**2. РОЗРОБКА ПРОГРАМИ**

**2.1 Аналіз**

Для реалізації потрібно визначитися з середовищем та мовою програмуванням. Одним із варіантів буде мова програмування Python та середовище – PyCharm. Сам код повинен містити токен, для активації бота, та бібліотеку, де знаходяться інструменти роботи.

Програма повинна виконувати такі дії:

* виконувати команди;
* видавати ролі;
* відтворювати музику;
* виводити інформацію про погоду;

Через команди, користувач може звертатися до бота та виконувати певні дії. Наприклад: команда help буде виводити список команд, ping – виводить швидкість оклику коду, join – підключатися до голосового каналу та інші.

Ролі повинні видаватися користувачу, якщо він виконав певну дію – встановлення реакції (емодзі) на конкрентному повідомленні. Для кожної реакції буде своя роль.

Для відтворення музики бот повинен отримати можливість підключення до голосового каналу, скачувати потрібну музику зі сайту та налагоджувати її без дефектів.

Інформацію про погоду користувач повинен вказувати назву міста, а програма – обробити та виводити інформацію. Для цього буде використаний сайт OpenWeather.

**2.2 Програмування**

В цьому підрозділі описано створення бота, основні пакети, команди та виведення інформації.

**2.2.1 Створення бота**

Для створення бота потрібно мати профіль в Discord. Потім перейти до сайту Discord Developer Portal та створити новий застосунок, який буде мати назву «Smart bot» (рисунок 2.1).

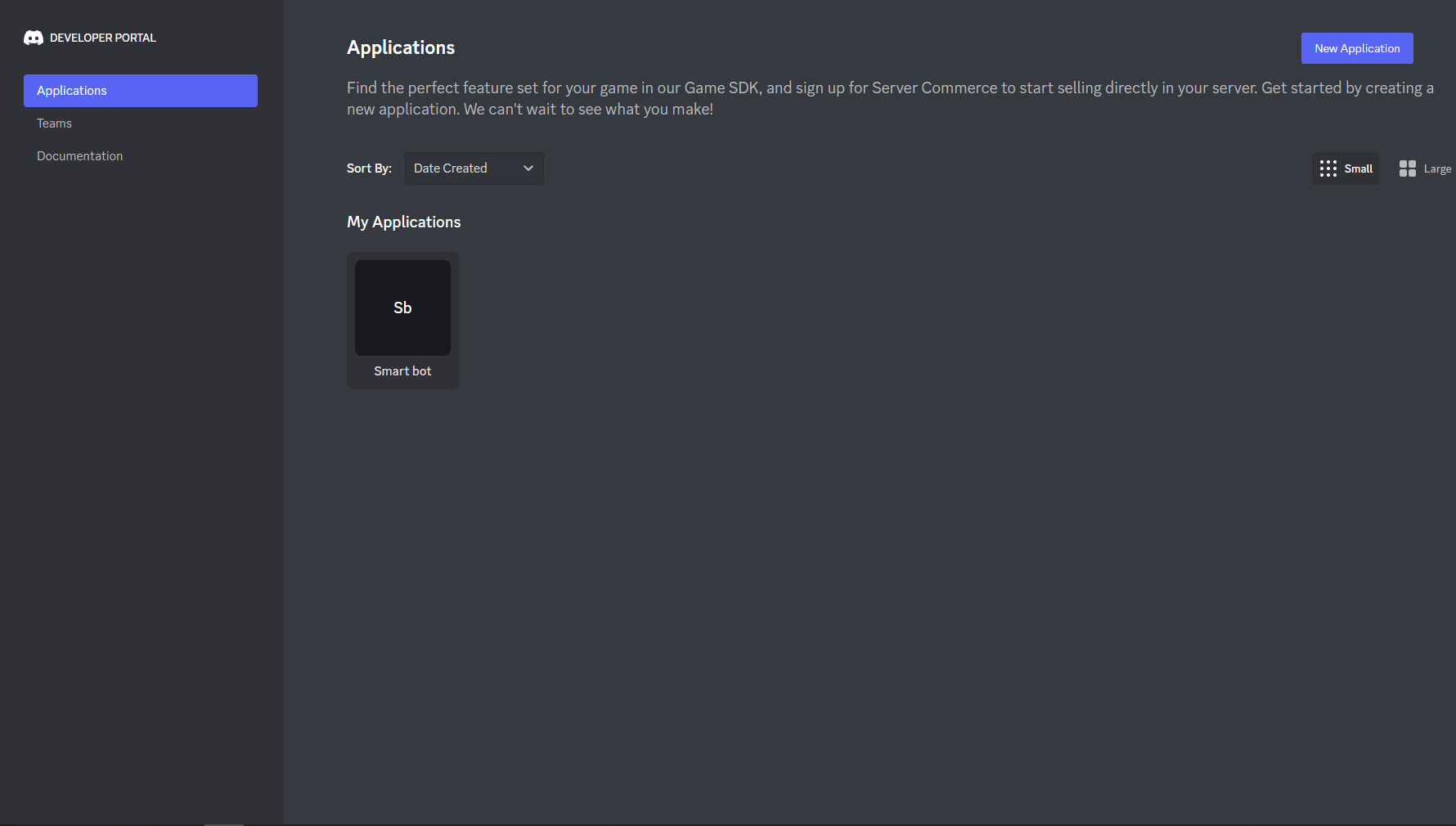


Рисунок 2.1 – Створений застосунок «Smart bot»

Наступним кроком буде створення самого бота в цьому застосунку, використовуючи пункт «Bot» (рисунок 2.2). В ньому створюється профіль, який має ім’я, зображення та особливий токен, який буде використовуватися при його програмуванні, тому його потрібно ховати від інших користувачів.

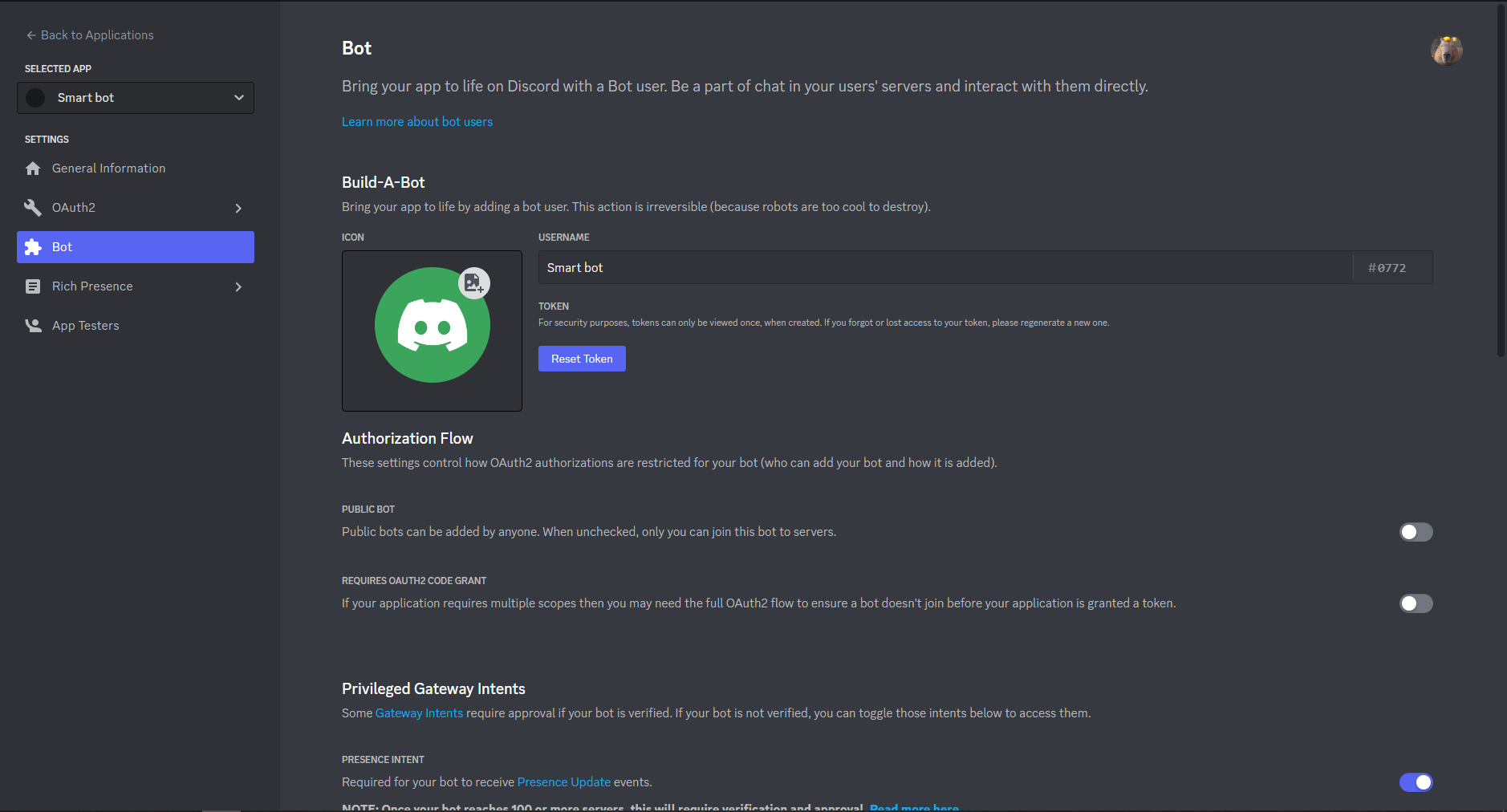


Рисунок 2.2 – Пункт «Bot»

Далі потрібно перейти в пункт «OAuth2», де створюється посилання для запрошення бота на свій сервер (рисунок 2.3). Тут потрібно вказати в можливості пункт «bot» та додати йому права. Безпечніше буде дати права до не модераторских дій (считувати повідомлення, відправляти повідомлення, виконувати підключення до голосового каналу та інші), якщо він буде працювати в сервері з багатьма людьми, але сервер для тестування буде пустим, тому даємо права адміністратора.

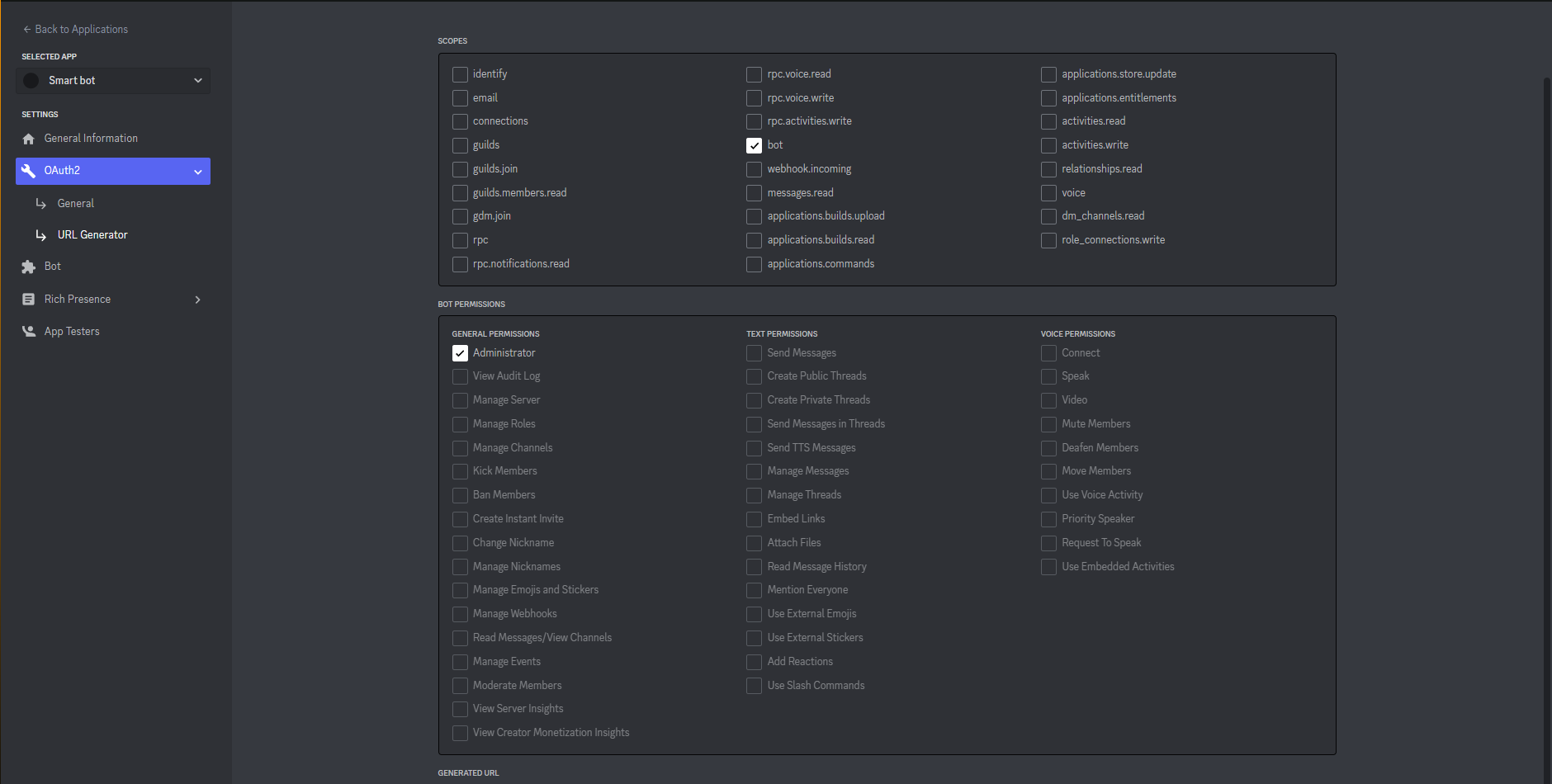


Рисунок 2.3 – Пункт «OAuth2»

На створене посилання потрібно перейти та додати профіль до сервера. На рисунку 2.4 він буде доданий до серверу «Test server». Після цього, бот з’явиться в нашому сервері.

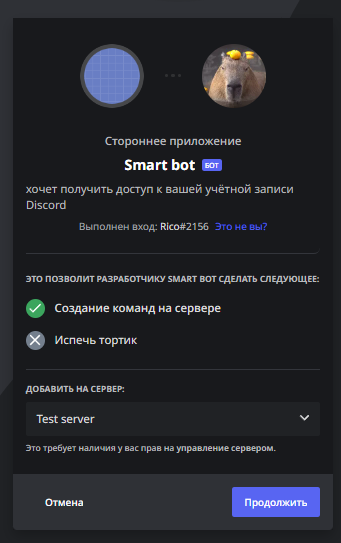


Рисунок 2.4 – Додавання бота до серверу

**2.2.2 Пакети для програмування та включення бота**

В середовищі програмування PyCharm використувується Python ver. 3.9.13 та такі пакети:

* discord.py ver. 1.7.3;
* youtube-dl ver. 2021.12.17;
* ffmpeg ver. 1.4;

discord.py – основний пакет для програми. За допомогою нього виконується програмування бота через його токен.

youtube-dl використовується для скачування відео з сайту youtube.com з потрібними параметрами. ffmpeg – пакет, який дозволяє боту відтворити медіа файл.

Для включення бота в коді потрібно прописати такий код:

intents = discord.Intents.default()

intents.members = True

bot = commands.Bot(command\_prefix=”!”, intents=intents)

bot.run(TOKEN)

bot.run(“Токен”) виконує запуск бота через токен. До цієї команди виконується конфігурація для того, щоб програма бачила користувачів серверу та дозвіл використовування префіксу для виконання команд. При запуску коду, бот буде відмічений «в мережі», як інші користувачі (рисунок 2.5).

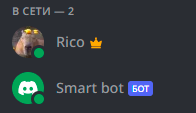


Рисунок 2.5 – Статус «в мережі»

**2.2.3 Отримання ролі та команди**

Для отримання ролі, користувач повинен виконати конкретну подію. Бот може відстежувати дії кожних користувачів та виконувати для них код. В коді вони позначаються, як @bot.event. В даному випадку користувач повинен поставити реакцію на повідомлення для отримання ролі. На рисунку 2.6 наведено повідомлення, де потрібно встановити реакцію.

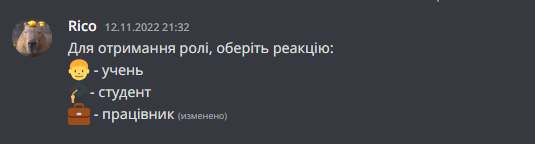


Рисунок 2.6 – Повідомлення для встановлення реакції

Для відстеження події додавання реакції використовується метод on\_raw\_reaction\_add(payload). Після цього бот відстежує повідомлення та збирає інформацію про користувача, який додав емодзі: якщо він додав один із трьох, які написані в повідомлені, йому буде додано роль, якщо він додав не із списку – роль не буде видано.

Для зворотної дії використовується метод on\_raw\_reaction\_remove(payload). Процес виконання коду такий самий, як і у додавання емодзі, але навпаки: якщо користувач видаляє свою реакцію – роль з нього видаляється.

Команди потрібні для керуванням ботом через повідомлення користувачів, які починаються з префіксів. В коді вони позначаються, як @bot.command(). Для цього бота використовується префікс «!». Програма має такий список команд:

* help;
* ping;
* join;
* leave;
* play «url»;
* pause;
* resume;
* sd;
* weather «місто».

!help – виводить список доступних команд користувачу (рисунок 2.7).

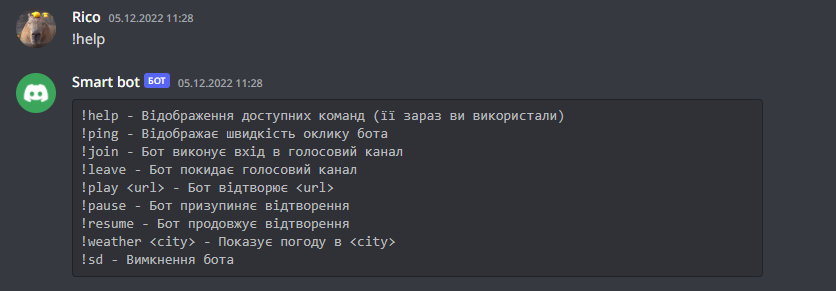


Рисунок 2.7 – Команда help

!ping – виводить швидкість оклика бота на команди (рисунок 2.8).

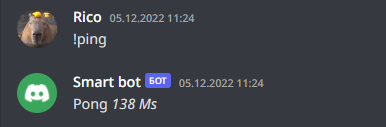


Рисунок 2.8 – Команда ping

!join – підключається до голосового каналу. Ця команда потрібна для подальшого відтворення музики. Перед виконанням команди, користувач повинен бути в цьому каналі.

!play «url» - відтворює посилання до сайта Youtube, який надіслав користувач, в голосовому каналі. Якщо це виконується вперше – програма повинна завантажити відео та сконвертувати на «найкаще аудіо». Процес відтворення виводиться в консоль (рисунок 2.9). Бот при цьому повинен бути в голосовому каналі.

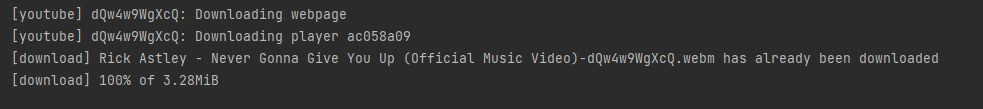


Рисунок 2.9 – Процес завантаження та відтворення музики

!pause, !resume та !leave – призупиняє, відновлює роботу та покидає голосовий канал бот.

!sd – альтернативний спосіб вимкнення бота.

!weather «місто» - виводить інформацію про погоду в місті. Для цього бот виконує вхід до сайту OpenWeather, отримає інформацію про погоду та виводить користувачу в читабельній формі. Перед цим потрібно зареєструватися для отримання API ключа. Він потрібен для створення посилання боту, де знаходиться значення, які потрібні розшифрувати. Прикладом є рисунок 2.10, де бот виводить погоду в Києві та Львові. Місто обов’язково повинно бути написано на англійській мові.



Рисунок 2.10 – Виконання команди weather

**ВИСНОВКИ**

В даному курсовому проекті було розроблено мережеве програмне забезпечення «Бот для месенджера Discord». Дана програма автоматизує додавання ролей користувачу та виконує команди відтворення музики, перегляд погоди в місті, виведення список команд та швидкість оклику. Програма була створена на мові Python в середовищі PyCharm. Для роботи коду використовувалися основний пакет Python, пакет для роботи бота в месенджері discord.py та пакети встановлення і відтворення музики youtube-dl та ffmpeg.

В коді програми бот може реагувати як на події (@bot.event), так і на створені команди (@bot.command()). Через подію встановлення та видалення реакції користувач може отримати та видалити роль. Через команди – інформацію про погоду, швидкість оклику бота та відтворити музику за посиланням. Програма звертається до сайтів Youtube (для скачування аудіо) та OpenWeather (для отримання інформації по погоду).

**ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. Боты для дискорда: рейтинг Топ 10 лучших. URL: <https://highload.today/boty-dlya-discord> (дата звернення: 12.12.2022).
2. Creating a Bot Account. URL: <https://discordpy.readthedocs.io/en/stable/discord.html> (дата звернення: 12.12.2022).
3. API Reference. URL: <https://discordpy.readthedocs.io/en/stable/api.html> (дата звернення: 12.12.2022).

# Додатки

Додаток А

Лістинг файлу main.py

import discord  
import requests  
import Config  
import Music  
from discord.ext import commands  
  
intents = discord.Intents.default()  
intents.members = True  
bot = commands.Bot(command\_prefix=Config.Prefix, intents=intents)  
  
  
@bot.event # Повідомляє в консоль про працездатність бота  
async def on\_ready():  
 print('{0.user} зараз працює'.format(bot))  
  
  
@bot.command() # Команда ping - показує час оклику бота  
async def ping(ctx):  
 await ctx.send(f"Pong \*{round(bot.latency \* 1000)} Ms\*")  
  
  
bot.remove\_command('help')  
  
  
@bot.command() # Команда help - виводить список команд бота  
async def help(ctx):  
 await ctx.send(Config.help)  
  
  
@bot.command() # Команда join - приєднується до голосового каналу де знаходиться користувач  
async def join(ctx):  
 channel = ctx.message.author.voice.channel  
 await channel.connect()  
  
  
@bot.command() # Команда leave - виходить з голосового каналу  
async def leave(ctx):  
 ctx.voice\_client.stop()  
 await ctx.voice\_client.disconnect()  
  
  
@bot.command() # Команда play <url> - (виконується коли бот знаходиться в голосовому каналі) відтворює звуковий youtube url запит в голосовому каналі  
async def play(ctx, \*, url):  
 ctx.voice\_client.stop()  
 async with ctx.typing():  
 player = await Music.YTDLSource.from\_url(url, loop=bot.loop)  
 ctx.voice\_client.play(player, after=lambda e: print(f'Player error: {e}') if e else None)  
 await ctx.send(f'Now playing: {player.title}')  
  
  
@bot.command() # Команда pause - призупиняє відтворенню url запита  
async def pause(ctx):  
 ctx.voice\_client.pause()  
  
  
@bot.command() # Команда pause - відновляє відтворення url запита  
async def resume(ctx):  
 ctx.voice\_client.resume()  
  
  
@bot.command() # Команда weather <city> - виводить інформацію про погоду в <city> (місто повинно бути написано на англ мові)  
async def weather(ctx, \*, City: str):  
 def kelvin\_to\_celsius(kelvin):  
 celsius = kelvin - 273.15  
 return celsius  
  
 url = Config.weather\_url + "appid=" + Config.API\_key + "&q=" + City # Причина, чому потрібно писати місто на англ мові - він створює url запить для отримання інформації  
 response = requests.get(url).json()  
 weather = response['weather'][0]['main']  
 temp\_kelvin = response['main']['temp']  
 temp\_celsius = kelvin\_to\_celsius(temp\_kelvin)  
 feels\_like\_kelvin = response['main']['feels\_like']  
 feels\_like\_celsius = kelvin\_to\_celsius(feels\_like\_kelvin)  
 wind\_speed = response['wind']['speed']  
 humidity = response['main']['humidity']  
 cloud = response['clouds']['all']  
 await ctx.send(f"""  
 ```  
Погода: {weather}  
Температура: {temp\_celsius:.2f}°C  
За відчуттям: {feels\_like\_celsius:.2f}°C  
Вологість: {humidity}%  
Хмарність: {cloud}%  
Швидкість повітря: {wind\_speed} м/c ``` """)  
 print(url)  
  
@bot.command() # Команда sd - вимкнення бота через дискорд (якщо не хочеться вимикати через консоль )  
async def sd(ctx):  
 await ctx.send("Виконується викнення бота")  
 await bot.logout()  
  
  
# Події отримання ролі (список знаходиться в файлі Config, клас Roles())  
@bot.event # Виконання події при якій якщо користувач додає "реакцію" (або іншим словом емодзі) під конкрентним повідомленням, він отримає роль, яка привязана до цієї реакції  
async def on\_raw\_reaction\_add(payload):  
 if payload.message\_id == Config.Post\_ID:  
 guild\_id = payload.guild\_id  
 guild = discord.utils.find(lambda g: g.id == guild\_id, bot.guilds)  
 role = Config.Roles(payload, guild).Role()  
 if role is not None:  
 member = discord.utils.find(lambda m: m.id == payload.user\_id, guild.members)  
 if member is not None:  
 await payload.member.add\_roles(role)  
 print("Додано роль {0} для {1}.".format(role, member))  
 else:  
 print("Користувача не знайдено.")  
 else:  
 print("Роль не знайдено.")  
  
  
@bot.event # Виконання події при якій якщо користувач видаляє "реакцію" (або іншим словом емодзі) під конкрентним повідомленням, з нього видаляється роль, яка привязана до цієї реакції  
async def on\_raw\_reaction\_remove(payload):  
 if payload.message\_id == Config.Post\_ID:  
 guild\_id = payload.guild\_id  
 guild = discord.utils.find(lambda g: g.id == guild\_id, bot.guilds)  
 role = Config.Roles(payload, guild).Role()  
 if role is not None:  
 member = discord.utils.find(lambda m: m.id == payload.user\_id, guild.members)  
 if member is not None:  
 await member.remove\_roles(role)  
 print("Віднято роль {0} у {1}.".format(role, member))  
 else:  
 print("Користувача не найдено.")  
 else:  
 print("Роль не знайдено.")  
  
  
bot.run(Config.TOKEN) # Запуск бота

Додаток Б

Лістинг файлу Config.py

# Тут знаходяться значення, які можна змінювати для іншого бота  
import discord  
  
TOKEN = 'TOKEN'# Спеціальний ключ для запуску бота в сервері  
Prefix = '!' # Префікс команд  
Post\_ID = 1041072950184530070 # IP повідомлення, де потрібно вставити "реакцію" (подія отримання ролей)  
FFMPEG\_OPTIONS = {'options': '-vn'} # Налагодження для відтворення музики (звуків) (команда play)  
YDL\_OPTIONS = {'format': 'bestaudio'} # Налагодження для встановлення музики (звуків) (команда play)  
API\_key = "Key" # Спеціальний ключ користувача OpenWeather (команда weather)  
weather\_url = "http://api.openweathermap.org/data/2.5/weather?" # Посилання OpenWeather для створення посилання міста (команда weather)  
help = """   
 ```  
!help - Відображення доступних команд (її зараз ви використали)  
!ping - Відображає швидкість оклику бота  
!join - Бот виконує вхід в голосовий канал  
!leave - Бот покидає голосовий канал   
!play <url> - Бот відтворює <url>   
!pause - Бот призупиняє відтворення   
!resume - Бот продовжує відтворення   
!weather <city> - Показує погоду в <city>  
!sd - Вимкнення бота ``` """ # Список, який виводить команда help  
  
  
class Roles(): # Список емодзі та прив'язаних до них ролі (подія отримання ролей)  
 def \_\_init\_\_(self, payload, guild):  
 self.payload = payload  
 self.guild = guild  
  
 def Role(self):  
 if self.payload.emoji.name == '👦':  
 role = discord.utils.get(self.guild.roles, name='Учень')  
 elif self.payload.emoji.name == '🎓':  
 role = discord.utils.get(self.guild.roles, name='Студент')  
 elif self.payload.emoji.name == '💼':  
 role = discord.utils.get(self.guild.roles, name='Працівник')  
 else:  
 role = None  
 return role

Додаток В

Лістинг файлу Music.py

#Файл потрібен для скачування аудіо файлу з сайту Youtube

import discord  
import asyncio  
import youtube\_dl  
import Config  
  
ytdl = youtube\_dl.YoutubeDL(Config.YDL\_OPTIONS)  
  
  
class YTDLSource(discord.PCMVolumeTransformer):  
 def \_\_init\_\_(self, source, \*, data, volume=0.5):  
 super().\_\_init\_\_(source, volume)  
 self.data = data  
 self.title = data.get('title')  
 self.url = data.get('url')  
  
 @classmethod  
 async def from\_url(cls, url, \*, loop=None, stream=False):  
 loop = loop or asyncio.get\_event\_loop()  
 data = await loop.run\_in\_executor(None, lambda: ytdl.extract\_info(url, download=not stream))  
  
 if 'entries' in data:  
 data = data['entries'][0]  
  
 filename = data['url'] if stream else ytdl.prepare\_filename(data)  
 return cls(discord.FFmpegPCMAudio(filename, \*\*Config.FFMPEG\_OPTIONS), data=data)