УДК 004

Создание бота VK на Python

Романов Даниил Алексеевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
Целью данной статьи является, создание программы чат бота, способного на некоторые взаимодействия с пользователем при помощи api и дополнительных библиотек языка Python. Программа написана на языке программирования Python, с использованием модулей vk_api, json и sqlite3. Результатом исследования станет готовая программа с подробным описанием ее реализации.
Ключевые слова: бот, VK, Python, vk_api, json, sqlite3

Creating a VK bot in Python

Romanov Daniil Alekseevich
Sholom-Aleichem Priamursky State University
Student

Abstract
The purpose of this article is to create a chatbot program capable of some user interaction using the api and additional Python libraries. The program is written in Python programming language, using vk_api, json and sqlite3 modules. The result of the study will be a ready-made program with a detailed description of its implementation.
Keywords: bot, VK, Python, vk_api, json, sqlite3

1 Введение

1.1 Актуальность

Язык программирования Python является поистине многозадачным языком программирования способным выполнять множество задач. В данной статье рассматривается лишь одна из многочисленных возможностей данного языка программирования. С помощью библиотеки vk_api можно создать полноценного чат-бота для взаимодействий с пользователем. Бот является довольно полезной программой для многих сфер т.к. с его помощью можно автоматизировать рутинные задачи. Они нужны там, где есть много похожих действий, которые сейчас выполняются руками. Бот может принять заказ в интернет-магазине, прислать подборку видеороликов, посоветовать подходящие концерты или ближайший ресторан.

1.2 Обзор исследований

В своей работе Д.В. Гриднев, М.Н. Иванов, В.А. Кирилкин описывали создание чат-ботов для различных социальных сетей, методы создания, а также возможности применения [1]. M. Vorontsov, S.I. Radmir в своём исследовании показали ипользование языка программирования Python для взаимодействия с API [2]. ДИ Биков продемонстрировал способы обработки запросов для чат-бота при помощи vk_api [3].

1.3 Цель исследования

Цель исследования – создать программу чат-бота на языке программирования Python способного на определённые взаимодействовия с пользователем.

2 Материалы и методы

Для создания программы потребуется несколько вещей. Во-первых, это язык программирования Pynthon [4], библиотеки vk_api [5], json [6], sqlite3 [7], среда программирования PyCharm [8] и наличие аккаунта в социальной сети VK [9].

3 Результаты и обсуждение

В первую очередь создаём новый проект в PyCharm. Устанавливаем и импортируем необходимые модули vk_api, json и sqlite3, котрый будет использоваться для хранения данных пользователя. Из vk_api импортируем VkLongPoll и VkEventType (рис.1).

```
import vk_api, json
import sqlite3
from vk_api.longpoll import VkLongPoll, VkEventType
```

Рисунок 1 - Импорт необходимых библиотек и компонентов

Заходим на главную страницу VK и выбираем вкладку “сообщество”. Нажимаем кнопку “создать сообщество”, заполняем все параметры и ещё раз нажимаем “создать сообщество”. В самом сообществе нажимаем “настройки” с правой стороны и выбираем “работа с API”. Нажимаем на кнопку “создать ключ” и выбираем только необходимые права для бота, например, доступ к сообщениям сообщества. Другие параметры лучше не выставлять, т.к. если кто-то узнает ключ сообщества, он получит доступ ко всем функциям сообщества (рис.2).

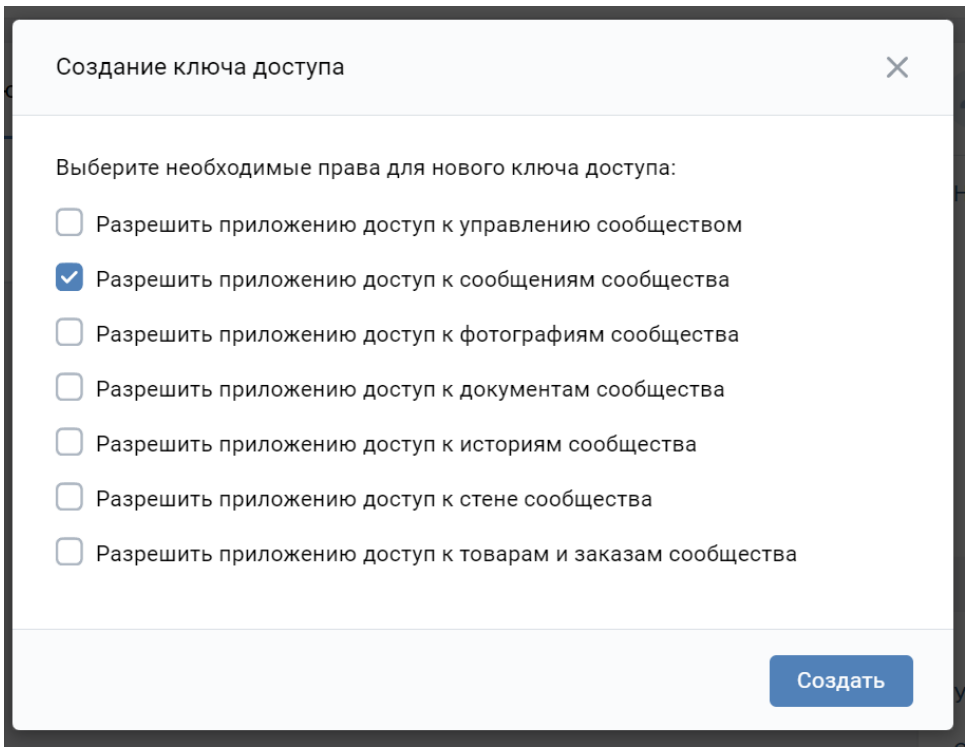

Рисунок 2 - Создание ключа сообщества

Далее создаём сессию используя метод VkApi в который передаём ключ для доступа к сообществу (рис.3).

```
vk_session = vk_api.VkApi(token='ваш ключ')
```

Рисунок 3 - Использование API ключа

Открываем vk_session используя метод get_api(). Создаём объект longpool, используя vk_session, он будет работать с событиями сообщества в реальном времени (рис.4).

```
session_api = vk_session.get_api()
longpool = VkLongPoll(vk_session)
```

Рисунок 4 - Подготовка объектов

С помощью sqlite3 создаём файл, в котором будет храниться база данных. Далее создаём объект курсора и вызываем его, с помощью него бот будет делать запросы к базе данных. Используя синтаксис SQL, создаём таблицу, в которой прописываем какие данные будут храниться в базе данных, для этого вызываем метод “execute”, нужный для обращения к базе данных. После любого изменения в таблице надо у переменной вызвать метод “commit” для сохранения данных и создать переменную “userAct” (рис.5).

```
db = sqlite3.connect('action.db')
sql = db.cursor()
sql.execute("""CREATE TABLE IF NOT EXISTS users (
    userId BIGINT,
    act TEXT,
    fio TEXT,
    gender TEXT,
    age TEXT
)""")
db.commit()
userAct = '0'
```

Рисунок 5 - Работа с хранением данных

Теперь создаём функцию sendMsg в которую передаём id и переменную some_text. Используя vk_session и метод messages.send указываем id пользователя, которому необходимо передать сообщение и само сообщение, которое должно прийти пользователю. Далее указываем уникальный идентификатор сообщения для того, чтобы не было дублирования одного и того же сообщения (рис.6).

```
def sendMsg(id, some_text):
    vk_session.method("messages.send", {"user_id": id, "message": some_text, "random_id": 0})
```

Рисунок 6 - Функция отправки сообщений пользователю

Затем создадим функцию fixMsg. Она нужна для того, чтобы оборачивать некоторые фрагменты текста в кавычки что позволит заносить их базу данных (рис.7).

```
def fixMsg(msg):
    msg = "'"+msg+"'"
    return msg
```

Рисунок 7 - Функция fixMsg

Далее создаём цикл, который будет проверять каждое событие из longpool, если событие является новым сообщением и адресовано боту, текст сообщения сохраняется в нижнем регистре переменной msg, а в id сохраняем id пользователя. В данном цикле будет описан основной функционал бота. С помощью метода “execute” находим id пользователя в таблице и, если его нет в базе данных бот предлагает провести регистрацию, записывая его данные в

базу данных. Если пользователь уже есть в базе данных, то бот не реагирует. (рис. 8).

```
for event in longpool.listen():
    if event.type == VkEventType.MESSAGE_NEW and event.to_me:
        msg = event.text.lower()
        id = event.user_id
        sql.execute(f"SELECT userId FROM users WHERE userId = '{id}'")
        if sql.fetchone() is None:
            sql.execute("INSERT INTO users VALUES (?, ?, ?, ?, ?)", (id, "newUser", "0", "0", "0"))
            db.commit()
            sendMsg(id, "Привет, напиши 'магазин', что бы просмотреть доступные товары.")
        else:
            userAct = sql.execute(f"SELECT act FROM users WHERE userId = '{id}'").fetchone()[0]
            if userAct == "newUser" and msg == "магазин":
                sendMsg(id, "Товары доступные без регистрации: Хлеб, вода, овощи, фрукты. Отправь 'рег' для регистрации.")
            elif userAct == "newUser" and msg == "рег":
                sql.execute(f"UPDATE users SET act = 'getFio' WHERE userId = {id}")
                db.commit()
                sendMsg(id, "Напиши свое ФИО")
            elif userAct == "getFio":
                sql.execute(f"UPDATE users SET fio = {fixMsg(msg)} WHERE userId = {id}")
                sql.execute(f"UPDATE users SET act = 'getGender' WHERE userId = {id}")
                db.commit()
                sendMsg(id, "Твой пол?")
            elif userAct == "getGender":
                sql.execute(f"UPDATE users SET gender = {fixMsg(msg)} WHERE userId = {id}")
                sql.execute(f"UPDATE users SET act = 'getAge' WHERE userId = {id}")
                db.commit()
                sendMsg(id, "Сколько тебе лет?")
            elif userAct == "getAge":
                sql.execute(f"UPDATE users SET age = {fixMsg(msg)} WHERE userId = {id}")
                sql.execute(f"UPDATE users SET act = 'full' WHERE userId = {id}")
                db.commit()
                sendMsg(id, "Регистрация прошла успешно!")
            elif userAct == "full" and msg == "магазин":
                sendMsg(id, "Ты зарегистрированный пользователь, тебе доступно все: Хлеб, вода, овощи, фрукты, алкоголь, оружие, лекарства.")
```

Рисунок 8 - Функционал бота

Запустив код и написав любое сообщение в сообщество бот ответит “Привет, напиши ‘магазин’, чтобы посмотреть доступные товары”. Пишем “магазин” и бот показывает какие товары нам доступны, а заодно предлагая зарегистрироваться: “Товары доступные без регистрации: Хлеб, вода, овощи, фрукты. Отправь ‘рег’ для регистрации.”. Если после регистрации снова ввести ‘рег’ бот не отреагирует, т.к. пользователь уже находится в базе данных. Вводим “Магазин” и бот отвечает “Ты зарегистрированный пользователь, тебе доступно все: Хлеб, вода, овощи, фрукты, алкоголь, оружие, лекарства.” (рис.7).

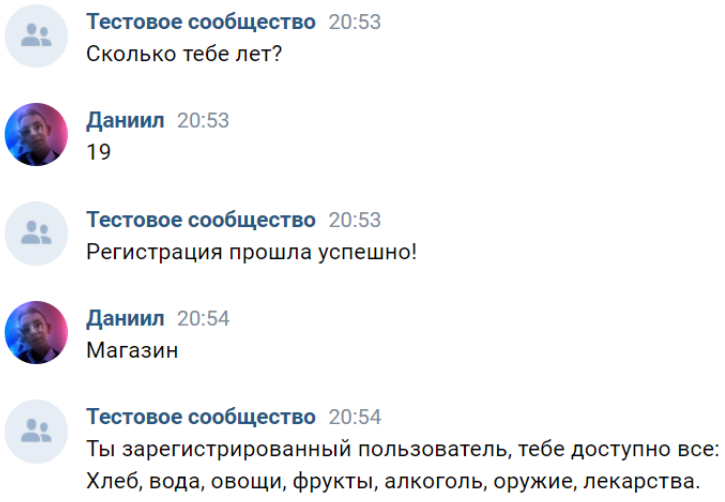

Рисунок 7 - Итог работы

Выводы

В данной работе была создана программа в виде бота для социальной сети VK. Бот способен на определённые взаимодействия с пользователем, среди которых есть возможность внесения данных пользователя в базу данных и проверка наличия данных пользователя в базе данных. Программу можно использовать как шаблон для создания других приложений с более продвинутым функционалом.

Библиографический список

1. Гриднев Д. В., Иванов М. Н., Кирилкин В. А. Разработка ботов ВКонтакте и телеграм для расписания университета // Наука. Информатизация. Технологии. Образование. 2020. С. 35-42.
2. Vorontsov M., Radmir S. I. Automation of Message Sending Processes Using Specialized Software //2021 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus). IEEE, 2021. С. 746-748.
3. Биков Д. И. Способы обработки запросов для чат-бота при помощи инструментов VK API //Приоритетные направления инновационной деятельности в промышленности. 2020. С. 35-36.
4. Python URL: https://www.python.org/downloads
5. vk_api URL: https://pypi.org/project/vk-api
6. json URL: https://pypi.org/project/jsons
7. sqlite3 URL: https://pypi.org/project/pysqlite3
8. PyCharm URL: https://www.jetbrains.com/ru-ru/pycharm/download
9. VK URL: https://vk.com