпопулярніші серед яких - Apple Siri, Microsoft Cortana, Amazon Alexa, Google Assistant і IBM Watson. Вони мають підключення до Інтернету та, на відміну від своїх попередників, здатні створювати швидкі та осмислені відповіді.

З 2016 року і до сьогодні спостерігається стрімкий зріст використання чат-ботів, особливо через їхню інтеграцію в різноманітні соціальні платформи, такі як Facebook Messenger, Telegram, Viber, Skype і інші.

1.2 Класифікація чат-ботів

Чат-боти можна класифікувати за кількома основними критеріями (Рисунок 1.1):

- Залежно від сфери застосування:

- Інформаційні чат-боти: ці чат-боти призначені для надання корисної інформації користувачам. Вони можуть включати в себе новинні боти, консультативні системи та інші, які спрямовані на передачу конкретних знань.

- Транзакційні чат-боти: цей тип ботів спрямований на виконання різних операцій та транзакцій. Вони можуть включати платіжні боти, сервіси бронювання та інші, які забезпечують реалізацію певних операцій.
- Інтерактивні чат-боти: ці чат-боти орієнтовані на розважальний чи освітній контент, надаючи користувачам можливість взаємодії за допомогою цікавих та навчальних елементів.

• За типом комунікації:

- Текстові чат-боти: взаємодія з цими чат-ботами відбувається через текстові повідомлення. Користувачі вводять запитання або команди, і боти реагують текстовими відповідями.
- Голосові чат-боти: цей тип чат-ботів використовує голосові команди та надає голосові відповіді. Користувачі можуть взаємодіяти з ботом, використовуючи свій голос.

• За технічною класифікацією:

Сучасні інструменти розробки чат-ботів переважно базуються на наступних основних типах:

- Лінгвістичні (чат-боти на основі правил): чат-боти на основі правил надають точно налаштований контроль і гнучкість, властиві чат-ботам машинного навчання. Вони використовують логіку «якщо-тоді» (if/then) для створення розмовних потоків та можуть бути налаштовані для обробки мовних умов, синонімів та інших аспектів для забезпечення консистентних відповідей.
- На моделях машинного навчання (чат-боти ШІ): чат-боти машинного навчання (ШІ) є більш складними та розмовними, здатними до прогнозування та обробки даних. Вони використовують інтелектуальні дані для персоналізації взаємодії та можуть навіть враховувати контекст розмови.
- Гібридний підхід: об'єднує переваги обох типів чат-ботів, дозволяючи створювати складні інтерактивні рішення. Він забезпечує прозорість, консистентність та можливість реагувати на широкий спектр сценаріїв, роблячи

його оптимальним вибором для розробки додатків ШІ, які впливають на клієнтський досвід і прибутки підприємств.

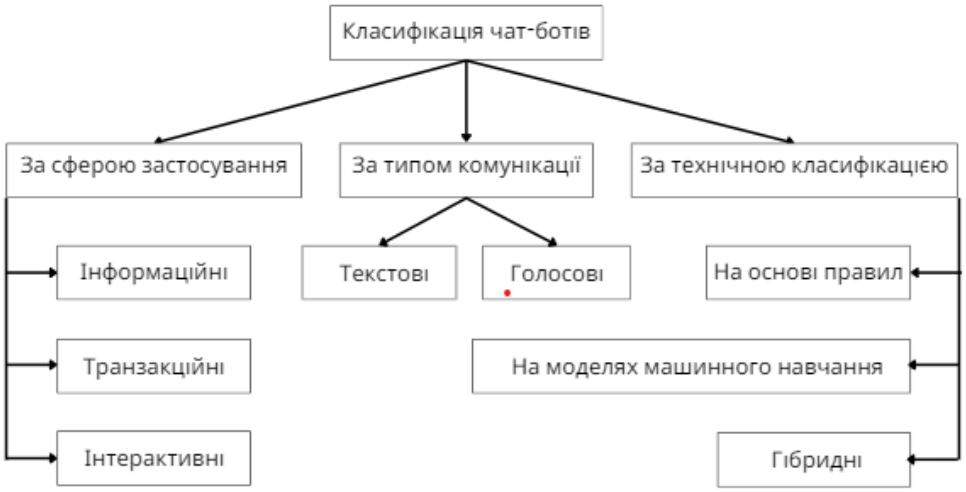


Рисунок 1.1 - Класифікація чат-ботів

Порівняння чат-ботів на основі правил та чат-ботів на основі ШІ наведено у Таблиці 1.1.

Таблиця 1.1 - Порівняння чат-ботів на основі правил та чат-ботів на основі ШІ

Питання	**Чат-боти на основі правил**	**Чат-боти на основі ШІ**
Здатність до навчання	Статичний - не можна навчатися на основі взаємодії користувача.	Динамічний - постійно вчиться та вдосконалюється завдяки взаємодії з користувачем.
Гнучкість реагування	Обмежений - може відповідати лише на попередньо визначені запити.	Універсальний - може розуміти та реагувати на широкий спектр введених користувачами, навіть якщо вони не були запрограмовані.
Розмовний потік	Жорсткий - слідкує за лінійним потоком розмови.	Природний - імітує людську розмову, забезпечуючи більш плавну та органічну взаємодію.

Продовження таблиці 1.1 - Порівняння чат-ботів на основі правил та чат-ботів на основі ШІ

Складність запитів	Базовий - може обробляти прості, зрозумілі запити.	Просунутий - здатний обробляти складні запитання, зміни контексту та багатоетапні розмови.
Інтеграційні можливості	Базовий - обмежений певними попередньо визначеними інтеграціями.	Розширений - можна інтегрувати з безліччю інструментів, баз даних та інших систем.
Масштабованість	Обмежено - для оновлення або масштабування потрібне ручне втручання.	Автоматизовано - може легко масштабуватися та розвиватися в міру зростання змін у потребах.
Досвід користувача	Передбачуваний - неодноразово пропонує ту саму взаємодію.	Персоналізований - пропонує індивідуальну взаємодію на основі поведінки та вподобань користувачів.
Технічне обслуговування	Часто - потребує регулярних оновлень вручну для задоволення нових запитів.	Мінімальний - з часом самопокращується, зменшуючи потребу у частих оновленнях вручну.
Вартість з часом	Вищі - регулярні оновлення вручну можуть з часом збільшити витрати.	Економічно ефективні - початкове налаштування може бути більш витратним, але довгострокові витрати зменшуються завдяки самовдосконаленню та масштабованості.

Таким чином, можна зробити певні висновки: Чат-боти, засновані на суто лінгвістичній моделі (на основі правил), можуть бути жорсткими та повільними для розробки через дуже трудомісткий підхід.

Хоча ці типи чат-ботів використовують обробку природної мови, взаємодія з ними досить специфічна та структурована. Цей тип чат-ботів, як правило, нагадує інтерактивні поширені запитання, і їхні можливості є базовими.

Проте, на даний час, це найпоширеніший тип ботів, які використовуються у багатьох сферах — у чатах, на веб-сайтах чи в месенджерах.

Чат-боти на основі ШІ, як правило, більш складні, інтерактивні та персоналізовані. Та згодом при роботі з даними вони стають більш обізнаними щодо контексту та використовують розуміння природної мови і застосовують інтелектуальні дані для персоналізації взаємодії з користувачем.

Проте, для роботи навіть на самому елементарному рівні такі системи часто потребують великої кількості навчальних даних і висококваліфікованих спеціалістів.

Використання гібридного підходу пропонує найкраще з обох рішень і пропонує можливість створювати складніші розмовні рішення чат-ботів ШІ.

Гібридний підхід має кілька ключових переваг перед обома альтернативами. У порівнянні з методами машинного навчання він дозволяє створювати розмовні системи навіть без даних, забезпечує прозорість у тому, як працює система, дозволяє користувачам зрозуміти програму та гарантує, що зберігається послідовна індивідуальність і узгоджена поведінка з очікуваннями.

Цей підхід дозволяє розробляти складні розмовні рішення, поєднуючи прозорість інтерфейсу та індивідуальність взаємодії..

1.3 Дизайн і розробка чат-ботів

Проектуючи та розробляючи чат-бота, важливо враховувати не лише технічні аспекти, але й елементи дизайну, які забезпечать зручність та

задоволення від користування.

1) Визначення мети та аудиторії

Першим кроком у розробці чат-бота є чітке визначення його мети та цільової аудиторії. Різні бізнеси можуть використовувати чат-ботів для різних цілей, таких як підтримка клієнтів, продажі чи розваги. Розуміння потреб цільової аудиторії визначить функціональність та дизайн бота.

2) Інтуїтивний інтерфейс

Інтуїтивно зрозумілий інтерфейс є ключовим елементом в успішному дизайні чат-бота. Користувач повинен легко орієнтуватися та взаємодіяти з ботом. Використання зрозумілої лексики та лаконічних команд сприяє створенню природної та ефективної взаємодії.

3) Персоналізація та гуманізація

Додавання персонального елементу в чат-бота може покращити користувацький досвід. Використання імен користувачів та гуманізація мови може зробити взаємодію більш привітною та приємною. Заздалегідь вивчені переваги користувача дозволяють створити індивідуальний підхід до кожного користувача.

4) Контекстно-орієнтована взаємодія

Чат-боти повинні бути здатні визначати контекст розмови для послідовності та змістовності відповідей. Врахування попередніх повідомлень користувача та узгодження їх із новими запитаннями дозволяє створити природні та логічні діалоги.

5) Візуальний елемент

Якщо чат-бот інтегрований в веб-сайт чи мобільний додаток, важливо враховувати візуальний аспект. Чітке відображення тексту, використання емодзі та графічних елементів може поліпшити сприйняття та зробити взаємодію більш цікавою.

6) Тестування та оптимізація

Не менш важливим етапом є тестування чат-бота з реальними користувачами. Зібрані дані дозволять виявити слабкі місця та внести відповідні корективи для покращення функціональності та ефективності бота.

Дизайн та розробка чат-ботів вимагає глибокого розуміння потреб користувачів та бізнес-цілей. Інтеграція елементів дизайну, таких як інтуїтивний інтерфейс та гуманізована взаємодія, разом з технічною функціональністю, дозволяє створити чат-бота, який ефективно вирішує завдання та надає задоволення від користування.

Взаємодію користувача і чат-бота можна подати у наступний спосіб - обмін інформацією в архітектурі чат-бота відбувається за шаблоном "запит-відповідь". Ось типовий порядок подій:

1) Взаємодія може початися як з боку користувача, так і з боку чат-бота (наприклад, активне привітання на бізнес-сторінці). Інтерфейс користувача фіксує введення користувача, що може бути текстовим, голосовим або іншим типом повідомлення.

2) Інтерфейс користувача передає дані користувача модулю розуміння природної мови (NLU).

3) Модуль NLU надсилає структуровану інформацію до Менеджера діалогу (Dialog Manager).

4) Менеджер діалогу аналізує контекст розмови, наміри користувача та витягнуті сутності для визначення відповідної дії чи відповіді. В рамках встановлення контексту Менеджер діалогу може розглядати інформацію про особу користувача, обліковий запис та попередні розмови.

5) Потім Менеджер діалогу викликає компонент генерації дії/відповіді для створення відповіді, яка надсилається користувачеві.

6) Генерація дії/відповіді надсилає відповідь до інтерфейсу користувача для відображення її користувачеві або виконання необхідної дії.

7) Якщо чат-боту потрібно взаємодіяти з зовнішніми системами чи API, компонент Integration and APIs обробляє зв’язок.

Цей потік спілкування забезпечує плавну та змістовну розмову між чат-ботом і користувачем.

1.4 Слабкі сторони та загрози чат-ботів

В останні роки чат-боти стали необхідною частиною бізнес-стратегій, спрямованих на поліпшення обслуговування клієнтів та оптимізацію бізнес-процесів. Однак, разом із зростанням популярності цих інноваційних інструментів, виявляються певні недоліки, які варто враховувати.

1) Взаємодія з користувачем не завжди ідеальна

Однією з ключових проблем є взаємодія чат-ботів з користувачами. Навіть з передовими технологіями штучного інтелекту, є випадки, коли розмови стають не ідеальними. Недостатня автентичність розмов може призводити до втрати інтересу та позитивного враження від взаємодії.

2) Обмежене розпізнавання мови

Багатомовність та різноманіття мовних виразів створюють виклики для чат-ботів у точному розпізнаванні та адекватній відповіді на різноманітні діалекти та вирази. Особливо це стає актуальним в глобальному бізнес-середовищі, де користувачі використовують різні мови та лінгвістичні особливості.

3) Проблеми з безпекою та приватністю

Використання чат-ботів вимагає великої уваги до захисту особистих даних. Якщо компанії надають окрему програму чат-бота, вони несуть відповідальність за адекватний захист та обробку даних клієнтів. Щоправда, у випадках, коли чат-боти розміщені на сторонніх сайтах, може виникнути питання стосовно передачі даних.

4) Потреба в навчанні та обслуговуванні

Створення та утримання чат-ботів потребує великого обсягу роботи. Розробники повинні вдосконалювати алгоритми, навчати моделі та підтримувати їх актуальними. Це вимагає наявності висококваліфікованих фахівців та фінансових ресурсів для вирішення технічних питань.

5) Відсутність емпатії

Чат-боти не завжди здатні виявити емоційний стан користувача. Брак емпатії може зробити взаємодію холодною та відчуженою, оскільки вони не враховують субтильності емоційних виразів, таких як гумор, сарказм чи вираження почуттів.

Враховуючи ці недоліки, важливо визнати, що чат-боти продовжують розвиватися. Активна робота над вирішенням цих проблем може забезпечити подальший розвиток та удосконалення чат-ботів, зробивши їх більш ефективними та корисними для користувачів.

1.5 Месенжер Telegram, чат-боти в месенжері Telegram

Telegram - це месенджер, який завоював світ своєю функціональністю, приватністю та інноваційними підходами, ставши однією з найпопулярніших платформ для спілкування та обміну інформацією.

Однією з ключових особливостей Telegram є його фокус на захисті приватності користувачів. Месенджер використовує потужне шифрування для захисту повідомлень від несанкціонованого доступу. Крім того, Telegram пропонує функцію "секретних чатів", де повідомлення можуть автоматично видалитись через певний час після читання.

Telegram дозволяє створювати канали та групи для спільного обміну інформацією. Це стало корисним інструментом для організації подій, обговорення тем або навіть ведення бізнесу. Крім того, в Telegram існує можливість

створювати групи з великою кількістю учасників, що робить його ефективним для великих спільнот.

Telegram дозволяє користувачам редагувати вже відправлені повідомлення, видаляти їх або навіть встановлювати таймер на автовидалення. Це дає більше контролю над тим, яким чином відбувається обмін інформацією та які дані залишаються в месенджері.

Telegram не обмежується тільки текстовими повідомленнями. В ньому можна відправляти фотографії, відео, аудіо та навіть створювати власні стікери. Широкі можливості для обміну мультимедійним вмістом роблять комунікацію більш виразною та цікавою.

Telegram відомий своєю відкритістю до ботів та інтеграцій. Ця функція робить Telegram не просто месенджером, а повноцінною екосистемою для різноманітних завдань.

Деякі загальні технічні аспекти месенжеру Telegram, які можна визначити:

1) Шифрування:

Telegram використовує протокол шифрування MTProto для захисту конфіденційності повідомлень. Цей протокол був розроблений самою командою Telegram і відрізняється від інших широковживаних протоколів шифрування.

В основі протоколу MTProto (Рисунок 1.2) лежить оригінальна комбінація симетричного алгоритму шифрування AES (в режимі IGE), протоколу Діффі-Хеллмана для обміну 2048-бітними RSA-ключами між двома пристроями та ряду хеш-функцій. Протокол допускає використання шифрування end-to-end з опціональним звірянням ключів.

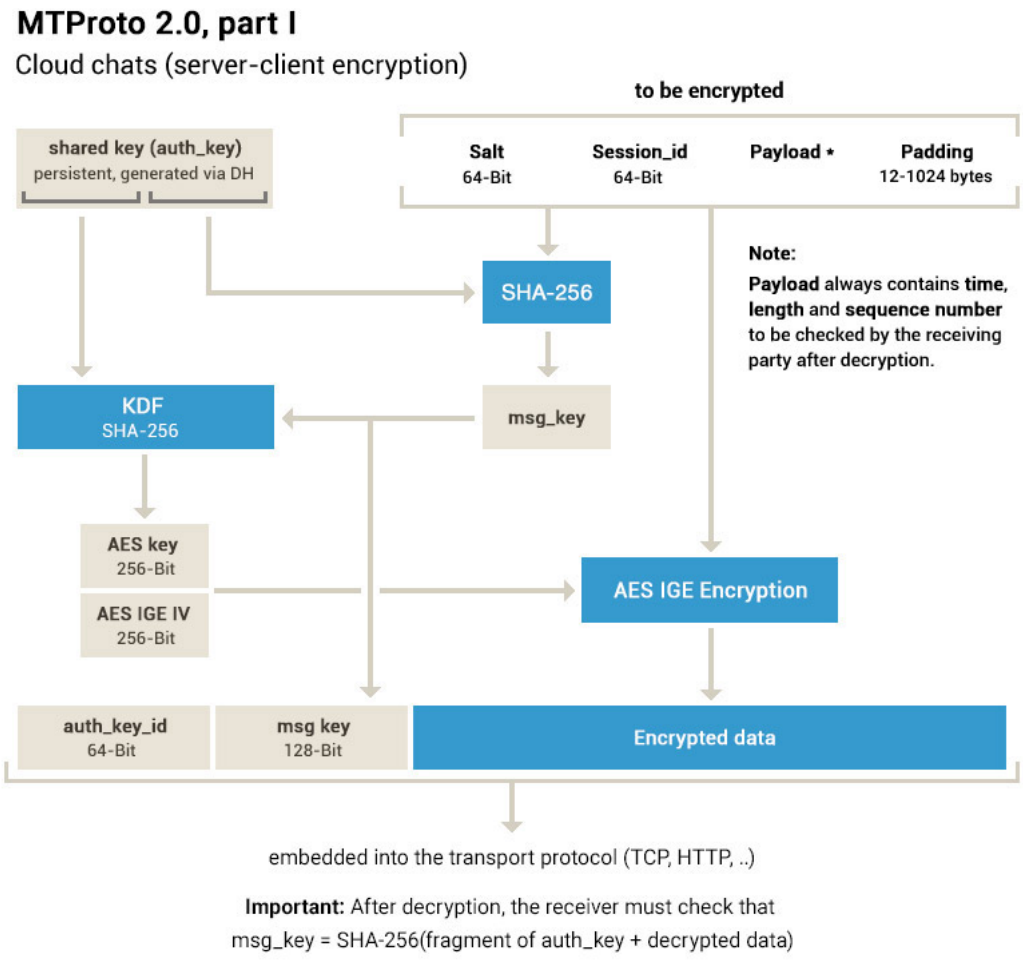


Рисунок 1.2 - Схема роботи алгоритму шифрування MTProto

2) Секретні чати:

Telegram надає можливість проводити "секретні чати" з додатковими заходами безпеки, такими як автодеструкція повідомлень.

3) Боти та API:

Telegram має відкрите API, що дозволяє розробникам створювати ботів та інтегрувати Telegram в інші додатки та сервіси.

4) Групи та канали:

Для обробки великого обсягу повідомлень у групах та каналах Telegram використовує потужні сервери.

5) Мультимедійні можливості:

Telegram дозволяє відправляти різноманітні мультимедійні файли, включаючи фотографії, відео, аудіо, геолокацію тощо.

6) Хмарне збереження даних:

Зображення та інші мультимедійні файли можна зберігати у хмарі, щоб користувачі могли легко отримувати доступ до них з різних пристроїв.

7) Механізми видалення повідомлень:

Користувачі можуть видаляти відправлені повідомлення, а також встановлювати таймер автовидалення для них.

8) Багатомовність:

Telegram підтримує багатомовність, що дозволяє користувачам обирати мову інтерфейсу та комунікації.

9) Мобільні та десктопні версії:

Telegram надає клієнти для різних платформ, включаючи мобільні (iOS та Android) та десктопні (Windows, macOS, Linux) операційні системи.

Telegram, як платформа для обміну повідомленнями, приділяє велику увагу захисту особистих даних користувачів. Згідно політики конфіденційності, компанія пропонує два ключових принципи: відсутність використання особистих даних у рекламних цілях та застосування політики мінімізації даних. Основним юридичним обґрунтуванням обробки даних є надання послуг для кінцевих користувачів та боротьба із шахрайством.

Категорії даних, які використовуються, включають номер мобільного телефону, ім'я профілю, зображення, дані облікового запису, адресу електронної пошти для двоетапної перевірки, повідомлення, телефонні контакти, дані про місцезнаходження та файли cookie. З метою захисту прав користувачів Telegram забезпечує користувачам право запитувати, видаляти, змінювати свої дані, а також висловлюватись проти обробки особистих даних та подавати скарги до органів захисту даних.

Незважаючи на зручну політику конфіденційності, умови використання Telegram встановлюють чіткі обмеження з метою переконання, що платформа не використовується для незаконних дій, таких як спам, обман, пропаганда

насильства. Це підтверджує Telegram як сервіс, який активно працює для забезпечення безпеки та етичного використання своїх послуг відповідно до законів та етичних норм.

Чат-боти в Telegram стали ключовою складовою інтернет-комунікацій та послуг. Їхня історія розвитку охоплює численні етапи, від перших спроб до сучасних інновацій. Давайте поглянемо на етапи їхнього становлення та еволюції.

Запуск Telegram Bot API у 2015 році визначив початок епохи чат-ботів в цьому месенджері. Цей інтерфейс програмування дозволяє розробникам створювати власних ботів та інтегрувати їх функціонал в Telegram.

Протягом наступних років користувачі почали ширше використовувати ботів у різних сферах: від інтерактивних ігор до новинних розсилок та онлайн-комерції. Популярність Telegram серед розробників сприяла активному створенню нових і цікавих ботів.

У 2017 році Telegram впроваджує Inline-режим для ботів, що дозволяє їм взаємодіяти з користувачами прямо в чатах, без необхідності запускати окремі діалоги. Також додаються можливості розширеної клавіатури для зручної навігації.

Протягом 2018-2020 років Telegram вдосконалює Bot API, додаючи нові можливості, такі як опитування, оновлені медіа-вкладення та адаптація під бізнес-цілі. Чат-боти стають важливим інструментом для підтримки бізнес-процесів та взаємодії з аудиторією.

Останні роки характеризуються впровадженням інновацій, таких як підтримка мовного розпізнавання та інтеграція із системами штучного інтелекту. Це розширює можливості чат-ботів, дозволяючи їм взаємодіяти з користувачами більш природним та інтелектуальним способом.

Переваги використання чат-ботів в Telegram:

Доступність: користувачі можуть отримати доступ до сервісів та інформації, не залишаючи месенджер.

Зручність взаємодії: взаємодія відбувається у звичному чатовому форматі, що робить її природною та легкою для багатьох користувачів.

Автоматизація завдань: чат-боти можуть автоматизувати рутинні операції, що полегшує роботу як для користувачів, так і для бізнесу.

Як використовувати чат-боти в Telegram:

Клієнтське Обслуговування: боти можуть використовуватися для швидкого та ефективного вирішення запитань користувачів.

Онлайн-продажі: боти можуть допомагати в здійсненні покупок, надавати інформацію про товари та послуги.

Навчання та розваги: створення ботів для навчання, проведення тестів чи розважальних ігор.

Інтеграція з сервісами: використання ботів для отримання інформації з інших сервісів чи оновлення стану замовлень.

Чат-боти в Telegram стають все більш важливим інструментом для спрощення комунікації та автоматизації процесів. Їхні можливості широкі, і їх використання може бути корисним як для користувачів, так і для бізнесу, що прагне покращити якість обслуговування та оптимізувати рутинні завдання.

Висновки до розділу 1

Отже, провівши аналіз даних першого розділу, виявлено, що чат-боти можна класифікувати за функціональністю, розрізняючи активні та пасивні, та за їхнім призначенням, таким як інфраструктурні, асистенти, торгові та інші.

Зараз чат-боти здобувають популярність завдяки доступності та можливості працювати на різних платформах, що робить їх привабливими для користувачів. Використання чат-ботів розширюється у сферах обслуговування клієнтів, електронної комерції та освіти, що свідчить про їхню важливість для бізнесу та задоволення потреб користувачів. У майбутньому очікується подальший розвиток чат-ботів, поліпшення їхньої інтелектуальної здатності та здатності натурально взаємодіяти з користувачами.

Однією за найпопулярніших платформ для інтеграції чат-ботів є месенжер Telegram, який виявляється зручною платформою для розробки чат-ботів через відкритий підхід для розробників та зручне API. Це дозволяє ботам надавати різні функції, від надсилання повідомлень до роботи з мультимедіа та створення інтерактивних взаємодій. Тому на основі аналізу отриманих результатів було вирішено створити чат-бот саме на базі месенджеру Telegram.

2 ЗАСОБИ РЕАЛІЗАЦІЇ ЧАТ-БОТУ

Для реалізації дипломного проекту було обрано мову програмування Python та бібліотеку aiogram та магічні фільтри, @BotFather – бот від Telegram месенжер.

2.1 Telegram Bot API

Bot API — це інтерфейс на основі HTTP, створений для роботи з ботами в Telegram.

Кожному боту під час створення надається унікальний маркер автентифікації . Маркер виглядає приблизно так:

123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11

Всі запити до API Telegram Bot повинні обслуговуватися через протокол HTTPS та подаватися у наступній формі:
https://api.telegram.org/bot<token>/METHOD_NAME.

Наприклад:https://api.telegram.org/bot123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11/getMe

Передача запитів підтримується за допомогою методів HTTP GET і POST. Є чотири способи передачі параметрів у запитах API ботів:

1. Рядок запиту URL-адреси.
2. application/x-www-form-urlencoded.
3. application/json (за винятком завантаження файлів).
4. multipart/form-data (використовуйте для завантаження файлів).

Відповідь містить об’єкт JSON, який завжди має логічне поле «ok» і може містити необов'язкове поле String «опис» із зрозумілим описом результату. Якщо «ок» дорівнює True, запит виконано успішно, і результат можна знайти в полі «результат». У разі невдалого запиту «ок» дорівнює false, а помилка пояснюється в полі «опис». Також повертається ціле поле 'error_code', але його вміст може

змінюватися в майбутньому. Деякі помилки також можуть містити необов'язкове поле «параметри» типу ResponseParameters, яке може допомогти автоматично обробити помилку.

Усі методи в API бота нечутливі до регістру. Усі запити повинні виконуватися за допомогою UTF-8.

Є два взаємовиключних способи отримання оновлень для чат-бота:

1. метод getUpdates
2. веб-хуки.

Вхідні оновлення зберігаються на сервері, поки бот не отримає їх у той чи інший спосіб, але вони зберігатимуться не довше 24 годин.

У відповідь отримуємо JSON-серіалізовані об’єкти оновлення – Update.

Лише один із необов’язкових параметрів може бути присутнім у будь-якому оновленні (Таблиця 2.1).

Таблиця 2.1 – Параметри об’єкту Update

Поле	**Тип**	**Опис**
update_id	Ціле число	Унікальний ідентифікатор оновлення. Ідентифікатори оновлення починаються з певного додатного числа та послідовно збільшуються. Цей ідентифікатор стає особливо зручним, якщо ви використовуєте вебхуки , оскільки він дозволяє ігнорувати повторювані оновлення або відновлювати правильну послідовність оновлень, якщо вони виходять з ладу. Якщо не буде нових оновлень протягом принаймні тижня, то ідентифікатор наступного оновлення буде вибрано випадково, а не послідовно.

Продовження таблиці 2.1 - Параметри об’єкту Update

message	повідомлення	Необов'язковий. Нове вхідне повідомлення будь-якого типу - текст, фото, стікер тощо.
edited_message	повідомлення	Необов'язковий. Нова версія повідомлення, яка відома боту та була відредагована
channel_post	повідомлення	Необов'язковий. Нова вхідна публікація каналу будь-якого типу - текст, фото, наклейка тощо.
edited_channel_post	повідомлення	Необов'язковий. Нова версія публікації на каналі, яка відома боту та була відредагована
inline_query	InlineQuery	Необов'язковий. Новий вхідний вбудований запит
chosen_inline_result	ChosenInlineResult	Необов'язковий. Результат вбудованого запиту, який вибрав користувач і надіслав своєму партнеру в чаті. Перегляньте нашу документацію щодо збору відгуків , щоб дізнатися, як увімкнути ці оновлення для вашого бота.

Продовження таблиці 2.1 - Параметри об’єкту Update

callback_query	CallbackQuery	Необов'язковий. Новий вхідний запит зворотного виклику
shipping_query	ShippingQuery	Необов'язковий. Новий вхідний запит на доставку. Лише для рахунків із гнучкою ціною
pre_checkout_query	PreCheckoutQuery	Необов'язковий. Новий вхідний запит перед розрахунком. Містить повну інформацію про оформлення замовлення
poll	Опитування	Необов'язковий. Новий стан опитування. Боти отримують лише оновлення про зупинені опитування та опитування, які надсилає бот
poll_answer	PollAnswer	Необов'язковий. Користувач змінив свою відповідь у неанонімному опитуванні. Боти отримують нові голоси лише в опитуваннях, надісланих самим ботом.
my_chat_member	ChatMemberUpdated	Необов'язковий. У чаті бота оновлено статус учасника чату. Для приватних чатів це оновлення отримується лише тоді, коли бот заблоковано або розблоковано користувачем.

Продовження таблиці 2.1 - Параметри об’єкту Update

chat_member	ChatMemberUpdated	Необов'язковий. У чаті оновлено статус учасника чату. Бот має бути адміністратором у чаті та має явно вказати "chat_member"у списку дозволених_оновлень отримання цих оновлень.
chat_join_request	ChatJoinRequest	Необов'язковий. Запит на приєднання до чату надіслано. Щоб отримувати ці оновлення , бот повинен мати права адміністратора can_invite_users у чаті.

Telegram Bot API підтримує декілька методів, за допомогою яких можна працювати з телеграм-ботом:

- getUpdates – використовується для отримання вхідних оновлень за допомогою довгого опитування. Повертає масив об’єктів оновлення. Цей метод не працюватиме, якщо налаштовано вихідний вебхук.
- setWebhook – використовується, щоб указати URL-адресу та отримувати вхідні оновлення через вихідний вебхук. Щоразу, коли з’являється оновлення для бота, надсилається HTTPS-запит POST на вказану URL-адресу, що містить JSON-серіалізоване оновлення. У разі успіху повертає True . Оновлення не можливо отримати за допомогою getUpdates , поки налаштовано вихідний вебхук. Порти, які наразі підтримуються для веб-хуків : 443, 80, 88, 8443 .
- deleteWebhook – використовується, щоб видалити інтеграцію webhook, якщо є необхідність повернутися до getUpdates . У разі успіху повертає True .

- getWebhookInfo – використовується, щоб отримати поточний статус вебхуку. Не вимагає параметрів. У разі успіху повертає об’єкт WebhookInfo . Якщо бот використовує getUpdates , поверне об’єкт із порожнім полем url.
- WebhookInfo – описує поточний статус вебхуку.

Основні параметри наведених методів:

Параметри методу getUpdates наведено у Таблиці2.2.

Таблиця 2.2 – Параметри методу getUpdates

Параметр	**Тип**	**Необхідність**	**Опис**
limit	Ціле число	Необов'язковий	Обмежує кількість оновлень, які потрібно отримати. Приймаються значення від 1 до 100. За замовчуванням 100.
offset	Ціле число	Необов'язковий	Ідентифікатор першого оновлення, яке буде повернуто. Має бути на одиницю більшим за найвищий серед ідентифікаторів раніше отриманих оновлень. За замовчуванням повертаються оновлення, починаючи з самого раннього непідтвердженого оновлення. Оновлення вважається підтвердженим, щойно getUpdates викликається зі зсувом, вищим за його update_id .

Продовження таблиці 2.2 – Параметри методу getUpdates

allowed_updates	Масив рядків	Необов'язковий	Серіалізований у форматі JSON список типів оновлень, які має отримувати ваш бот. Наприклад, вкажіть ["message", "edited_channel_post", "callback_query"]отримувати лише оновлення цих типів. Повний список доступних типів оновлень див . у розділі Оновлення . Укажіть порожній список, щоб отримати всі типи оновлень, крім chat_member (за замовчуванням). Якщо не вказано, буде використано попереднє налаштування.
timeout	Ціле число	Необов'язковий	Тайм-аут у секундах для тривалого опитування. За замовчуванням 0, тобто звичайне коротке опитування. У разі позитивного результату короткий опитування слід використовувати лише для тестування.

Параметри методу setWebhook наведено у Таблиці 2.3

Таблиця 2.3 – Параметри методу setWebhook

Параметр	**Тип**	**Необхідність**	**Опис**
url	Рядок	Так	URL-адреса HTTPS для надсилання оновлень. Використовуйте порожній рядок, щоб видалити інтеграцію webhook
certificate	InputFile	Необов'язковий	Завантажте свій сертифікат відкритого ключа, щоб можна було перевірити кореневий сертифікат, який використовується. Додаткову інформацію дивіться в нашому підписаному посібнику .
ip_address	Рядок	Необов'язковий	Фіксована IP-адреса, яка використовуватиметься для надсилання запитів на вебхук замість IP-адреси, визначеної через DNS
drop_pending_updates	Логічний	Необов'язковий	Передайте *True* , щоб видалити всі незавершені оновлення
max_connections	Ціле число	Необов'язковий	Максимально дозволена кількість одночасних підключень HTTPS до вебхука для доставки оновлень: 1–100.

Продовження таблиці 2.3 – Параметри методу setWebhook

			За замовчуванням 40 . Використовуйте нижчі значення, щоб обмежити навантаження на сервер вашого бота, і вищі значення, щоб збільшити пропускну здатність вашого бота.
allowed_updates	Масив рядків	Необов'язковий	Серіалізований у форматі JSON список типів оновлень, які має отримувати ваш бот. Наприклад, вкажіть ["message", "edited_channel_post", "callback_query"]отримувати лише оновлення цих типів. Укажіть порожній список, щоб отримати всі типи оновлень, крім chat_member (за замовчуванням). Якщо не вказано, буде використано попереднє налаштування. Зауважте, що цей параметр не впливає на оновлення, створені до виклику setWebhook, тому протягом короткого періоду часу можуть надходити небажані оновлення.

Продовження таблиці 2.3 – Параметри методу setWebhook

secret_token	Рядок	Необов'язковий	Секретний маркер, який надсилається в заголовку «X-Telegram-Bot-Api-Secret-Token» у кожному запиті на вебхук, 1–256 символів. Допускаються лише символи A-Z, a-z, 0-9, _ і -. Заголовок корисний, щоб переконатися, що запит надходить із встановленого вами вебхуку.

Параметри методу deleteWebhook наведено у Таблиці 2.4

Таблиця 2.4 – Параметри методу deleteWebhook

Параметр	**Тип**	**Необхідність**	**Опис**
drop_pending_updates	Логічний	Необов'язковий	Передайте *True* , щоб видалити всі незавершені оновлення

Параметри методу WebhookInfo наведено у Таблиці 2.5

Таблиця 2.5 – Параметри методу WebhookInfo

Поле	**Тип**	**Опис**
url	Рядок	URL-адреса вебхуку може бути порожньою, якщо вебхук не налаштовано

Продовження таблиці 2.5 – Параметри методу WebhookInfo

has_custom_certificate	Логічний	*True* , якщо для перевірки сертифікатів webhook було надано спеціальний сертифікат
pending_update_count	Ціле число	Кількість оновлень, що очікують на доставку
ip_address	Рядок	Необов'язковий. Поточна IP-адреса вебхука
last_error_date	Ціле число	Необов'язковий. Час Unix для останньої помилки, яка сталася під час спроби доставити оновлення через вебхук
last_error_message	Рядок	Необов'язковий. Повідомлення про помилку в зручному для читання форматі для останньої помилки, яка сталася під час спроби доставити оновлення через вебхук
last_synchronization_error_date	Ціле число	Необов'язковий. Час останньої помилки Unix, яка сталася під час спроби синхронізувати доступні оновлення з центрами обробки даних Telegram

Продовження таблиці 2.5 – Параметри методу WebhookInfo

max_connections	Ціле число	Необов'язковий. Максимально дозволена кількість одночасних підключень HTTPS до вебхуку для доставки оновлень
allowed_updates	Масив рядків	Необов'язковий. Список типів оновлень, на які підписаний бот. За замовчуванням для всіх типів оновлень, крім *chat_member*

2.2 Створення чат-боту за допомогою чату @BotFather

@BotFather – це бот від Telegram, з яким можна взаємодіяти так само, як і з будь-яким іншим користувачем платформи. Цей інструмент дозволяє створювати нових ботів, керувати існуючими ботами та видаляти їх, а також налаштовувати своїх ботів за допомогою різноманітних параметрів, таких як нові команди, описи та зображення профілю.

Головні можливості @BotFather:

1) Створення ботів: основна функція @BotFather - це можливість створювати нових ботів. Користувачам просто потрібно слідувати інструкціям, щоб визначити ім'я та інші параметри свого бота.

2) Управління ботами: можливість покращувати та налаштовувати свої боти за допомогою команд @BotFather. Змінювати опис, додавати нові команди, та налаштовувати зображення профілю.

3) Видалення ботів: якщо бот став зайвим або його потрібно замінити, @BotFather дозволяє легко видаляти неактивні боти.

Покрокова інструкція зі створення Telegram-бота

Крок 1. Розпочинаємо чат із @BotFather

По-перше, відкриваємо програму Telegram і знаходимо «@BotFather» у рядку пошуку вгорі. Знайшовши, починаємо з ним новий чат (Рисунок 2.1).

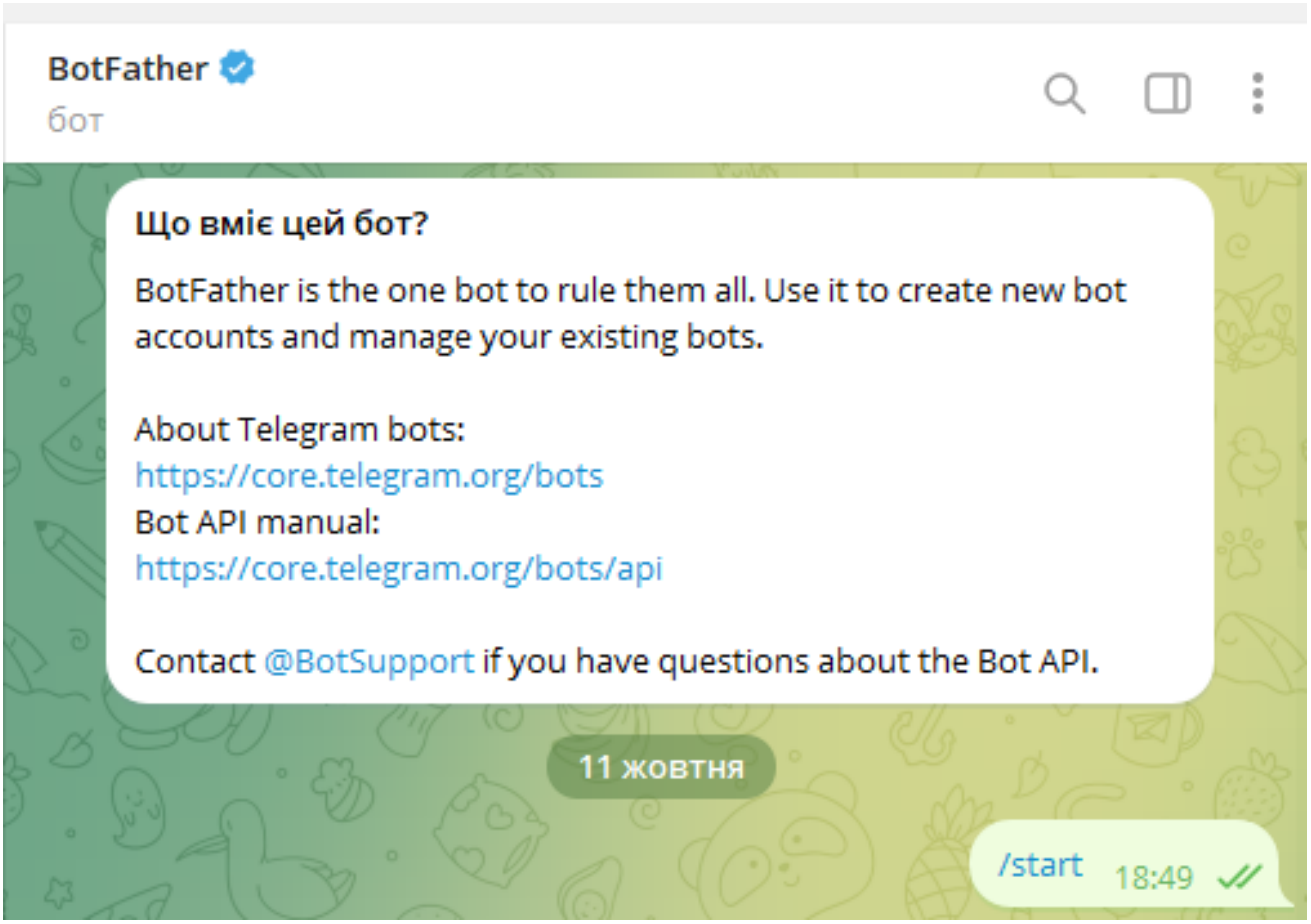


Рисунок 2.1 – Початок роботи з @BotFather

Крок 2: Створення нового бота

Щоб створити нового бота, вводимо "/newbot" і надсилаємо повідомлення. Після цього @BotFather пропонує вибрати ім’я для свого бота. Це ім’я може бути будь-яким, яке користувачі будуть бачити, взаємодіючи із ботом.

Далі обираємо ім’я користувача для бота. Це ім’я користувача має закінчуватися на «bot» (наприклад, chat_bot) і має бути унікальним на платформі.

Після виконання цих кроків @BotFather створює бота та надає маркер. Цей маркер є ключовим, оскільки він використовуватиметься для доступу до HTTP API (Рисунок 2.2).

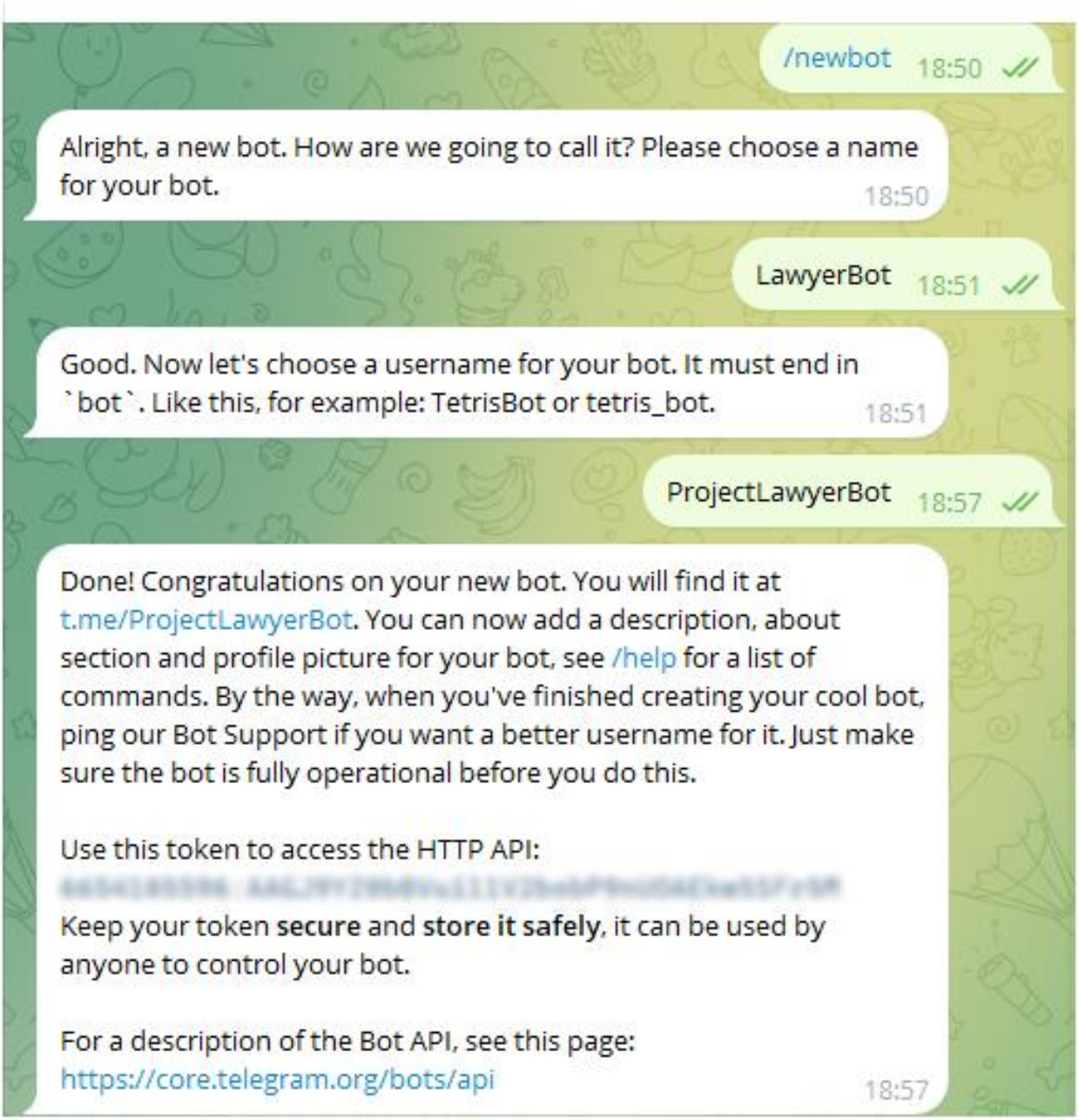


Рисунок 2.2 – Створення нового бота

Крок 3: Налаштування бота

За допомогою команди /help можна подивитись команди для налаштування чат-боту (Рисунок 2.3).

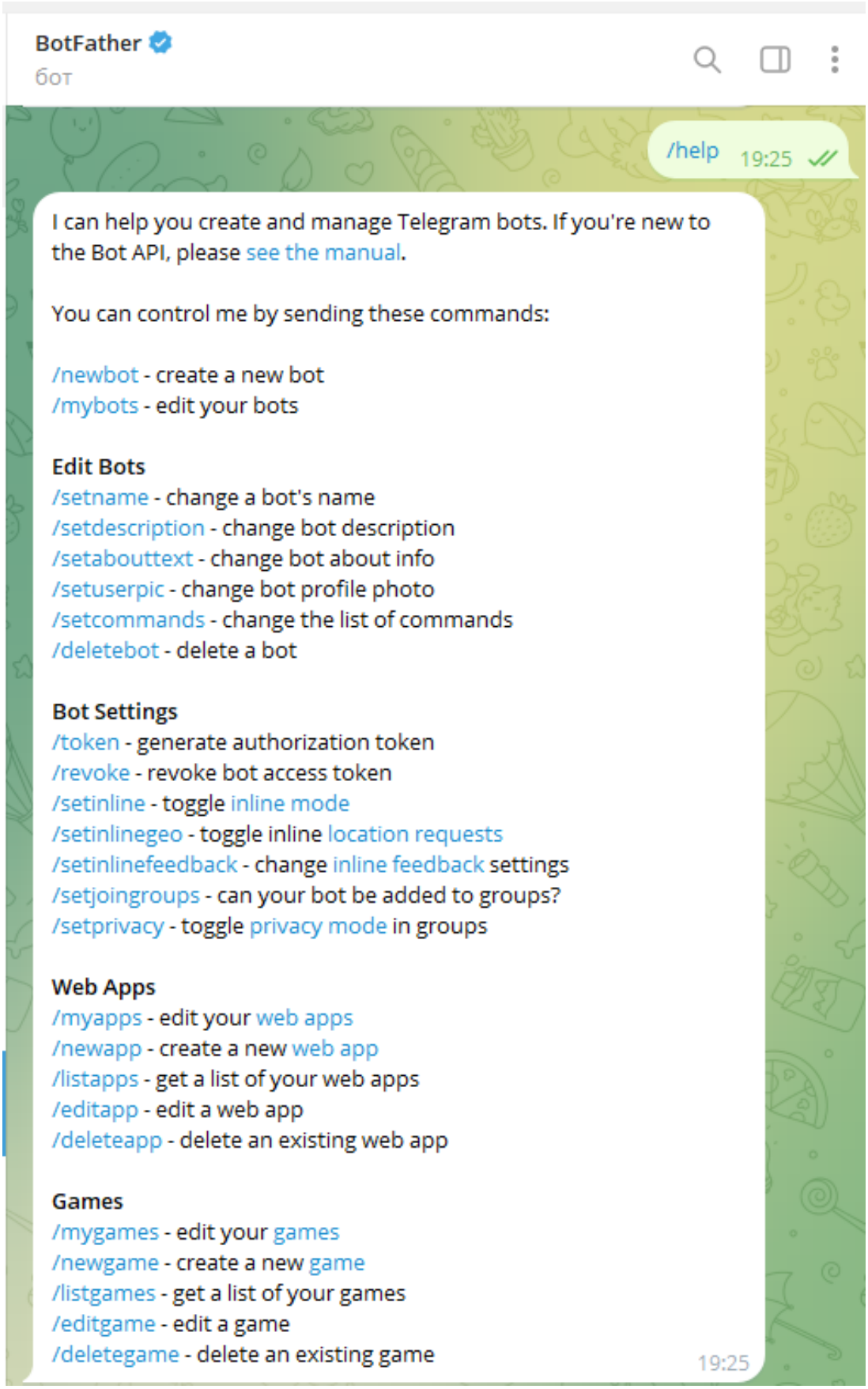


Рисунок 2.3 – Команди налаштування чат-бота

Базові команди:

Ці команди починають етапи створення нового та редагування існуючих ботів.

- "/newbot" — створює новий.
- "/mybots" — дозволяє переглядати список існуючих і керувати налаштуваннями. Остання команда відкриє список усіх доступних ботів, які можна редагувати. Там знаходяться такі функції.

- Маркер API — покаже дійсний маркер.

- Редагування — зміна всіх основних налаштувань, наприклад, імені або аватара.

- Налаштування — додаткові налаштування

- Передати права — за допомогою цієї функції можна передати бота іншому користувачеві Telegram.

Редагування створеного бота:

У цьому блоці відбувається основна настройка бота в Telegram через BotFather. Команди запускаються, якщо попередньо обрати об’єкт за допомогою команди «/mybots».

- "/setname" — задає назву. Він буде відображатися вгорі.
- "/setdescription" — задає опис. Ви можете знайти його в профілі.
- "/setabouttext" — задає текст, який бачить користувач перед натисканням кнопки "СТАРТ".
- "/setuserpic" — встановлює аватар. Вам потрібно надіслати файл як звичайне стиснене зображення будь-якого формату. Тобто спочатку йде команда, а потім ім'я, яке буде відображатися в меню бота. Однак якщо вони не вказані в Telegram «BotFather», вони все одно будуть працювати, але користувач не зможе дізнатися про їх існування.
- "/deletebot" — дозволяє видалити бота. Щоб його видалити, вам потрібно буде підтвердити серйозність своїх намірів, ввівши ключове слово. Він буде надісланий у відповідь після введення команди.

Налаштування:

Цей блок дозволяє налаштувати основні параметри бота. Наприклад, можна створити новий токен, якщо він був втрачений або потрапив до рук зловмисників.

"/token" — відображає поточний токен.

"/revoke" — генерує новий маркер. Потрібно бути обережним з цією командою, тому що вона миттєво генерує нову. Тобто не вимагає додаткових підтверджень.

2.3 Мова програмування Python

2.3.1 Технічні можливості

Python - це високорівнева, інтерпретована мова програмування, яка здобула широку популярність серед розробників. Вона відзначається простотою синтаксису, що робить її дуже доступною для початківців, а також потужною для професіоналів. Ось деякі технічні аспекти Python:

1) Синтаксис: простий і читабельний

Однією з ключових особливостей Python є його простий і лаконічний синтаксис. Код на Python зазвичай коротший і легше читається, що сприяє швидкій розробці та обслуговуванню проектів.

```
# Приклад: Вивід "Hello, World!" на Python
print("Hello, World!")
```

2) Інтерпретованість і крос-платформенність

Python є інтерпретованою мовою, що означає, що код можна виконувати без попередньої компіляції. Це спрощує розробку та тестування коду. Крім того, Python є крос-платформеним, тобто можна запускати код на різних операційних системах без змін.

3) Обширна бібліотека

Python має велику стандартну бібліотеку, яка включає модулі для роботи з різними завданнями, такими як робота з мережами, обробка даних, графічний дизайн, і багато іншого. Це дозволяє розробникам швидко вирішувати завдання без необхідності написання великої кількості коду з нуля.

4) Розширюваність і інтеграція

Python легко розширюється за допомогою модулів, написаних на C або C++, що дозволяє використовувати існуючий код на цих мовах або взаємодіяти з ними. Велика кількість бібліотек, які працюють з Python, також забезпечують його інтеграцію з різними технологіями.

5) Підтримка ООП

Python повністю підтримує об'єктно-орієнтоване програмування (ООП), що дозволяє створювати код, орієнтований на об'єкти, забезпечуючи чистий та структурований дизайн.

```
# Приклад: створення класу в Python
class Car:
    def __init__(self, brand, model):
        self.brand = brand
        self.model = model

    def display_info(self):
        print(f"{self.brand} {self.model}")
```

6) Асинхронність і здатність до роботи з багатьма задачами

Однією з сильних сторін Python є його підтримка асинхронного програмування. Асинхронність дозволяє виконувати асинхронні функції, що не блокують виконання інших частин програми. Це особливо корисно для розробки

програм, які потребують роботи з багатьма задачами одночасно, такими як веб-сервери, обробники подій та асинхронні запити до мережі.

Наприклад:

```
import asyncio
async def example_async_function():
    print("Start")
    await asyncio.sleep(2)
    print("End")

# Запуск асинхронної функції
asyncio.run(example_async_function())
```

Цей код дозволяє функції example_async_function виконуватися асинхронно, тобто програма може виконувати інші завдання, поки чекає на результат asyncio.sleep.

7) Бібліотеки для асинхронного програмування

У Python існують різні бібліотеки для асинхронного програмування, такі як asyncio, aiohttp, та async/await конструкції мови. Ці засоби дозволяють розробникам створювати ефективні та швидкодіючі програми, які можуть ефективно обробляти велику кількість одночасних запитів.

8) Зручні інструменти для асинхронного коду

Python надає зручні інструменти для роботи з асинхронним кодом, такі як async та await, які полегшують створення та управління асинхронними функціями. Це забезпечує зрозумілість та легкість у розробці асинхронного коду.

```
async def fetch_data(url):
    async with aiohttp.ClientSession() as session:
        async with session.get(url) as response:
            return await response.text()
```

У цьому прикладі async/await використовується для створення асинхронного HTTP-запиту.

2.3.2 Використання мови Python для написання чат-ботів

Python є популярним вибором для написання чат-ботів завдяки своїй простоті, широкій підтримці бібліотек та інструментів для роботи з текстом і обробки мовного інтерфейсу. Ось деякі аспекти, які роблять Python ідеальним для розробки чат-ботів:

- Бібліотеки для роботи з чат-ботами:

- NLTK (Natural Language Toolkit): є потужною бібліотекою для обробки природної мови, що дозволяє аналізувати та розуміти текст.

- spaCy: це бібліотека для обробки природної мови та виявлення іменованих сутностей, що може бути корисно для аналізу повідомлень.

ChatterBot: це проста бібліотека для створення чат-ботів, яка базується на правилах та машинному навчанні. Вона використовує машинне навчання для генерації відповідей на основі введення користувача.

- Aiogram: це асинхронна бібліотека для створення чат-ботів у Telegram. Вона надає потужний фреймворк для роботи з API Telegram, а також зручні інструменти для обробки повідомлень та інших подій.

- Telegram Bot API: це офіційний API Telegram для створення чат-ботів. Хоча це не бібліотека Python, але вона надає простий інтерфейс для взаємодії з Telegram.

- python-telegram-bot: ця бібліотека є обгорткою для Telegram Bot API та надає зручний інтерфейс для розробки чат-ботів. Вона підтримує асинхронне програмування та має велику кількість функцій.

- Microsoft Bot Framework: бібліотека від Microsoft, яка дозволяє створювати чат-боти для різних платформ, таких як Skype, Microsoft Teams, Slack тощо.

- Rasa: це відкритий фреймворк для розробки чат-ботів та вирішення задач у галузі обробки природної мови (NLP).
- Discord.py: бібліотека для створення чат-ботів у Discord, написана на Python. Вона надає зручний інтерфейс для роботи з API Discord.

Це лише кілька прикладів бібліотек, існує багато інших, специфічних для різних платформ та фреймворків. Вибір бібліотеки залежить від конкретного використання та платформи, на якій необхідно розгорнути чат-бота.

- Інтеграція з платформами:

Python легко інтегрується з різними платформами для чат-ботів, такими як Telegram, Facebook Messenger, Slack тощо. Деякі бібліотеки, як python-telegram-bot чи facebook-sdk, полегшують роботу з цими платформами.

- Можливості асинхронного програмування:

Асинхронність в Python може бути корисною для розробки чат-ботів, оскільки вони часто повинні обробляти одночасні події, такі як введення запиту від користувача та відповіді на повідомлення.

- Машинне навчання та штучний інтелект:

Python має багато бібліотек для машинного навчання, таких як TensorFlow, PyTorch, та Scikit-learn. Це може бути використано для створення більш інтелектуальних чат-ботів, які можуть вивчати відповіді на питання чи адаптуватися до стилів спілкування користувачів.

- Зручність для прототипування:

Python є зручним для прототипування, що дозволяє швидко створювати та тестувати різні аспекти чат-бота перед його реалізацією в робочому середовищі.

Приклади використання бібліотек:

1) Приклад використання бібліотеки ChatterBot

```
from chatterbot import ChatBot
from chatterbot.trainers import ChatterBotCorpusTrainer
```

```
# Створення інстанції чат-бота
bot = ChatBot('MyBot')

# Використання вбудованих даних для навчання чат-бота
trainer = ChatterBotCorpusTrainer(bot)
trainer.train('chatterbot.corpus.english')

# Отримання відповіді від чат-бота
response = bot.get_response('Hello, how are you?')
print(response)
```

Цей код створює чат-бота, який використовує вбудовані дані для навчання та може відповідати на різні питання. Python надає простий та зрозумілий спосіб створення чат-ботів для різних застосувань.

2) Приклад використання бібліотеки aiogram.

```
import logging
from aiogram import Bot, Dispatcher, types
from aiogram.types import ParseMode
from aiogram.utils import executor

# Встановлення рівня логування
logging.basicConfig(level=logging.INFO)

# Замініть 'YOUR_BOT_TOKEN' на фактичний токен вашого бота
bot = Bot(token='YOUR_BOT_TOKEN')
dp = Dispatcher(bot)
```

```
# Команда для обробки /start
@dp.message_handler(commands=['start'])
async def start(message: types.Message):
    await message.answer("Привіт! Я твій чат-бот.")

# Команда для обробки текстових повідомлень
@dp.message_handler()
async def echo(message: types.Message):
    await message.answer(f"Ти написав: {message.text}")

# Обробник для виведення фотографії при отриманні команди /photo
@dp.message_handler(commands=['photo'])
async def send_photo(message: types.Message):
    photo_url = "https://example.com/example.jpg"
    caption = "Ось твоє фото!"
    await bot.send_photo(message.chat.id, photo_url, caption=caption)

# Обробник для невідомих команд
@dp.message_handler()
async def unknown(message: types.Message):
    await message.answer("Вибач, я не розумію цю команду.")

# Запуск бота
if __name__ == '__main__':
    from aiogram import executor
    executor.start_polling(dp, skip_updates=True)
```

У цьому прикладі бот реагує на команду /start, виводить текстові повідомлення, обробляє команду /photo для виведення фотографії, і відповідає на

всі інші текстові повідомлення. Бот використовує бібліотеку aiogram для зручного роботи з Telegram API.

У прикладі необхідно замінити 'YOUR_BOT_TOKEN' на фактичний токен бота, який можна отримати, створивши бота через @BotFather в Telegram.

2.3.3 Області використання Python

Мова програмування Python використовується в різних галузях і сферах діяльності через свою зручність, простоту синтаксису та багатий набір бібліотек. Ось деякі з основних областей використання Python:

- Веб-розробка:

Python часто використовується для розробки веб-додатків і фреймворки, такі як Django та Flask, допомагають розробникам швидко створювати та підтримувати високоякісні веб-сайти.

Приклад:

Django Framework - використовується для розробки великих та потужних веб-додатків. Наприклад, Instagram базується на Django для свого серверного застосування.

- Аналіз даних та штучний інтелект:

Python є популярним вибором для аналізу даних та машинного навчання. Бібліотеки, такі як NumPy, Pandas, TensorFlow та PyTorch, забезпечують потужні інструменти для обробки даних та розробки моделей машинного навчання.

Приклад:

SciPy та Scikit-learn - використовуються для аналізу даних та розв'язання завдань машинного навчання. Наприклад, Scikit-learn може бути використаний для класифікації даних та навчання моделей.

Pandas та NumPy: великі компанії, такі як Netflix, використовують Python для аналізу великих обсягів даних.

TensorFlow та PyTorch: Google та Facebook використовують ці бібліотеки для розробки моделей машинного навчання.

- Інтернет речей (IoT):

Python використовується для розробки програмного забезпечення для пристроїв Інтернет речей та взаємодії з ними.

Приклад: MicroPython - дозволяє використовувати Python для програмування мікроконтролерів, що використовуються у вбудованих системах та проектах Інтернету речей.

- Наукові дослідження та обробка природних мов:

У сферах наукових досліджень Python використовується для обробки даних, моделювання та аналізу результатів. Також він є популярним для проектів з обробкою природної мови.

Приклад: Jupyter Notebooks та NLTK

Jupyter Notebooks дозволяє вченим проводити наукові дослідження та обмінюватися результатами. NLTK (Natural Language Toolkit) використовується для обробки природної мови.

SciPy - використовується для наукових досліджень та обробки природної мови. Наприклад, NASA використовує SciPy для обробки наукових даних.

- Бази даних та робота з Big Data:

Python забезпечує бібліотеки, такі як SQLAlchemy для роботи з реляційними базами даних (Dropbox використовує SQLAlchemy для роботи з базою даних), та взаємодіє з системами обробки великих обсягів даних, такими як Apache Spark.

- Автоматизація та сценарії:

Python використовується для створення автоматизованих сценаріїв і скриптів для різних завдань, включаючи адміністрування систем, роботу з файлами, взаємодію з API та інше.

Приклад: Ansible та Selenium

Ansible використовує Python для автоматизації задач системного адміністрування. Selenium використовується для автоматизації веб-браузерів. Spotify використовує Ansible для автоматизації своїх серверів.

- Графічний дизайн та ігрова розробка:

У галузі графічного дизайну Python використовується для створення та редагування зображень, а також для розробки ігор за допомогою бібліотек, таких як Pygame.

- Системне адміністрування:

Python використовується для автоматизації рутинних задач системного адміністрування, таких як моніторинг, резервне копіювання та обслуговування серверів.

Приклад: SaltStack та Fabric

SaltStack використовує Python для автоматизації конфігурації та управління серверами. Fabric - для виконання задач системного адміністрування через SSH.

- Робототехніка та вбудовані системи:

Python застосовується для програмування мікроконтролерів та вбудованих систем, таких як Raspberry Pi або Arduino, для створення різноманітних електронних проектів.

Це лише декілька прикладів, і використання Python продовжує розширюватися в нових областях завдяки своїй гнучкості та спільноті розробників.

Це лише кілька прикладів, і використання Python розповсюджується в багатьох інших областях, таких як фінанси, охорона здоров'я, енергетика, тестування програмного забезпечення, веб-сервери та інше. Python стає все більш універсальною мовою програмування і використовується в різних індустріях.

2.3.4 Спільнота та підтримка

Python має велику та активну спільноту розробників, що сприяє обміну досвідом, наданню порад та розвитку мови. Офіційна документація Python також є високоякісною та зручною для використання.

Ці аспекти роблять Python потужним інструментом для широкого спектру завдань від простих сценаріїв до складних веб-програм та наукових досліджень. Незалежно від вашого досвіду в програмуванні, Python може виявитися корисним для вас завдяки своїм зручним інструментам та багатий функціональності.

2.4 Бібліотека aiogram

Aiogram — це платформа на основі Python, спеціально розроблена для створення ботів Telegram. Aiogram надає повний набір інструментів і функцій, які спрощують процес розробки бота. Одним із ключових понять в aiogram є стан і система обробки. Система стану дозволяє визначати стан чат-бота та керувати ним, а система обробки дозволяє обробляти різні типи подій і команд. Розуміючи ці основні поняття, можна створювати ботів, які зможуть розумно реагувати на взаємодії користувача.

2.4.1 Ключові особливості Aiogram.

Aiogram пропонує широкий спектр функцій, які спрощують розробку ботів. По-перше, він надає простий та інтуїтивно зрозумілий API, який дозволяє безперешкодно взаємодіяти з API Telegram Bot, незалежно від того, чи потрібно надсилати повідомлення, редагувати повідомлення чи обробляти вбудовані запити. Крім того, aiogram підтримує вбудовані клавіатури, які дозволяють створювати інтерактивні меню та кнопки у чат-боті. Ця функція може значно покращити взаємодію з користувачем і зробити чат-бота більш привабливим.

2.4.2 Налаштування Aiogram і Telegram API

Перш ніж почати створювати бота за допомогою Aiogram, потрібно створити бота на платформі Telegram (див. розділ 2.1). Після створення бота отримано маркер API, який використовується для автентифікації чат-бота за допомогою API Telegram. Далі потрібно встановити Aiogram, виконавши таку команду в терміналі або командному рядку:

pip install aiogram

Після встановлення Aiogram можна запустити новий скрипт Python та імпортувати необхідні модулі:

```
Імпорт журналювання з aiogram
importlogging
from aiogram import Bot, Dispatcher, executor, types
Налаштування журналювання
logging.basicConfig(level=logging.INFO)
Ініціалізація бота та диспетчера
bot = Bot(token="YOUR_API_TOKEN")
storage = MemoryStorage()
dp = Dispatcher(bot, storage=storage)
```

2.4.3 Основні функції Aiogram

Асинхронність: Aiogram використовує асинхронний підхід для зручного та ефективного управління різними аспектами бота. Це дозволяє створювати швидкі та відгукні боти.

Високорівневий API: бібліотека надає високорівневий API, що дозволяє розробникам легко виконувати основні операції, такі як відправлення повідомлень, робота з клавіатурами та обробка подій.

Обробка подій: Aiogram надає зручний механізм обробки різних подій, таких як отримання повідомлень, клавіатур, запитань та інших.

Робота з медіа: вбудована підтримка для роботи з медіа-файлами, включаючи зображення, аудіо, відео та файли. Це дозволяє легко обробляти та взаємодіяти з різними типами контенту.

Система команд: Aiogram має вбудовану систему команд, яка спрощує взаємодію з користувачами через текстові команди.

2.4.4 Вивчення стану та системи обробки Aiogram

Система стану та обробки — це фундаментальний аспект Aiogram, який дозволяє керувати станом чат-бота та обробляти різні типи подій і команд. В Aiogram стан представляє поточний стан бота, а обробник — це функція, яка виконується, коли запускається певна подія чи команда. Ефективно використовуючи стан і систему обробки, можна створювати ботів, які розумно реагують на взаємодії користувача.

Щоб визначити стан в Aiogram, можна використати декоратор dp.state і надати унікальну назву для стану.

Наприклад:

@dp.message_handler(state="MY_STATE") async def handle_message(message: types.Message): # Обробка повідомлення в стані "MY_STATE"

await message.answer("Привіт від MY_STATE")

Для обробки команд можна використовувати декоратор dp.message_handler і вказати команду, яку необхідно обробити.

Наприклад:

@dp.message_handler(commands=["start"]) async def handle_start(message: types.Message): # Команда обробки "/start" очікує

message.answer("Ласкаво просимо до мого бота!")

2.4.5 Робота з методами API Aiogram

Aiogram надає повний набір методів API, які дозволяють взаємодіяти з API Telegram Bot. Ці методи дозволяють надсилати повідомлення, редагувати повідомлення, обробляти вбудовані запити тощо. Розглянемо деякі з найбільш часто використовуваних методів API в Aiogram і як їх можна використовувати для покращення функціональності чат-бота.

Щоб надіслати повідомлення за допомогою Aiogram, можна скористатися методом bot.send_message.

Наприклад:

```
await bot.send_message(chat_id, text="Привіт, світ!")
```

Для редагування повідомлення можна скористатися методом bot.edit_message_text. Це дозволяє змінити текст наявного повідомлення.

Наприклад:

```
await bot.edit_message_text(chat_id, message_id, text="Новий текст")
```

Розширені функції Aiogram — вбудовані клавіатури, вбудовані запити тощо.

Aiogram пропонує низку розширених функцій, які можуть вивести чат-бота на новий рівень. Однією з таких функцій є можливість створювати вбудовані клавіатури. Вбудовані клавіатури дозволяють відображати інтерактивні кнопки в повідомленнях чат-бота, даючи користувачам можливість робити вибір або запускати дії одним дотиком.

Щоб створити вбудовану клавіатуру, ви можна використовувати клас types.InlineKeyboardMarkup. Цей клас дозволяє визначати рядки та кнопки на клавіатурі.

Наприклад:

```
keyboard = types.InlineKeyboardMarkup()
button1 = types.InlineKeyboardButton(text="Button 1",
callback_data="button1")
```

```
button2 = types.InlineKeyboardButton(text="Button 2",
callback_data="button2")
 keyboard.row(button1, button2)
```

2.4.6 Обробка помилок

Розробляючи бота Telegram, дуже важливо виправляти помилки. Aiogram надає механізми обробки помилок, які дозволяють виловлювати та обробляти винятки, які можуть виникнути під час роботи бота.

Для цього можна використовувати декоратор @dp.errors_handler, щоб визначити функцію, яка буде викликана, коли станеться помилка. Усередині функції можна зареєструвати помилку, надіслати повідомлення користувачеві або виконати будь-яку іншу дію, щоб належним чином усунути помилку.

2.4.7 Тестування та налагодження

Тестування та налагодження є ключовими кроками в процесі розробки, щоб переконатися, що чат-бот працює належним чином. Aiogram надає утиліти для тестування, які дозволяють імітувати взаємодію користувача та перевіряти поведінку бота.

Можна використовувати модуль aiogram.contrib.testing для створення тестів і затвердження очікуваних результатів. Крім того, Aiogram надає функцію журналювання, яка дозволяє реєструвати повідомлення та налагоджувати бота під час розробки.

2.4.8 Розгортання та масштабованість

Aiogram підтримує різні варіанти розгортання, включаючи запуск бота на локальній машині, розгортання його на хмарному сервері або використання

безсерверних платформ. Можна вибрати варіант розгортання, який найкраще відповідає потребам, враховуючи такі фактори, як вартість, масштабованість і вимоги до обслуговування.

2.4.9 Приклади використання

Надсилання привітання новим користувачам:

```
from aiogram import Bot, Dispatcher, types
from aiogram.types import Message
API_TOKEN = 'your_token'
bot = Bot(token=API_TOKEN)
dp = Dispatcher(bot)
@dp.message_handler(commands=['start'])
async def send_welcome(message: Message):
    await message.answer("Привіт! Я твій новий бот. Я готовий відповідати на твої команди.")

if __name__ == '__main__':
    from aiogram import executor
    executor.start_polling(dp, skip_updates=True)
```

Відповідь на повідомлення з зображенням:

```
@dp.message_handler(content_types=types.ContentType.PHOTO)
async def handle_photo(message: Message):
await message.answer("Дякую за фото! Я знаю, що з ним робити.")
```

Висновок щодо використання бібліотеки Aiogram для створення чат-ботів дозволяє визначити її як потужний та зручний інструмент для розробки чат-ботів на платформі Telegram. Aiogram виділяється своєю простотою та зручністю використання, забезпечуючи розробникам інтуїтивний інтерфейс для взаємодії з

Telegram API. Підтримка асинхронного програмування, розширений функціонал для роботи з різними елементами, активна спільнота та постійні оновлення роблять Aiogram ефективним інструментом з відкритими можливостями та стабільною підтримкою.

2.5 Використання магічних фільтрів

Проведено дослідження використання магічних фільтрів для покращення взаємодії чат-ботів з користувачами, які опубліковано у статті «Як Python підвищує ефективність чат-ботів: покращення взаємодії з користувачем за допомогою магічних фільтрів в aiogram» [25].

Відповідно до дослідження можна зазначити, що використання магічних фільтрів у структурах чат-ботів значно покращує взаємодію та зробить роботу з користувачем ефективнішою. Розробники почали впроваджувати магічні фільтри у своїх структурах чат-ботів, зокрема для:

Розпізнавання сутностей: чат-бот може ефективно виявляти та виділяти конкретні елементи з повідомлень користувача за допомогою спеціальних фільтрів. Наприклад, у чат-бота для служби доставки їжі може бути застосований магічний фільтр, який визначатиме продукти харчування, згадані у повідомленні користувача, і витягуватиме відповідні деталі, такі як кількість та уподобання. Це дозволяє чат-боту надавати персоналізовані рекомендації та спрощувати процес замовлення.

Аналіз настрою: магічні фільтри також можна використовувати для визначення емоційного настрою повідомлень користувачів. Використовуючи алгоритми аналізу настроїв, чат-бот може визначити, чи є повідомлення користувача позитивним, негативним чи нейтральним. Цю інформацію можна використовувати для налаштування відповідей чат-бота, забезпечуючи чуйну та відповідну підтримку користувачам на основі їхніх емоцій.

Розпізнавання мови: у світі, де взаємодія здійснюється кількома мовами, чат-боти можуть використовувати магічні фільтри для автоматичного визначення мови повідомлення користувача та відповідного надання відповідей. Це дозволяє уникнути необхідності явно вказувати мову спілкування з користувачем, забезпечуючи безперебійну та зручну взаємодію.

MagicFilters — це зовнішній пакет, який підтримує основна бібліотека aiogram. За замовчуванням він інсталюється разом із aiogram, а також доступний у PyPi - magic-filter (https://pypi.org/project/magic-filter/) .

Це означає, що можна інсталювати та використовувати його з іншими бібліотеками та у своїх власних проектах, незалежно від того, чи інстальована бібліотека aiogram. Проте, слід зазначити, що aiogram має невелике розширення над magic-filter. Якщо є необхідність використовувати це розширення, то слід імпортувати його з aiogram замість пакета magic_filter:

from aiogram import F

замість

from magic_filter import F

2.5.1 Вивчення можливих дій за допомогою магічних фільтрів

Об’єкт MagicFilter підтримує кілька основних логічних операцій над атрибутами об’єкта.

- Існує або не існує

Дія за замовчуванням перевіряє, існує чи ні певний атрибут.

Наприклад:

F.photo перевіряє наявність атрибута photo в об’єкті.

- Дорівнює

Оператор == перевіряє, чи дорівнює атрибут певному значенню.

Наприклад:

F.text == 'hello' перевіряє, чи текст атрибута дорівнює рядку 'hello'.

- Є одним із

Метод in_ або оператор @ можна використовувати, щоб перевірити, чи є атрибут одним із значень у ітерації.

Наприклад:

F.from_user.id.in_({42, 1000, 123123}) перевіряє, чи дорівнює атрибут from_user.id будь-якому зі значень у наборі {42, 1000, 123123}.

- Містить

Метод contains перевіряє, чи містить атрибут рядка певний підрядок.

Наприклад:

F.text.contains('foo') перевіряє, чи текст атрибута містить підрядок 'foo'.

- Startswith/Endswith

Методи startswith і endswith можна застосовувати лише до текстових атрибутів.

Наприклад:

F.text.startswith('foo') перевіряє, чи починається текст атрибута з рядка 'foo'.

- Регулярний вираз

Метод regexp дозволяє використовувати регулярні вирази для зіставлення значень атрибутів.

Наприклад:

F.text.regexp(r'Hello, .+') перевіряє, чи текст атрибута відповідає шаблону регулярного виразу 'Hello, .+'.

- Власні функції

Можна використовувати власну функцію як дію.

Наприклад:

F.chat.func(lambda chat: chat.id == -42) перевіряє, чи дорівнює атрибут chat.id -42 за допомогою власної функції.

- Інвертування результату

Будь-яку доступну операцію можна інвертувати за допомогою оператора побітової інверсії ~.

Наприклад:

~(F.text == 'spam') інвертує результат операції F.text =='spam'.

- Комбінування дій

Усі операції можна комбінувати за допомогою побітових операторів і (&) та або (|). Наприклад:

(F.from_user.id == 42) & (F.text == 'admin') перевіряє, чи атрибути from_user.id і text задовольняють заданим умовам.

- Модифікатори атрибутів - маніпуляції рядками

Магічні фільтри також дозволяють використовувати модифікатори атрибутів, зокрема для рядкових атрибутів. Ці модифікатори можна використовувати для керування регістром атрибута рядка.

Наприклад:

F.text.lower() == 'test' перевіряє, чи текст атрибута дорівнює рядку 'test' у нижньому регістрі.

F.text.upper().in_({'FOO', 'BAR'}) перевіряє, чи версія тексту атрибута у верхньому регістрі дорівнює будь-якому з рядків у наборі {'FOO', 'BAR'}.

F.text.len() == 5 перевіряє, чи довжина тексту атрибута дорівнює 5.

- Отримати результат фільтра як аргумент обробника

Хоча ця функція недоступна безпосередньо у magic-filter, її можна використовувати в поєднанні з aiogram. Імпортувавши F з aiogram, ви можете використовувати цю функцію.

Наприклад:

```
from aiogram import F
...
@router.message(F.text.regexp(r"^(\d+)$").as_("digits"))
async def any_digits_handler(message: Message, digits: Match[str]):
await message.answer(html.quote(str(digits)))
```

У цьому прикладі фільтр F.text.regexp(r"^(\d+)$").as_("digits") використовується для отримання цифр із текстового атрибута повідомлення. Отримані цифри потім передаються як аргумент функції any_digits_handler.

Впровадження магічних фільтрів у чат-ботах може мати безліч переваг як для бізнесу, так і для клієнтів. Нижче перераховано декілька ключових плюсів:

Покращення взаємодії з користувачем: магічні фільтри надають можливість чат-ботам краще розуміти та обробляти введені користувачем дані, забезпечуючи більш персоналізовану та ефективну комунікацію. Це сприяє загальному підвищенню задоволеності та лояльності клієнтів.

Економія часу та коштів: завдяки автоматизації завдань, таких як розпізнавання об'єктів і аналіз настроїв, магічні фільтри значно скорочують час і зусилля, необхідні для обробки повідомлень користувачів. Це призводить до ефективної економії ресурсів та дозволяє чат-ботам швидше та точніше реагувати.

Покращена масштабованість і гнучкість: магічні фільтри легко адаптуються до змінних потреб і вподобань користувачів. По мірі розвитку чат-бота можна оновлювати та вдосконалювати магічні фільтри для обробки нових шаблонів або намірів, забезпечуючи актуальність та ефективність.

Статистика на основі даних: магічні фільтри генерують цінну інформацію про поведінку та вподобання користувачів. Аналізуючи ці дані, можна глибше розуміти своїх клієнтів і приймати рішення на основі даних для поліпшення продуктивності та ефективності чат-бота.

Магічні фільтри на основі Python і реалізовані в aiogram пропонують потужний набір інструментів для покращення взаємодії з чат-ботами. Використовуючи методи обробки природної мови, магічні фільтри дозволяють чат-ботам розуміти та інтерпретувати введені користувачем дані більш розумно та контекстно. Це призводить до більш ефективної та персоналізованої взаємодії, що зрештою призводить до підвищення рівня задоволеності та лояльності клієнтів.

Висновки до розділу 2

Виходячи з проведеного аналізу можна зробити висновки, що використання Telegram Bot API з @BotFather, в поєднанні з мовою програмування Python та бібліотекою Aiogram, створює зручний та ефективний інструмент для створення різноманітних чат-ботів на платформі Telegram.

Цей стек технологій дозволяє розробникам швидко та легко реалізувати функціональність ботів та забезпечує їхню стабільність та продуктивність та має кілька значущих переваг, зокрема:

Простота створення: Telegram Bot API надає простий інтерфейс для створення та керування ботами.

@BotFather дозволяє швидко отримати токен та здійснити базову конфігурацію бота.

Можливості взаємодії: Telegram Bot API надає широкі можливості взаємодії, включаючи роботу з повідомленнями, клавіатурами, зображеннями, файлами та багато іншого.

Aiogram допомагає спростити обробку різних типів повідомлень та подій.

Підтримка різних типів ботів: Telegram Bot API підтримує різні типи ботів, включаючи звичайних, адміністраторів та інлайн-ботів.

Aiogram робить розробку різних функціональностей для різних типів ботів більш зручною.

Асинхронність та продуктивність: мова програмування Python та бібліотека Aiogram підтримують асинхронний код, що дозволяє створювати ефективні та продуктивні боти, особливо при великій кількості одночасних запитань.

Спільнота та ресурси: Python має велику активну спільноту розробників, що дозволяє легко знаходити рішення для різних проблем.

Aiogram має добре документовану базу коду та ресурси для розробників.

Таким чином, на підставі зроблених висновків, було прийнято рішення щодо створення чат-боту для Telegram за допомогою мови програмування Python з використанням бібліотеки Aiogram.

3 ОПИС РОЗРОБКИ ЧАТ-БОТУ

3.1 Створення чат-боту у Telegram за допомогою @BotFather

Для початку запускаємо чат-бот @BotFather командою /start та виконуємо наступну послідовність дій:

1) За допомогою команди /newbot запускаємо процес створення нового бота: згідно з підказками вводимо ім’я, та унікальну назву, яка має закінчуватись на `bot` (Рисунок 3.1).

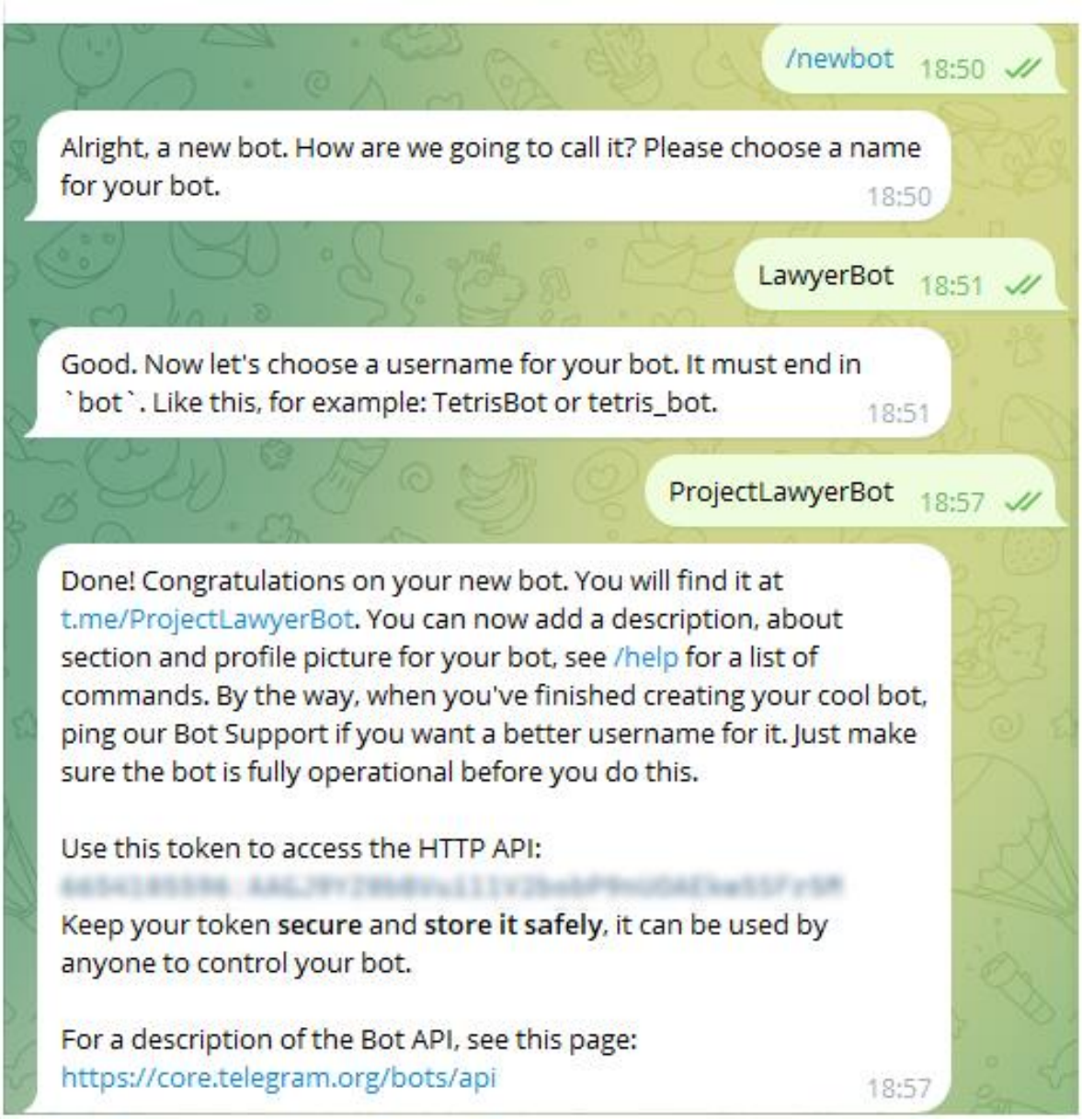


Рисунок 3.1 – Створення нового бота за допомогою @BotFather

2) За допомогою команди /setname можна змінити ім’я створеного бота (Рисунок 3.2).

Рисунок 3.2 – Зміна імені створеного чат-бота

3) За допомогою команди /setuserpic додаємо фото профілю створеного бота (Рисунок 3.3).

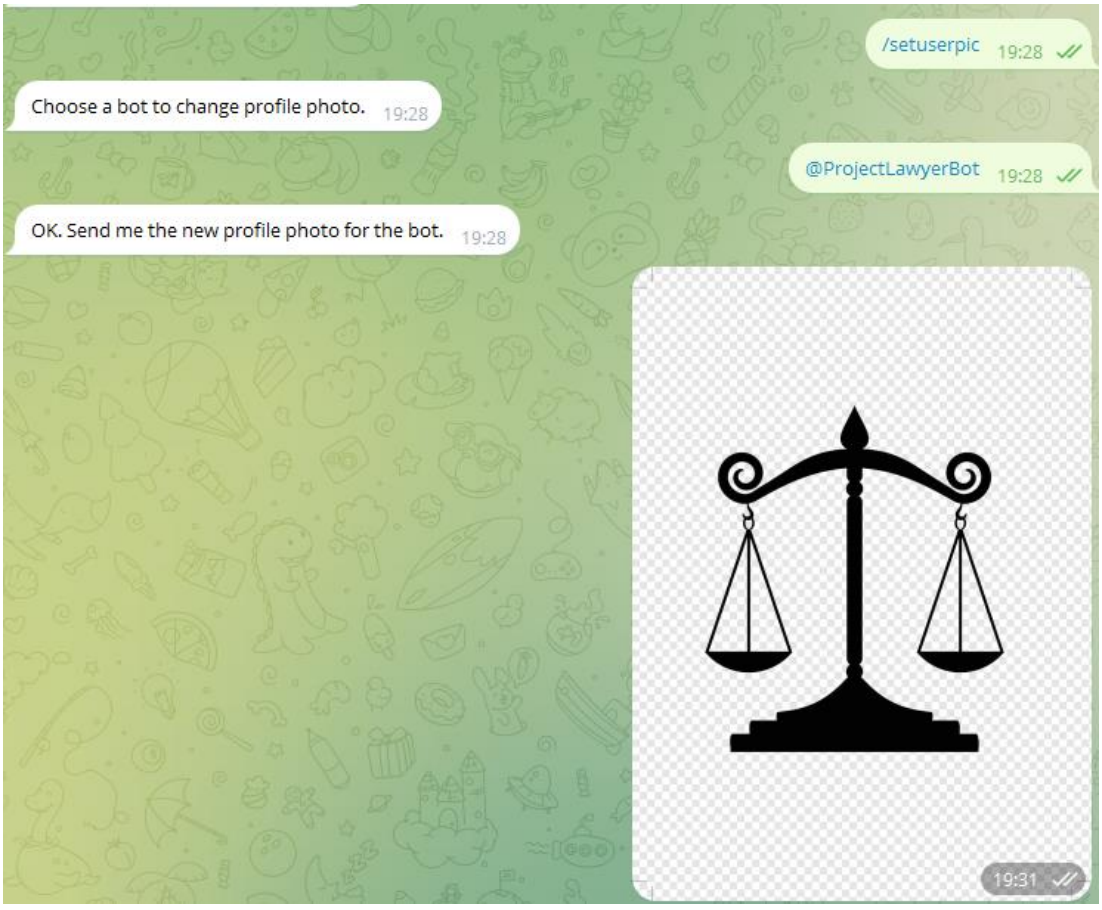


Рисунок 3.3 – Встановлення фото профілю створеного чат-бота

4) За допомогою команди /token отримаємо унікальний токен для створеногочта-бота. Токен - це унікальний ключ, який дозволяє отримати доступ до управління ботом та його функціоналу. Важливо вказати цей токен у програмному коді, оскільки без нього програмі буде невідомо, як саме взаємодіяти з створеним чат-ботом.

Щоб уникнути можливого захоплення керування чат-ботом третіми особами (шахрайство), не можна передавати токен нікому. Токен є унікальним шифром для створеного чат-бота.

Виконавши ці дії, створено чат-бота на ім’я Lawyer TeleBot (Рисунок 3.4), проте, для того щоб навчити його виконувати будь-які функції, потрібно розробити відповідний програмний код.

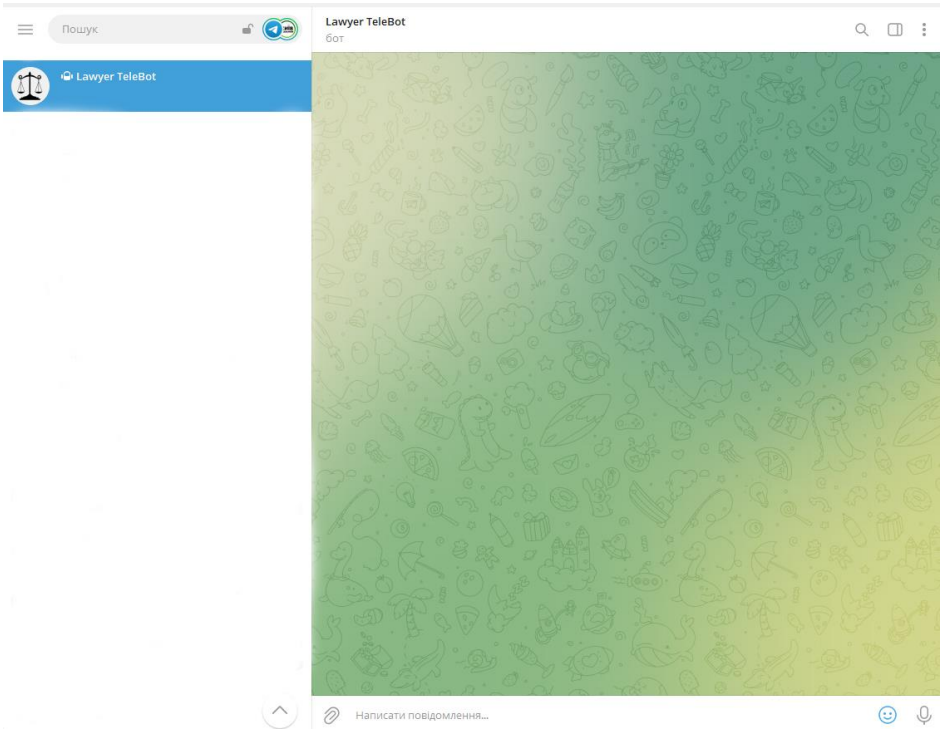


Рисунок 3.4 – Створений чат-бот за допомогою @BotFather

3.2 Структура та основні задачі чат-боту

Основними задачами чат-боту є наступні (Рисунок 3.5):

• Реєстрація користувача: користувач автоматично передає свій ID для реєстрації у чат-боті.

• Доступ до шаблонів документів: зареєстрований користувач отримує можливість користуватися шаблонами юридичних документів компанії, таких як акт, довіреність, шаблон NDA.

• Запит на консультацію: зареєстрований користувач може надіслати запит на телефонний дзвінок або особисту зустріч для консультації з юристом компанії.

• Відстеження роботи з документами: зареєстрований користувач може направити документ на розгляд юристам компанії та слідкувати за його статусом в роботі.

• Резервування номера документу: зареєстрований користувач компанії має можливість зарезервувати унікальний номер для юридичного документу.

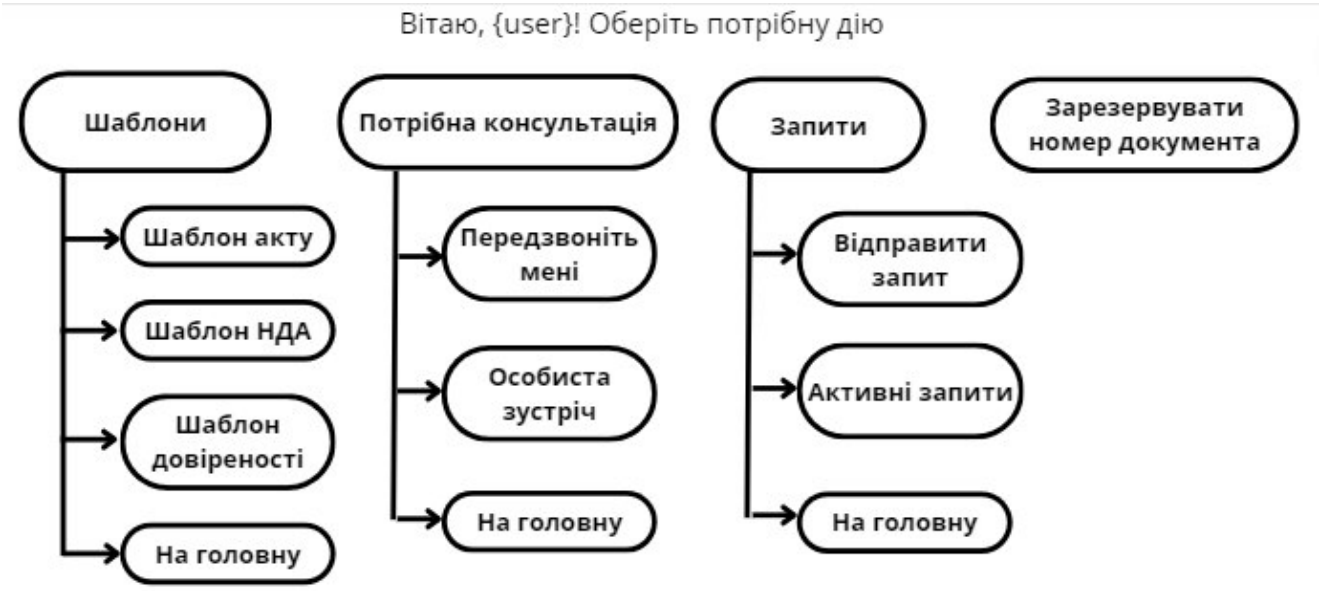


Рисунок 3.5 – Блок-схема основних задач чат-боту

• Обробка запитів: юристи компанії автоматично отримують відповідну інформацію від користувачів та мають доступ до документів, що надіслані на розгляд.

3.3 Розробка чат-боту

Для реалізації чат-боту використовувались мова програмування Python та PyCharm Community Edition 2023.2.2 - інтегроване середовище розробки для мови програмування Python.

Проект включає два файли:

1) keyboard.py – містить назви та розміщення кнопок у чат-боті (Додаток 1)

2) main.py – містить опис дій, що виконуються при натисканні користувачем на відповідну кнопку (Додаток 2).

3.4 Тестування чат-боту

Процедура тестування виконувалась за наступними кроками:

1) За допомогою пошуку, знаходимо у мессенжері Telegram чат-бот @ProjectLawyerBot (Рисунок 3.6).

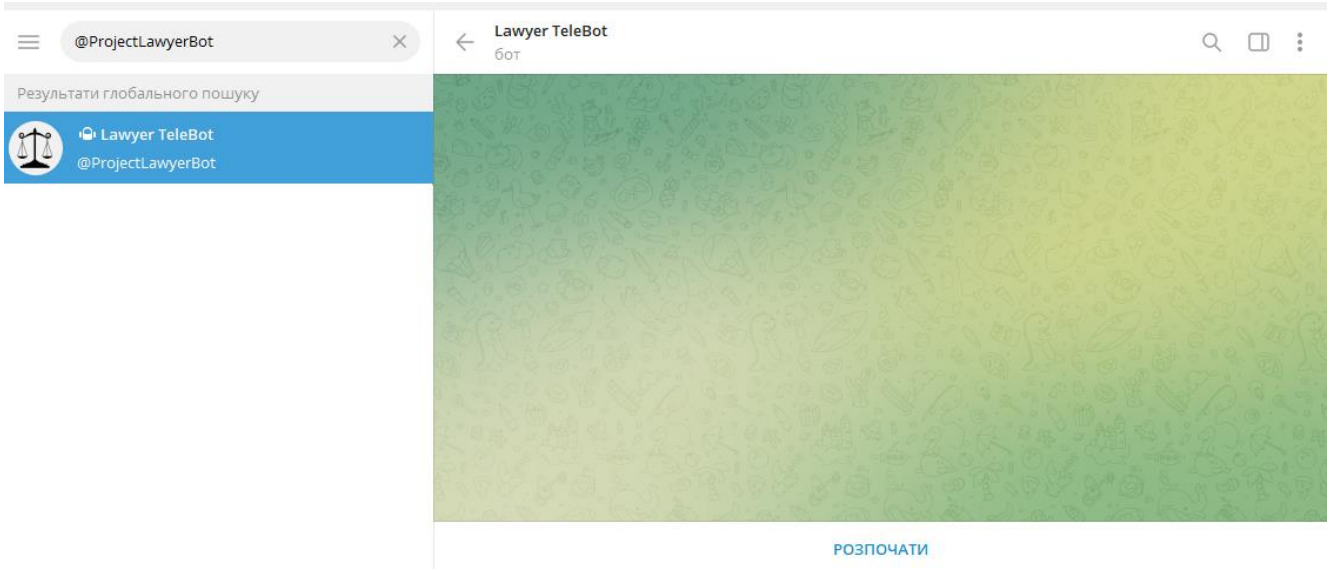


Рисунок 3.6 – Пошук створеного чат-боту

2) Починаємо діалог з чат-ботом натиснувши кнопку «Розпочати» та отримуємо запрос пройти реєстрацію (Рисунок 3.7).

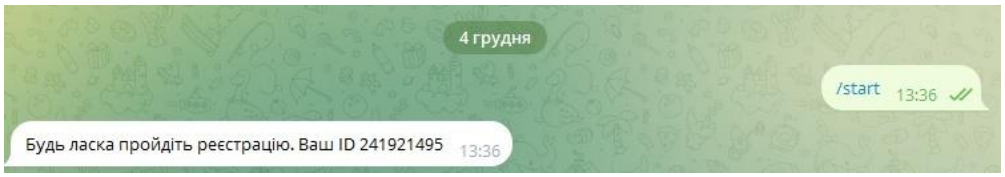


Рисунок 3.7 – Початок діалогу з створеним чат-ботом

3) Адміністратор чат-боту отримує повідомлення про нового користувача та додає його до користувачів чат-боту (Рисунок 3.8).

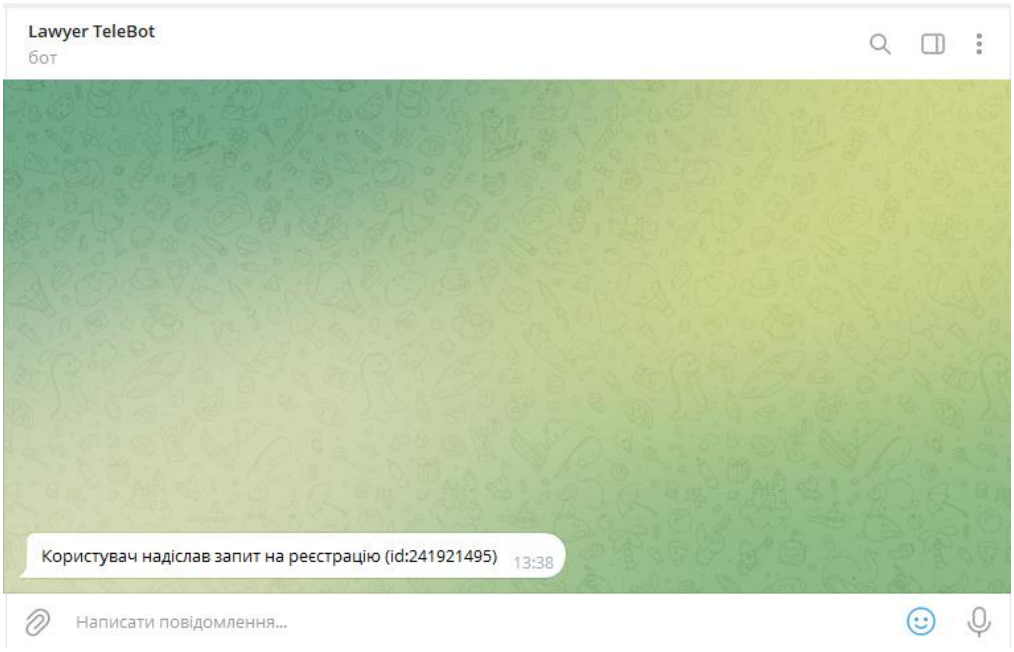


Рисунок 3.8 – Реєстрація нового користувача

4) Після проходження реєстрації отримуємо доступ до можливостей чат-боту (Рисунок 3.9).

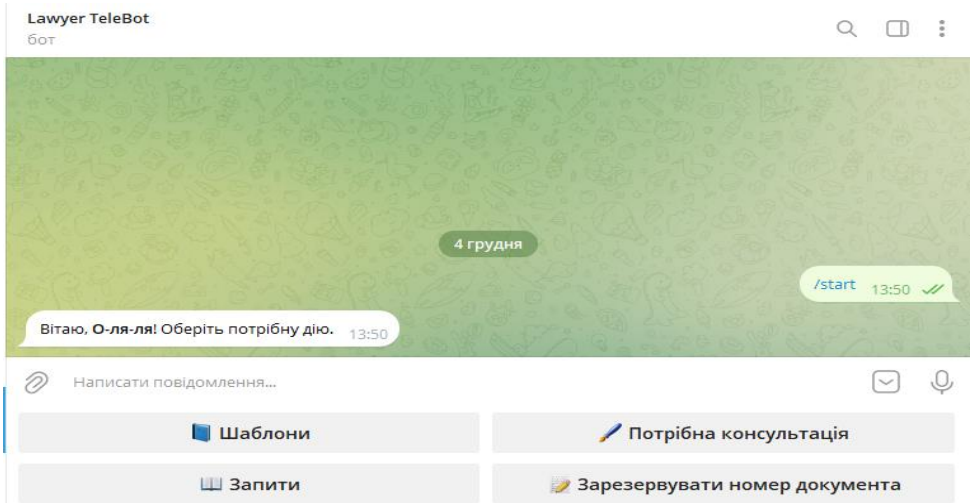


Рисунок 3.9 – Початок використання функцій чат-бота

5) Натиснувши кнопку «Шаблони» заходимо в підменю та надсилаємо запити на надання шаблону акту, довіреності та шаблону NDA і отримуємо відповідні документи (Рисунок 3.10).

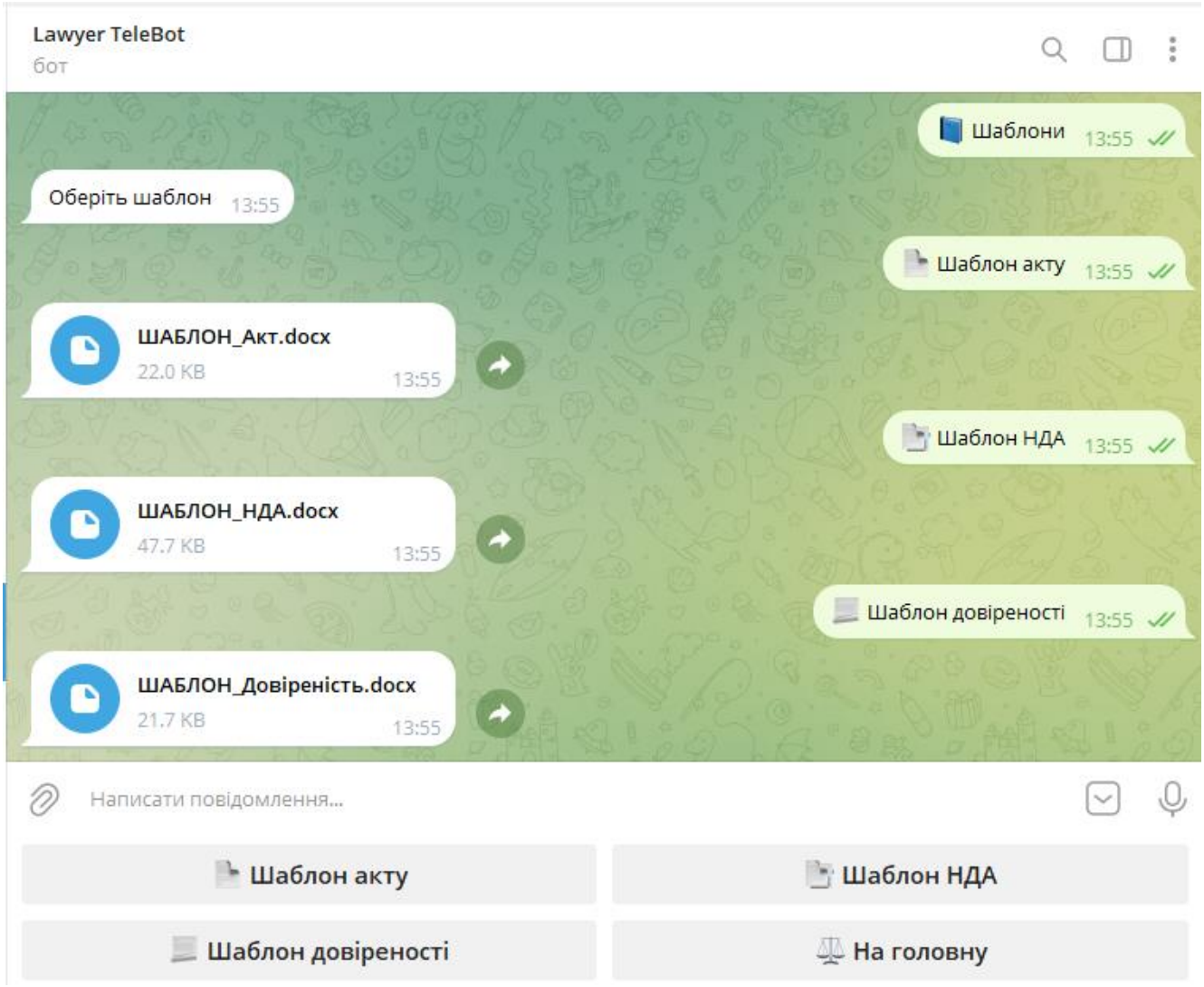


Рисунок 3.10 – Функції меню «Шаблони»

6) Повертаємось на головну сторінку.

7) Натискуючи кнопку «Потрібна консультація», ми входимо в підменю та натискаємо кнопки «Передзвоніть мені», автоматично відправляючи номер телефону для зв'язку, та «Особиста зустріч», надсилаючи запит юридичному відділу щодо потреби в особистій зустрічі (Рисунок 3.11). Юристи компанії отримують відповідні запити з номером телефону за яким можна зв’язатись з користувачем та запит на особисту зустріч.

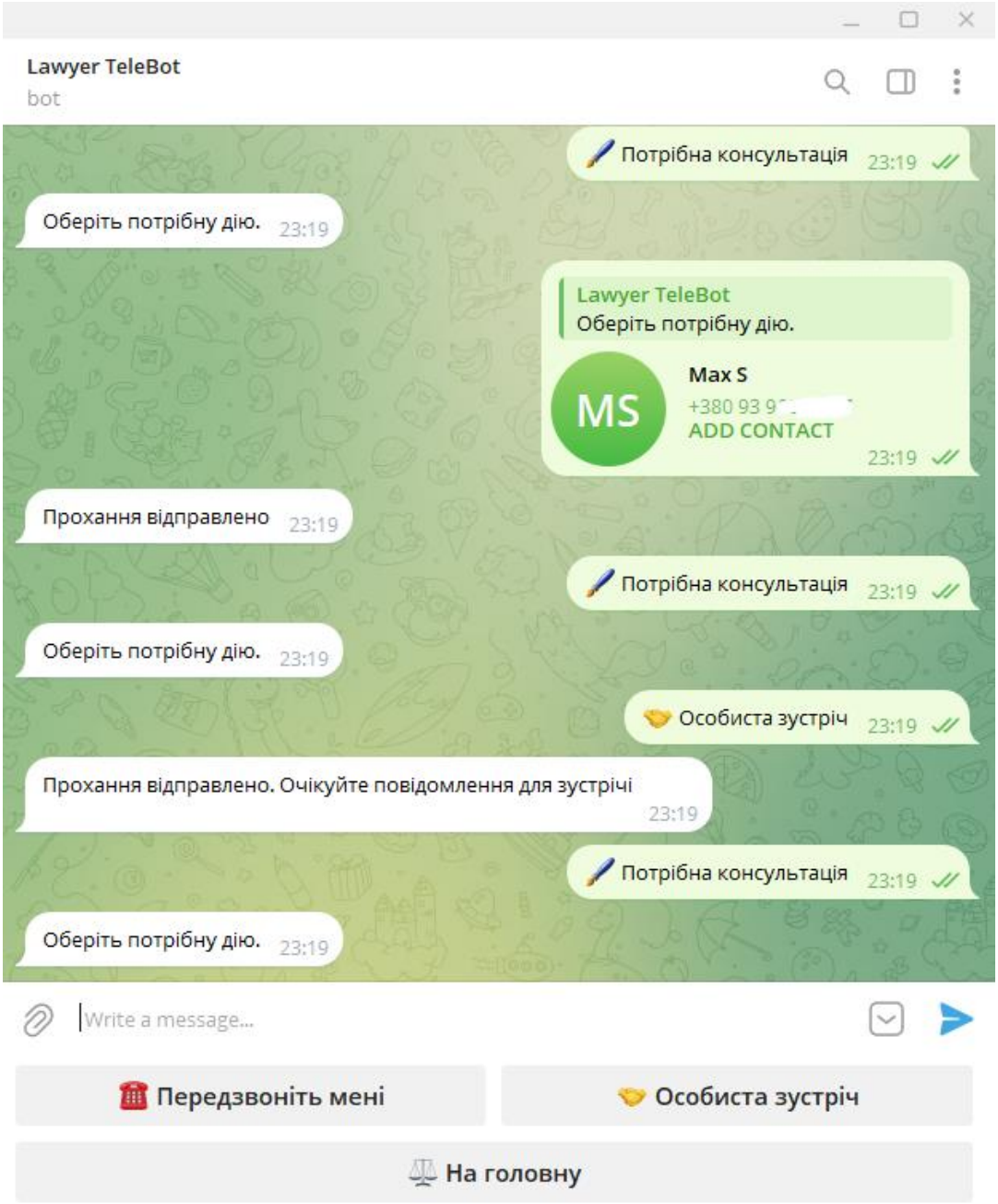


Рисунок 3.11 – Функції меню «Потрібна консультація»

8) Натиснувши кнопку «Запити» заходимо в підменю та натискаємо кнопку «Відправити запит» (Рисунок 3.12). Відповідно до повідомлення «Відправте файл в чат» відправляємо потрібний файл, який автоматично загружається для обробки юридичним відділом, присвоюючи йому ID користувача та назву документу (Рисунок 3.13).

Натиснувши кнопку «Активні запити» дивимось, які надіслані файли ще знаходяться в роботі (Рисунок 3.12).

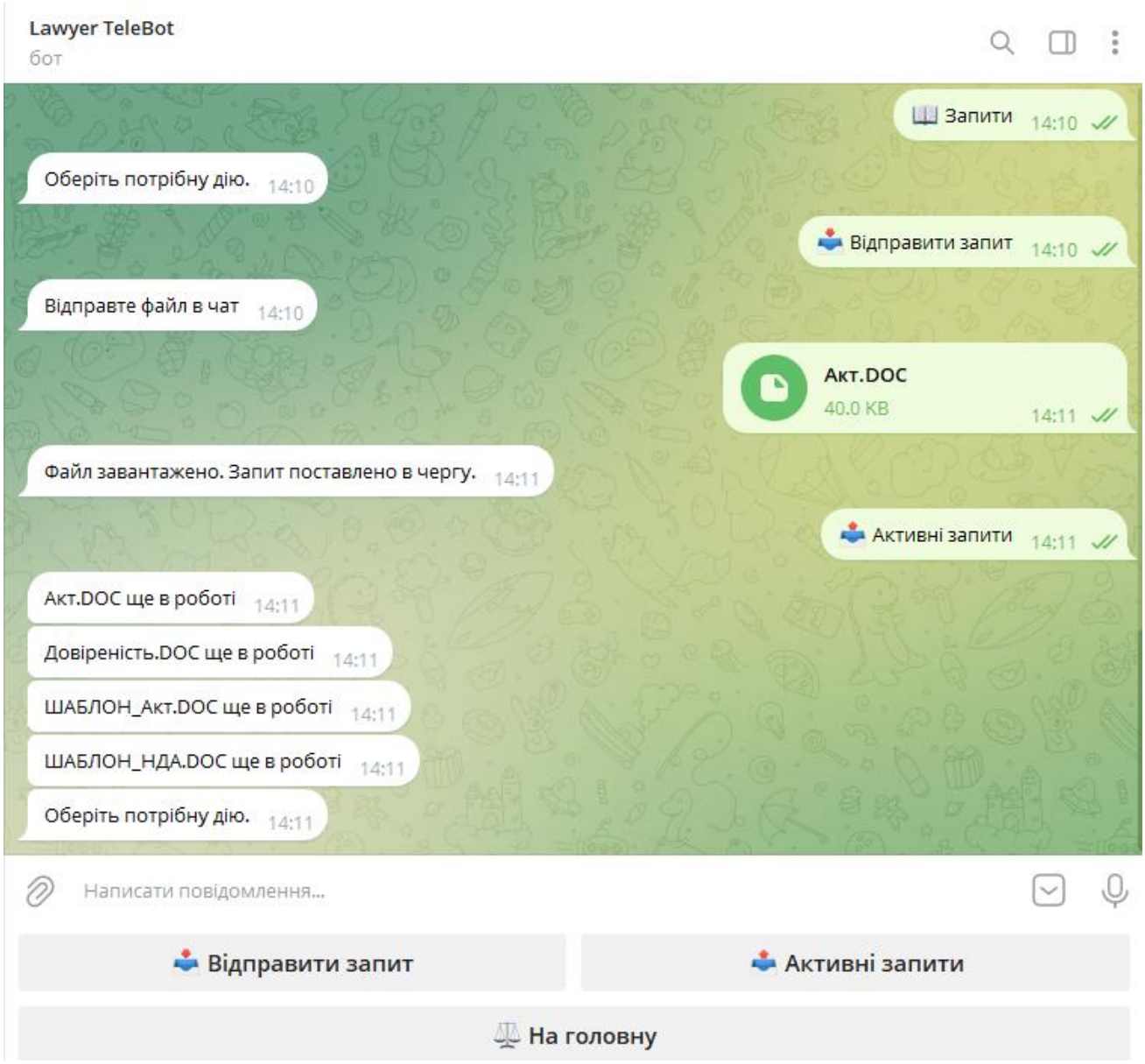
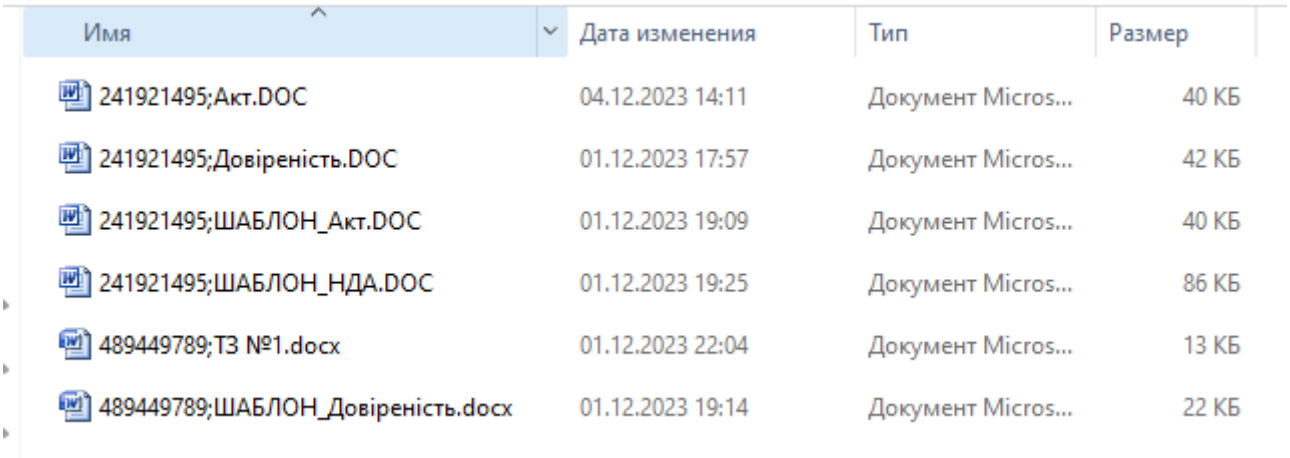


Рисунок 3.12 – Функції меню «Запити»



Имя	Дата изменения	Тип	Размер
241921495;Акт.DOC	04.12.2023 14:11	Документ Micros...	40 КБ
241921495;Довіреність.DOC	01.12.2023 17:57	Документ Micros...	42 КБ
241921495;ШАБЛОН_Акт.DOC	01.12.2023 19:09	Документ Micros...	40 КБ
241921495;ШАБЛОН_НДА.DOC	01.12.2023 19:25	Документ Micros...	86 КБ
489449789;ТЗ №1.docx	01.12.2023 22:04	Документ Micros...	13 КБ
489449789;ШАБЛОН_Довіреність.docx	01.12.2023 19:14	Документ Micros...	22 КБ

Рисунок 3.13 – Файли, які завантажені користувачами

9) Повертаємось на головну сторінку.

10) Натискаємо на кнопку «Зарезервувати номер документу» та отримуємо унікальний порядковий номер (Рисунок 3.14).

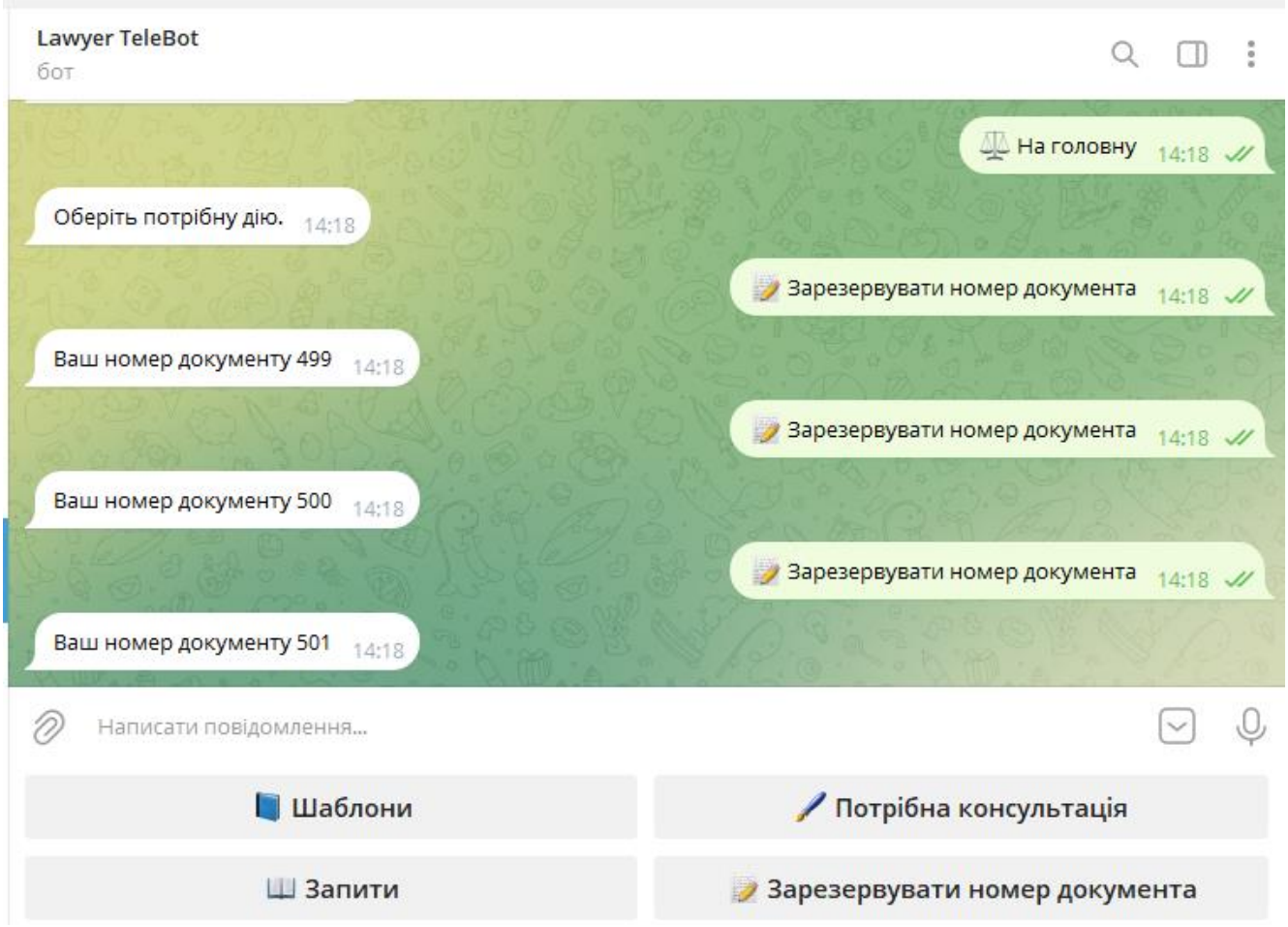


Рисунок 3.14 – Отримання унікального порядкового номеру документа

Юристи компанії отримують відповідне повідомлення, про те, що користувач зарезервував номер документу (Рисунок 3.16).

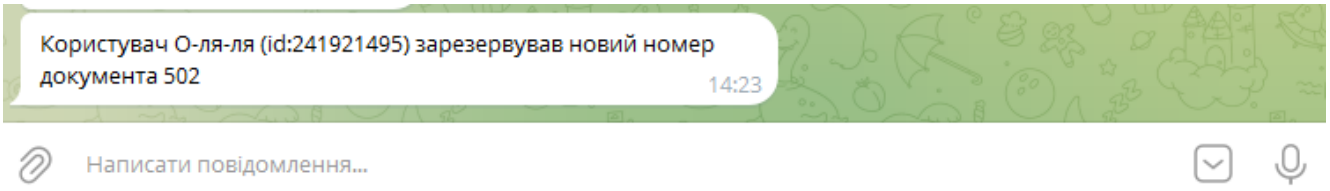


Рисунок 3.15 – Повідомлення про резервування номеру документу

Висновки до розділу 3

Як результат, у даному розділі представлено розробку чат-боту для юридичного відділу компанії, який відзначається високою ефективністю та можливістю забезпечити потрібний рівень обслуговування. Результати тестування підтверджують, що чат-бот успішно виконує свої функції та відповідає поставленим вимогам.

За потреби внесення нових завдань чи додавання додаткових функцій, функціонал чат-боту готовий до легкого розширення. Це робить його гнучким і готовим відповідати зростаючим потребам користувачів. Додатково, можливість розширення функціоналу забезпечує довгострокову усталеність та використання чат-боту як інструменту, що забезпечує підтримку та зручність у роботі з юридичним відділом.

ВИСНОВКИ

Під час виконання дипломного проекту був проведений детальний аналіз предметної області та визначені ключові принципи функціонування чат-ботів. Також розглянуті основні методики для їх створення. Проведено аналіз загальних принципів і інструментів для побудови чат-ботів у месенджері Telegram та програмних інструментів для їх розробки.

В якості основних засобів розроблення були обрані мова програмування Python, бібліотека Aiogram та Telegram Bot API. Ці інструменти визначились як оптимальні для створення чат-бота з урахуванням потреб користувачів та функціоналу чат-боту.

Як результат проектування, був успішно створений чат-бот для юридичного відділу компанії. За допомогою цього бота користувачі можуть автоматично отримувати шаблони документів, надсилати свої документи на розгляд та надавати запити на консультації від юристів компанії. Це забезпечує зручність та ефективність взаємодії користувачів з юридичним відділом через цифровий канал.

Проведений аналіз та створення чат-бота для юридичного відділу виявили важливість та актуальність використання такого інноваційного інструменту в сучасному бізнес-середовищі. З врахуванням зростання популярності месенджерів та зручності їх використання, чат-бот стає важливим інструментом для покращення внутрішньої та зовнішньої комунікації в компанії.

Використання чат-бота для надання шаблонів документів та консультацій з юридичних питань дозволяє користувачам отримувати швидку та ефективну допомогу безпосередньо через месенджер.

Такий чат-бот також забезпечує стабільний та однаковий доступ до необхідних ресурсів для всіх користувачів, покращуючи ефективність роботи юридичного відділу. Це рішення є актуальним у контексті стрімкого розвитку цифрових технологій та росту важливості автоматизації процесів у сфері юридичних послуг.

СПИСОК ЛІТЕРАТУРИ

1. Telegram. Офіційний сайти [Електронний ресурс]. – Режим доступу: https://telegram.org/
2. MTProto Mobile Protocol [Електронний ресурс]. – Режим доступу: https://core.telegram.org/mtproto
3. Telegram FAQ [Електронний ресурс]. – Режим доступу: https://telegram.org/faq
4. Telegram Bot API [Електронний ресурс]. – Режим доступу: https://core.telegram.org/bots/api
5. Dialogue Manager (DM) [Електронний ресурс]. – Режим доступу: https://www.hyro.ai/glossary/dialogue-manager-dm/
6. Chatbot Development: Designing Dialog Management [Електронний ресурс]. – Режим доступу: https://blog.vsoftconsulting.com/blog/chatbot-development-designing-dialog-management
7. Types of chatbots – How to choose the best for your business? [Електронний ресурс]. – Режим доступу: https://yellow.ai/blog/types-of-chatbots/
8. Benefits Of Using Telegram Bot [Електронний ресурс]. – Режим доступу: https://botsify.com/blog/hospitality-industry-using-telegram-bot/
9. Building Machine Learning Chatbots: Choose the Right Platform and Applications [Електронний ресурс]. – Режим доступу: https://neptune.ai/blog/building-machine-learning-chatbots-platforms-and-applications
10. What is a Chatbot? Types of Chatbots and How They Work [Електронний ресурс]. – Режим доступу: https://www.shopify.com/blog/chatbots
11. How do Chatbots Work? A Guide to Chatbot Architecture [Електронний ресурс]. – Режим доступу: https://marutitech.com/chatbots-work-guide-chatbot-architecture/

12. Types of Chatbot Technology [Електронний ресурс]. – Режим доступу: https://medium.com/voice-tech-podcast/types-of-chatbot-technology-72d095df2540

13. An Architectural Overview of Task-Oriented Dialog Systems (TODS) - Building Effective Virtual Agent Chatbots [Електронний ресурс]. – Режим доступу: https://www.linkedin.com/pulse/architectural-overview-task-oriented-dialog-systems-tods-mendiratta

14. What is Telegram? How This App Can Make Your Team More Productive [Електронний ресурс]. – Режим доступу: https://mailchimp.com/resources/what-is-telegram/

15. How to Create a Chatbot in Telegram [Електронний ресурс]. – Режим доступу: https://sendpulse.com/knowledge-base/chatbot/telegram/create-telegram-chatbot

16. What is a Telegram Chatbot And How To Create It? [Електронний ресурс]. – Режим доступу: https://chatimize.com/telegram-chatbots/

17. A brief history of Chatbots [Електронний ресурс]. – Режимдоступу:https://chatbotslife.com/a-brief-history-of-chatbots-d5a8689cf52f

18. A Visual History Of Chatbots [Електронний ресурс]. – Режим доступу: https://chatbotsmagazine.com/a-visual-history-of-chatbots-8bf3b31dbfb2

19. History and License - Python documentation [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/license.html

20. Путівник мовою програмування Python [Електронний ресурс]. – Режим доступу: https://pythonguide.rozh2sch.org.ua/

21. Python Enhancement Proposals [Електронний ресурс]. – Режим доступу: https://peps.python.org

22. Asynchronous Programming in Python [Електронний ресурс]. – Режим доступу: https://superfastpython.com/python-asynchronous-programming/

23. aiogram Documentation Release 3.2.0 [Електронний ресурс]. – Режим доступу: https://readthedocs.org/projects/aiogram/downloads/pdf/latest/

24. Документація та приклади, надані командою aiogram [Електронний ресурс]. – Режим доступу: https://docs.aiogram.dev/en/dev-3.x/dispatcher/filters/magic_filters.html

25. «Як Python підвищує ефективність чат-ботів: покращення взаємодії з користувачем за допомогою магічних фільтрів в aiogram» . – Режим доступу: журнал «Зв'язок» за грудень 2023 року (сторінки 24-28)

Додаток 1

Зміст файлу keyboard.py

```
from aiogram.types import ReplyKeyboardMarkup, KeyboardButton
main_keyboard = ReplyKeyboardMarkup(keyboard=[
    [
        KeyboardButton(text="□ Шаблони" ),
        KeyboardButton(text="□ Потрібна консультація"),

    ],
    [
        KeyboardButton(text="□ Запити"),
        KeyboardButton(text="□ Зарезервувати номер документа"),

    ],

], resize_keyboard=True)


template_keyboard = ReplyKeyboardMarkup(keyboard=[
    [
        KeyboardButton(text="□ Шаблон акту" ),
        KeyboardButton(text="□ Шаблон NDA " ),
    ],
    [
        KeyboardButton(text="□ Шаблон довіреності"),
        KeyboardButton(text="□ На головну"),
    ],

], resize_keyboard=True)
```

```
contact_keyboard = ReplyKeyboardMarkup(keyboard=[

    [
        KeyboardButton(text="□ Передзвоніть мені", request_contact=True ),
        KeyboardButton(text="□ Особиста зустріч" ),
    ],
    [
        KeyboardButton(text="□ На головну"),
    ],

], resize_keyboard=True)

req_keyboard = ReplyKeyboardMarkup(keyboard=[

    [
        KeyboardButton(text="□ Відправити запит"),
        KeyboardButton(text="□ Активні запити"),
    ],
    [
        KeyboardButton(text="□ На головну"),
    ],

], resize_keyboard=True)
```

Додаток 2

Зміст файлу main.py

```
import asyncio
import os

from aiogram import Bot, Dispatcher, types, F
from aiogram.enums import ParseMode
from aiogram.types import Message
from aiogram.utils.markdown import hbold

from keyboard import main_keyboard, template_keyboard,
contact_keyboard,req_keyboard

TOKEN = ''
admin_id = {}
user_id = {}

bot = Bot(token=TOKEN)
dp = Dispatcher()

@dp.message(F.text == "/start")
async def command_start_handler(message: Message) -> None:
    if message.from_user.id in user_id:
        await message.answer(f"Вітаю, {hbold(message.from_user.full_name)}!
Оберіть потрібну дію.", reply_markup=main_keyboard)

        @dp.message((F.text == "□ На головну") &
(F.from_user.id.in_(user_id)))
        async def main_kb(message: Message) -> None:
```

```
        await message.answer("Оберіть потрібну дію.",
reply_markup=main_keyboard)

    @dp.message((F.text == "□ Шаблони") & (F.from_user.id.in_(user_id)))
    async def get_templates(message: Message) -> None:
        await message.answer("Оберіть шаблон",
reply_markup=template_keyboard)

    @dp.message((F.text == "□ Потрібна консультація") &
(F.from_user.id.in_(user_id)))
    async def main_kb(message: Message) -> None:
        await message.answer("Оберіть потрібну дію.",
reply_markup=contact_keyboard)

    @dp.message((F.text.in_({'□ Шаблон акту', '□ Шаблон NDA', "□
Шаблон довіреності"})) & (F.from_user.id.in_(user_id)) )
    async def send_file(message: Message) -> None:

        file = {
          "□ Шаблон акту"   :   r"template\ШАБЛОН_Акт.docx",
          "□ Шаблон NDA "  :   r"template\ШАБЛОН_НДА.docx",
          "□ Шаблон довіреності": r"template\ШАБЛОН_Довіреність.docx",
        }

        doc = types.FSInputFile(path=file[message.text])
        await message.answer_document(document=doc)

    #відправка контакту для консультації по телефону
    @dp.message((F.contact) & (F.from_user.id.in_(user_id)))
    async def call_me(message: Message) -> None:
```

```
        await message.answer("Прохання відправлено",
reply_markup=main_keyboard)
        for t in admin_id:
            await bot.send_message(t, f"Користувач
{message.contact.first_name} {message.contact.last_name} просить
перетелефонувти за номером {message.contact.phone_number}")

    @dp.message((F.text == "□ Особиста зустріч") &
(F.from_user.id.in_(user_id)))
    async def meet(message: Message) -> None:
        await message.answer("Прохання відправлено. Очікуйте
повідомлення для зустрічі", reply_markup=main_keyboard)
        for t in admin_id:
            #await bot.send_message(t, str(message.from_user.id))
            await bot.send_message(t, f"Користувач
{(message.from_user.full_name)} бажає зустрітись
(id:{message.from_user.id})")

    @dp.message((F.text.contains('устріч;')) &
(F.from_user.id.in_(admin_id)))
    async def meet_admin(message: Message) -> None:
        try:
            user = int(message.text.split(' ')[0])
            text = message.text.split(';')[1]

            await bot.send_message(user, text)
            for t in admin_id:
                await bot.send_message(t, message.text)
```

```
        except:
            await message.answer("Помилка вводу")

        @dp.message((F.text == "□ Запити") & (F.from_user.id.in_(user_id)))
        async def req_main(message: Message) -> None:
            await message.answer("Оберіть потрібну дію.",
reply_markup=req_keyboard)

        @dp.message((F.text == "□ Відправити запит") &
(F.from_user.id.in_(user_id)))
        async def req_sendreq(message: Message) -> None:
            await message.answer("Відправте файл в чат",
reply_markup=req_keyboard)

        @dp.message((F.document ) & (F.from_user.id.in_(user_id)))
        async def get_doc(message: Message) -> None:
            file_id = message.document.file_id
            file = await bot.get_file(file_id)
            file_path = file.file_path
            await bot.download_file(file_path, r"docs\\" + str(message.from_user.id)
+ ";" + message.document.file_name)
            await message.answer("Файл завантажено. Запит поставлено в
чергу.")

        @dp.message((F.text == "□ Активні запити") &
(F.from_user.id.in_(user_id)))
        async def req_checkreq(message: Message) -> None:
            no_file = True
            for _, _, files in os.walk(r"docs"):
                for f in files:
```

```
            if str(message.from_user.id) + ";" in f:
                    no_file = False
                    await message.answer(f.split(";")[1] + " ще в роботі",
reply_markup=req_keyboard)

            if no_file:
                await message.answer("Документів в роботі відсутні",
reply_markup=req_keyboard)

            await message.answer("Оберіть потрібну дію.",
reply_markup=req_keyboard)

        @dp.message((F.text == "□ Зарезервувати номер документа") &
(F.from_user.id.in_(user_id)))
        async def meet(message: Message) -> None:
            with open(r'number\number.txt', 'r') as file:
                number = int(file.read())
            number = number+1
            with open(r'number\number.txt', 'w') as file:
                file.write(str(number))
            await message.answer(f"Ваш номер документу {number}",
reply_markup=main_keyboard)
            for t in admin_id:
                await bot.send_message(t, f"Користувач
{(message.from_user.full_name)} (id:{message.from_user.id}) зарезервував
новий номер документа {number}")
    else:
        await message.answer(f"Будь ласка пройдіть реєстрацію. Ваш ID
{message.from_user.id}")
        for t in admin_id:
```

```
        await bot.send_message(t, f"Користувач надіслав запит на реєстрацію (id:{message.from_user.id})")
async def main() -> None:

# Initialize Bot instance with a default parse mode which will be passed to all API calls
    bot = Bot(TOKEN, parse_mode=ParseMode.HTML)
    # And the run events dispatching
    await dp.start_polling(bot)

if __name__ == "__main__":
    asyncio.run(main())
```

Державний університет інформаційно-комунікаційних технологій

Кафедра Інженерії програмного забезпечення автоматизованих систем

КВАЛІФІКАЦІЙНА РОБОТА
на тему:
“РОЗРОБКА ЧАТ БОТУ ДЛЯ АВТОМАТИЗАЦІЇ ЮРИДИЧНИХ ПРОЦЕСІВ В КОМПАНІЇ”

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Виконав: здобувач вищої освіти гр. ІСДМ-63
Максим Сметана
Керівник: доцент кафедри ТЦР Аронов А.О.

Київ - 2023

- ***Актуальність теми:*** Новий підхід до вирішення завдань юридичного відділу. Ця робота пропонує інноваційний метод застосування чат-боту для автоматизації рутинних та повсякденних завдань, що передбачаються юридичним відділом. Розроблений чат-бот спеціально адаптований до потреб юридичного середовища та має здатність ефективно взаємодіяти з користувачами.

- ***Об`єкт дослідження:*** є процеси та задачі, пов'язані з розробкою та впровадженням чат-боту для оптимізації та полегшення роботи юридичного відділу/департаменту компанії.

- ***Предмет дослідження:*** чат-бот.

- ***Мета дослідження:*** розробка та впровадження чат-боту для автоматизації рутинних завдань та полегшення роботи юридичного відділу компанії.

- ***Завдання дослідження:***

- Вивчення сучасних підходів та визначення рутинних операцій, які можуть бути автоматизовані за допомогою чат-боту;
- Вивчення існуючих рішень та технологій в області чат-ботів. Історія та аналіз існуючих платформ та програмних засобів для створення чат-ботів з метою визначення найбільш ефективних рішень для реалізації цієї роботи у контексті юридичної автоматизації.
- Розробка концепції чат-боту для юридичних процесів. Формулювання вимог до чат-боту, враховуючи особливості та потреби певного юридичного середовища компанії.
- Реалізація прототипу чат-боту. Створення функціонального прототипу на основі визначених вимог, з урахуванням можливостей обраної технології.
- Тестування та вдосконалення чат-боту. Проведення тестів для оцінки ефективності та коректності роботи чат-боту, а також внесення необхідних корекцій для підвищення його функціональності.
- Аналіз результатів тестування та оцінка впливу чат-боту на ефективність юридичних процесів в організації.

В процесі дослідження вирішувалися ці завдання для досягнення мети розроблення чат-боту, спрямованого на автоматизацію та підвищення ефективності юридичних процесів у компанії.

Чат-бот (chatbots) - це програма, що взаємодіє з користувачем чи клієнтом за допомогою текстових або голосових повідомлень. Це різновид програм штучного інтелекту (ШІ), створений для автоматизації комунікації та виконання завдань через чатові інтерфейси.

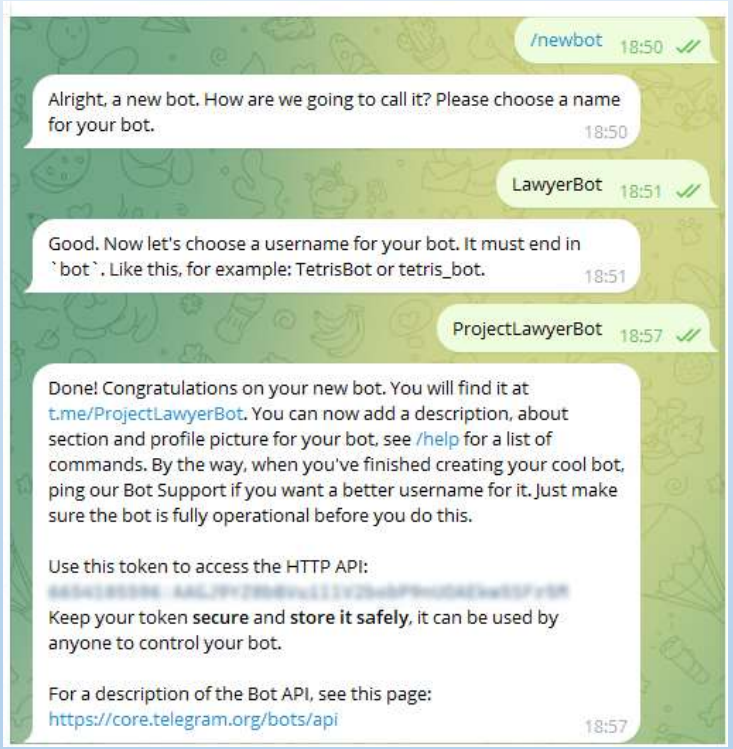
Основна ідея чат-ботів полягає в тому, щоб надавати користувачам можливість отримувати інформацію, задавати питання, отримувати рекомендації чи виконувати конкретні завдання, використовуючи природний мовний інтерфейс. Це може бути використано в різних сферах, таких як обслуговування клієнтів, освіта, розваги, медицина, фінанси та інші галузі.

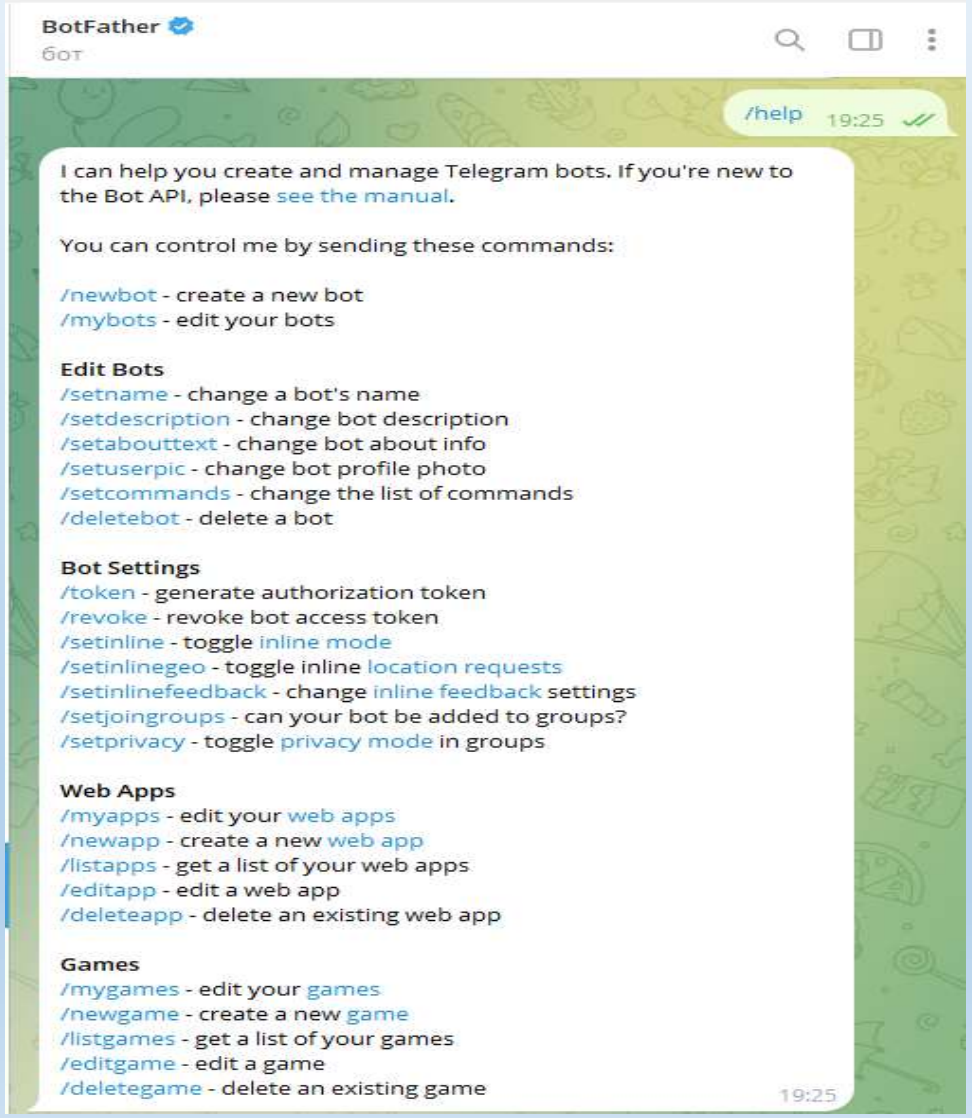
Дизайн і розробка чат-ботів

- Визначення мети та аудиторії
- Інтуїтивний інтерфейс
- Персоналізація та гуманізація
- Контекстно-орієнтована взаємодія
- Візуальний елемент
- Тестування та Оптимізація

ЗАСОБИ РЕАЛІЗАЦІЇ ЧАТ-БОТУ

Для реалізації дипломного проекту було обрано мову програмування Python та бібліотеку aiogram, @BotFather – бот від Telegram месенжер





Python - це високорівнева, інтерпретована мова програмування, яка здобула широку популярність серед розробників. Вона відзначається простотою синтаксису, що робить її дуже доступною для початківців, а також потужною для професіоналів.

- Синтаксис: простий і читабельний
- Інтерпретованість і крос-платформенність
- Обширна бібліотека

Aiogram, ChatterBot, Telegram Bot API, python-telegram-bot та багато інших

- Розширюваність і інтеграція
- Підтримка ООП
- Асинхронність і здатність до роботи з багатьма задачами

Python легко інтегрується з різними платформами для чат-ботів, такими як Telegram, Facebook Messenger, Slack тощо. Деякі бібліотеки, як python-telegram-bot чи facebook-sdk, полегшують роботу з цими платформами

Aiogram

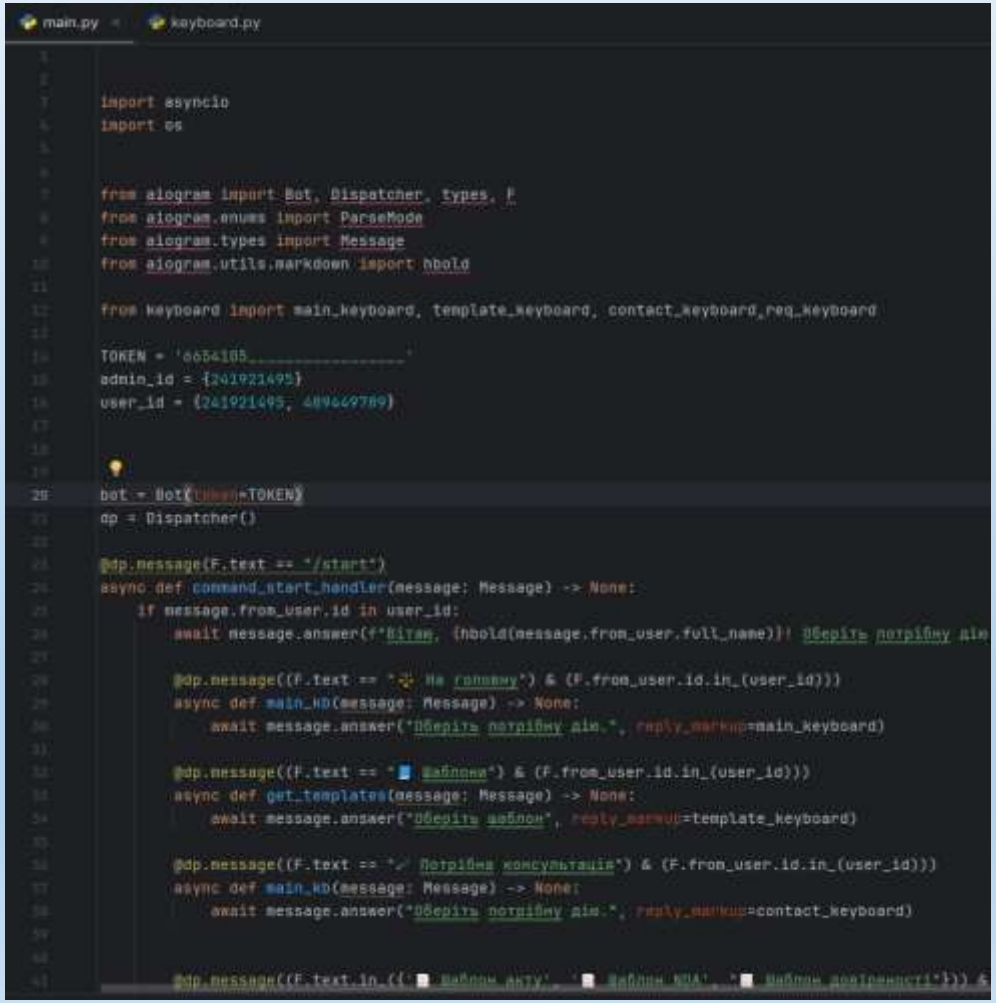
- Aiogram пропонує широкий спектр функцій, які спрощують розробку ботів.
- По-перше, він надає простий та інтуїтивно зрозумілий API, який дозволяє безперешкодно взаємодіяти з API Telegram Bot, незалежно від того, чи потрібно надсилати повідомлення, редагувати повідомлення чи обробляти вбудовані запити.
- Крім того, aiogram підтримує вбудовані клавіатури, які дозволяють створювати інтерактивні меню та кнопки у чат-боті. Ця функція може значно покращити взаємодію з користувачем і зробити чат-бота більш привабливим.

Структура та основні задачі чат-боту

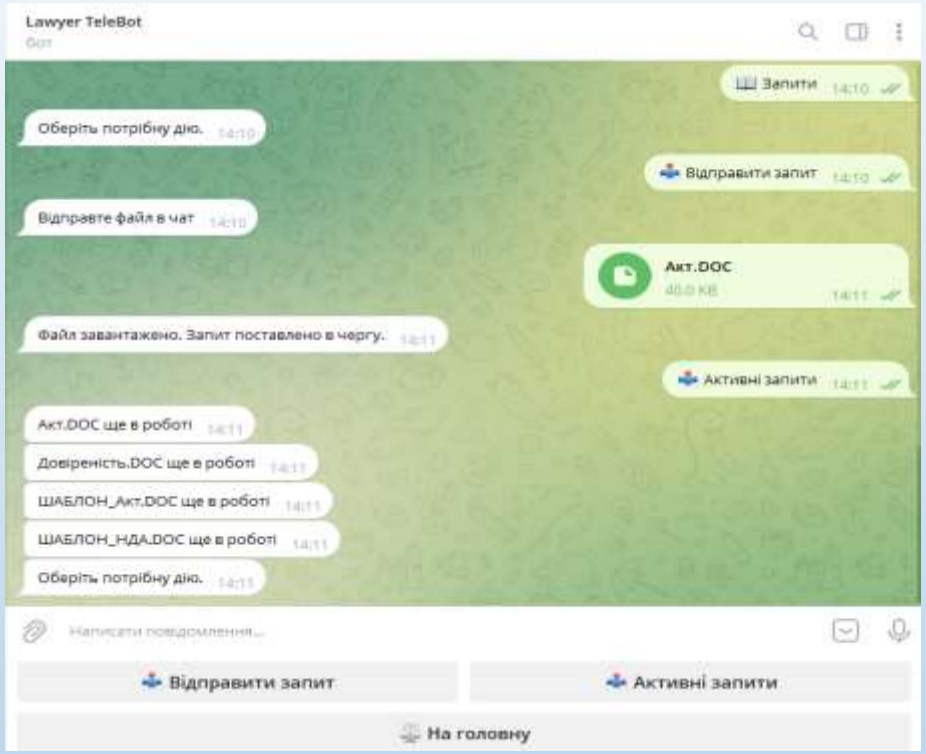
- Реєстрація користувача: користувач автоматично передає свій ID для реєстрації у чат-боті.
- • Доступ до шаблонів документів: зареєстрований користувач отримує можливість користуватися шаблонами юридичних документів компанії, таких як акт, довіреність, шаблон NDA.
- • Запит на консультацію: зареєстрований користувач може надіслати запит на телефонний дзвінок або особисту зустріч для консультації з юристом компанії.
- • Відстеження роботи з документами: зареєстрований користувач може направити документ на розгляд юристам компанії та слідкувати за його статусом в роботі.
- • Резервування номера документу: зареєстрований користувач компанії має можливість зарезервувати унікальний номер для юридичного документу.
- Обробка запитів: юристи компанії автоматично отримують відповідну інформацію від користувачів та мають доступ до документів, що надіслані на розгляд.

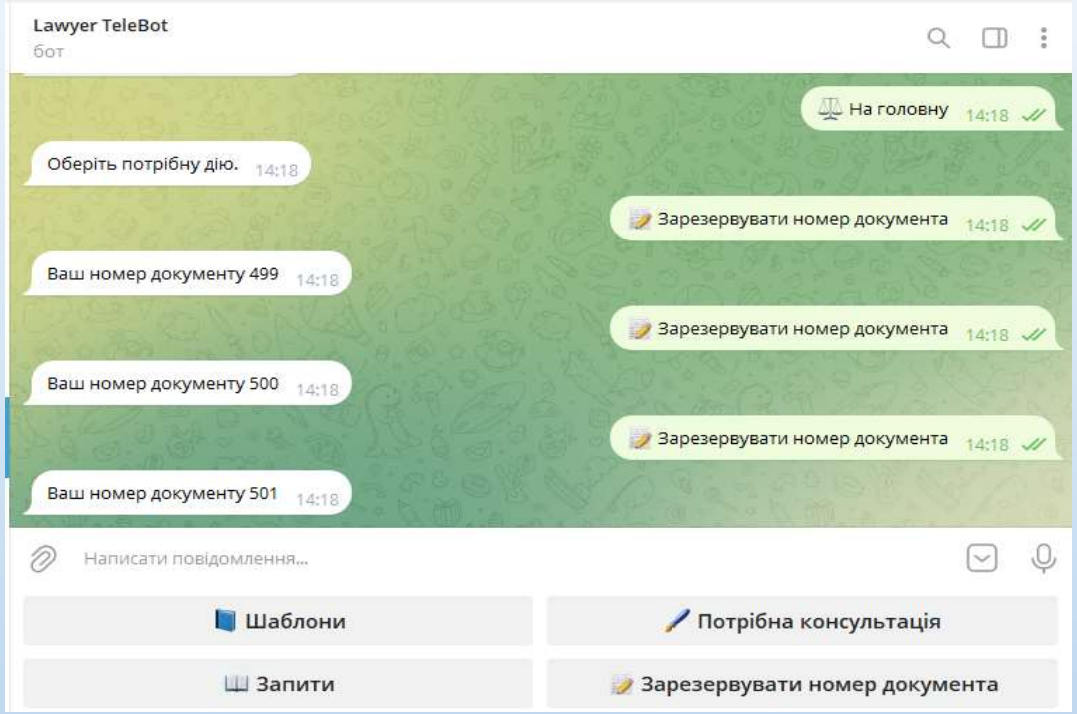
Проект включає два файли:

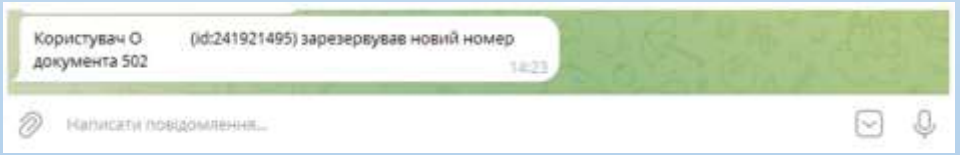
- keyboard.py – містить назви та розміщення кнопок у чат-боті
- main.py – містить опис дій, що виконуються при натисканні користувачем на відповідну кнопку



Робота чат-боту: 1) Інформування 2) Запис 3) Документообіг







користувачі можуть автоматично отримувати шаблони документів, надсилати свої документи на розгляд та надавати запити на консультації від юристів компанії. Це забезпечує зручність та ефективність взаємодії користувачів з юридичним відділом через цифровий канал

Демонстрація роботи чат-боту

ВИСНОВКИ

- Під час виконання дипломного проекту був проведений детальний аналіз предметної області та визначені ключові принципи функціонування чат-ботів. Також розглянуті основні методики для їх створення. Проведено аналіз загальних принципів і інструментів для побудови чат-ботів у месенджері Telegram та програмних інструментів для їх розробки.
- В якості основних засобів розроблення були обрані мова програмування Python, бібліотека Aiogram та Telegram Bot API. Ці інструменти визначились як оптимальні для створення чат-бота з урахуванням потреб користувачів та функціоналу чат-боту.
- Як результат проектування, був успішно створений чат-бот для юридичного відділу компанії. За допомогою цього бота користувачі можуть автоматично отримувати шаблони документів, надсилати свої документи на розгляд та надавати запити на консультації від юристів компанії. Це забезпечує зручність та ефективність взаємодії користувачів з юридичним відділом через цифровий канал.
- Проведений аналіз та створення чат-бота для юридичного відділу виявили важливість та актуальність використання такого інноваційного інструменту в сучасному бізнес-середовищі. З врахуванням зростання популярності месенджерів та зручності їх використання, чат-бот стає важливим інструментом для покращення внутрішньої та зовнішньої комунікації в компанії.

ДЯКУЮ ЗА УВАГУ!

Стаття: «Як Python підвищує ефективність чат-ботів: покращення взаємодії з користувачем за допомогою магічних фільтрів в aiogram» Режим доступу: журнал «Зв'язок» (сторінки 24-28)