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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

**До захисту допущено:**
**Завідувач кафедри**
_____ Сергій СТІРЕНКО
(підпис)

“__” _____2022 р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**

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

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

на тему: Бот для взаємодії між мессенджером та іншим сайтом

Виконав : студент 4 курсу, групи ІО-83
(шифр групи)

Палеха Богдан Петрович
(прізвище, ім’я, по батькові) (підпис)

Керівник асистент Пономаренко А. М.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

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

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

Київ – 2022 р.

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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалавр)

Освітньо-професійна програма

Освітньо-професійна програма “Комп’ютерні системи та мережі”

спеціальності 123 “Комп'ютерна інженерія

**ЗАТВЕРДЖУЮ**
**Завідувач кафедри**
_____ Сергій СТІРЕНКО
(підпис)

“__” _____2022 р.

**ЗАВДАННЯ**
на бакалаврський дипломний проєкт студента

Палехи Богдана Петровича

1. Тема проєкту Бот для взаємодії між мессенджером та іншим сайтом керівник проєкту Пономаренко Артем Миколайович, асистент,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від 2022 року №

2. Термін здачі студентом закінченого проєкту 11 *червня 2022 р.*

3. Вихідні дані до проєкту *технічна документація, теоретичні дані.*

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

Розділ 1. Опис предметної області.

Розділ 2. Огляд алгоритмів та технологій для розробки системи.

Розділ 3. Вибір оптимальних технологій для розробки проєкту та їх обгрунтування

Розділ 4. Дослідження та аналіз розробленої системи.

5. Перелік графічного матеріалу (з точним позначенням обов'язкових креслень) структурна схема системи, функціональна схема (діаграма класів), алгоритм дій програмного забезпечення.

6. Консультанта проєкту, з вказівкою розділів проєкту, які до них вносяться

| Розділ | Консультант | Підпис, дата | |
|---|---|---|---|
| | | Завдання видав | Завдання прийняв |
| Нормоконтроль | Сімоненко В. П. | | |
| | | | |

7. Дата видачі завдання «20» гр 2021 р.

Календарний план

| № п/п | Найменування етапів дипломного проєкту | Терміни виконання етапів проєкту | Примітки |
|---|---|---|---|
| 1. | *Затвердження теми проєкту* | *10.12.2021-15.12.2021* | |
| *2.* | *Вивчення та аналіз завдання* | *15.12.2020-15.03.2021* | |
| *3.* | *Розробка архітектури та загальної структури системи* | *15.03.2022-25.03.2022* | |
| *4.* | *Розробка структур окремих підсистем* | *25.03.2022-5.04.2022* | |
| *5.* | *Програмна реалізація системи* | *5.05.2022-15.05.2022* | |
| 6. | *Оформлення пояснювальної записки* | *15.05.2022-20.05.2022* | |
| 7. | *Захист програмного продукту* | *25.05.2022* | |
| *8.* | *Передзахист* | *11.06.2022* | |
| *9.* | *Захист* | *21.06.2022* | |

Студент-дипломник _____ Палеха Богдан
(підпис)

Керівник проєкту _____ Артем Пономаренко
(підпис)

## АНОТАЦІЯ

В цьому проекті для бакалавра ми розглянули месенджери, хмарні системи способи, їхні API та способи їх взаємодії для створення програми яка б дозволяв проводити взяємодію між ними. Після аналізу та перевірки було вирішено розробити чат-бота для месенджера Telegram, а хмарними сервісами зберігання даних з якими чат-бот буде працювати стали Google Drive та Mega.nz. Мовою розробки була вибрана Python, платформою для баз даних стала Heroku, а сам програмний продукт був запушений за допомогою Goorm IDE.

Результатом цього проекту став чат на Telegram який може проводити операції з даними як на дисковому просторі користувача так і в хмарних системах зберігання даних.

## ANNOTATION

In this project for Bachelor’s degree, we looked at messengers, cloud systems, their APIs, and how they work together to create an application that allows you to have interactions between them. After analysis and research, it was decided to develop a chatbot for Telegram messenger, and cloud storage services with which the chatbot will work was chosen to be Google Drive and Mega.nz. Python was chosen as the development language, Heroku became the platform for the databases, and the software product itself was дездщнув using the Goorm IDE

The result of this project was a chat on Telegram that can perform data operations both on the user's disk space and in cloud storage systems.

| справки | Формат | *Значення* | *Найменування* | Кіл. листів | № екземпляр | *Додаток* |
|---|---|---|---|---|---|---|
| | | | Документація загальна | | | |
| | | | Знову розроблена | | | |
| | | | | | | |
| | ***А4*** | ***ІАЛЦ.467200.002 ТЗ*** | Бот для взаємодії | 3 | | |
| | | | між мессенджером | | | |
| | | | та іншим сайтом | | | |
| | | | Технічне завдання | | | |
| | | | | | | |
| | ***А4*** | ***ІАЛЦ.467200.003 ПЗ*** | Бот для взаємодії | 61 | | |
| | | | між мессенджером | | | |
| | | | та іншим сайтом | | | |
| | | | Пояснювальна записка | | | |
| | | | | | | |
| | ***А4*** | ***ІАЛЦ.467200.004 Д1*** | Бот для взаємодії | 1 | | |
| | | | між мессенджером | | | |
| | | | та іншим сайтом | | | |
| | | | Структурна схема | | | |
| | | | системи | | | |
| | | | | | | |
| | ***А4*** | ***ІАЛЦ.4672008.005 Д2*** | Бот для взаємодії | 1 | | |
| | | | між мессенджером | | | |
| | | | та іншим сайтом | | | |
| | | | Функціональна схема | | | |
| | | | (діаграма класів) | | | |
| | | | | | | |
| | ***А4*** | ***ІАЛЦ.467200.006 Д3*** | Бот для взаємодії | 1 | | |
| | | | між мессенджером | | | |
| | | | та іншим сайтом | | | |
| | | | Алгоритм дій програмного | | | |
| | | | забезпечення | | | |
| | | | | | | |
| | ***А4*** | ***ІАЛЦ.467200.007 Д4*** | Бот для взаємодії | 8 | | |
| | | | між мессенджером | | | |
| | | | та іншим сайтом | | | |
| | | | Текст програмного коду | | | |
| | | | | | | |
| | | | | | | |

| | | | | | ***ІАЛЦ.467200.001 ОА*** | | |
|---|---|---|---|---|---|---|---|
| *Зм* | ***Лист*** | ***№ докум.*** | ***Підп*** | ***Дата*** | | | |
| ***Розроб*** | | Палеха Б.П | | | *Бот для взаємодії між мессенджером та іншим сайтом* Опис альбому | Літ. | Аркуш 1 / Аркушів 1 |
| ***Перев*** | | ПономаренкоА.М. | | | | | |
| | | | | | | ***КПІ ім. Ігоря Сікорського ФІОТ ІО-83*** | |

# ТЕХНІЧНЕ ЗАВДАННЯ
# ДО ДИПЛОМНОГО ПРОЕКТУ

на тему: «*Бот для взаємодії між мессенджером та іншим сайтом*»

Київ – 2022

# ЗМІСТ

# 1 НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку чат-боту для Telegram,які надасть можливість взаємодіяти з хмарними сервісами збереження даних

Область застосування: маніпуляції з даними

# 2 ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки даної системи є завдання для виконання роботи кваліфікаційно-освітнього рівня «бакалавр інженерії програмного забезпечення», який був затверджений факультетом “Інформатики та обчислювальної техніки” кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

# 3 МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою та призначенням даної роботи є розробка чат-бота для взяємодії між Telegram та хмарними сервісами даних

# 4 ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки даного дипломного проекту є офіційні документації, публікації та статті в мережі Інтернет на дану тему, науково-технічна література.

# 5 ТЕХНІЧНІ ВИМОГИ

## 5.1. Вимоги до розробленого продукту

Розроблена система має виконувати такі вимоги:

- Простий та зрозумілий інтерфейс та функціонал
- Можливість взаємодіяти з даними в хмарних сервісах

## 5.2. Вимоги до програмного забезпечення

- ОС Windows, Mac чи Linux.
- Windows/Linux/MacOS PyCharm IDE
- Встановлений Telegram.

## 5.3. Вимоги до апаратної частини

- ЦП не менше ніж Intel® Core (TM) i3-2100T.
- ROM не менше ніж 64 ГБ.
- RAM не менше ніж 4 ГБ.

| Назва етапів виконання | Термін виконання |
|---|---|
| Затвердження теми роботи | 10.12.2021-15.12.2021 |
| Вивчення та аналіз завдання | 15.12.2021-15.03.2022 |
| Розробка архітектури та загальної структури системи | 15.03.2022-25.03.2022 |
| Розробка структур окремих частин системи | 25.03.2022-5.05.2022 |
| Програмна реалізація системи | 5.05.2022-15.05.2022 |
| Виправлення помилок | 15.05.2022-20.05.2022 |
| Оформлення пояснювальної записки | 25.05.2022 |

# 6 ЕТАПИ РОЗРОБКИ

# ПОЯСНЮВАЛЬНА ЗАПИСКА
# ДО ДИПЛОМНОГО ПРОЄКТУ

на тему: «*Бот для взаємодії між мессенджером та іншим сайтом*»

Київ – 2022

ЗМІСТ

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

| | |
|---|---|
| HTML (HyperText Markup Language) | мова розмітки гіпертексту |
| IDE (Integrated Development Environment) | комплексне програмне рішення для розробки програм |
| JSON (JavaScript Object Notation) | формат обміну та зберігання даних |
| URL (Uniform Resource Locator) | стандартизована адреса певного ресурсу |
| XML (Extensible Markup Language) | розширювана мова розмітки |
| ОС | операційна система |

# ВСТУП

З розвитком технологій постійно змінюються або з'являються нові методи зв'язку і передачі даних. Такі технології як радіо, телебачення та телефоні мережі значно збільшили і спростили способи зв'язку та отримання інформації. Звичайно ж тільки на цих видах зв'язку наш розвиток технологій не закінчився і була розроблення технологія інтернету, яка швидко розповсюдилась по світу утворивши цілу павутину зв'язку. Хоча інтернет не є собою заміною більш раніших методів зв'язку таких як радіо, телебачення та телефонних мереж, з збільшенням швидкості інтернету та збільшенням потужностей приладів почали заявлятися дешевші та зручніші альтернативи більш старішим технологіям передачі даних та навіть її збереження. Так ,наприклад, за допомогою інтернету можливо зберігати і передавати велику кількості данник на хмарних платформах, таких як, mega.nz та drive.google.com а функції телефонних розмов, смс та передачі медіа можуть виконувати месенджери.

Одним з таких месенджерів в Україні є Telegram, медіа всередині нього можна поділити на 3 частини: чати,групи та боти.

Чат – де є взаємодія між багатьма користувачами де кожен може обмінюватись повідомленнями , медіа та файлами

Групи — де іде взаємодія між адміністратором і багатьма користувачами в вигляді повідомлень та медіа які користувачі можуть реагувати за допомогою реакцій та коментаріїв

Боти — це програми всередині самого Telegram які були створені самими користувачем і виконують функцію сервісів та автоматичних відповідей на запити на які вони були запрограмовані.

На даний момент часу Telegram є одним із найбільш популярних месенджерів завдяки. На ранніх моментах існування Telegram мав за мету тільки обмін між користувачами, але з часом він зазнав значних змін і покращень.

Одним з таких покращень стали програми чат-боти, у стандарному сенсі програми які працюють всередині самого мессенджера. Ці сервіси набрали широкої популярності серед користувачів через як і розширення можливостей групп і каналів так і через економію часу шляхом автоматизації дії необхідних для користувача.

В основі цього бакалаврського проекту є створення чат-бота для покращення зручності передачі та отримання інформації через взаємодії між Telgram та хмарними сервісами збереження даних таких як drive.google.com та mega.nz.

# РОЗДІЛ 1
# АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

## 1.1 Загальний огляд telegram та месенджерів

### 1.1.1. Telegram-бот

**Telegram** – це месенджер який є на ПК, android, IOS Windows Phone та Linux , написаний на C++, з функціоналом обміну даними та повідомленнями. Який був запущений наприкінці 2013 року і зараз має близько 550 мільйонів користувачів щомісяця. База користувачів Telegram має тенденцію зростати, коли скандал з конфіденційністю торкається одного з його більших конкурентів. Звичайно ж серверна частина має закритий код доступ до якого має тільки сама компанія, самі ж сервера розташовані в США та ФРГ(рис.1.1)

Рисунок 1.1 – Знак Telegram

При досліджені різних месенджерів, їх перваг та недоліків наступні переваги отримав телеграм:

- конфіденційність – шифрування, функціонал захищених чатів, опція самоліквідація акаунта та повідомлень через деякий час та
- швидкість – час приходження повідомлення менший чим у конкурентів та їх аналогів.

- розподіл – через те що сервера месенджера знаходяться в різних кутках світі він має більше стійкості до збоїв.
- відкритість – великим бонусом є також протокли MTProto та API.
- присутність функціоналу спілкування аудіоповідомлення та відеорозмов
- формування папок для окремих чатів

Сам месенджер був розроблений на багато різних ОС, серед них Windows, macOS, Linux, Android, iOS, його апаратними платформами є x86-64, ARMv7, AMD64.

Хоча сам Telegram має свій початок же в 2013, в самому початку він тільки мобільні платформи за мету, а вже потім з його подальшою еволюцією він перейшов нв інші платформи та навіть

Самі додатки на всіх ОС хоч і мають деякі відмінності в цілому він залижаеться незмінним незалежно від ОС, для прикладу буде наданий інтерфейс с додатків на ОС Windows 10(рис. 1.2) та Android 12(рис. 1.3).

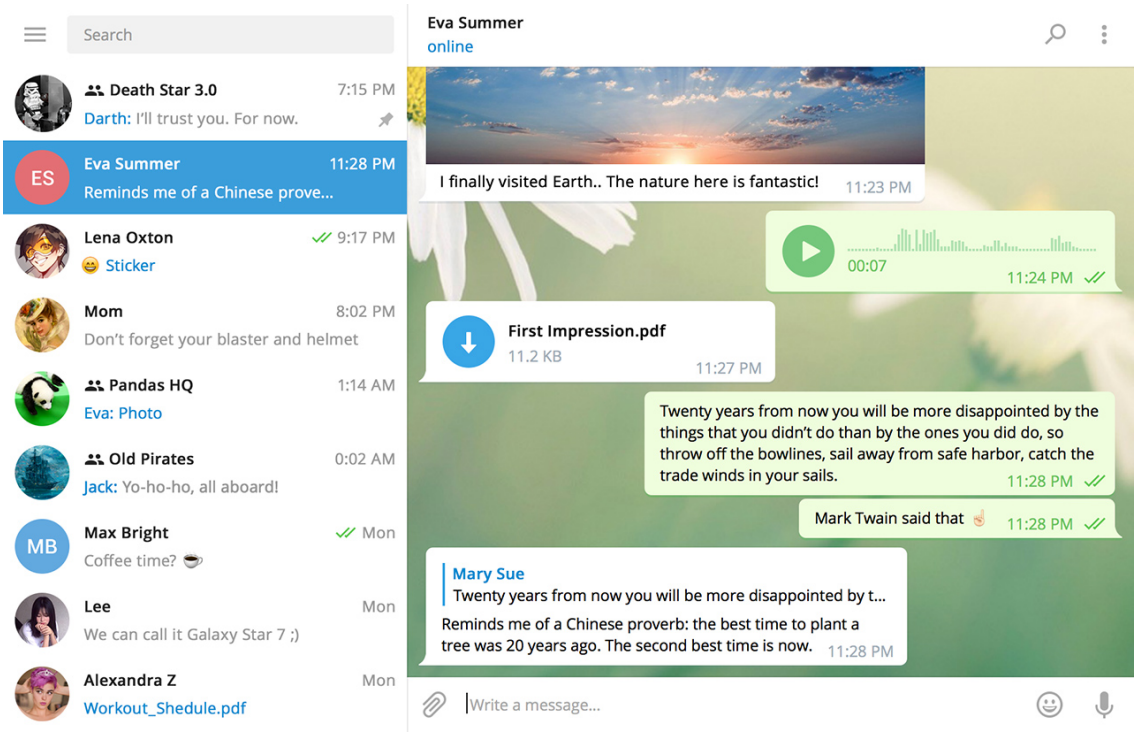



Рисунок 1.2 – інтерфейс мобільного додатку на Windows

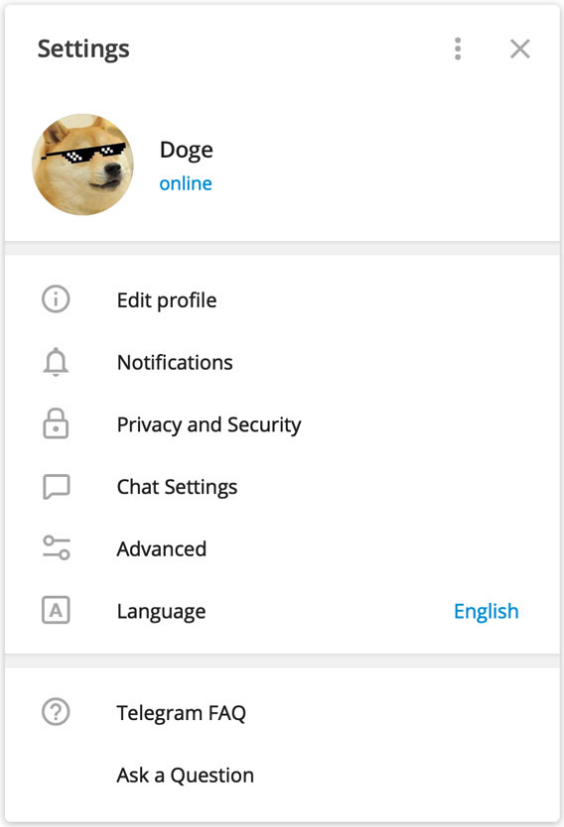



Рисунок 1.3 – інтерфейс мобільного додатку на Android

### 1.1.2. Telegram-бот

Чат-бот Telegram [1] — це програма, яка отримує інформацію від користувачів для формування правильних і логічних відповідей. Сьогодні є всі види чат-ботів. Нижче наведено деякі можливі варіанти:

- чат-ігри ;
- боти для реклами ;
- боти для новин ;
- боти для обслуговування та здійснення підтримки клієнтів (Банки,ДТЕК);
- боти для музики .

Боти являють собою спеціальні програми, які виконують багато різних запитів від юзера в чаті. Чати з ботами мають такий самий вигляд як і чати с іншими користувачами , але взаємодія йде не з людиною, а з програмами, які були написанні на певні функції. Невдовзі після їх створення вони отримали чималу популярність серед користувачів через їхню зручність, велику різномність та автоматизацію. На даний момент часу телеграм розповсюдженний навіть у бізнесі який не напряму зладнаний з інтернетом

Однією з особливостей Telegram-ботів є наявність можливостей виконання певних команд в чаті , а пізніше ініціювати дії безпосередньо або зробити запит на інформацію. В приклад можна взяти базову команду«/help», надіславши її ви отримаєте список можливих команд для цього чат-бота, у формі текстового зворотного зв'язку. В приклад візьмемо наступний список команд (рис. 1.4)

- /start ;
- /stop ;
- /reverse ;
- /word_reverse .

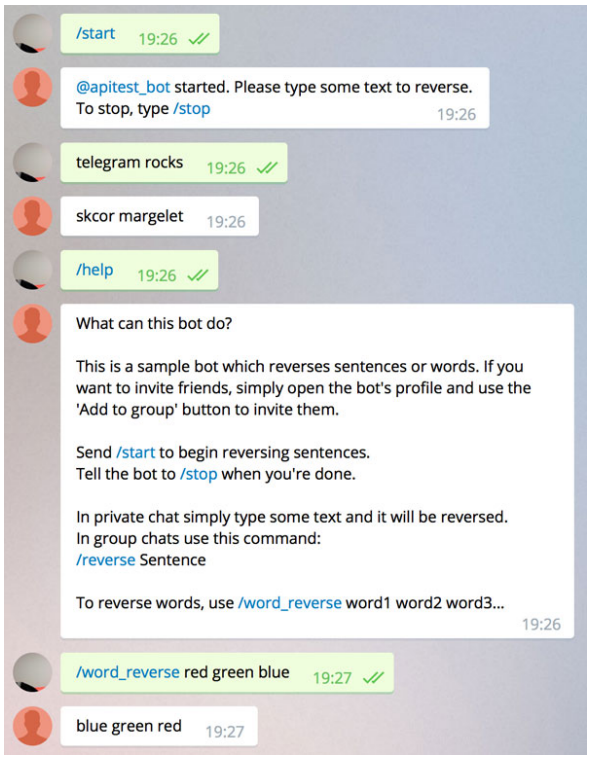



Рисунок 1.4 Список команд бота

Telegram зберігає всі дані, всі особисті налаштування, тобто всі чати, а також ботів та зашивровує їх в хмарі, тому не обов'язково зовнішньо резервне копіювати . Зареєстровані користувачі можуть використовувати всв свої дані на різних платформах у будь-який час і в будь-якому місці. Проте конфіденційні дані і зовнішні команди можна зберігати наприклад і серверах та базах даних. Також не всі боти є в загальній доступності, деякі можуть бути доступними тільки для деяких юзерів якщо вони мають спеціально налаштовані канал зв'язку з ботом. Такий канал є приватним, і є щось на кшталт групи с ботом до якої входять сам бот і ті користувачі які були схваленні. Також такою схемою управління можна керувати хто отримує можливість користуватись функціями бота .Також ботів можна завжди ідентифікувати за приставкою бот, наприклад: @TriviaBot, @tchannelsbot, @ImageBot тв інші.

### 1.1.3. Етапи створення Telegram-бота

Створити власного бота зазвичай дуже просто. Вам допоможе віртуальний користувач Botfather (Рисунок 1.5), центральний інструмент для створення ботів Telegram. Отже, спочатку вам потрібно знайти цього користувача в Telegram і поговорити з ним. Переконайтеся, що ви вибрали підтвердженого користувача (виділено синім кольором).

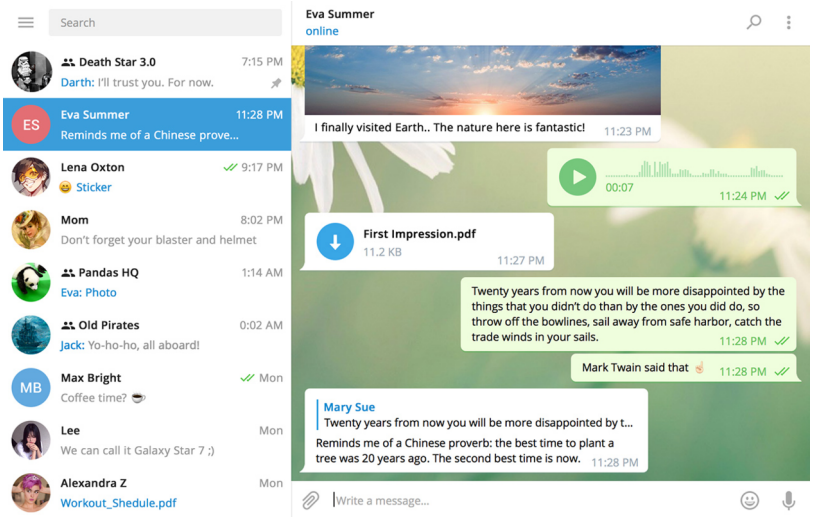



Рис 1.5 –Чатбот BotFather для створення чатботів

Заснований на романі Маріо П’юзо і успішно знятий як «Хрещений батько» з Марлоном Брандо в 1972 році, Батько-бот із трьох частин (названий на честь мафіозної саги «Хрещений батько») насправді є телеграм-ботом, який відповідає на накази. Він допомагає створювати власних ботів, організовує унікальні імена ботів і відкриває доступ до програмування.

Для програмування ботів доступні гативно не всі, мови програмування,хоча в принципі, ви можете використовувати всі запущені на сервері мови програмування і відповідати на запити через HTTPS. Вибір популярних мов програмування, які використовуються для розробки інтернет-додатків:

- Python;
- JavaScript (node.js);
- PHP
- Ruby.
- Swift;
- Java / Kotlin;
- C#;

При створенні чат-бота, як і для створення, потрібно розібрати всю задачу на деяку кількість маленьких та пройти зробити їх поетапно.

1) Спочатку дізнайтеся, чи підходить чат-бот для вашого бізнесу.Чат-боти є гарною ідеєю, якщо:

- Ваша команда виконує багато рутинних і повторюваних завдань;
- Ви отримаєте багато запитань на ту саму тему;
- Маєте за мету зниження вартості обслуговувань клієнті
- Ваша клієнтська база має в собі людей с різними мовами
- Присутня багатоканальна підтримка
- Є бажання обслуговувати користувачів та клієнтів, хоч і частково цілодобово без перерви та неробочих днів
- На меті є реклама та подальше просування продукту чи послуг;

- Можливість привабити додаткових користувачів та клієнтів, чим самим збільшити прибутки;
- На меті більша зручність і функціонал чми у конкурентів, а отже і більша приваблюваність;
- Бажання не відстувати від аналогів та своїх конкурентів

2) Якщо ж все таки були поставлене завдання створити бота, то потрібно прміркувати над метою самого чатбота.

3) Якщо було визначено що чат-бот все таки необхідний то потрібно продовжити роздуми щодо планів на бота:

Повинно матись розуміння для чого цей проект робится. Одним із найкращиз спосіб є концепція управлівння ідеями та завданнями з абревіатурою SMART.

4) Як тільки ми закінчили ставити задачі перед самим ботом, нам переходити до визначення областей, які може обробляти чат-бот. Вам потрібно проаналізувати та скласти список поточних трудомістких і рутинних процесів.

Як правило, найкращими областями для чат-ботів є задачі с обслуговуванням людей, оскільки ним треба часто надавати ті самі послуги. Наприклад, такі процеси продажу, як обслуговування клієнтів, технічна підтримка,пошук нових клієнтів.

5) Далі нам потрібно вибрати куди робити бота на на якій мові його писати.

6) Знайдіть час, щоб подумати та записати можливу інтеграцію для вашого чат-бота.

7) Виберіть потрібну компанію для розробки платформи для чат-ботів або чат-ботів.

## 1.2 Загальний огляд хмарних сервісів зберігання даних

### 1.2.1. Хмарні сховища даних

Хмарне сховище — це модель зберігання комп'ютерних даних, де цифрові дані зберігаються в логічних пулах, які називаються «хмарами». Фізична пам'ять охоплює багато серверів (іноді в кількох місцях), а фізичне середовище зазвичай належить і керується хостинговою компанією. Ці постачальники хмарних сховищ відповідають за доступність і доступність даних, а також за підтримку, захист і експлуатацію фізичного середовища. Окремі особи та організації купують або орендують пам'ять у постачальників для зберігання інформації про користувачів, організації чи програми.(Рис 1.6)

| Назва | Dropbox | Google Drive | Mega | Bitcasa |
|---|---|---|---|---|
| Безкоштовний обсяг | 2 GB + 36 GB — за виконання завдань | 15 GB | 50 GB | 20 GB |

| Назва | Copy.com | Yunpan 360 | 4shared | OneDrive |
|---|---|---|---|---|
| Безкоштовний обсяг | 15 GB + 7 GB — за додаткові завдання | 36 TB | 15 GB | 7 GB |

Рис 1.6 – розцінки і ємність сервісів

Доступ до служб хмарного сховища можна отримати через програми, які використовують API, наприклад службу хмарних обчислень, інтерфейс програмування веб-сервісів (API) або пам'ять хмарного робочого столу, шлюз хмарного сховища або веб-системи керування вмістом.

Хмарне сховище базується на високовіртуалізованій інфраструктурі і подібне до більш широких хмарних обчислень з точки зору інтерфейсів, практично миттєвої гнучкості та масштабованості, багаторазового оренди та вимірних ресурсів. Служби хмарного сховища можна використовувати із зовнішнього сервісу (Amazon S3) або розгорнути локально (ViON Capacity Services).

Існує три типи хмарного сховища: служба зберігання розміщених об’єктів, сховище файлів і блочне сховище. Кожен із цих типів хмарного сховища має унікальні переваги.

Служби зберігання об’єктів, які можна розгорнути та реалізувати за допомогою функцій хмарного сховища, включають системи зберігання об’єктів, такі як Amazon S3, Oracle Cloud Storage і Microsoft Azure Storage і розподілені продуктів. Дослідницькі проекти зі сховища, такі як OceanStore] і VISION Cloud.

Служби зберігання файлів, такі як Amazon Elastic File System (EFS) і Qumulo Core, використовуються для програм, яким потрібен доступ до спільних файлів і потрібна файлова система. Це сховище часто підтримується сервером мережевого додаткового сховища (NAS), який використовується для великих сховищ вмісту, середовищ розробки, медіа-магазинів або домашніх папок користувачів.

Служби блочного зберігання, такі як Amazon Elastic Block Store (EBS), використовуються для інших корпоративних програм, таких як бази даних, і часто потребують певного періоду затримки для кожного хоста. У деяких аспектах це можна прямо порівняти з додатковим сховищем (DAS) або мережею зберігання даних (SAN).

Хмарна пам'ять складається з багатьох розподілених ресурсів, але все ж або федерація , або спільне сховище відіграє роль в архітектурі хмари.

Дуже толерантний до помилок через надмірність і перерозподіл інформації

Висока довговічність при створенні версійних копій

Як правило, це відповідає реплікам даних

### 1.2.2. Google drive

Google Drive являє собою рішення задачі зберігання своїх даних віддалено через хмарні сервіси від компанії Google(Рис 1.7). Запущений 24 квітня 2012 року Google Drive дозволяє користувачам синхронізувати та обмінюватися файлами в хмарі (на серверах Google) на пристроях. На додаток до веб-інтерфейсу, Google Drive пропонує програми з автономними можливостями для комп'ютерів Windows і MacOS, а також для смартфонів і планшетів Android і iOS. Переглядайте Google Диск, Google Документи, Google Таблиці та Google Презентації, документи, електронні таблиці, презентації, малюнки, форми, форми, форми тощо. Файли, створені та відредаговані.


Рис 1.7 – логотип google drive

Google Drive пропонує користувачам 15 ГБ безкоштовного сховища через Google One. Google One також пропонує 100 ГБ, 200 ГБ, 2 ТБ, які пропонуються через додаткові платні плани. Завантажені файли можуть мати розмір до 750 ГБ. Користувачі можуть змінювати налаштування конфіденційності для окремих документів і папок, включаючи налаштування конфіденційності для обміну з іншими користувачами або створення спільноти вмісту. На веб-сайті користувачі можуть шукати зображення за візуальним описом і використовувати його природною мовою, щоб знайти конкретні документи, наприклад «Знайди графік бюджету з грудня минулого року».

Веб-сайт і додаток для Android пропонують розділ резервного копіювання для резервного копіювання спеціальних папок на комп'ютері користувача, оскільки пристрої Android надають копію даних службі, а також повністю оновлену комп'ютерну програму, випущену в липні 2017 року. Функція швидкого входу дозволяє користувачам розумно передбачити необхідні документи.

Google Drive, ключовий компонент Google Workpace, — це щомісячна підписка Google для компаній і організацій до жовтня 2020 року. Як частина планів Choice Google Workpace, диск пропонує необмежений обсяг пам'яті, розширений звіт про перевірку файлів, розширені інструменти керування та додаткові інструменти для спільної роботи для команд.

З моменту запуску сервісу політика конфіденційності Google Drive була широко розкритикована деякими представниками ЗМІ. Google, набір умов надання послуг та угод про політику конфіденційності, які охоплюють усі її послуги, ймовірно, має широкий спектр прав на відтворення, використання та отримання вмісту, що зберігається на Google Диску компанії, мовою угод. Політика також викликає занепокоєння, що користувачі захищають свої права інтелектуальної власності, захист конфіденційності, ліцензії, налаштування Google реклами та інформації, наданої Google, а також право на використання та використання інформації. Навпаки, інші представники ЗМІ заявили, що угоди не гірші за конкуруючі послуги хмарного сховища, але конкуренція використовувала в угодах «більш художню мову» і «вимагала права на переміщення файлів по серверах, кешування ваших даних або створення мініатюр.Станом на липень 2018 року Google Drive мав понад мільярд активних користувачів, а станом на вересень 2015 року нараховувалося мільйон корпоративних користувачів платежів. Станом на травень 2017 року в сервісі зберігалося два трильйони файлів.

### 1.2.3. Mega

Mega (рекурсивна абревіатура від мегашифрованого глобального доступу) — це хмарне сховище та послуга розміщення файлів, що пропонується компанією, що базується в Окленді, Нова Зеландія, і компанією, що базується в Новій Зеландії. Послуга надається через веб-додатки. Мобільні програми Mega також доступні для Android та iOS. Mega відома як найбільше у світі повнофункціональне безкоштовне хмарне сховище з 20 ГБ безкоштовного розповсюдження для безкоштовних облікових записів. Веб-сайт і сервіс було запущено 19 січня 2013 року Кімом Доткомом, директором і співзасновником Матіасом Ортманном, директором з маркетингу Фінном Батато та Бремом Ван Дер Голком.(Рис 1.8)


Рис 1.8 – логотип mega.nz

Дані про мегасервіси шифруються на стороні клієнта за допомогою алгоритму AES . Оскільки вони не знають ключів шифрування мега-завантажених файлів, вони не можуть розшифрувати та переглянути вміст. Тому вони не можуть нести відповідальність за вміст завантажених файлів . Шифруючи файли, MEGA може співпрацювати з більшою кількістю компаній, що розміщують дані по всьому світу, і зменшує ймовірність захоплення урядом серверів методом Megaupload. MEGA також використовує технологію CloudRAID , що означає, що файли поділяються приблизно на рівні частини і зберігаються в різних країнах. Вони також зберігають іншу

частину в іншій країні і відзначають, що кількість бітів «1» у певній позиції у всіх частинах є парною або непарною. Це означає, що користувачі можуть скинути таку інформацію, навіть якщо одна з частин недоступна.

Протягом перших кількох тижнів після запуску Mega були виявлені різні проблеми безпеки, які, за словами дослідників, зловмисник міг використовувати для отримання доступу до файлів користувача . У відповідь Mega запустила програму винагороди за вразливість, яка пропонувала винагороду до 10 000 євро за повідомлення про проблеми безпеки в MEGA

Mega SDK , а також публікує вихідний код усіх клієнтських програм під ліцензією Mega Limited Code Review License. [19] Вихідний код був опублікований після того, як колишній директор Кім Дотком оголосив, що він «створить повністю відкритий і неприбутковий конкурент Mega» після того, як залишить Mega Ltd. MEGA також публікує щорічний звіт про прозорість, в якому публікуються статистичні дані щодо запитів на скасування, розкриття інформації про абонентів та пов’язаних з цим питань. Він спрямований на забезпечення прозорості та прозорості операційних процесів Mega для користувачів, регуляторів та постачальників з точки зору конфіденційності та дотримання законодавства . Згідно з останнім звітом за 2021 рік, MEGA наразі має понад 230 мільйонів зареєстрованих користувачів у більш ніж 200 країнах і територіях. Загалом користувачі Mega завантажили понад 100 мільярдів різних файлів .

## Висновки до розділу 1

Нами в даній часини проекта було розглянуто теоретичні поняття про месенджери та хмарні системи зберігання даних, їх вибір та особливості для бота для взаємодії між Telegram та сервісом хмарного збереження даних який буде розроблений в ході цієї дипломної роботи. Це дало можливість врахувати особливості систем їх недоліки та переваги що привезло до полегшення розробки програми.

На даний момент часу месенджери дуже сильно розповсюдились та переросли свої початкову ціль і функціонал як передача інформації у вигляді повідомлень та дзвінків та стали платформами для створення цілих інтернет-спільнот, деякі з них для розваг та музики, інші для новин та розповсюдження медіа контенту, і навіть для обслуговування клієнтів та ведення бізнесу. Все це стало можливо завдяки ботам які значно спростили та автоматизували багато різних дій та процесів , а надто Telegram які надали можливість користувачам їх легко створювати. Темою та метою даного дипломного проекту є створення бота для взаємодії між месенджером та хмарними сховищами даних для подальшого полегшення та автоматизації обміну та збереженню даних що робить даний проект корисним та актуальним.

# РОЗДІЛ 2
# АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РЕЛІЗАЦІЇ ПРОЄКТУ

## 2.1. IDE, платформа та вибрана мова для програмування бота.

### 2.1.1. IDE Python PyCharm

PyCharm — це крос-платформне середовище розробки, яке використовується для програмування на Python. Це одне з найкращих середовищ для розробки мови програмування Python, яке можна використовувати в Windows, macOS та Linux. Ця програма містить API, який може використовуватися розробниками для написання окремих плагінів Python, щоб вони могли покращувати певні функції. PyCharm (Малюнок 2.1) безпосередньо підтримує розробку Python. Тож ви можете просто відкрити новий файл і почати писати код. Ви також можете запускати та налаштовувати Python у PyCharm, і він підтримуватиме керування версіями та проектами. Існують також інструменти для роботи зі сховищами GIT у середовищі

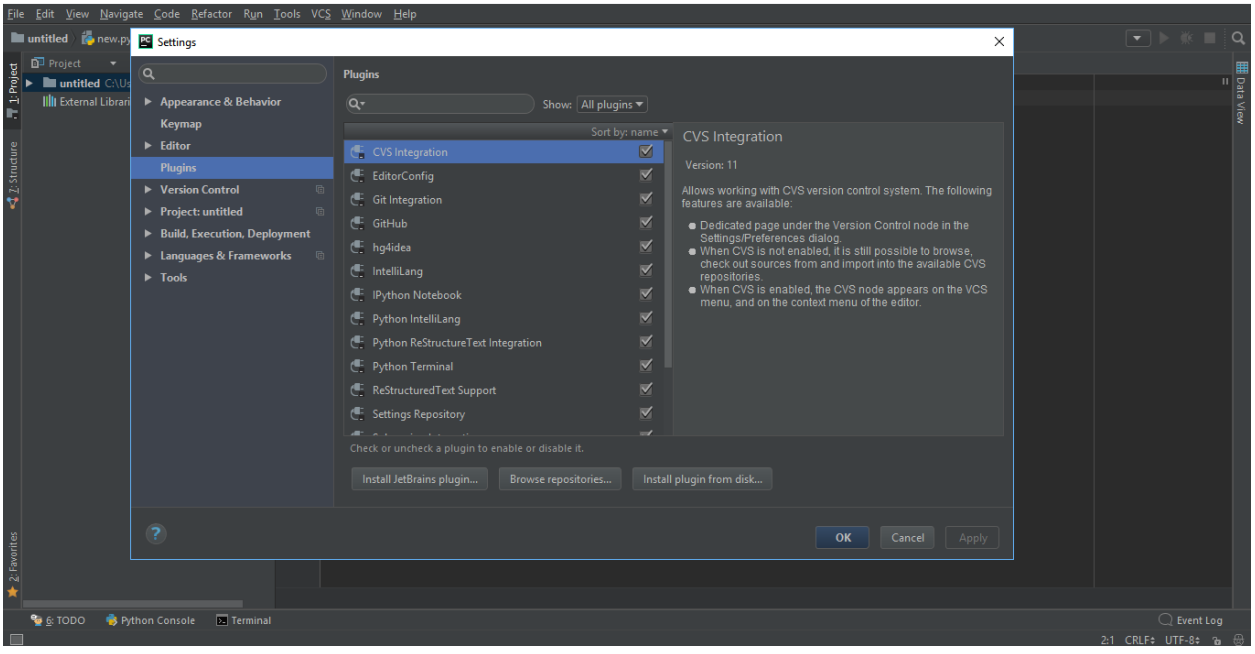



Рисунок 2.1 – Інтерфейс PyCharm на Windows 10

Особливості:

- Підтримується розумний редактор коду Python
- CoffeeScript, JavaScript, CSS і TypeScript
- Розумний пошук, щоб перейти до будь-якого класу, персонажа або файлу
- Інтелектуальна кодова навігація
- Безпечний і швидкий рефакторинг коду
- Надає доступ до кількох баз даних із ID

До недоліків можна віднести повільне завантаження середовища та налаштування стандартних параметрів для існуючих проектів.

### 2.1.2. Python

**Python** — це високорівнева об'єктно-орієнтована мова програмування з динамічною семантикою. Його структури даних високого рівня поєднуються з динамічним записом і закриттям, що робить його ще більш цікавим для створення динамічних додатків,


Рисунок 2.2 – Логотип Python

Простий і легкий у засвоєнні синтаксис Python підкреслює легкість читання та зменшує витрати на обслуговування програми. Python підтримує пакети та модулі, які сприяють модульності програмного забезпечення та повторному використанню коду. Перекладач Python і велика стандартна бібліотека доступні для всіх у безкоштовній первинній або двійковій формі на всіх основних платформах і можуть вільно поширюватися.

Програмісти часто люблять писати на даній мові оскільки вона є доволі ефективною для більшої ефективності, це стається через крок компіляції відсутній що робить звичайний цикл програмування і робить його більш ефективним та швидким. Налаштувати програми Python дуже просто: помилка або неправильний запис не дасть отримати помили в сегментації,. Просто при появі помилки, він викликає виняток. Якщо програма не заважає винятку - транслятор друкує трасу стека. виконувати покрокове виконання коду одночасно тощо.

Перекладач Python зараз використовується практично на всіх платформах і операційних системах, і це незаперечна відмінність і перевага перед іншими.

Це була оригінальна мова, але її типи даних на різних пристроях часто займають багато пам'яті, що ускладнює написання великих програм.

Також відмінною рисою є розширення мови, що також має велике значення. Це показує, що у зацікавлених розробників є можливість вдосконалити та покращити цю мову. Перекладач програмно реалізований мовою програмування C, а вихідний код відкритий для всіх що змінюється.

Ще один важливий плюс – наявність безлічі модулів, що реалізують різні функції підтримки. Ці модулі зазвичай програмуються на мовах C, Python і розробляються більш кваліфікованими спеціалістами. Нижче наведено найпопулярніші модулі як приклади:

- **Numpy —** популярна і широко використовувана бібліотека яка дозволяє проводити більш ефективні розрахунки в коді, також використовується в машинному навчанні.

- **Scikit-Learn** — є одною з найкращих бібліотек для управління складними даними,надає можливість використання елементів з картинок і текстів, перехрестна перевірка та багато алгоритмів проведення машинного навчання.

- **Bokeh** являє собою прекрасний модуль за допомогою якого ми можемо ортримати доступ до графіки, використовується для цього HTML та JavaScript. Цей модуль є дуже гарним рішенням для WEB розробок.

Нижче ми розглянемо використані модулі докладніше розробити програму для цієї роботи.для використання в якості мови сценаріїв або як мови для об'єднання існуючих компонентів (Малюнок 2.2).

## 2.2.Огляд модулів та елементів які були використані для розробки

### 2.2.1. Бібліотека Json

Json модуль із стандартної бібліотеки мови програмування Python є досить ефективним інструментом для взаємодії з файлами JavaScript Object Notation. За допомогою цього модуля ми можемо достатньо легко взаємодіяти з даними, працюючи з різними об’єктами JSON. Це значно полегшує створення веб-додатків на Python.

JSON – це дуже загальний текстовий формат, представлений у текстовій формі. Як випливає з назви, спочатку JSON був заснований на популярній мові JavaScript, але, тим не менш, добре взаємодіє з багатьма різними платформами через свої відносної простоти та сумісність. Цей формат дозволяє програмісту серіалізувати інформаційні структури для подальшої передачі між програмами. Наприклад, обмін текстовою інформацією про клієнтів в інтернет-магазині між сервером і браузером. Інформація у форматі JSON може бути представлена двома способами:послідовність ключів і пар відповідних значень - це просто набір впорядкованих значень.

Як правило, будь-яка мова програмування високого рівня підтримує такі структури даних. Значеннями, що передаються в JSON, можуть бути об’єкти , рядки, числа одновимірні масиви  і літерал(рис. 2.3) (істина, помилка, нуль). . Python підтримує формат JSON завдяки модулю json і методам кодування та дешифрування даних. Це полегшує читання та прийняття інформації.

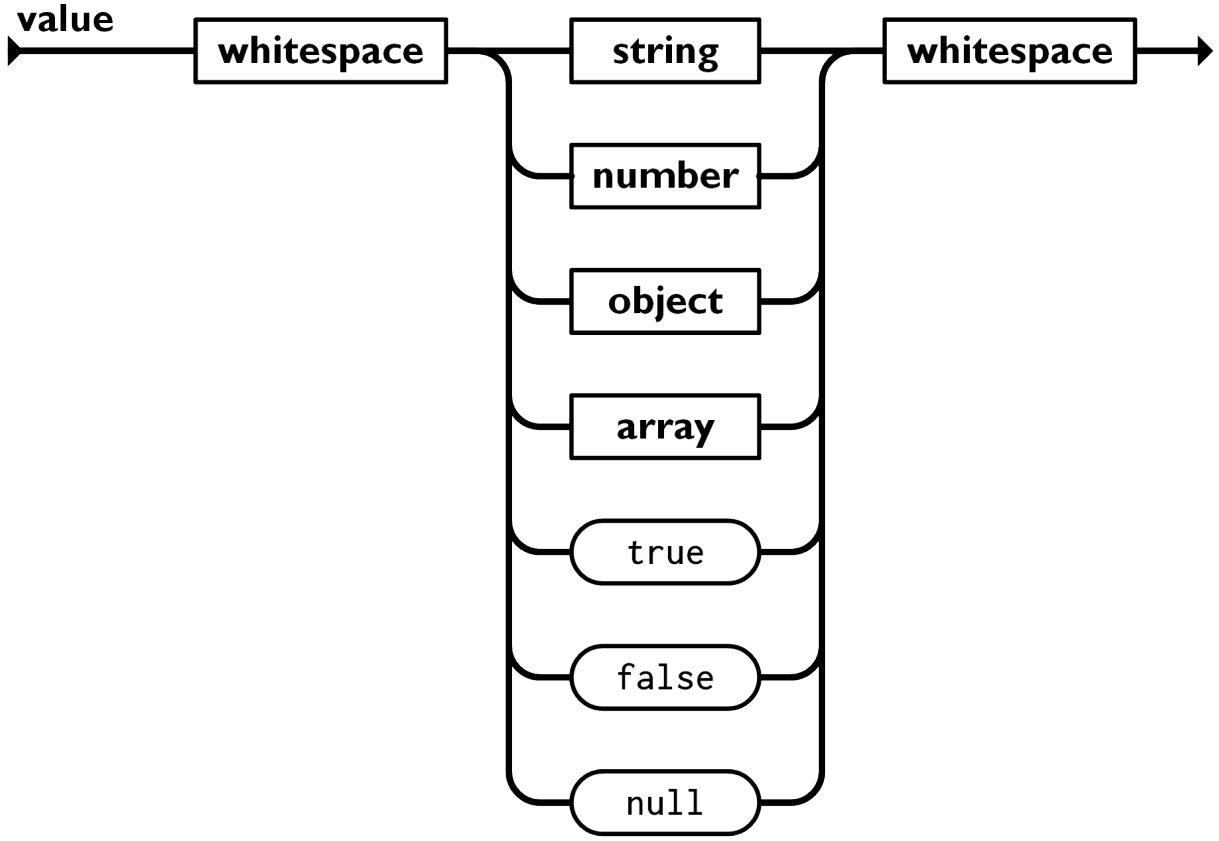



Рисунок 2.3 – літерал як схема

### 2.2.2. Бібліотека Google auth oauthlib

Ця бібліотека використрвується в ботів для керування акаунтом  в Goggle Drive та спрошення  доступа до їхнього API, f сама вона складается з двох частин:

**OAuthLib** бібліотеки для  імплементації стандарту OAuth для програм створених на Python. Одним з недоліків цього стандарту є певна складність та проблеми які виникають при спробах імплементації, і хоча є декілька інших

спроможних бібліотек для цього , перевагою OAuthLib є те що данна бібліотека імплементує логіку OAuth1 або OAuth2 без використання конкретного об’єкта запиту HTTP або веб-фреймворку. Його модна використувувати щоб прищепити підтримку клієнта OAuth у свою улюблену бібліотеку HTTP або надайте підтримку своїй улюбленій веб-фреймворку.

**Google Auth Python Library —** що являє собою бібліотеку яка спрощує різні використання механізмів міжсерверної аутентифікації Google для доступу до API Google.

Тож Google auth oauthlib собою об'єднання двух окремих бібліотек Google Auth Python Library та OAuthLib - Python Framework for OAuth1 & OAuth2 для спрощення використовування стандарту OAuth при користуванні Google API

### 2.2.3. Бібліотека TeleBot.

Бібліотека TeleBot — це бібліотека для взаємодії з TelegramBotAPI, яка використовується для спрощення та мінімізації програмного коду. Усі типи доступні в type.py. Усі вони точно такі ж, як і типи API Telegram, за винятком поля Message, яке називається from_user (оскільки це захищена піктограма Python). Такі атрибути, як message_id, адресуються безпосередньо, наприклад: message.message_id. При розробці та створенні програм, які не вимагають швидкості їх роботи, слід зазначити, що такий атрибут, як message.chat, можна застосувати як до користувача, так і до групової бесіди.

Усі методи API поміщені в клас TeleBot. Імена Python були перейменовані, щоб відповідати всім загальним умовам. Декоративною особливістю екземпляра TeleBot є обробник повідомлень. Формується одним або кількома фільтрами. Потім при оброблені певного повідомлення вони відправляють 0 чи 1, тобто True чи False, і при  True, менеджеру повідомлень надається можливість взаємодіяти з повідомленням.

### 2.2.4. Бібліотека asyncio

Бібліотека Asyncio — це бібліотека для асинхронного програмування-потокова програма / область користувача,де він керує ходом програми та переходом контексту, а не процесором. Бібліотека була створена як період подій, під час яких були активовані генератори, які підтримують стек, і могли створювати винятки.

Ця бібліотека досить сильна, тому Python вирішив зробити її стандартною бібліотекою. Ключове слово async було додано до синтаксису. Ключові слова розроблені, щоб зробити асинхронний код більш зрозумілим. Тому методи не змішуються з генераторами. Ключове слово async переходить до def, щоб вказати, що метод є асинхронним. Ключове слово await вказує на те, що програма очікується завершити.

Завдяки цій бібліотеці було вирішено такі проблеми потоку:

Зміна контексту процесора: Asyncio є асинхронним і використовує схему подій. Це дозволяє програмно змінити контекст

Статус гонки: оскільки Asyncio запускає лише одну спільну програму і перемикається лише в певні моменти, код не схильний до проблеми потоків перегонів.

Взаємне/активне блокування: вам не доведеться турбуватися про блокування тепер, коли немає гонки. Хоча взаємне блокування все ще може відбуватися в ситуації, коли дві спільні програми викликають одна одну

Вичерпання ресурсів: Вичерпати ресурси дуже важко, оскільки спільні програми працюють на одну тему і не вимагають додаткової пам’яті.

## 2.3. Платформа HEROKU

**Heroku** — хмарна платформа на основі керованої контейнерної системи з потужною системою для розгортання та запуску інтегрованих

інформаційних сервісів та сучасних додатків. Ця компанія була заснована 2007 році, та має декілька програмах рішення для хостингу даних клієнтів, а саме:

- The Heroku Platform
- Heroku Postgres
- Heroku Redis
- Heroku Teams
- Heroku Enterprise
- Heroku Connect
- Heroku Elements

Heroku — це прикладно-орієнтований підхід до надання програмного забезпечення, інтегрованого з найпопулярнішими інструментами та робочими процесами.

Heroku — класичний програмний хостинг. На відміну від звичайних точок доступу, він забезпечує непрямий доступ до віртуальної машини з набором інструментів для публікації програмного забезпечення в попередньо налаштованому середовищі. Heroku запускає програми всередині динозаврів - розумні контейнери в надійному, повністю керованому робочому середовищі. Розробники розміщують свій код різними мовами, і всі вони підтримують Heroku, набори відстежуються, редагуються й оновлюються, тому вони завжди готові й оновлюються. Підтримує програми, які запускаються без ручного втручання під час роботи.

### 2.4 База Даних PostgreSQL

**Heroku Postgres** — це служба хмарної бази даних (DBaaS) для Heroku на основі PostgreSQL. Heroku Postgres надає такі функції, як безперервний захист, відкат і висока доступність; також форки, фоловери та кліпи даних.

**PostgreSQL** також відома як Postgres, це безкоштовна система управління базою даних із відкритим вихідним кодом (RDBMS), яка

підкреслює масштабованість та сумісність із SQL.PostgreSQL обробляє властивості Atomic, Sequence, Isolation, Sustainability (ACID), автоматичне оновлення зображень, матеріалізовані зображення, тригери, зовнішні ключі та збережені процедури . Він призначений для обробки різноманітних робочих навантажень, від окремих машин до сховищ даних або кількох одночасних веб-сервісів. Це стандартна база даних для macOS Server , а також доступна для Windows, Linux, FreeBSD і OpenBSD.

PostgreSQL керує паралелізмом за допомогою багатовимірного контролю паралельності (MVCC), який надає «знімок» бази даних для кожної транзакції та дозволяє вносити зміни, не впливаючи на інші операції. Це значно усуває необхідність блокування читання та гарантує збереження принципів ACID в базі даних. PostgreSQL пропонує три рівні операційної ізоляції: Read Committed, Repeatable Read і Serializable. Оскільки PostgreSQL захищений від брудних читань, Read Uncommitted надає читання замість того, щоб вимагати рівня ізоляції транзакцій. PostgreSQL підтримує повну серіалізацію за допомогою ізоляції знімків, які можна серіалізувати (SSI).

PostgreSQL включає вбудовану підтримку звичайних індексів B-дерева та хеш-таблиць, а також чотири методи введення індексів: узагальнені дерева пошуку (GiST), узагальнені зворотні індекси (GIN), розділені пробілом GiST (SP-GiST) [34] та Блокові проміжні індекси (BRIN). Крім того, можна створити визначені користувачем методи індексації, хоча це дуже привабливий процес.

Індекси виразу можуть бути створені просто за індексом виразу або результатом функції замість значення стовпця.

Часткові індекси, які індексують лише якусь відсоток файлу можна створити додавши в кінці оператора Create Index, що надасть результат у вигляді більш легкого індекса.

Також при бажанні можливо взаємодіяти одразу з н кількістю індексів, щоб надавати складні запити, використовуючи операції індексу растрового

зображення в тимчасовій пам’яті (корисно для програм сховищ даних для об’єднання великих таблиць даних у менші таблиці, наприклад ті, що розташовані в зірковій діаграмі).

Індексування k-найближчих сусідів (k-NN) (також зване KNN-GiST ) забезпечує ефективний пошук «найближчих значень» до показаних, корисно для пошуку подібних слів, об’єктів чи місць, близьких до геопросторових. дані .Це досягається без повного вирівнювання цінностей.

Поодинокі сканування індексів часто дозволяють системі отримувати інформацію з індексів, навіть не входячи в основну таблицю.

У PostgreSQL схема містить усі об'єкти, за винятком ролей і табличних просторів. Схеми ефективно функціонують як простір імен, що дозволяє об’єктам з однаковою назвою співіснувати в одній базі даних. За замовчуванням новостворені бази даних мають схему, яка називається загальнодоступною, але можна додати будь-які додаткові схеми, а публічна схема є необов’язковою.Нові об'єкти створюються в шляху пошуку_ першими будь-якої дійсної схеми (на даний момент існуючої).Підтримуються багато локальних типів даних, зокрема:

- логічне значення
- Довільні точні цифри
- Символ (текст, varchar, char)
- двійковий
- Дата / час (штамп часу / часовий пояс / не час, дата, інтервал)
- Гроші
- Enum
- Розрядні рядки
- Тип текстового пошуку
- Композитний
- Hstore — це сховище ключових значень, яке дозволяє вам розширюватися в PostgreSQL

- Масиви до 1 ГБ загальної пам'яті
- Геометричні примітиви
- Адреси IPv4 та IPv6
- Блоки безкласової маршрутизації домену (CIDR) і MAC-адреси
- XML, який підтримує запити XPath
- Універсальний унікальний ідентифікатор (UUID)
- Нотація об'єктів JavaScript (JSON) і швидший двійковий JSONB (не те саме, що BSON)

Крім того, користувачі можуть створювати власні типи даних, які можна повністю індексувати, зазвичай через інфраструктуру індексування PostgreSQL - GiST, GIN, SP-GiST. До них належать, наприклад, типи даних геоінформаційної системи (ГІС) із проекту PostGIS для PostgreSQL.

Існує також тип інформації, який називається доменом, який такий самий, як і будь-який інший тип інформації, але з додатковими обмеженнями, встановленими творцем цього домену. Це означає, що будь-яка інформація, введена в стовпці з використанням домену, повинна відповідати обмеженням, встановленим як частина домену.

Один тип даних може використовуватися для представлення діапазону інформації, який називається типами діапазону. Це можуть бути дискретні діапазони (наприклад, всі цілі значення від 1 до 10) або безперервні діапазони (наприклад, будь-який час з 10:00 до 11:00). Доступні внутрішні типи діапазонів включають цілі, великі цілі , десяткові, позначки часу і дати чисел. Можна створити окремі типи діапазонів, щоб зробити доступними нові типи діапазонів, наприклад діапазони IP-адрес, які використовують тип inet як базовий, або діапазони з плавучою чисельністю, які використовують тип даних float як основу. Типи діапазонів підтримують включні та виключні межі діапазону за допомогою символів [/] та (/). Проміжні типи також включають перекриття, зберігання, закон тощо.

## Висновки до розділу 2

Під час розгляду відомостей про інструменти та дані для розробки в даному розділі були вибрані технології для створення бота, а саме мовою програмування стала Python а IDE було вибранно PyCharm IDE У мові програмування Python переглянула велику кількість бібліотек, які можна використовувати для створення чат-ботів, але визначила бібліотеки, які найбільше підходять для розробки: Google Auth Python Library, JSON,i Asyncio. Вибір обумовлений тим, що ці бібліотеки містять необхідні і відносно прості інструменти для створення будь-якої програми.

Після дослідів та аналізу також було вирішено що платформою сервера для нашого бота Telegram стане Heroku .

# РОЗДІЛ 3
# РОЗРОБКА ПРОГРАМНОГО ЗАБЕПЕЧЕННЯ

## 3.1 Створення боту за допомогую BotFather

Як вже було описано в попередніх розділах, процес створювання бота проходить через BotFather та є відносно легким (рис. 3.1)

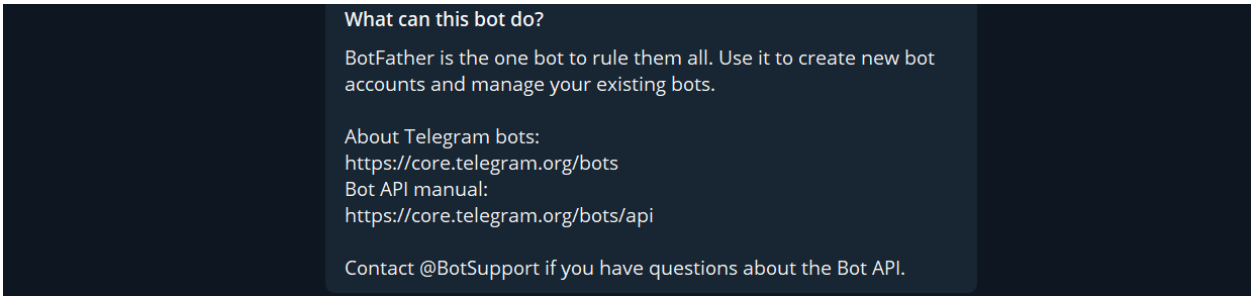



Рисунок 3.1 – початок роботи з BotFather

Спочатку починаємо процес за допомогою команди старт /start (рис. 3.2).

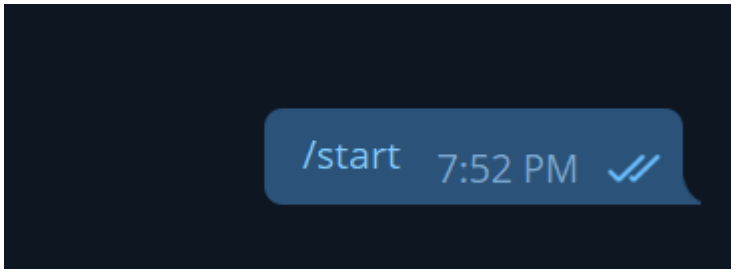



Рисунок 3.2 – Запуск бота для створення нового

Ми отримуємо список можливих команд для BotFather (рис. 3.3).

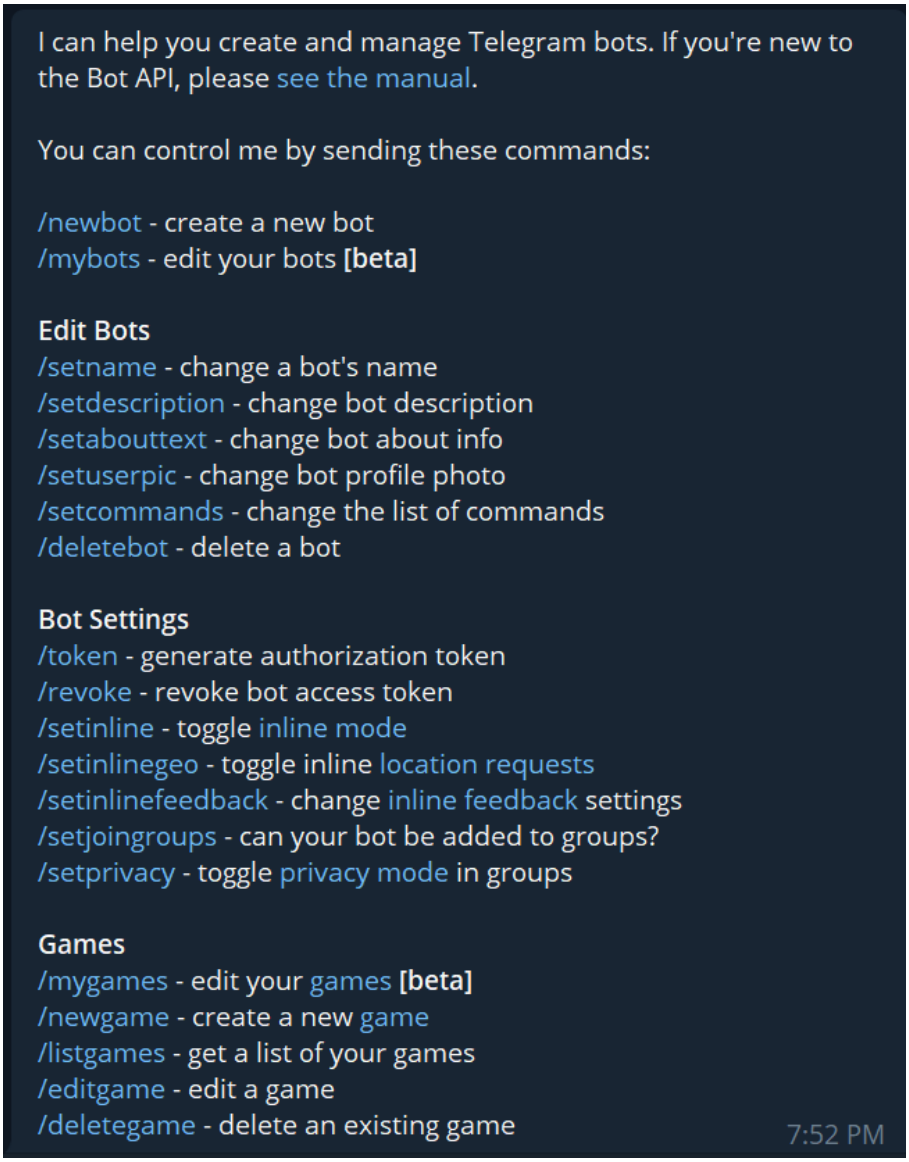



Рисунок 3.3 – Список команд

За допомогою команди /newbot створюємо бот, нам необхідно дати нашому боту ім'я та назву користувача. Сама назва користувача повинна в кінці містити _bot (рис. 3.4).

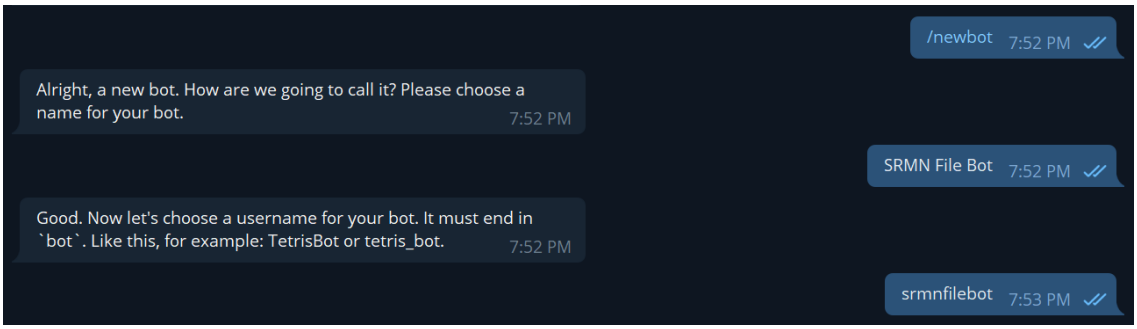



Рисунок 3.4 – створення бота з BotFather

В результаті даних операцій ми також отримуємо токен для доступу до нашого боту (рис. 3.5)

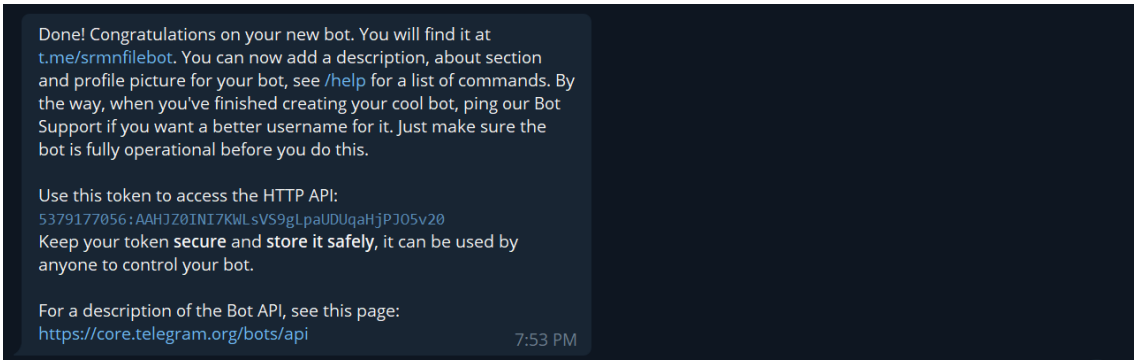



Рисунок 3.5 – отримання токену

Після цього також можна додати опис (рис. 3.6) та додаткову інформацію (рис. 3.7)

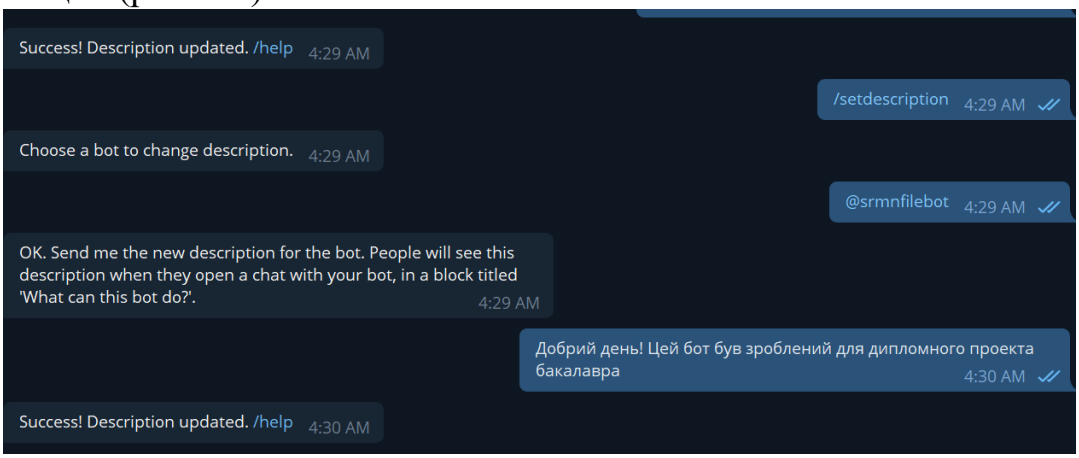



Рисунок 3.6 – додавання опису

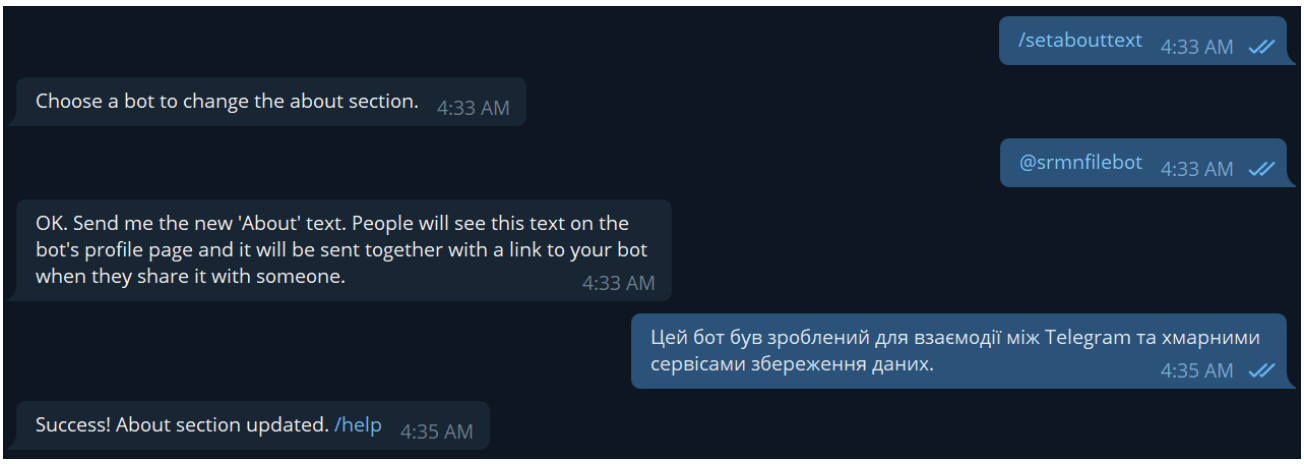



Рисунок 3.7 – додавання додаткової інформації

Наступним кроком надаємо BotFather команди для даного бота (рис. 3.8) з таблиці (таблиця 3.1)

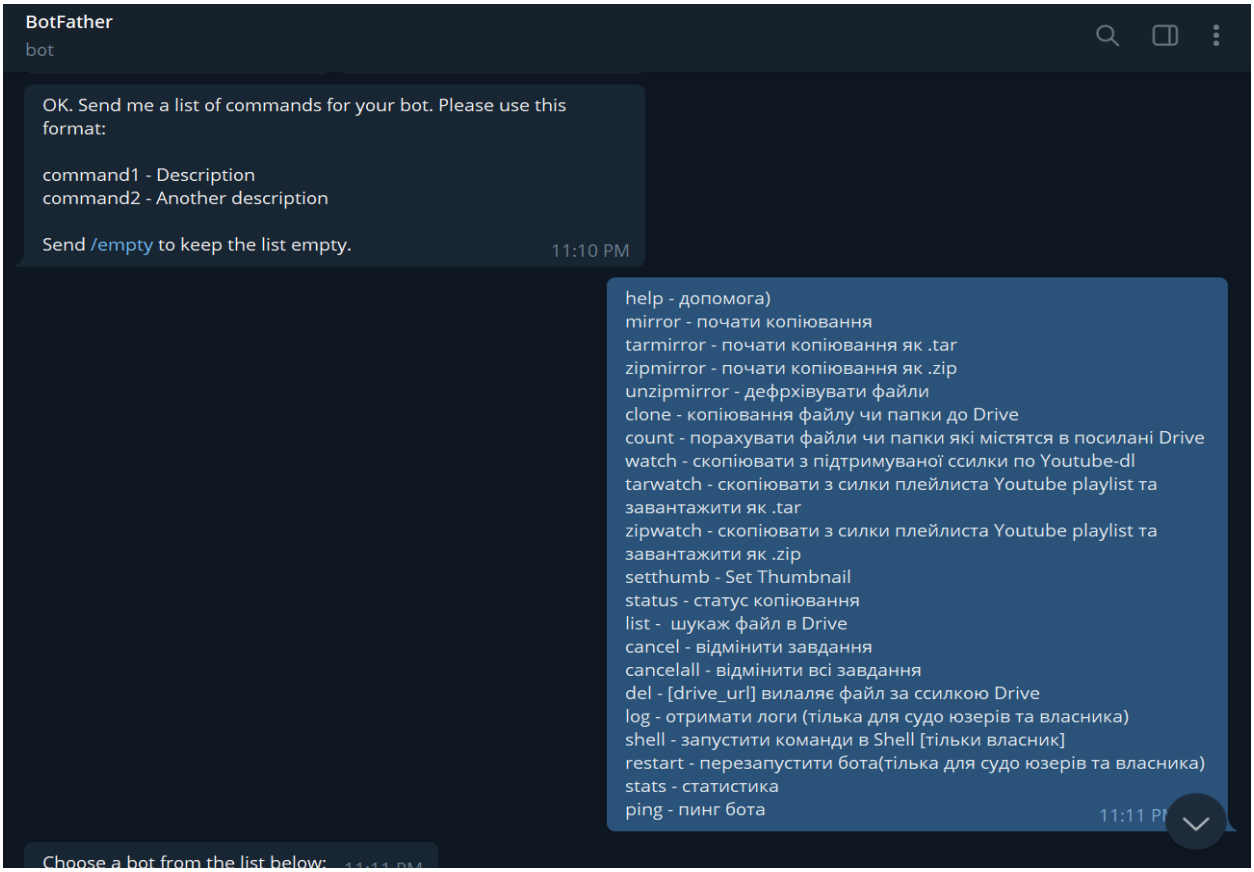



Рисунок 3.8 – додавання команд боту

Таблиця 3.1 – Список команд бота

| Команда | Функціонал |
|---|---|
| help | допомога) |
| mirror | почати копіювання |
| tarmirror | почати копіювання як .tar |
| zipmirror | почати копіювання як .zip |
| unzipmirror | дефрхівувати файли |
| clone | копіювання файлу чи папки до Drive |
| count | порахувати файли чи папки |
| watch | скопіювати з підтримуваної ссилки по Youtube-d |
| tarwatch | Скопіювати з силки плейлиста Youtube playlist та завантажити як .tar |
| zipwatch | Скопіювати з силки плейлиста Youtube playlist та завантажити як .zip |
| setthumb | Поставити ескізи |
| status | Статус копіювання |
| list | Шукаж файл в Drive |
| cancel | відмінити завдання |
| cancelall | відмінити всі завдання |
| del | вилаляє файл за ссилкою Drive |
| log | отримати логи |
| shell | запустити команди в Shell |
| restart | перезапустити бота |
| stats | статистика |

Оскільки для розробки цього бота було вибрано Python , однією з переваг якої є адаптивність, ми за допомогою плагінів та модулів можемо відносно просто створити якісного бота для Telegram. Також необхідним використати токен від Telegram в самій програмі щоб мати спроможність керувати нашим ботом.

Сам інструмент створення ботів BotFather має широкий список функцій деякі з них ми тут опишемо у вигляді таблиці

Таблиця 3.2 – Команди BotFather

| Команди | Функціонал |
|---|---|
| /setname | Надати ім'я |
| /setdescription | Надати текст для 1 раза коди юзер буде використовувати даний чат-бот |
| /setabouttext | Поставити текст в профілі бота |
| /setuserpic | Поставити фотографію в профілі бота |
| /setcommands | Надати список команд |
| /deletebot | Видалити |
| /token | Token для бота |
| /revoke | Знімає легітимність існуючого токену для чат-бота |
| /setinline | Налаштування опції додвангя бота до іншого |
| /cancel | Відмінити нинішню операцію |

Продовження таблиці 3.2

| Команди | Функціонал |
|---|---|
| /setinlinefeedback | Дозволяє отримувати статистику скільки і які команди використовуються користувачами |
| /setprivacy | Які повідомлення твій бот може бачити в групах |
| /setinlinegeo | Дозвіл на отримання гео локації |

Також в BotFather присутні команди для створення ігор в Telegram, цими командами є /newgame, /listgame, /editgame, /deletegame

## 3.2 Створення бази даних за допомогою Heroku PostgreSQL та її імплементація в коді

Спочатку ми створюємо новий додаток через головне меню та додаємо Heroku Postgress як основу. Це можна дуже легко зробити через кнопку створення яка знаходитися на екрані. Сам сервіс Heroku можна використовувати безплатно тому він може бракувати деяких можливостей його більш дорожчих аналогів, тому оскільки ми використовуємо безплатний план ми можемо створити тільки одні програму, чого нам цілком досить для нашого проекту(рис. 3.9).

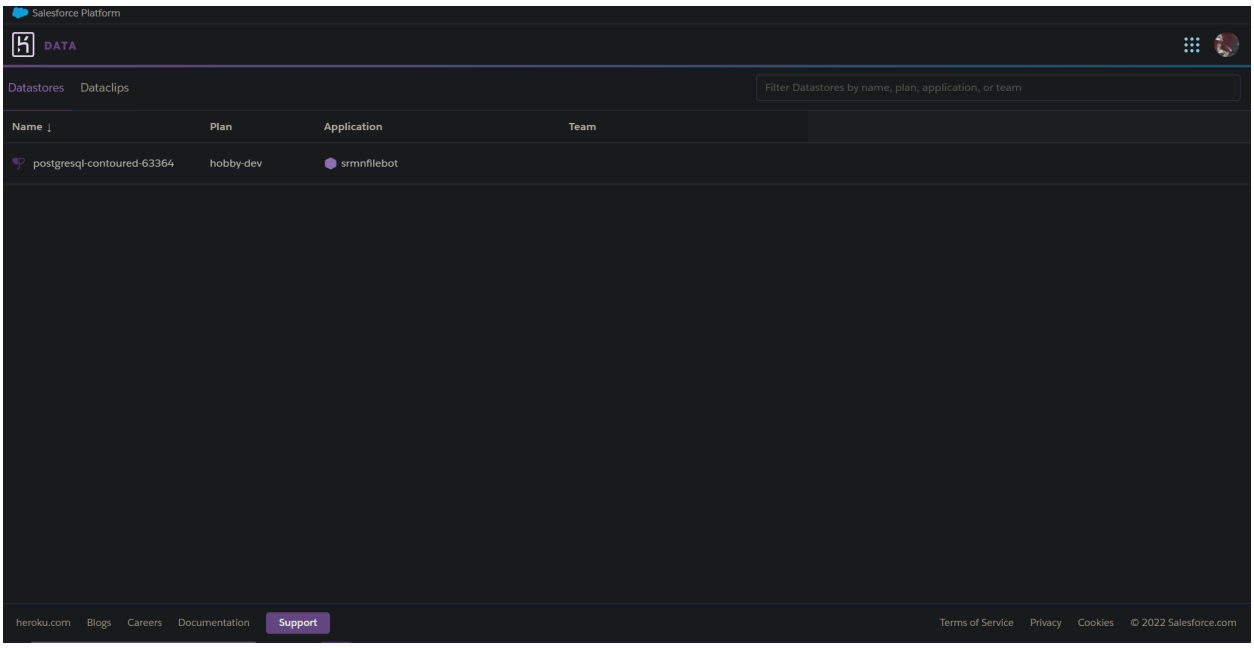



Рисунок 3.9 – створення нової програми

В процессі створення ми вибираємо найближний до нас сервер для хостингу нашої бази даних, для нас це на даний є сервером Європи.

Після цього ми даемо назву нашому додатку та вибираємо базу даних PostgreSQL, це можна зробити перейшовши на Resource, де це можна буде зробити за допомогою панелі. Там ми вводимо в пошук Heroku Postgres та вибтражмо нашу програму для цього та тарифний плна що є в нашому випадку безплатним планом з назвою Hobby Dev.

Звичайно ж щоб використовувати нашу базу даних яку ми тільки що створили ми повинні спочатку отримати облікові дані, також ми повинні отримати ссилку на сам додаток баз даних для коректної роботи нашого коду. Ці всі задачі модна виконати зайшовши до Resource та вибравши там Heroku Postgres. Це дасть нам можливість переглядати конфігурації бази даних через вкладку з налаштувань, де ми зможеио побачити облікові дані та ссилку додатка.(рис. 3.10)

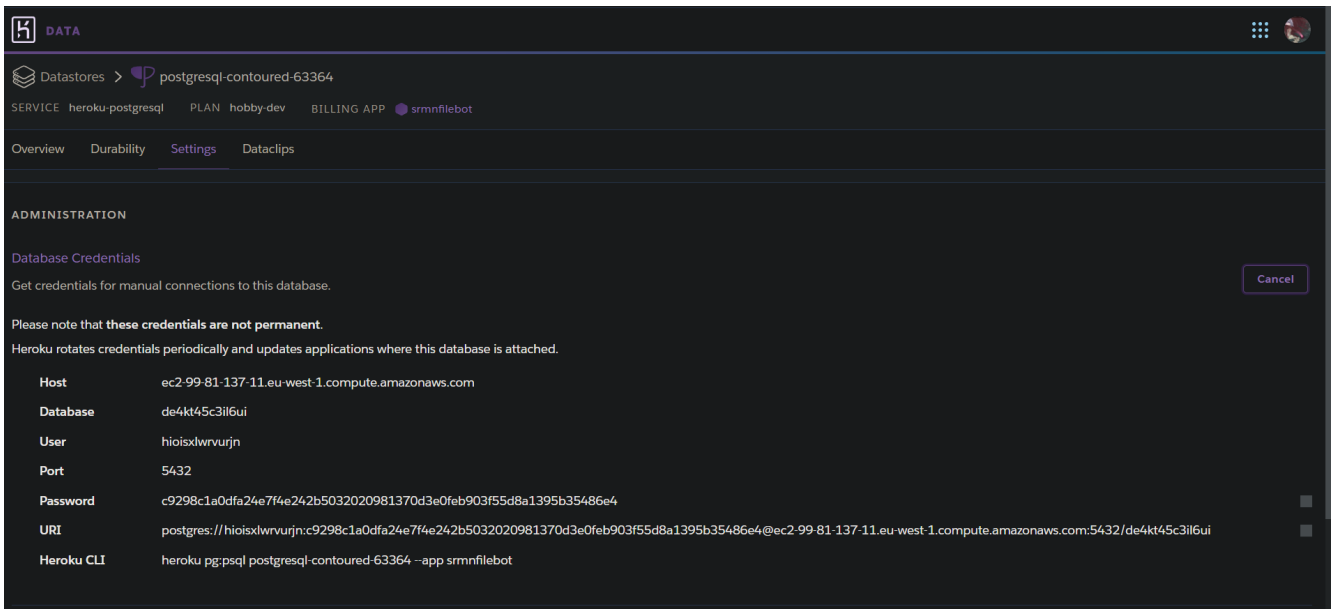



Рисунок 3.10 – облікові дані бази даних

Зараз ми вже дізнались всю необхідну інформацію роботи додатка з базою даних PostgreSQL,тобто ми тепер можемо імплементувати код для кліжнта для зєднання з базою даних.

Сама взаємодія між нашим ботом та базою даних проходить за допомогою модуля psycopg2. Psycopg2 — модуль який слугує як драйвер для мови програмування Python та PostgreSQL, його перевагами є малий розмір, швидкість та стабільність, саме тому він і був вибраний. Основними його командами в даному проекті є «conn» та «cur».

За допомогою команди «conn» ми підключаємось до вже існуючої таблиці, а за допомогою команди «cur» ми виконуємо операції над базою даних. Обидві ці команди ми використовуємо для створення таблиці в базі даних(рис. 3.11) та для можливості надавання юзерів бота правами sudo юзерів(рис. 3.12).

```python
def mktable():
    try:
        conn = psycopg2.connect(DB_URI)
        cur = conn.cursor()
        sql = "CREATE TABLE users (uid bigint, sudo boolean DEFAULT FALSE);"
        cur.execute(sql)
        conn.commit()
        logging.info("Created!")
    except Error as e:
        logging.error(e)
        exit(1)
```

Рисунок 3.11 – створення таблиці в базі даних

```python
try:
    DB_URI = getConfig('DATABASE_URL')
    if len(DB_URI) == 0:
        raise KeyError
except KeyError:
    DB_URI = None
if DB_URI is not None:
    try:
        conn = psycopg2.connect(DB_URI)
        cur = conn.cursor()
        sql = "SELECT * from users;"
        cur.execute(sql)
        rows = cur.fetchall()  #returns a list ==> (uid, sudo)
        for row in rows:
            AUTHORIZED_CHATS.add(row[0])
            if row[1]:
                SUDO_USERS.add(row[0])
    except Error as e:
        if 'relation "users" does not exist' in str(e):
            mktable()
        else:
            LOGGER.error(e)
            exit(1)
    finally:
        cur.close()
        conn.close()
```

Рисунок 3.12 – функція надання прав sudo

### 3.3 Генерування токена для GoogleDrive

API компанії Google для того щоб користувач і програма могли отримати доступ до автентифікації та авторизації, що потрібно щоб використовувати наприклад API для користування Google Drive, використовується протокол OAuth 2.0 для процесів по автентифікації та авторизації(рис. 3.13).

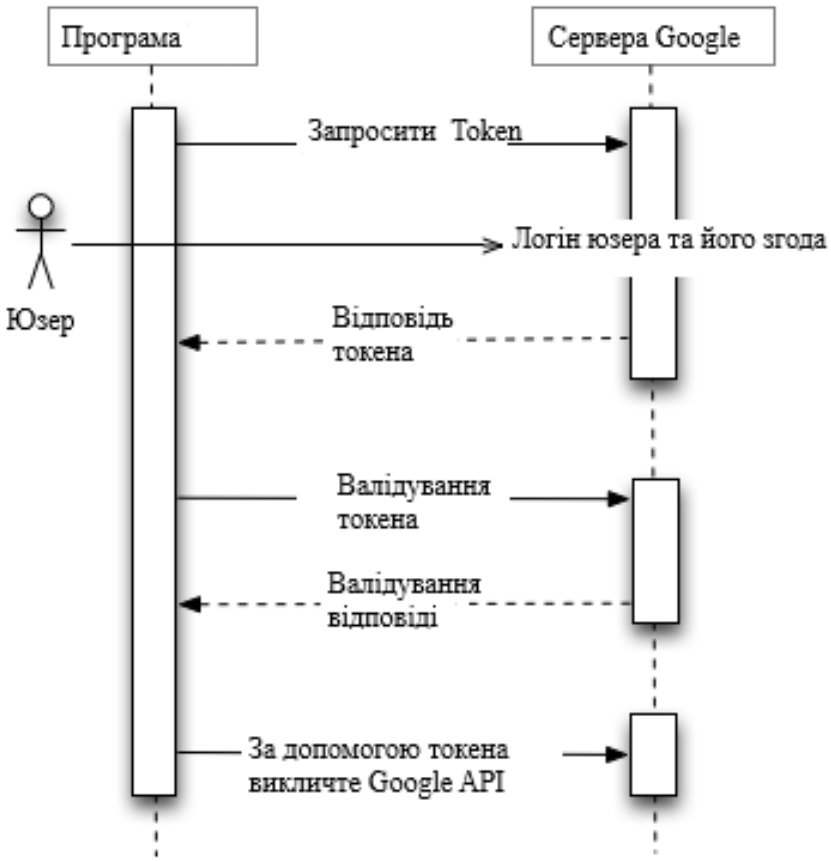



Рисунок 3.15 – принцип роботи токена

Є всього 5 простих кроків в усіх випадках використання Google API за допомогою протоколу OAuth 2.0. :

- Спочатку нам потрібно потрібно отримати наші облікові дані. Це можна зробити на сайті API в налаштуваннях

- Потім нам потрібно отримати токен від серверів Google, в даному проекті це виконується завдяки скрипту. Спочатку ми автентифікуємся, після цього отримуємо код який ми вводимо, також оскільки наша програма має статус тестування, то ми повинні надати на сайті API собі доступ.
- Третім кроком ми повинні дослідити наскільки юзер надає нам доступ, це необхідно тому що сфера, включене у ваш запит, може не збігатися з сферою у вашій відповіді, навіть якщо користувач надає все необхідне сфери.
- Четвертим буде надсилання токену доступу до серверів Google у вигляді запита на авторизацію
- Останнім, п'ятим кроком є оновлення токена для доступа якщо це необхідно, постільки токен доступу від Google має ліміт свого існування коли він закінчується а програмі треба отримати доступ до Google API, то це дозволяє отримати оновлений токен.

Всі ці дії ми робимо за допомогою модулів google_auth_oauthlib.flow та google.auth.transport.requests ми можемо запрошувати наші облікові дані для Google Drive,якщо їх нема або якщо вони недійсні та зберігати їх на наступний сеанс(рис. 3.14).

```python
import pickle
import os
from google_auth_oauthlib.flow import InstalledAppFlow
from google.auth.transport.requests import Request

credentials = None
__G_DRIVE_TOKEN_FILE = "token.pickle"
__OAUTH_SCOPE = ["https://www.googleapis.com/auth/drive"]
if os.path.exists(__G_DRIVE_TOKEN_FILE):
    with open(__G_DRIVE_TOKEN_FILE, 'rb') as f:
        credentials = pickle.load(f)
        if (
            (credentials is None or not credentials.valid)
            and credentials
            and credentials.expired
            and credentials.refresh_token
        ):
            credentials.refresh(Request())
else:
    flow = InstalledAppFlow.from_client_secrets_file(
        'credentials.json', __OAUTH_SCOPE)
    credentials = flow.run_local_server(port=0)

# Save the credentials for the next run
with open(__G_DRIVE_TOKEN_FILE, 'wb') as token:
    pickle.dump(credentials, token)
```

Рисунок 3.14 – імплементація керування токеном

### 3.4 Функція логування для бота

Функція логування є дуже важливим явищем при розробці ПО. Завдяки ній стало можливе більш детальний аналіз та дослідження програми в робочому стані. Також ця функція дозволяє отримати більш детально інформацію про помилки коді, що дуже стало в нагоді та дозволило знайти помилки в самій програмі та помилки в конфігураційних файлах. Логування показало себе дуже бажанною функцією навіть в нашому невеличкому проекті, в великих же проектах ця функція просто стає необхідною навіть і після виходу додатка щоб його підтримувати. Самі ж логи розділяються на 5 рівнів:

- Debug — дебаг, детальна інформація, частіше всього необхідна для діагностування проблеми.
- Info — інформація, менш детальна інформація про те як працює програма.
- Warning — попередження про або можливу проблему в майбутньому або про те що сталась ситуація яке не була запрограмована та задумана.
- Error — помилка, з'явилась ситуація в якій програма не може виконати якусь певну свою функцію.
- Critical — критична помилка, сталась ситуація в якій програма не може більше виконуватись.

Також при бажані можна представити логи в виді файла с форматом .log та відформатувати як саме текст подається в логі при наявному бажанні.

Нижче буде наведено схему роботи логінга(рис. 3.15).

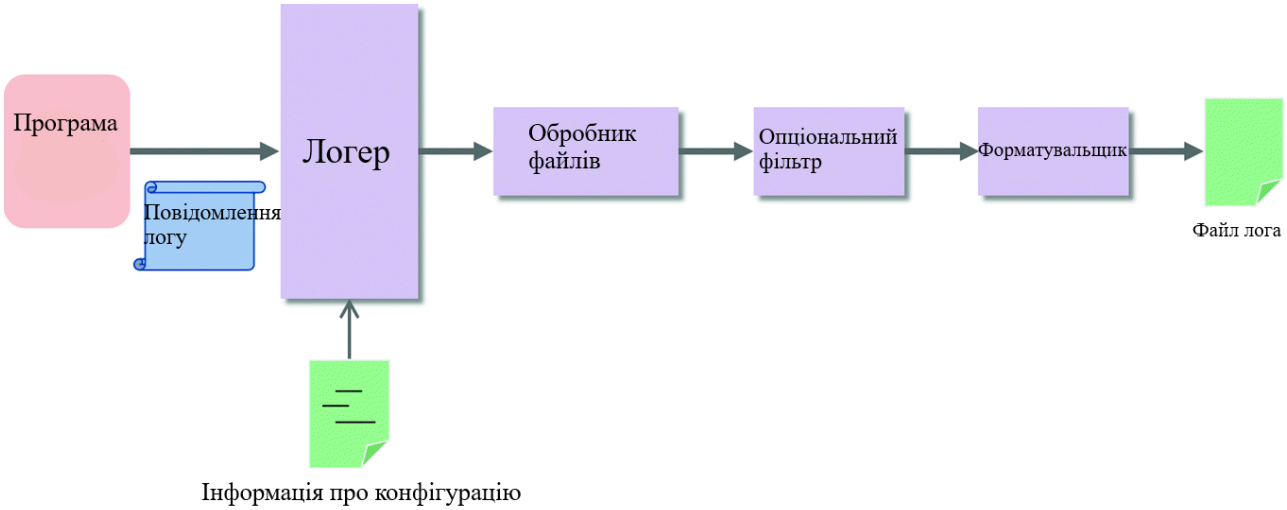



Рисунок 3.15 – імплементація керування токеном

Функцію логінга для перевірки та дослідження як працює наша програма було імплементовано за допомогою модуля logging. Приклади імплементації даного модуля буде представлено у виглядах налаштуваннь самого логера(рис. 3.16) та очікуваних помилках для цього бота при його взаємодії з mega.nz(рис. 3.17).

```
logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',
                    handlers=[logging.FileHandler('log.txt'), logging.StreamHandler()],
                    level=logging.INFO)

LOGGER = logging.getLogger(__name__)

CONFIG_FILE_URL = os.environ.get('CONFIG_FILE_URL', None)
if CONFIG_FILE_URL is not None:
    res = requests.get(CONFIG_FILE_URL)
    if res.status_code == 200:
        with open('config.env', 'wb+') as f:
            f.write(res.content)
            f.close()
    else:
        logging.error(res.status_code)

load_dotenv('config.env')
```

Рисунок 3.16 – налаштування для логера

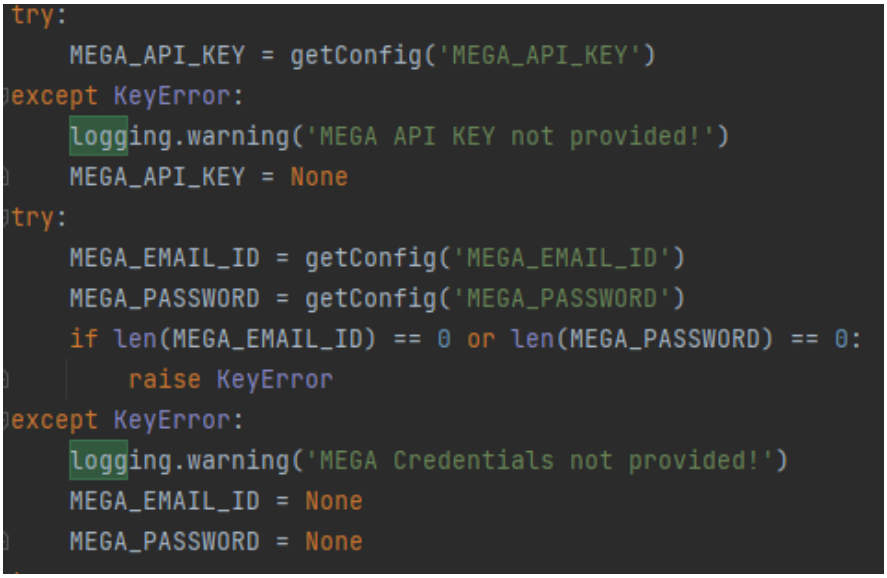



Рисунок 3.16 – очікувані попередження при роботі з mega.nz

## Висновки до розділу 3

В даному третьому розділі були реалізована детальний розбір кода бота для Telegram, всього його шляху розробки починаючи від його ініціювання за допомогою BotFather та закінчуючи готовим та робочим ботом.

Були розглянуті основні команди та функція для створення ботів в Telegram та розглянуто процес створення саме цього бота, його команди і основні команди BotFather були занесені в відповідні таблиці.

Також було досліджено процес створення баз даних за допомогою Heroku PostgreSQL на прикладі бази даних для даного бакалаврського проекту

Робота токенів для доступу до Google API також була продемонстрована та досліджена, а саме 5 її основних кроків, як саме працює система з доступом та її імплементація в коді цієї програми.

Імплементація логування також була розглянута, також було досліджено як само працює даний функціонал і були розглянуті приклади його імплементації в коді

# РОЗДІЛ 4

# Досліждення роботи та показ бота

## 4.1 Тестування та дослідження роботи бота SRMN

Результатом роботи даного бакалаврського проекту став чат-бот для Telegram з назвою SRMN File Bot з юзернеймом @srmnfilebot , демонстрація його функціонала та тест його функцій проводиться через ручне тестування в чаті даного бота в операційній системі Windows 10 Pro версії збірки 19044.1706 в додатку Telegram Desktop версії 3.7.5 beta.

Саме наше тестування починається з пошуку нашого бота в Telegram та його активування командою /start (рис. 4.1)

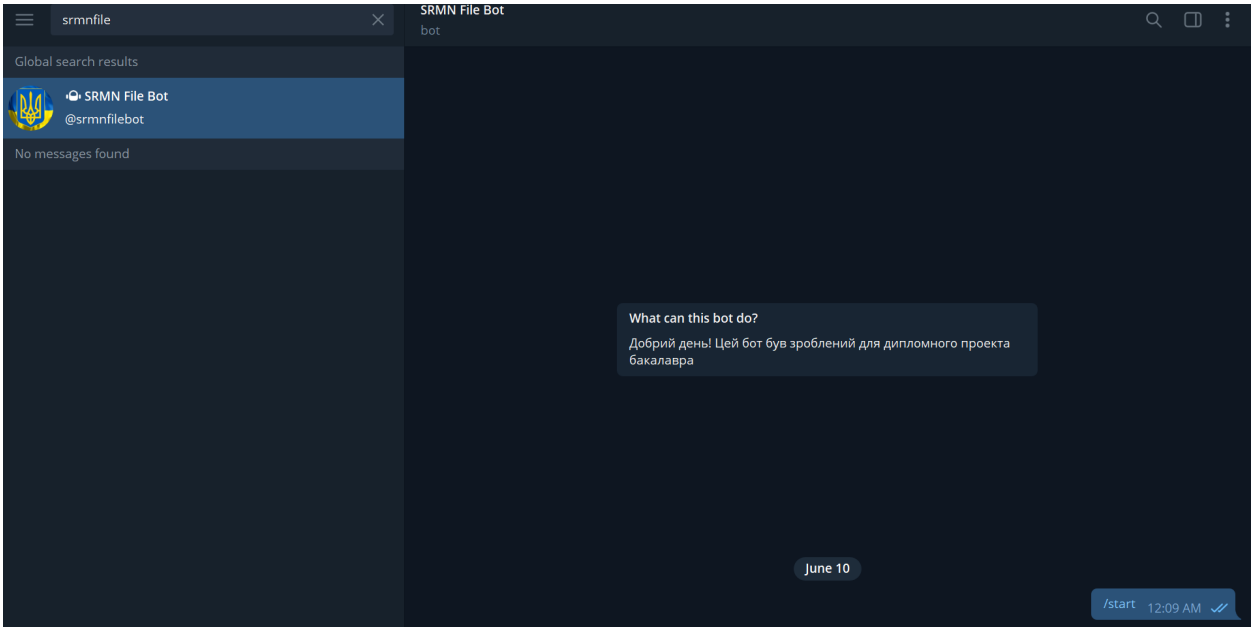



Рис.4.1 Початок використання бота

Далі за допомогою кнопки Menu ,яка знаходиться в нижній частині екрану біля поля для написання повідомлення, ми можемо отримати список всіх команд для бота які були записані через BotFather(рис. 4.2).

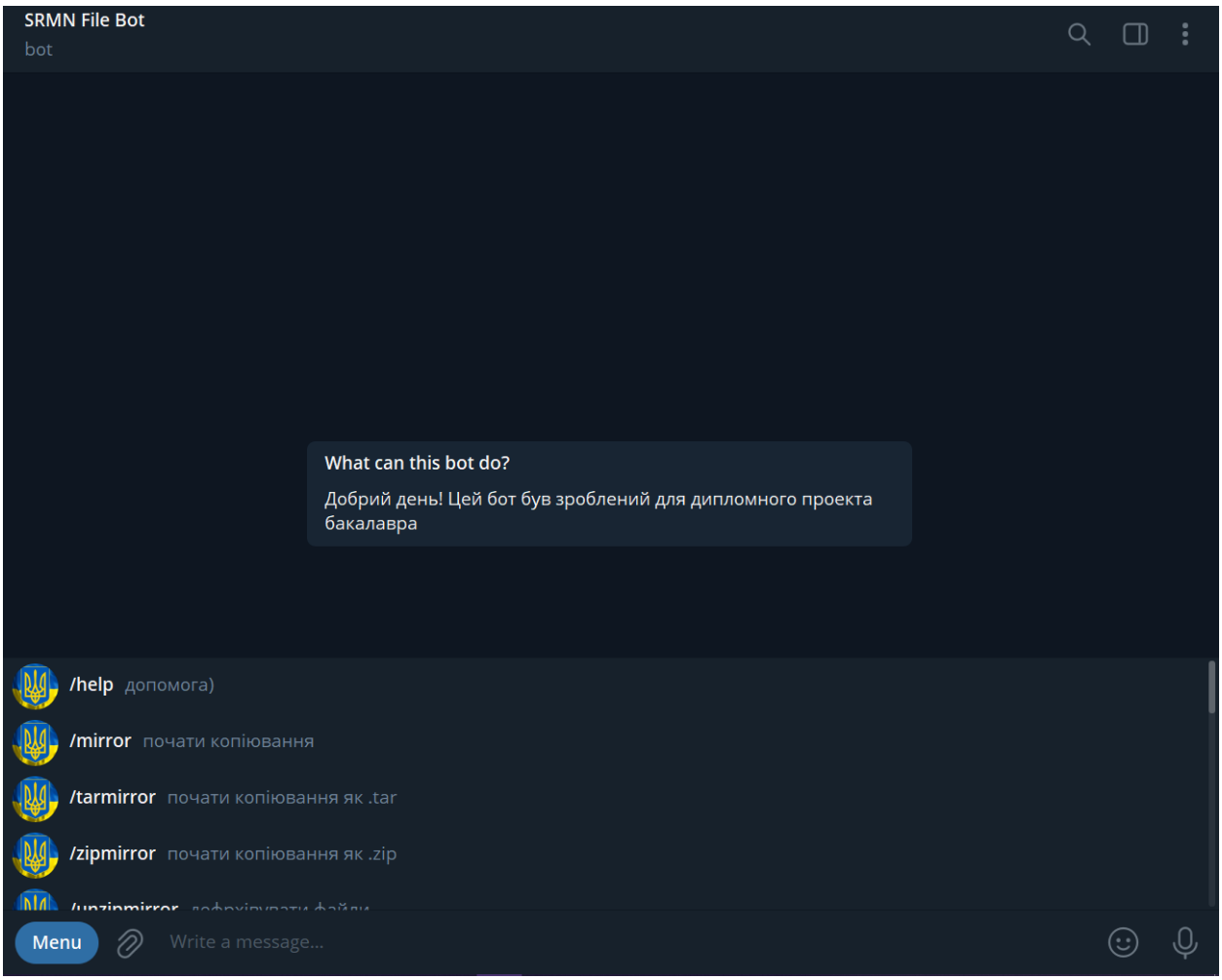



Рис.4.2 Використання функції Menu

Також через функцію /help ми можемо отримати пояснення команд але вже на англійській мові(рис. 4.3)


Рис.4.3 Команда /help

За допомогою команди /clone ми клонуємо файли з заданої папки, яка надається у вигляді ссилки (рис. 4.4)

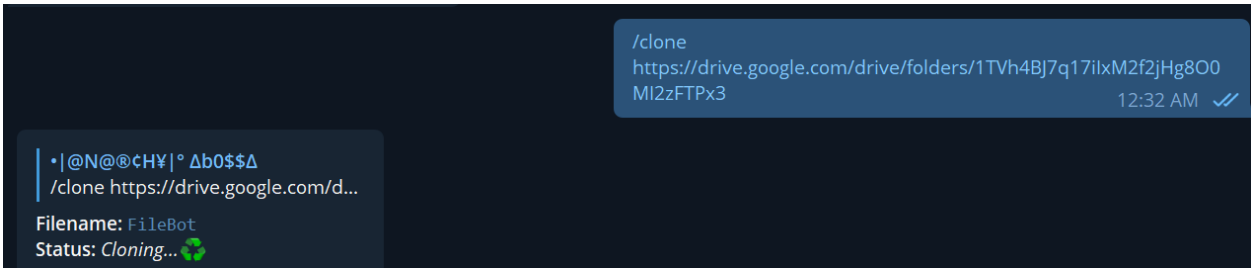



Рис.4.4 Команда /clone

Також є аналоги команді /clone у вигляді команд /tarmirror та /zipmirorr яка копіює файли з заданої по ссилці папки у вигляді файла архіва у форматі .tar(рис. 4.5), а команда /zipmirorr відповідно також копіює файли з заданої по ссил папки у вигляді файла архіва але вже у вигляді формата .zip(рис. 4.6).

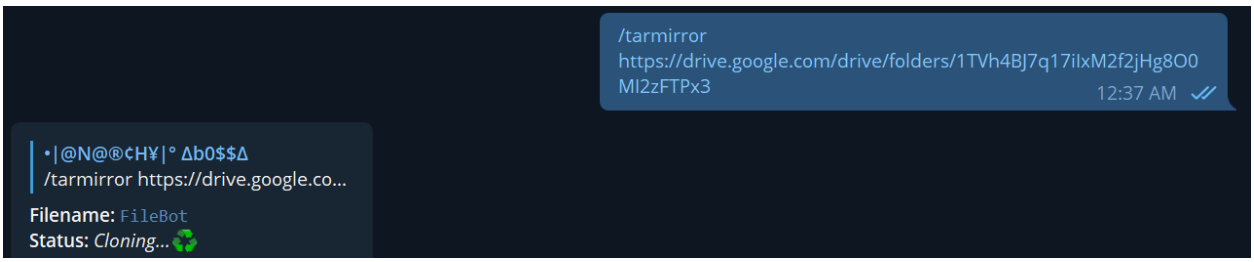



Рис.4.5 Команда /tarmirror

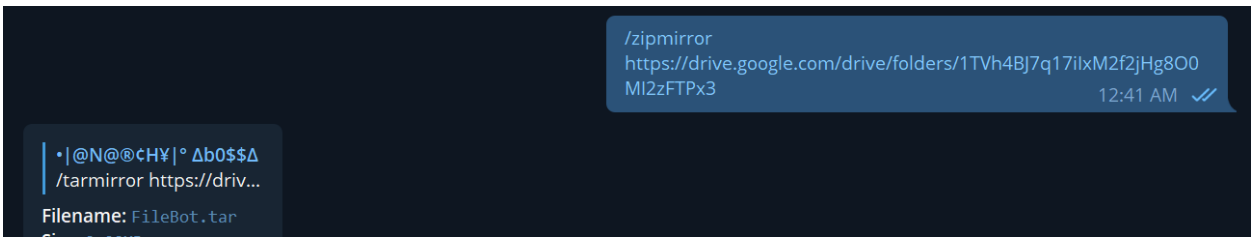



Рис.4.6 Команда /zipmirror

Функціонал команди /list дозволяє шукати шукати файли по їхньому імені у всьому хмарному сховищі наших даних(рис. 4.7).


Рис.4.7 Команда /list

Також присутній функціонал обрахування кількості файлів в заданій папці яка була задана через ссилку за допомогою команди /cout/(рис. 4.8).

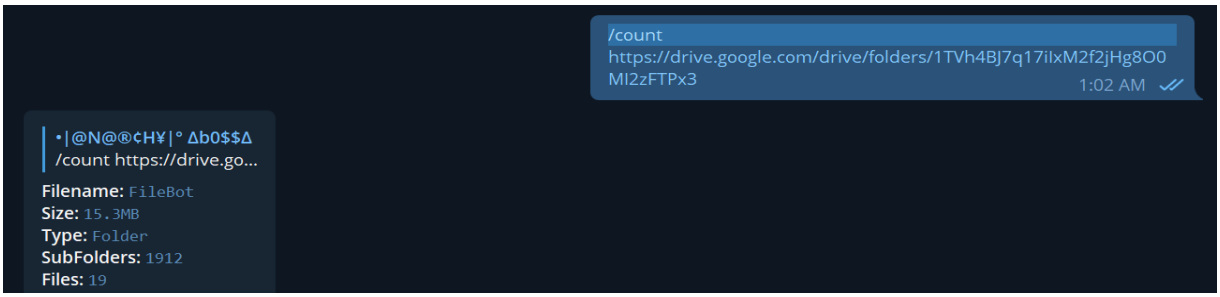



Рис.4.7 Команда /cout

Якщо користувач забажає завершити свою операцію до її виконання то він це може зробити це за допомогою команди /cancel(рис. 4.8), також якщо виконується декілька команд одночасно їх можливо відмінити за допомогою команди /cancelall(рис. 4.9)

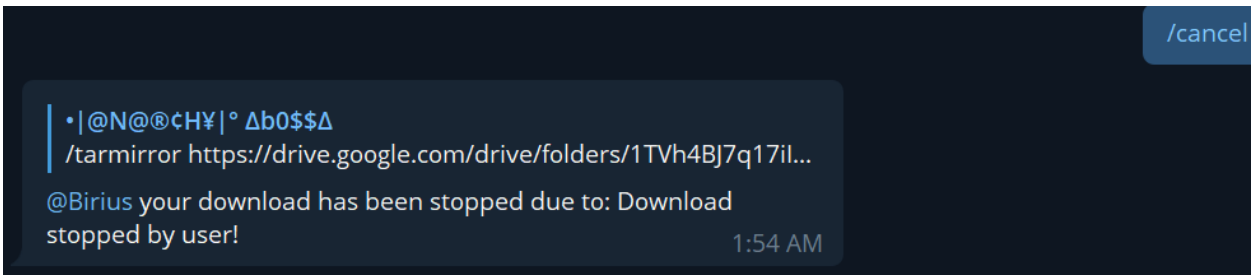



Рис.4.8 Команда /cancel


Рис.4.9 Команда /cancelall

Також присутній функціонал видалення об'єктів за наданною ссилкою у формі команди /del (рис. 4.10)

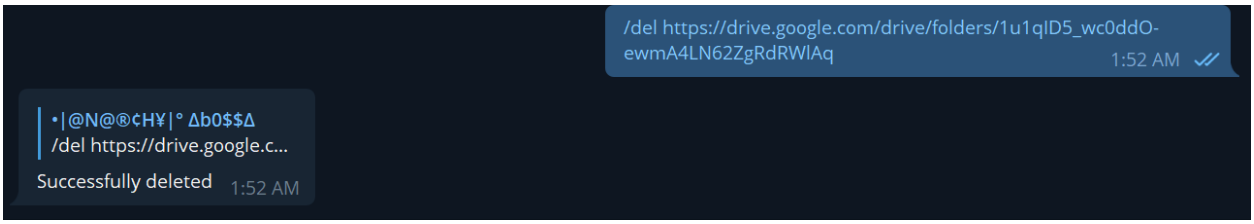



Рис.4.9 Команда /del

Якщо у користувачі з'явиться потреба у отриманні інформацію статуса всіх нинішних операцій то вівн це може зробити через команду /status. Ця команда покаже всі нинішні операції, а якщо на дантй момень часу ніяких операцій немає то бот відправить про це повідомленя в чат(рис. 4.10)


Рис.4.10 Команда /status

Для дослідження роботи та допомоги у виправлені помилок було імплементовано функцію логування, сам файл лога можливо отримати у самому чаті з ботом за допомогою команди /log/, але дана команда є

доступною тіль-ки для sudo юзерів, тобто власника бота та користувачів яким він надасть такі повноваження в конфігурації(рис. 4.11).


Рис.4.11 Команда /log

Також якщо користувач захоче додаткову інформації, він може за допомогою команди /status він може продивитися скільки часу вже пропрацював бот, скільки було завантажено та відправлено ним даних, а за допомогую команди /ping, юзер модно отримати інформацію про пінг від бота(рис. 4.12)


Рис.4.12 Команда /ping

Остання команда яка буде розглянута стане команда /restart яка перезапускає самого бота, доступна вона тільки судо користувачам та власнику(рис. 4.13).


Рис.4.13Команда /restart

## Висновки до розділу 4

В даному четвертому розділі було перевірено, продемонстровано та досліджено результати розробки цього бакалаврського проекту, а саме чат-бот для взаємодії між Telegram та сервісами хмарного збереження даних.

Сама перевірка відбулась вручну, через введення команд в чат бота та спостереження за результатами роботи самого бота в самому чаті та читання файлу лога який був отриманий в результаті роботи даного бота.

Також результати роботи було продемонстровано в вигляді скріншотів з самого бот, а саме скріншоти з введеними командами та результату дій бота з приводу цієї команди.

В процесі дослідження даного бота було доведено що його запланована і заявлена, в даному бакалаврському проекті, функціональність присутня і робоча доказами чого стали взаємодія між Telegram та хмарними системами зберігання даних та скріншоти на яких були показані результати цієї взаємодії.

# ВИСНОВОК

Даний бакалаврський проект мав на меті створення зручного та багатофункціонального чат-бота для взаємодії між мессенджеромTelegram та хмарних сервісами на зразок Google  Drive та Mega.

Сам месенджер Telegram був вибраний в результаті огляду самих популярних месенджерів. Він має одну з найбільших аудиторій в Україні, зручний інтерфейс, чудовий набір функцій та гарний і відносно легкий спосіб створення ботів з багатьма прикладами готових ботів, проаналізувавши які ми змогли створити більш функціональний та зручний бот.

Мовою програмування даного бота стала Python через велику кількість плагінів та бібліотек за допомогою яких можна більш ефективно та якісно написати код для нашого чат-бота. IDE для цього проекту стала PyCharm через її зручність та багатофункціональність.

Для багатофункціональності даного проекту були використана достатня кількість плагінів, до них входять такі плагіни як  Asyncio, Telebot, logging, та oogle auth oauthlib.

В результаті розробки цей бот отримав такі основні функції для взаємодії з сервісами хмарного збереження інформації: копіювання в різних форматах, завантаження файлів на сервер, видалення файлів та обрахування кількості файлів.

# СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


What is Telegram Bot? [Електронний ресурс]. – Режим доступу: https://www.opc-router.com/what-is-a-telegram-bot/

Хмарний сервіс зберігання даних [Електронний ресурс]. – Режим доступу: http://elartu.tntu.edu.ua/bitstream/lib/21688/2/X_VSNTK_2017v1_Shevchenko_M-Cloud_services_data_104-105.pdf

Quick Access: Building a Smart Experience for Google Drive [Електронний ресурс]. – Режим доступу: https://dl.acm.org/doi/abs/10.1145/3097983.3098048

Digital forensic analysis of cloud storage data in IDrive and Mega cloud drive [Електронний ресурс]. – Режим доступу: https://ieeexplore.ieee.org/abstract/document/7830159

The Development of Telegram BOT Through Short Story[Електронний ресурс].–Режим доступу: https://doi.org/10.2991/assehr.k.201021.049

What is Python? [Електронний ресурс]. – Режим доступу: https://www.python.org/doc/essays/blurb/\

Mastering PyCharm [Електронний ресурс].–Режим доступу:https://books.google.com.ua/books?hl=uk&lr=&id=MPh_CwAAQBAJ&oi=fnd&pg=PP1&dq=pycharm&ots=GWM--hUCe3&sig=R4Eb6S9LiI_GIaZ9FDRSkElF-nk&redir_esc=y#v=onepage&q=pycharm&f=false

Foundations of JSON Schema [Електронний ресурс]. – Режим доступу: https://doi.org/10.1145/2872427.2883029

Python reference manual [Електронний ресурс]. – Режим доступу: https://ir.cwi.nl/pub/5008

Aiogram’s documentation [Електронний ресурс]. – Режим доступу:https://docs.aiogram.dev/

Data Exchange Service using Google Drive API [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/profile/Pande-Made-Risky-Dinatha/publication/

310762064_Data_Exchange_Service_using_Google_Drive_API/links/5caa0e534585157bd3294542/Data-Exchange-Service-using-Google-Drive-API.pdf

Heroku [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Heroku

PostgreSQL: Introduction and Concepts [Електронний ресурс]. – Режимдоступу: https://momjian.us/main/writings/pgsql/other/bookfigs.pdf

The OAuth 2.0 authorization framework [Електронний ресурс]. – Режимдоступу: https://www.hjp.at/doc/rfc/rfc6749.html

SQL: The Complete Reference Електронний ресурс]. – Режимдоступу: http://englishonlineclub.com/pdf/SQL%20-%20The%20Complete%20Reference%20[EnglishOnlineClub.com].pdf

Додаток 1

Бот для взаємодії між мессенджером та іншим сайтом

**Структурна схема системи**

ІАЛЦ.467200.004 Д1

Аркушів 1

**Київ 2022 р**


| | | | | | ІАЛЦ.467200.004 Д1 | | |
|---|---|---|---|---|---|---|---|
| Зм. | Арк. | № докум. | Підпис | Дата | | | |
| Розробив | | Палеха Б.П. | | | **Бот для взаємодії між мессенджером та іншим сайтом** Структурна схема системи | Літ. | Аркуш 1 / Аркушів 1 |
| Перевірив | | Пономаренко А.М | | | | | |
| Реценз. | | | | | | КПІ ім. Ігоря Сікорського, ФІОТ, ІО-83 | |
| Н. Контр. | | Сімоненко В. П. | | | | | |
| Затвердив | | | | | | | |

Додаток 2

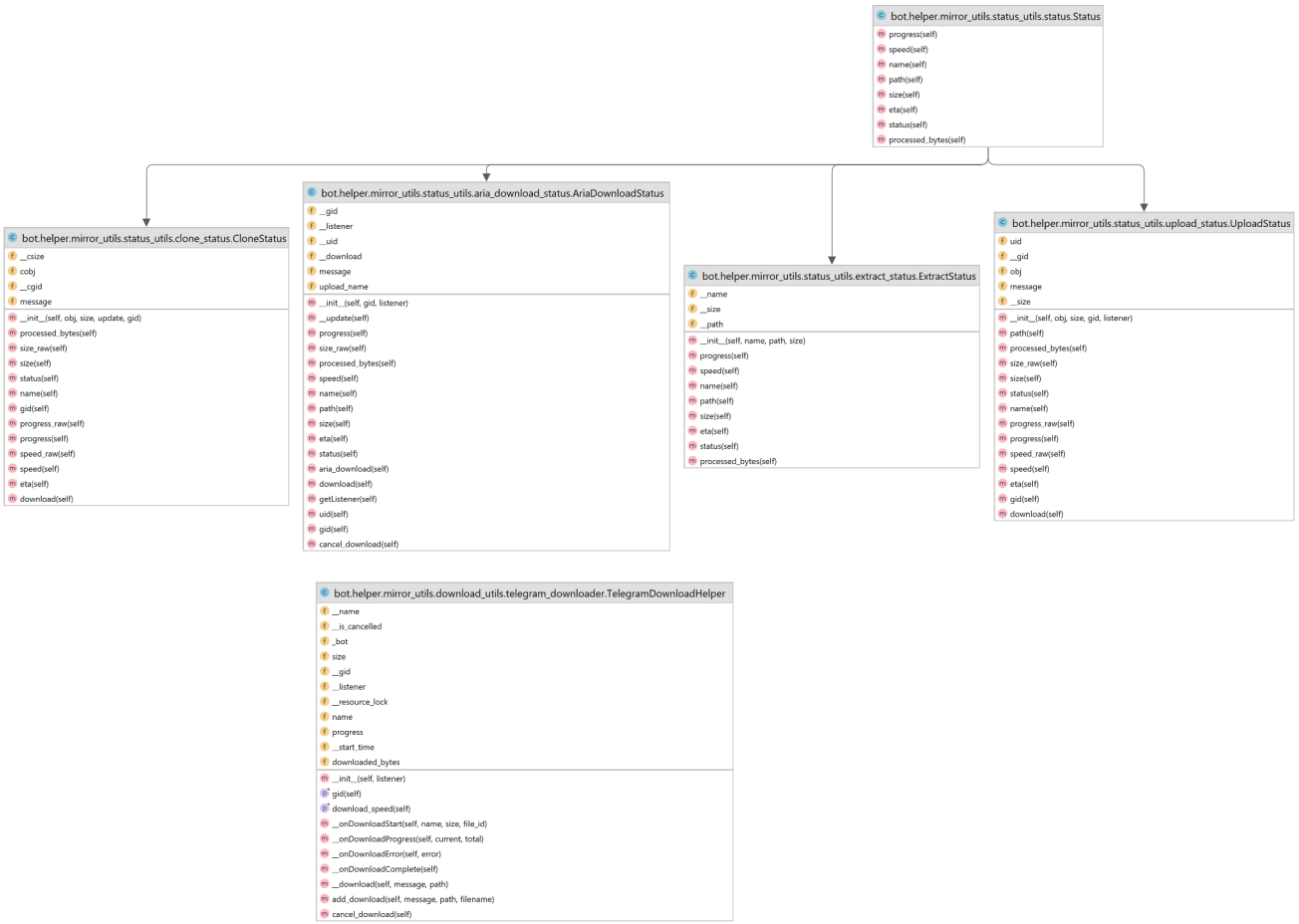
Бот для взаємодії між мессенджером та іншим сайтом

**Функціональна схема (діаграма класів)**

ІАЛЦ.467200.005 Д2

Аркушів 1

**Київ 2022 р**




| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467200.005 Д2 | |
| Зм. | Арк. | № докум. | Підпис | Дата | | |
| Розробив | | Палеха Б.П. | | | **Бот для взаємодії між мессенджером та іншим сайтом** Функціональна схема | Літ. / Аркуш 1 / Аркушів 1 |
| Перевірив | | Пономаренко А.М | | | | |
| Реценз. | | | | | | КПІ ім. Ігоря Сікорського, ФІОТ, ІО-83 |
| Н. Контр. | | Сімоненко В. П. | | | | |
| Затвердив | | | | | | |

# Додаток 3

Бот для взаємодії між мессенджером та іншим сайтом

## Алгоритм дій програмного забезпечення

ІАЛЦ.467200.006 Д3

Аркушів 1

**Київ 2022 р**


| | | | | | ІАЛЦ.467200.006 ДЗ | | |
|---|---|---|---|---|---|---|---|
| Зм. | Арк. | № докум. | Підпис | Дата | | | |
| Розробив | | Палеха Б.П. | | | **Бот для взаємодії між мессенджером та іншим сайтом** Алгоритм дій програмного забезпечення | Літ. | Аркуш 1 / Аркушів 1 |
| Перевірив | | Пономаренко А.М. | | | | | |
| Реценз. | | | | | | КПІ ім. Ігоря Сікорського, ФІОТ, ІО-83 | |
| Н. Контр. | | Сімоненко В. П. | | | | | |
| Затвердив | | | | | | | |

# Додаток 4

Бот для взаємодії між мессенджером та іншим сайтом

Текст програмного коду

ІАЛЦ.467200.007 Д4

Аркушів 8

**Київ 2022 р**

_main_.py

```
import shutil, psutil
import signal
import os
import asyncio

from pyrogram import idle
from sys import executable

from telegram import ParseMode
from telegram.ext import CommandHandler
from telegraph import Telegraph
from wserver import start_server_async
from bot import bot, app, dispatcher, updater, botStartTime,
IGNORE_PENDING_REQUESTS, IS_VPS, PORT, alive, web, OWNER_ID,
AUTHORIZED_CHATS, telegraph_token
from bot.helper.ext_utils import fs_utils
from bot.helper.telegram_helper.bot_commands import BotCommands
from bot.helper.telegram_helper.message_utils import *
from .helper.ext_utils.bot_utils import get_readable_file_size, get_readable_time
from .helper.telegram_helper.filters import CustomFilters
from bot.helper.telegram_helper import button_build
from .modules import authorize, list, cancel_mirror, mirror_status, mirror, clone,
watch,
shell, eval, torrent_search, delete, speedtest, count, leech_settings


def stats(update, context):
    currentTime = get_readable_time(time.time() - botStartTime)
    total, used, free = shutil.disk_usage('.')
    total = get_readable_file_size(total)
    used = get_readable_file_size(used)
    free = get_readable_file_size(free)
    sent = get_readable_file_size(psutil.net_io_counters().bytes_sent)
    recv = get_readable_file_size(psutil.net_io_counters().bytes_recv)
    cpuUsage = psutil.cpu_percent(interval=0.5)
    memory = psutil.virtual_memory().percent
    disk = psutil.disk_usage('/').percent
    stats = f'<b>Bot Uptime:</b> <code>{currentTime}</code>\n' \
            f'<b>Total Disk Space:</b> <code>{total}</code>\n' \
```

```
f'<b>Used:</b> <code>{used}</code> ' \
        f'<b>Free:</b> <code>{free}</code>\n\n' \
        f'<b>Upload:</b> <code>{sent}</code>\n' \
        f'<b>Download:</b> <code>{recv}</code>\n\n' \
        f'<b>CPU:</b> <code>{cpuUsage}%</code> ' \
        f'<b>RAM:</b> <code>{memory}%</code> ' \
        f'<b>DISK:</b> <code>{disk}%</code>'
    sendMessage(stats, context.bot, update)


def start(update, context):
    buttons = button_build.ButtonMaker()
    buttons.buildbutton("Repo", "https://github.com/SlamDevs/slam-mirrorbot")
    buttons.buildbutton("Channel", "https://t.me/SlamMirrorUpdates")
    reply_markup = InlineKeyboardMarkup(buttons.build_menu(2))
    if CustomFilters.authorized_user(update) or CustomFilters.authorized_chat(update):
        start_string = f'''
This bot can mirror all your links to Google Drive!
Type /{BotCommands.HelpCommand} to get a list of available commands
'''
        sendMarkup(start_string, context.bot, update, reply_markup)
    else:
        sendMarkup(
            'Oops! not a Authorized user.\nPlease deploy your own
<b>slam-mirrorbot</b>.',
            context.bot,
            update,
            reply_markup,
        )


def restart(update, context):
    restart_message = sendMessage("Restarting, Please wait!", context.bot, update)
    # Save restart message object in order to reply to it after restarting
    with open(".restartmsg", "w") as f:
        f.truncate(0)
        f.write(f"{restart_message.chat.id}\n{restart_message.message_id}\n")
    fs_utils.clean_all()
    alive.terminate()
    web.terminate()
    os.execl(executable, executable, "-m", "bot")
```

```
def ping(update, context):
    start_time = int(round(time.time() * 1000))
    reply = sendMessage("Starting Ping", context.bot, update)
    end_time = int(round(time.time() * 1000))
    editMessage(f'{end_time - start_time} ms', reply)


def log(update, context):
    sendLogFile(context.bot, update)

def main():
    fs_utils.start_cleanup()
    if IS_VPS:
        asyncio.get_event_loop().run_until_complete(start_server_async(PORT))
    # Check if the bot is restarting
    if os.path.isfile(".restartmsg"):
        with open(".restartmsg") as f:
            chat_id, msg_id = map(int, f)
        bot.edit_message_text("Restarted successfully!", chat_id, msg_id)
        os.remove(".restartmsg")
    elif OWNER_ID:
        try:
            text = "<b>Bot Restarted!</b>"
            bot.sendMessage(chat_id=OWNER_ID, text=text,
parse_mode=ParseMode.HTML)
            if AUTHORIZED_CHATS:
                for i in AUTHORIZED_CHATS:
                    bot.sendMessage(chat_id=i, text=text, parse_mode=ParseMode.HTML)
        except Exception as e:
            LOGGER.warning(e)
    # bot.set_my_commands(botcmds)
    start_handler = CommandHandler(BotCommands.StartCommand, start,
run_async=True)
    ping_handler = CommandHandler(BotCommands.PingCommand, ping,
                                  filters=CustomFilters.authorized_chat |
CustomFilters.authorized_user, run_async=True)
    restart_handler = CommandHandler(BotCommands.RestartCommand, restart,
                                     filters=CustomFilters.owner_filter | CustomFilters.sudo_user,
run_async=True)
```

```
  help_handler = CommandHandler(BotCommands.HelpCommand,
                                bot_help, filters=CustomFilters.authorized_chat |



 CustomFilters.authorized_user, run_async=True)
    stats_handler = CommandHandler(BotCommands.StatsCommand,
                                   stats, filters=CustomFilters.authorized_chat |
CustomFilters.authorized_user, run_async=True)
    log_handler = CommandHandler(BotCommands.LogCommand, log,
filters=CustomFilters.owner_filter | CustomFilters.sudo_user, run_async=True)
    dispatcher.add_handler(start_handler)
    dispatcher.add_handler(ping_handler)
    dispatcher.add_handler(restart_handler)
    dispatcher.add_handler(help_handler)
    dispatcher.add_handler(stats_handler)
    dispatcher.add_handler(log_handler)
    updater.start_polling(drop_pending_updates=IGNORE_PENDING_REQUESTS)
    LOGGER.info("Bot Started!")
    signal.signal(signal.SIGINT, fs_utils.exit_clean_up)

app.start()
main()
idle()
```

_init_.py

```
    import logging
    import os
    import threading
    import time
    import random
      import string
      import subprocess
     import requests


import telegram.ext as tg
```

```
        from dotenv import load_dotenv
from pyrogram import Client
from telegraph import Telegraph

import psycopg2
from psycopg2 import Error



import socket
import faulthandler
faulthandler.enable()

socket.setdefaulttimeout(600)

botStartTime = time.time()
if os.path.exists('log.txt'):
    with open('log.txt', 'r+') as f:
        f.truncate(0)

logging.basicConfig(format='%(asctime)s - %(name)s - %(levelname)s - %
(message)s',
                    handlers=[logging.FileHandler('log.txt'), logging.StreamHandler()],
                    level=logging.INFO)

LOGGER = logging.getLogger(__name__)

CONFIG_FILE_URL = os.environ.get('CONFIG_FILE_URL', None)
if CONFIG_FILE_URL is not None:
    res = requests.get(CONFIG_FILE_URL)
    if res.status_code == 200:
        with open('config.env', 'wb+') as f:
            f.write(res.content)
            f.close()
    else:
        logging.error(res.status_code)

load_dotenv('config.env')
```

```
    SERVER_PORT = os.environ.get('SERVER_PORT', None)
    PORT = os.environ.get('PORT', SERVER_PORT)
    web = subprocess.Popen([f"gunicorn wserver:start_server --bind 0.0.0.0:{PORT} --
worker-class aiohttp.GunicornWebWorker"], shell=True)
    time.sleep(1)
    alive = subprocess.Popen(["python3", "alive.py"])
    subprocess.run(["mkdir", "-p", "qBittorrent/config"])
    subprocess.run(["cp", "qBittorrent.conf", "qBittorrent/config/qBittorrent.conf"])
    subprocess.run(["qbittorrent-nox", "-d", "--profile=."])
    Interval = []
    DRIVES_NAMES = []
    DRIVES_IDS = []




    INDEX_URLS = []
    def getConfig(name: str):
        return os.environ[name]

    def mktable():
        try:
            conn = psycopg2.connect(DB_URI)
            cur = conn.cursor()
            sql = "CREATE TABLE users (uid bigint, sudo boolean DEFAULT FALSE);"
            cur.execute(sql)
            conn.commit()
            logging.info("Created!")
        except Error as e:
            logging.error(e)
            exit(1)

    try:
        if bool(getConfig('_____REMOVE_THIS_LINE_____')):
            logging.error('The README.md file there to be read! Exiting now!')
            exit()
    except KeyError:
        pass

    aria2 = aria2p.API(
```

```
 aria2p.Client(
        host="http://localhost",
        port=6800,
        secret="",
    )
)


def get_client() -> qba.TorrentsAPIMixIn:
    qb_client = qba.Client(host="localhost", port=8090, username="admin",
password="adminadmin")
    try:
        qb_client.auth_log_in()
        #qb_client.application.set_preferences({"disk_cache":64,
 "incomplete_files_ext":True, "max_connec":3000, "max_connec_per_torrent":300,




"async_io_threads":8, "preallocate_all":True, "upnp":True, "dl_limit":-1, "up_limit":-
1,
 "dht":True, "pex":True, "lsd":True, "encryption":0, "queueing_enabled":True,




"max_active_downloads":15, "max_active_torrents":50,
"dont_count_slow_torrents":True,


 "bittorrent_protocol":0, "recheck_completed_torrents":True,
"enable_multi_connections_from_same_ip":True,
"slow_torrent_dl_rate_threshold":100,"slow_torrent_inactive_timer":600})
        return qb_client
    except qba.LoginFailed as e:
        logging.error(str(e))
        return None
```

generate_drive_token.py

```
import pickle
import os
from google_auth_oauthlib.flow import InstalledAppFlow
from google.auth.transport.requests import Request

credentials = None
__G_DRIVE_TOKEN_FILE = "token.pickle"
__OAUTH_SCOPE = ["https://www.googleapis.com/auth/drive"]
if os.path.exists(__G_DRIVE_TOKEN_FILE):
    with open(__G_DRIVE_TOKEN_FILE, 'rb') as f:
        credentials = pickle.load(f)
        if (
            (credentials is None or not credentials.valid)
            and credentials
            and credentials.expired
            and credentials.refresh_token
        ):
```