Санкт-Петербургский государственный университет

Факультет прикладной математики – процессов управления

**Отчет №1**

по дисциплине «Функциональное программирование»

**Разработка многофункционального чат-бота с использованием Spotify и OpenWeather API**

Автор работы: Дацык Р.В.

Группа: 22.Б15-пу

Преподаватель: Киямов Ж. У.

Санкт-Петербург, 2023

**Оглавление**

[**1.** **Цель работы** 3](#_Toc153233350)

[**2.** **Задачи** 3](#_Toc153233351)

[**3.** **Теоретическая часть** 3](#_Toc153233352)

[**4.** **Алгоритм метода** 4](#_Toc153233353)

[**5.** **Описание программы** 5](#_Toc153233354)

[**6.** **Рекомендации пользователя** 10](#_Toc153233355)

[**7.** **Рекомендации программиста** 11](#_Toc153233356)

[**8.** **Контрольный пример** 11](#_Toc153233357)

[**9.** **Вывод** 13](#_Toc153233358)

[**10.** **Список использованной литературы** 13](#_Toc153233359)

# **Цель работы**

Целью данной работы является разработка музыкально-информационного бота, способного взаимодействовать с сервисами Spotify для воспроизведения музыки и OpenWeather для получения информации о погоде.

# **Задачи**

Для достижения цели поставлены следующие задачи:

1. Изучение и использование Spotify API для управления музыки.
2. Реализация интеграции с OpenWeather API для получения данных о погоде в различных городах.
3. Создание Telegram-бота с использованием Telegram Bot API для взаимодействия с пользователями.
4. Разработка функциональности воспроизведения музыки и отображения информации о погоде.
5. Проверка работоспособности бота в реальных условиях.

# **Теоретическая часть**

1. **Spotify API**

С использованием библиотеки spotipy была разработана функциональность для взаимодействия с Spotify API. Пользователю предоставляется выбор плейлистов, а также управление воспроизведением (воспроизведение, пауза, следующий/предыдущий трек).

1. **OpenWeather API**

Для получения информации о погоде был использован OpenWeather API. Пользователь может запросить погоду в конкретном городе, а также просмотреть текущую погоду в ранее выбранном городе.

1. **Интеграция с Telegram Bot API**

Для обеспечения удобного взаимодействия с пользователем был создан бот, использующий Telegram Bot API. Пользователь может выбирать различные действия, такие как воспроизведение музыки, получение информации о погоде, а также возвращение в основное меню.

# **Алгоритм метода**

Алгоритм работы бота представлен следующим образом:

1. **Инициализация:**
   * Бот создается с использованием библиотеки python-telegram-bot и получает токен для взаимодействия с Telegram Bot API.
   * Получается токен от Spotify для взаимодействия с их API.
2. **Главное меню:**
   * Пользователю предоставляется главное меню с двумя разделами: "Spotify" и "Погода".
   * Пользователь может выбрать один из разделов.
3. **Spotify:**
   * В разделе "Spotify" пользователь может выбрать действия, такие как воспроизведение музыки ("Микс дня", "Любимое") или вернуться в главное меню.
   * При выборе определенного плейлиста бот использует Spotify API для воспроизведения соответствующего плейлиста.
4. **Погода:**
   * В разделе "Погода" пользователь может выбрать действия, такие как выбор города для получения прогноза или просмотр текущей погоды в ранее выбранном городе.
   * При выборе города бот использует OpenWeather API для получения информации о погоде.
5. **Обработка команд:**
   * Бот обрабатывает команды пользователя и выполняет соответствующие действия (воспроизведение музыки, получение погоды, возвращение в главное меню и т.д.).
   * Для управления Spotify используется spotipy для взаимодействия с Spotify API.
6. **Обработка сообщений:**
   * Бот также обрабатывает текстовые сообщения пользователя и предоставляет ответы, например, когда пользователь вводит название города для получения погоды.
7. **Интеграция с OpenWeather и Spotify:**
   * Разработаны функции для взаимодействия с OpenWeather API для получения данных о погоде в выбранных городах.
   * Реализованы функции для взаимодействия с Spotify API для получения рекомендаций и воспроизведения музыки.
8. **Обработка ошибок:**
   * Реализована обработка ошибок, например, если не удалось получить токен от Spotify или информацию о погоде от OpenWeather.
9. **Остановка бота:**
   * Бот остается в активном состоянии, ожидая команды пользователя, пока не завершится процесс работы.

Этот алгоритм обеспечивает пользователю интерактивное взаимодействие с ботом, позволяя ему управлять воспроизведением музыки, получать рекомендации и информацию о погоде.

# **Описание программы**

Программа представляет собой музыкально-информационного бота для платформы Telegram, который интегрирован с Spotify и OpenWeather API. Бот предоставляет пользователю функционал по воспроизведению музыки из Spotify и получению информации о погоде в различных городах. Пользователь может выбирать плейлисты для прослушивания, управлять воспроизведением, а также получать прогноз погоды для конкретного города.

Таблица 5.1. Описание методов

|  |  |
| --- | --- |
| Название функции | Назначение функции |
| **get\_spotify\_object()** | * **Описание:** Получает объект Spotify API для взаимодействия с сервисом. * **Возвращаемое значение:** Объект spotipy.Spotify или None в случае ошибки. |
| **get\_spotify\_recommendations()** | * **Описание:** Получает рекомендации по музыке из Spotify. * **Возвращаемое значение:** Список рекомендаций или None в случае ошибки. |
| **get\_weather(city)** | * **Описание:** Получает информацию о погоде в указанном городе с использованием OpenWeather API. * **Аргументы:** city - название города. * **Возвращаемое значение:** Строка с информацией о погоде или сообщение об ошибке. |
| **start\_weather(update: Update, context: CallbackContext) -> int** | * **Описание:** Обработчик команды для начала запроса погоды в определенном городе. * **Возвращаемое значение:** Состояние выбора города. |
| **handle\_city\_input(update: Update, context: CallbackContext) -> None** | * **Описание:** Обрабатывает ввод названия города для получения погоды и отправляет результат пользователю. * **Возвращаемое значение:** None. |
| **handle\_show\_weather(update: Update, context: CallbackContext) -> None** | * **Описание:** Обрабатывает команду пользователя для отображения текущей погоды в ранее выбранном городе. * **Возвращаемое значение:** None. |
| **cancel\_weather(update: Update, context: CallbackContext) -> int** | * **Описание:** Обработчик команды отмены запроса погоды. * **Возвращаемое значение:** Конечное состояние разговора. |
| **weather\_handler() -> ConversationHandler** | **Описание:** Создает обработчик для взаимодействия с пользователем по запросу погоды. |
| **play(update: Update, context: CallbackContext) -> None** | * **Описание:** Начинает воспроизведение музыки с использованием Spotify API. * **Возвращаемое значение:** None. |
| **pause(update: Update, context: CallbackContext) -> None** | * **Описание:** Ставит воспроизведение музыки на паузу. * **Возвращаемое значение:** None. |
| **next\_track(update: Update, context: CallbackContext) -> None** | * **Описание:** Переключается на следующий трек в списке воспроизведения. * **Возвращаемое значение:** None. |
| **prev\_track(update: Update, context: CallbackContext) -> None** | * **Описание:** Переключается на предыдущий трек в списке воспроизведения. * **Возвращаемое значение:** None. |
| **spotify\_menu(update: Update, context: CallbackContext) -> int** | * **Описание:** Отображает подменю для выбора плейлиста в Spotify. * **Возвращаемое значение:** Конечное состояние разговора. |
| **handle\_messages(update: Update, context: CallbackContext) -> None** | * **Описание:** Обрабатывает текстовые сообщения пользователя и выполняет соответствующие действия. * **Возвращаемое значение:** None. |
| **playback\_buttons(update: Update) -> None** | * **Описание:** Показывает кнопки управления воспроизведением. * **Возвращаемое значение:** None. |
| **play\_playlist(update: Update, playlist\_url: str) -> None** | * **Описание:** Начинает воспроизведение выбранного плейлиста из Spotify. * **Аргументы:** playlist\_url - URL плейлиста. * **Возвращаемое значение:** None. |
| **show\_actions\_menu(update: Update) -> None** | * **Описание:** Показывает кнопки управления воспроизведением после начала воспроизведения плейлиста. * **Возвращаемое значение:** None. |
| **get\_playlist\_id\_from\_url(playlist\_url: str) -> str** | * **Описание:** Извлекает идентификатор плейлиста из URL. * **Аргументы:** playlist\_url - URL плейлиста. * **Возвращаемое значение:** Идентификатор плейлиста. |
| **start(update: Update, context: CallbackContext) -> None** | * **Описание:** Обработчик команды /start. Приветствует пользователя. * **Возвращаемое значение:** None. |
| **help\_command(update: Update, context: CallbackContext) -> None** | * **Описание:** Обработчик команды /help. Предоставляет пользователю список доступных команд. * **Возвращаемое значение:** None. |
| **handle\_action\_selection(update: Update, context: CallbackContext) -> int:** | * **Описание:** Обработчик выбора действия из главного меню. * **Возвращаемое значение:** Конечное состояние разговора. |
| **main\_menu(update: Update, context: CallbackContext) -> int** | * **Описание:** Отображает главное меню с выбором разделов. * **Возвращаемое значение:** Конечное состояние разговора. |
| **main() -> None** | * **Описание:** Основная функция для запуска бота. Инициализирует бота, добавляет обработчики команд и сообщений, начинает опрос для взаимодействия с Telegram API. * **Возвращаемое значение:** None. |

# **Рекомендации пользователя**

Для запуска программы достаточно перейти по ссылке ведущей к чат-боту и нажать кнопку «СТАРТ». В главном меню присутствуют 2 кнопки: «Spotify», «Погода», ответственные за соответствующие действия. В «Spotify» пользователю предоставляется выбор между плейлистами, в последствии появляются кнопки управления музыкой. При выборе «Погода» пользователь должен ввести интересующий его город.

# **Рекомендации программиста**

Для запуска программы необходима 64-битная операционная система Windows, Linux или macOS. Для работы с кодом необходима среда разработки, совместимая с python 3.10. Также требуется установить следующие библиотеки: spotipy, telegram, requests.

Исходный код программы доступен по ссылке:

<https://github.com/CapTopGrade/Functional_Programming/blob/main/%D0%A4%D0%9F/%D0%A4%D0%9F%20%D0%B0%D1%81%D0%B8%D0%BD%D1%85%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B9%20%D1%87%D0%B0%D1%82/ACHAT%20server.py>

# **Контрольный пример**

В данном разделе представлен контрольный пример, демонстрирующий работу программы.

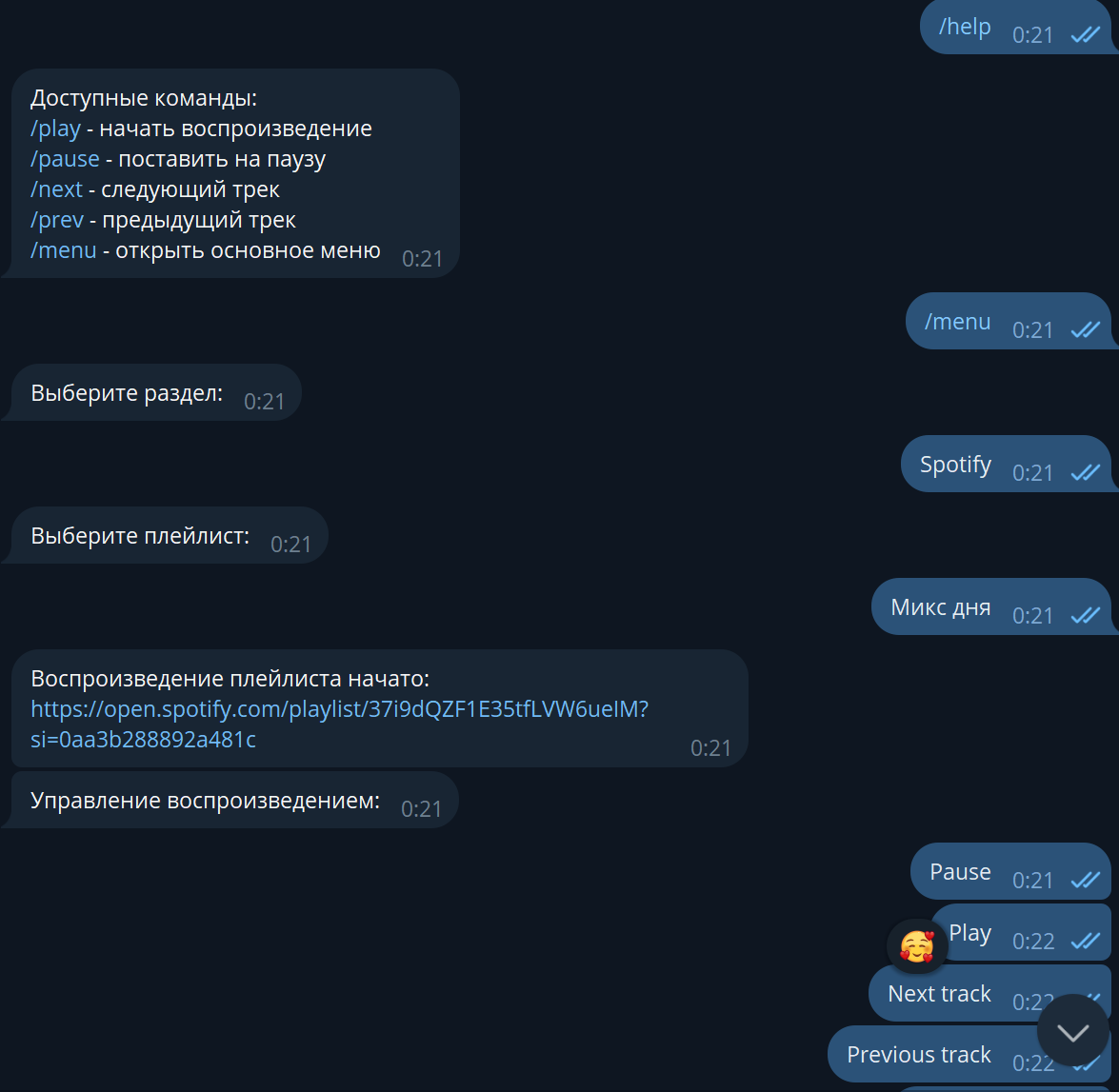


Рисунок 8.1, интерфейс для десктопной версии телеграм

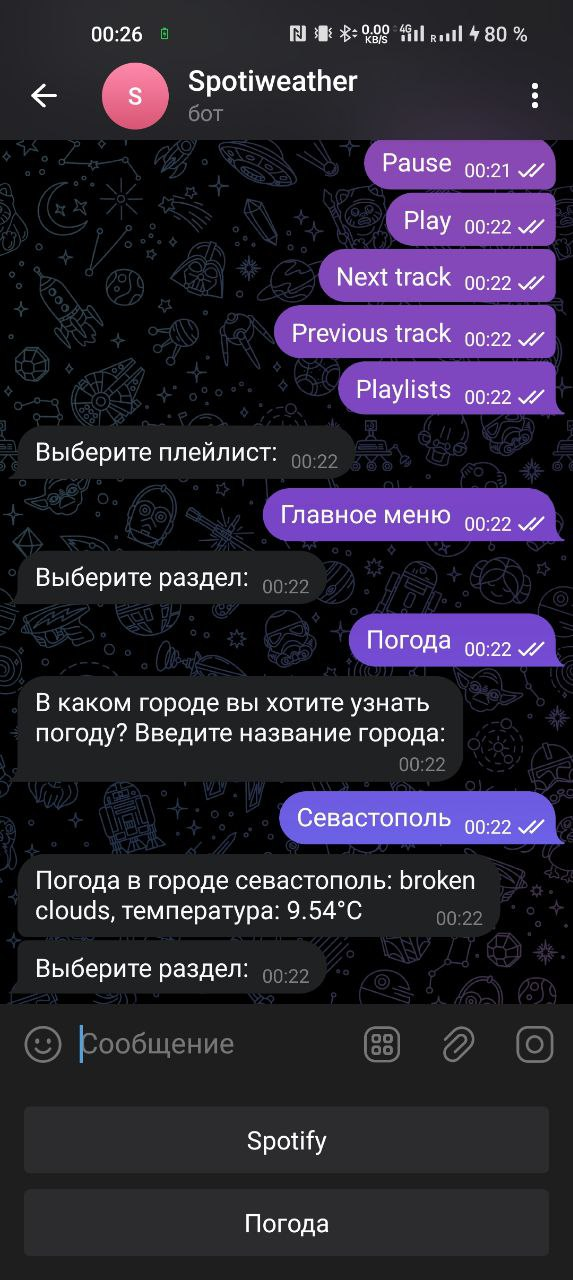


Рисунок 8.2, Пример результата работы программы

# **Вывод**

Разработанный музыкально-информационный бот успешно интегрирует функциональность Spotify и OpenWeather API, предоставляя пользователям удобный интерфейс через Telegram. Бот предоставляет функциональность воспроизведения музыки и получения информации о погоде, что делает его универсальным инструментом для пользователей.

# **Список использованной литературы**

1. Spotify API Documentation: <https://developer.spotify.com/documentation/web-api/>
2. OpenWeather API Documentation: <https://openweathermap.org/api>
3. Telegram Bot API Documentation: <https://core.telegram.org/bots/api>