УДК 004

Создание бота управления в мессенджере Discord

Эрдман Александр Алексеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В статье рассмотрен бот написанный на языке программирования Python, с помощью которого можно управлять некоторым функционалом сервера в мессенджере Discord.
Ключевые слова: python, discord бот, создание бота

Creating a management bot in the Discord messenger

Erdman Alexander Alekseevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The article discusses a bot written in the Python programming language, with which you can control some server functionality in the Discord messenger.
Keywords: python, discord bot, creating a bot

1 Введение

1.1 Актуальность

О важности и актуальности ботов для мессенджеров уже написано довольно много статей, начиная от обычного эхо бота и заканчивая ботом с продвинутым искусственным интеллектом. В данной статье будет рассмотрен бот в среде мессенджера Discord. Часто на практике бывают моменты, когда владельцы крупных Discord серверов сталкиваются с проблемой непосредственного управления сервером. Одной из таких проблем является распределение ролей на сервере. С одной стороны ничего сложного в этом нет, но когда на сервере находятся от 100 человек и более – вот тут и возникают трудности. Чтобы решить данную проблему можно нанять дополнительных администраторов сервера, которые будут заниматься распределением ролей, но можно вместо многочисленных администраторов сделать одного единственного бота, который будет управлять ролями. В статье будет рассмотрен второй вариант решения проблемы – создание бота управления.

1.2 Обзор исследований

Н.М Молчанов рассмотрел создание ботов при помощи Discord API [1]. М.М Москвин и Д.А. Литвинов рассмотрели создание Discord бота на Python

[2]. Е.А. Прокопенко рассмотрел метод создания чат-бота для мессенджера Discord [3]. С.А Кочкарев и И.Р. Жуков рассмотрели общий принцип создания онлайн чат-ботов [4]. Ш.Р. Миннивалиев, А.Н. Карамышева и Е.В Абросимова изучили главные аспекты ботов управления [5]. М.С Мелитонян и Е.В. Фешина показали явные преимущества мессенджера Discord среди конкурентов [6].

1.3 Цель исследования

Целью исследования является создание бота управления, написанного на языке программирования Python, на сервере мессенджера Discord для автоматизации работы администраторов и как следствие повышение эффективности самого сервера.

2 Материалы и методы

Для создания бота будет использоваться язык программирования Python, среда программирования PyCharm, необходимый модуль для создания ботов discord.py, непосредственно сам мессенджер Discord, а также любой браузер.

3 Результаты и обсуждения

Для начала работы необходимо создать сервер Discord. Для создания бота необходимо первым делом установить модуль для работы с Discord «discord.py» После команды «pip install discord.py» нужно проверить его работоспособность, для этого нужно написать простого бота, который будет возвращать все сообщения пользователя. Но сперва необходимо добавить бота на созданный сервер. Для этого нужно перейти в браузере по адресу «http://discordappcom/developers». На сайте нужно создать приложение, которое будет нашим ботом, при создании необходимо указать имя приложения (рис. 1). После создания приложения, нужно перейти в раздел «Bot» и скопировать токен нашего бота, который необходимо вставить в код (рис. 2). После всех действий тестовый бот готов, но для начала работы необходимо добавить бота на сервер. Для этого на сайте нужно перейти в раздел «OAuth2», выставить соответствующие параметры для бота и вставить появившуюся ссылку в адресную строку, после чего будет возможность добавить бота на сервер (рис. 3). После того, как бот добавлен, можно провести тест (рис. 4).

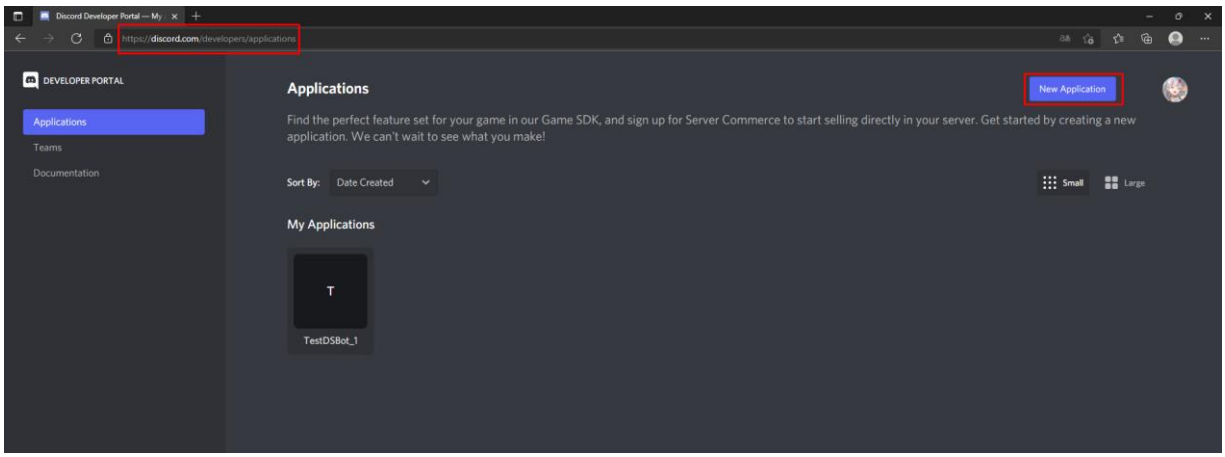

Рис.1. Создание приложения

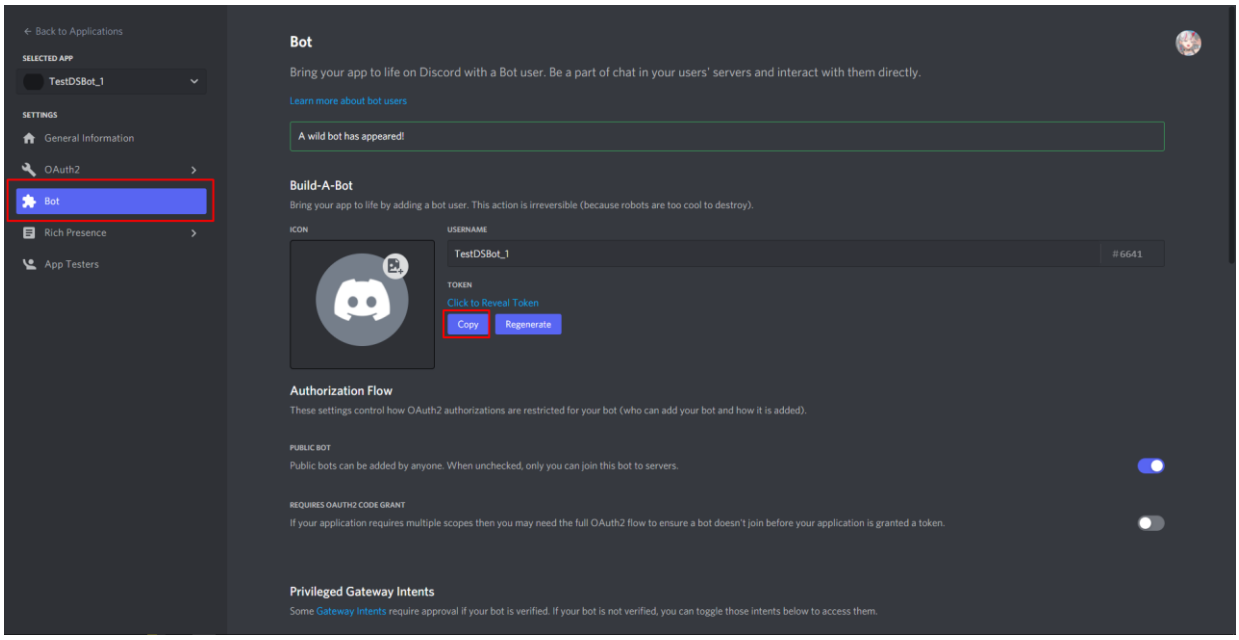

Рис. 2. Получение токена бота

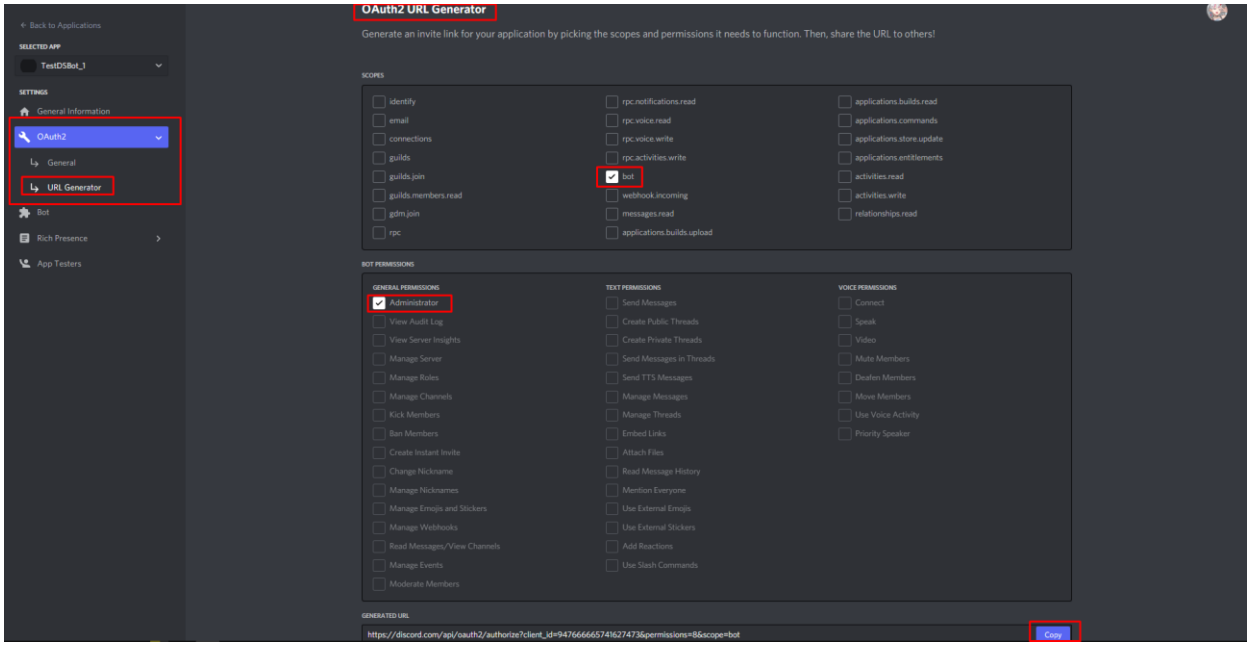

Рис. 3. Выставление параметров бота и получение ссылки на подключение

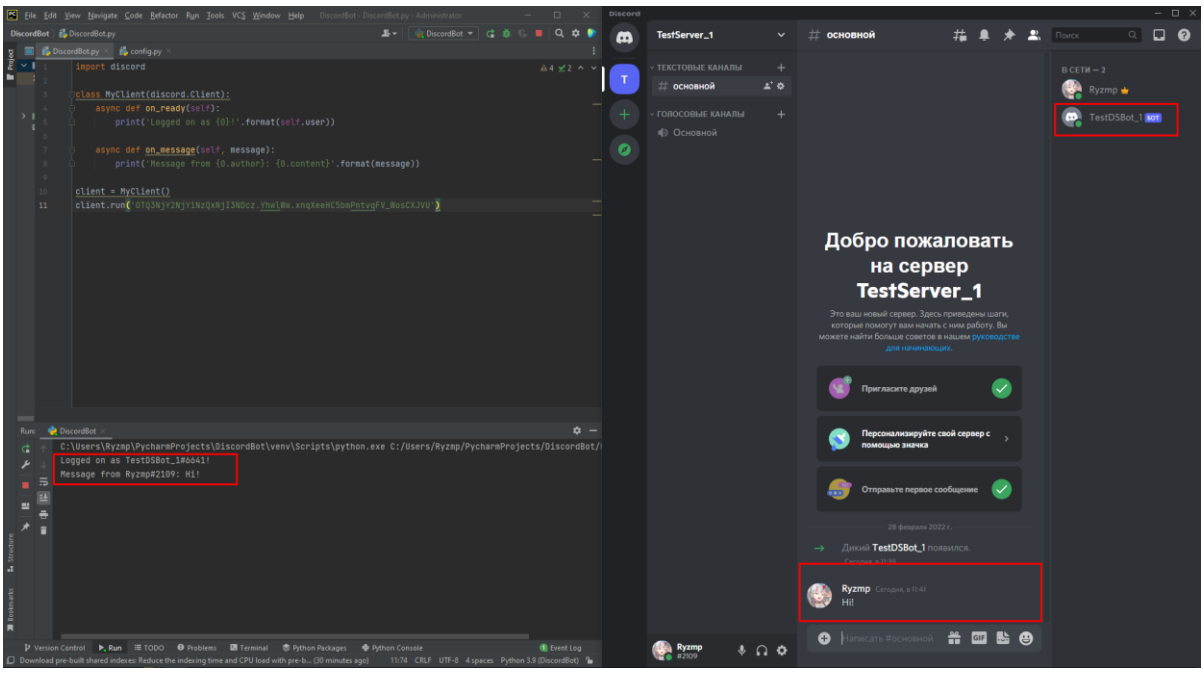

Рис. 4. Тест работоспособности бота

После успешного тестирования, можно переходить к написанию бота управления. Бот должен предоставлять роли для выбора, а также выдавать их пользователям по их нажатию. Сперва на сервере нужно создать чат, в котором непосредственно будет сообщения с перечнем ролей и их непосредственная возможность выбора (рис. 5). Также сразу нужно создать роли, в данном случае их будет 4. Для корректной работы бота необходимо в списке ролей поместить его вверх списка. Это обусловлено тем, что бот не может давать роли тем, кто стоит выше его по списку. Далее создадим два файла .py – DiscordBot.py и config.py. В первом файле будет находиться непосредственно код бота, а второй файл будет с настройками. Для удобства сперва пропишем код в config.py. Здесь нужно добавить id ролей, а также id поста с ними. Id можно скопировать в настройках сервера, а пост в чате. Также здесь нужно вставить скопированный ранее токен бота. К id ролей необходимо добавить реакции, через которые непосредственно будет осуществляться выдача ролей. Готовый config.py выглядит следующим образом (рис. 6).

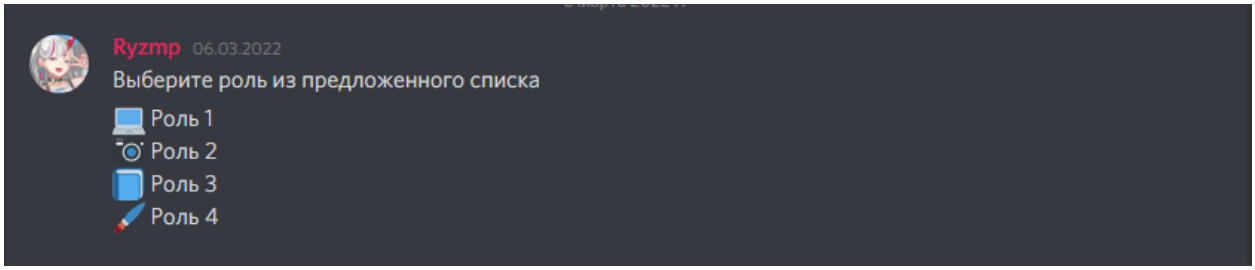

Рис. 5. Создание поста с ролями

DiscordBot.py config.py

```
TOKEN = 'OTQ3NjY2NjY1NzQxNjI3NDcz.YhwlWw.PjK54AItcV_lQL-RwA8PLj_oKj8' # токен бота

POST_ID =949816612666617936 # id поста, с котрого будут считываться реакции

ROLES = {
    '💻': 947693996015251556, # Роль 1
    '📷': 947694403705786409, # Роль 2
    '🟦': 947694466737786881, # Роль 3
    '🖌️': 947694552054124585, # Роль 4
} # список ролей с соответсвием эмоджи

MAX_ROLES_PER_USER = 2 # максимальное количество ролей, которое может взять пользовтель
```

Рис. 5. Содержимое файла config.py

После можно приступить к написанию непосредственно самого бота управления ролями (рис. 7).

DiscordBot.py config.py

```
import discord
from discord import utils

import config


class MyClient(discord.Client):
    async def on_ready(self):
        print('Logged on as {0}!'.format(self.user))

    async def on_raw_reaction_add(self, payload):
        if payload.message_id == config.POST_ID:
            channel = self.get_channel(payload.channel_id)  # получаем объект канала
            message = await channel.fetch_message(payload.message_id)  # получаем объект сообщения
            member = utils.get(message.guild.members,
                               id=payload.user_id)  # получаем объект пользователя который поставил реакцию

            try:
                emoji = str(payload.emoji)  # эмоджик который выбрал юзер
                role = utils.get(message.guild.roles, id=config.ROLES[emoji])  # объект выбранной роли (если есть)

                if (len([i for i in member.roles if i.id not in config.EXCROLES]) <= config.MAX_ROLES_PER_USER):
                    await member.add_roles(role)
                    print('[SUCCESS] User {0.display_name} has been granted with role {1.name}'.format(member, role))
                else:
                    await message.remove_reaction(payload.emoji, member)
                    print('[ERROR] Too many roles for user {0.display_name}'.format(member))

            except KeyError as e:
                print('[ERROR] KeyError, no role found for ' + emoji)
            except Exception as e:
                print(repr(e))

    async def on_raw_reaction_remove(self, payload):
        channel = self.get_channel(payload.channel_id)  # получаем объект канала
        message = await channel.fetch_message(payload.message_id)  # получаем объект сообщения
        member = utils.get(message.guild.members,
                           id=payload.user_id)  # получаем объект пользователя который поставил реакцию

        try:
            emoji = str(payload.emoji)  # эмоджик который выбрал юзер
            role = utils.get(message.guild.roles, id=config.ROLES[emoji])  # объект выбранной роли (если есть)

            await member.remove_roles(role)
            print('[SUCCESS] Role {1.name} has been remove for user {0.display_name}'.format(member, role))

        except KeyError as e:
            print('[ERROR] KeyError, no role found for ' + emoji)
        except Exception as e:
            print(repr(e))


# RUN
client = MyClient()
client.run(config.TOKEN)
```

Рис. 7. Код бота управления ролями

После написания необходимо проверить работоспособность бота. Для этого нужно зайти на сервер Discord, перейти в чат с выдачей ролей, щёлкнуть правой кнопкой мыши по посту и выбрать соответствующую реакцию, которая определяет роль (рис. 8).

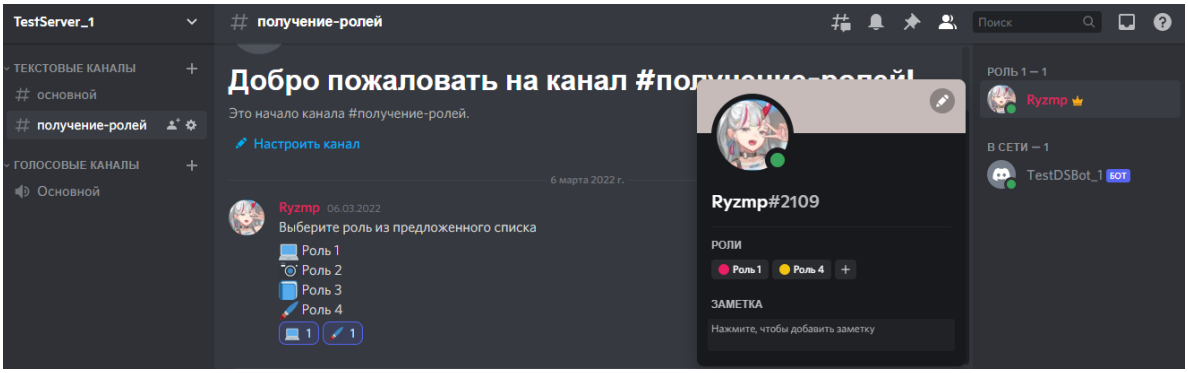

Рис. 8. Проверка

В результате исследования был создан бот управления ролями на сервере Discord, который в свою очередь существенно облегчает работу администраторов сервера, а также существенно повышает эффективность работы сервера с пользователями.

Библиографический список

1. Молчанов Н.М. Разработка ботов при помощи Discord API // В сборнике: Открытая городская научно-практическая конференция «Инженеры будущего». сборник работ победителей и призеров. Москва, 2020. С. 537-538.
2. Москвин М.М., Литвинов Д.А. Создание Discord бота на Python // В сборнике: Материалы студенческой научной конференции за 2020 год. В 2 частях. Воронежский государственный университет инженерных технологий. 2020. С. 68.
3. Прокопенко Е.А. Создание чат-бота для мессенджера Discord // В сборнике: Богатство России. сборник докладов. 2019. С. 87-88.
4. Кочкарев С.А., Жуков И.Р. Онлайн чат-бот // В сборнике: Открытая городская научно-практическая конференция «Инженеры будущего». сборник работ победителей и призеров. Москва, 2020. С. 481-484.
5. Миннивалиев Ш.Р., Карамышев А.Н., Абросимова Е.В. Приложение Telegram-Bot: "Система контроля состояния корпоративной сети" // Хроники объединенного фонда электронных ресурсов Наука и образование. 2017. № 10 (101). С. 28.
6. Мелитонян М.С., Фешина Е.В. Discord как средство онлайн общения и его преимущества среди других мессенджеров // Тенденции развития науки и образования. 2021. № 72-1. С. 73-76.