Московский государственный технический университет им. Н.Э. Баумана

Факультет ИУ
Кафедра ИУ5

Курс «Основы информатики»
Отчет по домашнему заданию

Выполнил студент группы ИУ5-33Б:
Хасанова К.М.
Подпись и дата:

Проверил преподаватель каф.:
Гапанюк Ю. Е.
Подпись и дата:

Москва, 2024 г

Постановка задачи

Реализовать небольшой проект на языке программирования, который ранее не был изучен.

1. Выбранный язык программирования

Python

2. Подробности реализации задачи

Разработать программу, которая обращается к внешнему API для получения списка отелей, доступных в указанном городе на определенную дату заезда.

Текст программы

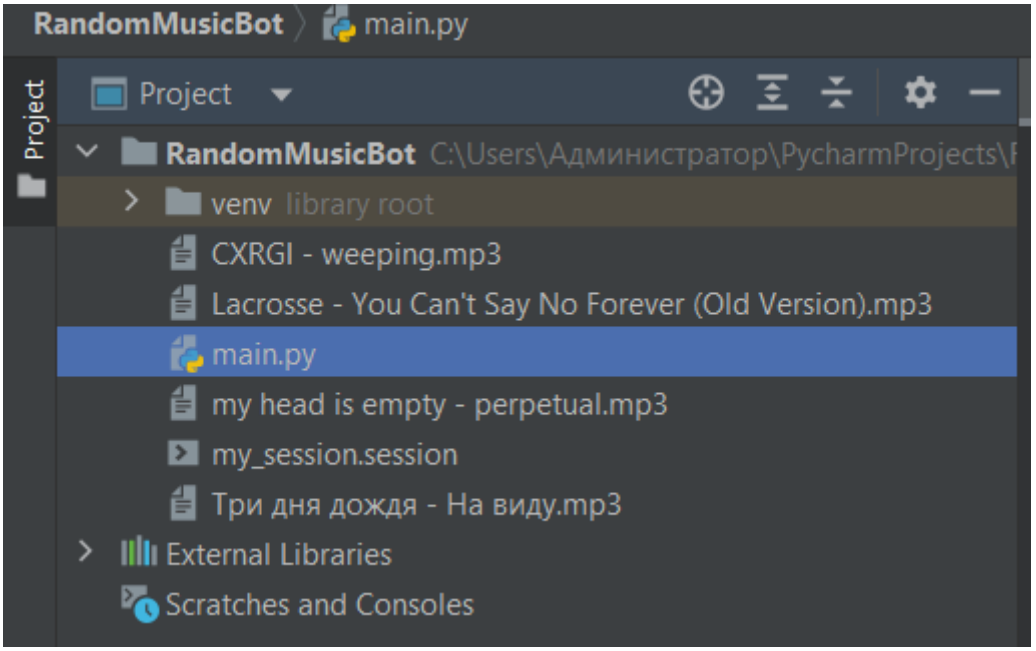
```
import random
import os
from typing import Final
from telegram import Update
from telegram.ext import Application, CommandHandler, MessageHandler, filters,
ContextTypes
from telethon import TelegramClient
from telethon.tl.functions.messages import GetHistoryRequest
from telethon.tl.types import MessageMediaDocument

# Константы
TOKEN: Final = '7366610208:AAFkUPBJJVy7RgvS5YAE87AeC4llhRgOOqc'
BOT_USERNAME: Final = '@miaumusicbot'
CHANNEL_USERNAME: Final = "@ejjejfnfnn"  # Юзернейм канала
API_ID: Final = '22766592'  # Укажите ваш API ID (my.telegram.org)
API_HASH: Final = '25a1510b124adc9de2ea37919a8e6dba'  # Укажите ваш API Hash
(my.telegram.org)

# Инициализация клиента Telethon
client = TelegramClient('my_session', API_ID, API_HASH).start()

# Функция для получения истории сообщений из канала
async def get_audio_messages():
    """Получает последние 50 аудиосообщений из указанного канала."""
    try:
        async with client:
            history = await client(GetHistoryRequest(
                peer=CHANNEL_USERNAME,
                limit=50,
                offset_date=None,
                offset_id=0,
                max_id=0,
                min_id=0,
                add_offset=0,
                hash=0
            ))
```

```
            # Фильтруем только аудиофайлы
            audio_messages = []
            for message in history.messages:
                if (message.media and
                        hasattr(message.media, 'document') and
                        message.media.document.mime_type.startswith('audio')):
                    audio_messages.append(message)

            return audio_messages
    except Exception as e:
        print(f"Ошибка при получении истории канала: {e}")
        return []


# Команда /start
async def start_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text('Привет, я Бот по отправке разной музыки!')


# Команда /help
async def help_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    await update.message.reply_text('Отправь мне команду /random или /last для
получения случайной или последней добавленной песни.')


# Команда /last
async def last_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """Отправляет последний добавленный аудиофайл из канала."""
    try:
        audio_messages = await get_audio_messages()

        if not audio_messages:
            await update.message.reply_text("В канале нет доступных
аудиофайлов!")
            return

        # Последнее сообщение
        last_audio = audio_messages[-1]

        # Проверяем, является ли сообщение аудиофайлом
        if isinstance(last_audio.media, MessageMediaDocument):
            # Локальная загрузка аудиофайла
            file_path = await client.download_media(last_audio.media)

            if not os.path.exists(file_path):
                await update.message.reply_text("Не удалось загрузить
аудиофайл.")
                return

            # Отправляем аудиофайл пользователю
            with open(file_path, 'rb') as audio_file:
                await context.bot.send_audio(chat_id=update.effective_chat.id,
audio=audio_file)

            # Удаляем локальный файл после отправки
            os.remove(file_path)
        else:
            await update.message.reply_text("Последнее сообщение не является
аудиофайлом.")
    except Exception as e:
        await update.message.reply_text(f"Произошла ошибка: {e}")
```

```python
# Команда /random
async def random_command(update: Update, context: ContextTypes.DEFAULT_TYPE):
    """Отправляет случайный аудиофайл из канала."""
    try:
        audio_messages = await get_audio_messages()

        if not audio_messages:
            await update.message.reply_text("В канале нет доступных
аудиофайлов!")
            return

        # Случайное сообщение
        random_audio = random.choice(audio_messages)

        if isinstance(random_audio.media, MessageMediaDocument):
            # Локальная загрузка аудиофайла
            file_path = await client.download_media(random_audio.media)

            if not os.path.exists(file_path):
                await update.message.reply_text("Не удалось загрузить
аудиофайл.")
                return

            # Отправляем аудиофайл пользователю
            with open(file_path, 'rb') as audio_file:
                await context.bot.send_audio(chat_id=update.effective_chat.id,
audio=audio_file)

            # Удаляем локальный файл после отправки
            os.remove(file_path)
        else:
            await update.message.reply_text("Выбранное сообщение не является
аудиофайлом.")
    except Exception as e:
        await update.message.reply_text(f"Произошла ошибка: {e}")


# Обработчик сообщений
async def handle_message(update: Update, context: ContextTypes.DEFAULT_TYPE):
    message_type: str = update.message.chat.type
    text: str = update.message.text

    print(f'User ({update.message.chat.id}) in {message_type}: "{text}"')

    if message_type == 'group' and BOT_USERNAME in text:
        new_text: str = text.replace(BOT_USERNAME, '').strip()
        response: str = handle_response(new_text)
    else:
        response: str = handle_response(text)

    print('Bot:', response)
    await update.message.reply_text(response)


# Обработчик текста (можно расширить)
def handle_response(text: str) -> str:
    return "Я пока не понимаю этот текст."
```

```python
# Обработчик ошибок
async def error(update: Update, context: ContextTypes.DEFAULT_TYPE):
    print(f'Update {update} caused error {context.error}')


# Запуск приложения
if __name__ == '__main__':
    print('Starting bot...')

    # Создаем приложение для Telegram бота
    app = Application.builder().token(TOKEN).build()

    # Добавляем команды
    app.add_handler(CommandHandler('start', start_command))
    app.add_handler(CommandHandler('help', help_command))
    app.add_handler(CommandHandler("last", last_command))
    app.add_handler(CommandHandler("random", random_command))

    # Обработчик сообщений
    app.add_handler(MessageHandler(filters.TEXT, handle_message))

    # Обработчик ошибок
    app.add_error_handler(error)

    # Запуск polling
    print('Polling...')
    app.run_polling(poll_interval=3)
```

Выполнение программы




23 декабря
/start 14:32
RB
Привет, я Бот по отправке разной музыки! 14:32
/help 14:32
RB
Отправь мне команду /random или /last для получения случайной или последней добавленной песни. 14:32
/last 14:32
RB
Произошла ошибка: Cannot send requests while disconnected 14:32
/random 14:32
/random 14:32
Произошла ошибка: Cannot send requests while disconnected 14:32
RB
Произошла ошибка: Cannot send requests while disconnected 14:32
/random 14:32
RB
Произошла ошибка: Cannot send requests while disconnected 14:32
Меню
Написать сообщение...