**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Парадигмы и конструкции языков программирования»

Отчет по лабораторной работе №5-6

«Вк Бот»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-25Б |  |  |
| Герасимович Н.И. |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г

**Задание:**

В этом коде реализован бот на Python, который использует различные библиотеки для выполнения нескольких функций, включая получение погоды, времени в разных городах, предоставление шуток и управление состоянием суммы. Код начинается с импорта необходимых библиотек, таких как requests для выполнения HTTP-запросов, bs4 для парсинга HTML, re для работы с регулярными выражениями, datetime и pytz для работы с датой и временем, а также random для генерации случайных чисел. Из строки test\_req извлекается URL для API погоды и API ключ с помощью регулярного выражения.

Основной класс бота называется VkBot, который инициализируется с идентификатором пользователя. При создании объекта бота происходит получение имени пользователя из VK и инициализация команд и шуток. Метод \_get\_user\_name\_from\_vk\_id получает имя пользователя из VK по его идентификатору, отправляя запрос и парсируя HTML-ответ для извлечения имени. Метод get\_random\_joke возвращает случайную шутку из предустановленного списка шуток. Метод add\_to\_sum увеличивает текущее значение суммы на заданное значение и возвращает обновленную сумму. Метод \_get\_time получает текущее время в указанном городе, используя словарь соответствия городов и часовых поясов, и, если город не найден, возвращает сообщение об ошибке. Метод \_get\_weather получает текущую погоду в указанном городе, обращаясь к OpenWeatherMap API, и возвращает температуру и описание погоды или сообщение об ошибке, если данные не могут быть получены. Метод \_clean\_all\_tag\_from\_str очищает строку от HTML-тегов, возвращая только текст. Метод new\_message обрабатывает входящие сообщения, проверяя, содержит ли сообщение одну из команд. В зависимости от команды бот выполняет соответствующее действие, например, приветствует пользователя, получает погоду или время, рассказывает шутку или увеличивает сумму, и возвращает ответ. Если команда не распознана, бот сообщает, что не понимает запрос. В целом, бот предоставляет пользователю возможность взаимодействовать с ним через текстовые команды, получая информацию о погоде, времени в разных городах и развлечение в виде шуток.

**Текст программы:**

**main.py**

import vk\_api  
from vk\_api.longpoll import VkLongPoll, VkEventType  
from vk\_bot import VkBot  
import random  
  
def write\_msg(user\_id, message):  
 vk.method('messages.send', {  
 'user\_id': user\_id,  
 'message': message,  
 'random\_id': random.randint(1, 2\*\*31) # Генерация случайного ID  
 })  
  
# API-ключ созданный ранее  
token = "vk1.a.F7bSNbVvaAk8unSm6\_jrd9iWXKTrY5C8I9qiGdkpRCBjEXxvsGEGWf7ZSlSXI61vrqjEtGq3dZTAz6BsRxwkrrBTIOEY\_ktT1n1ms9L1hy3IPViIR-k7ohozg3deHu\_VjugNpFpLDmQqBjeeTAMhJvQBSx5Y-QTSkrXMF8v6QH1mqXVsvMM41cpJSzFTGRpCGKVuXiMzg6Jx2IMNl9teaw"  
  
# Авторизуемся как сообщество  
vk = vk\_api.VkApi(token=token)  
  
# Работа с сообщениями  
longpoll = VkLongPoll(vk)  
  
# Создаем экземпляр бота один раз  
user\_id = None  
bot = VkBot(user\_id)  
  
# Основной цикл  
print("Server started")  
for event in longpoll.listen():  
 if event.type == VkEventType.MESSAGE\_NEW:  
 if event.to\_me:  
 print('New message:')  
 print(f'For me by: {event.user\_id}', end='')  
  
 # Устанавливаем user\_id для бота  
 bot.\_USER\_ID = event.user\_id # Обновляем user\_id в экземпляре бота  
  
 response\_message = bot.new\_message(event.text)  
 write\_msg(event.user\_id, response\_message)  
  
 print('Text: ', event.text)

**vk\_bot.py**

import requests  
import bs4  
import re  
from datetime import datetime  
import pytz  
import random  
  
  
  
test\_req = 'curl "https://api.openweathermap.org/data/2.5/weather?q=Moscow&appid=9c285a2cce86a82497e6f39ac6fb207b"'  
  
# Extract the URL and API key from the test\_req string  
match = re.search(r'curl\s\*"(https://api\.openweathermap\.org/data/2\.5/weather\?q=Moscow&appid=[^"]+)"', test\_req)  
if match:  
 weather\_api\_url = match.group(1)  
else:  
 raise ValueError("Invalid test\_req string")  
  
class VkBot:  
  
 def \_\_init\_\_(self, user\_id):  
 print("Создан объект бота!")  
 self.\_USER\_ID = user\_id  
 self.\_USERNAME = self.\_get\_user\_name\_from\_vk\_id(user\_id)  
 self.base = 0  
 self.\_COMMANDS = ["ПРИВЕТ", "ПОГОДА", "ВРЕМЯ", "ПОКА", "АНЕКДОТ", "УВЕЛИЧЬ"]  
 self.\_WEATHER\_API\_KEY = "9c285a2cce86a82497e6f39ac6fb207b" # Замените на ваш API ключ  
 self.\_jokes = [  
 "Почему программисты не любят природу? Потому что там слишком много багов.",  
 "Какой язык программирования самый оптимистичный? C#, потому что он всегда ищет положительные числа.",  
 "Какой любимый напиток у программистов? Java!",  
 "Почему у программистов не бывает жены? Потому что они всегда ищут правильный алгоритм.",  
 "Как программист делает себе завтрак? Он просто компилирует яйца.",  
 "Почему в программировании так много ошибок? Потому что программисты не могут найти 'исправить' в меню.",  
 "Какой любимый праздник у программистов? Хэллоуин, потому что они могут нарядиться в свои любимые баги.",  
 "Почему программисты всегда берут с собой запасной блокнот? На случай, если их код не скомпилируется.",  
 "Как программист заказывает пиццу? Он просто говорит: 'Пожалуйста, сделайте это с нулевым багом.'",  
 "Почему программисты не могут играть в прятки? Потому что хорошие программисты всегда находятся в 'отладочном режиме'."  
 ]  
 self.state = {  
 'total\_sum': 0  
 }  
  
 def \_get\_user\_name\_from\_vk\_id(self, user\_id):  
 headers = {  
 'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/58.0.3029.110 Safari/537.3'  
 }  
 request = requests.get(f"https://vk.com/id{user\_id}", headers=headers)  
  
 if request.status\_code != 200:  
 return f"Ошибка доступа к VK: {request.status\_code}"  
  
 bs = bs4.BeautifulSoup(request.text, "html.parser")  
 titles = bs.findAll("title")  
 if titles:  
 user\_name = self.\_clean\_all\_tag\_from\_str(titles[0].get\_text())  
 return user\_name.split()[0]  
 else:  
 return "Пользователь не найден"  
  
  
 def get\_random\_joke(self):  
 return random.choice(self.\_jokes)  
  
 def add\_to\_sum(self, value: int) -> str:  
 self.state['total\_sum'] += value  
 return f"Текущая сумма: {self.state['total\_sum']}"  
  
 def \_get\_time(self, city: str) -> str:  
 # Словарь соответствия городов и часовых поясов  
 timezone\_mapping = {  
 'москва': 'Europe/Moscow',  
 'нью-йорк': 'America/New\_York',  
 'лондон': 'Europe/London',  
 'токио': 'Asia/Tokyo',  
 'париж': 'Europe/Paris',  
 'берлин': 'Europe/Berlin',  
 'мадрид': 'Europe/Madrid',  
 'рома': 'Europe/Rome',  
 'канберра': 'Australia/Sydney',  
 'оттава': 'America/Toronto',  
 'пекин': 'Asia/Shanghai',  
 'сеул': 'Asia/Seoul',  
 'новый-дели': 'Asia/Kolkata',  
 'бразилиа': 'America/Sao\_Paulo',  
 'мехико': 'America/Mexico\_City',  
 'буэнос-айрес': 'America/Argentina/Buenos\_Aires',  
 'сантьяго': 'America/Santiago',  
 'каир': 'Africa/Cairo',  
 'йоханнесбург': 'Africa/Johannesburg',  
 'наироби': 'Africa/Nairobi',  
 'анкары': 'Europe/Istanbul',  
 'абу-даби': 'Asia/Dubai',  
 'рияд': 'Asia/Riyadh',  
 'дока': 'Asia/Qatar',  
 'куала-лумпур': 'Asia/Kuala\_Lumpur',  
 'манила': 'Asia/Manila',  
 'ханой': 'Asia/Bangkok',  
 'исламабад': 'Asia/Karachi',  
 'тегеран': 'Asia/Tehran',  
 'стокгольм': 'Europe/Stockholm',  
 'осло': 'Europe/Oslo',  
  
 # Добавленные города Российской Федерации  
 'санкт-петербург': 'Europe/Moscow',  
 'новосибирск': 'Asia/Novosibirsk',  
 'екатеринбург': 'Asia/Yekaterinburg',  
 'нижний-новгород': 'Europe/Moscow',  
 'казань': 'Europe/Moscow',  
 'челябинск': 'Asia/Yekaterinburg',  
 'омск': 'Asia/Omsk',  
 'самара': 'Europe/Samara',  
 'ростов-на-дону': 'Europe/Moscow',  
 'уфа': 'Asia/Yekaterinburg',  
 'волгоград': 'Europe/Moscow',  
 'красноярск': 'Asia/Krasnoyarsk',  
 'иркутск': 'Asia/Irkutsk',  
 'якутск': 'Asia/Yakutsk',  
 'владивосток': 'Asia/Vladivostok',  
 'магадан': 'Asia/Magadan',  
 }  
  
 # Приводим название города к нижнему регистру  
 city = city.lower()  
  
 # Проверяем, есть ли у нас часовой пояс для введенного города  
 if city not in timezone\_mapping:  
 return "Извините, я не знаю, какой часовой пояс у этого города."  
  
 timezone = timezone\_mapping[city]  
  
 try:  
 # Получаем текущее время в указанном часовом поясе  
 tz = pytz.timezone(timezone)  
 current\_time = datetime.now(tz)  
 return f"Время в городе {city.capitalize()}: {current\_time.strftime('%Y-%m-%d %H:%M:%S')}"  
  
 except Exception as e:  
 return f"Ошибка при получении времени: {e}"  
  
  
 def \_get\_weather(self, city: str) -> str:  
 base\_url = "https://api.openweathermap.org/data/2.5/weather"  
 params = {  
 'q': city,  
 'appid': self.\_WEATHER\_API\_KEY,  
 'lang': 'ru',  
 'units': 'metric'  
 }  
 response = requests.get(base\_url, params=params)  
 data = response.json()  
  
 if response.status\_code == 200:  
 temperature = data['main']['temp']  
 weather\_description = data['weather'][0]['description']  
 return f"В городе {city} сейчас {temperature}°C с погодой: {weather\_description}."  
 else:  
 return f"Не удалось получить данные о погоде. Статус: {response.status\_code}"  
  
 @staticmethod  
 def \_clean\_all\_tag\_from\_str(string\_line):  
 if string\_line:  
 return ''.join(bs4.BeautifulSoup(string\_line, "html.parser").findAll(text=True))  
 return ""  
  
 def new\_message(self, message):  
 if message.upper().startswith(self.\_COMMANDS[0]):  
 return f"Привет-привет!"  
 elif message.upper().startswith(self.\_COMMANDS[1]):  
 # Extract the city name from the message  
 parts = message.split()  
 if len(parts) > 1:  
 city = ' '.join(parts[1:])  
 else:  
 city = 'Moscow' # Default city if none is provided  
 return self.\_get\_weather(city)  
 elif message.upper().startswith(self.\_COMMANDS[2]):  
 parts = message.split()  
 if len(parts) > 1:  
 city = ' '.join(parts[1:])  
 else:  
 city = 'Москва' # Default city if none is provided  
 return self.\_get\_time(city)  
 elif message.upper() == self.\_COMMANDS[3]:  
 return f"Пока-пока!"  
 elif message.upper() == self.\_COMMANDS[4]:  
 return self.get\_random\_joke()  
 elif message.upper().startswith(self.\_COMMANDS[5]):  
 parts = message.split()  
 if len(parts) > 1 :  
 plus = int(' '.join(parts[2:]))  
 else:  
 plus = 1# Default city if none is provided  
 return self.add\_to\_sum(plus)  
 else:  
 return "Не понимаю о чем вы..."