ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ  
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»  
МОСКОВСКИЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ

**РУКОВОДСТВО РАЗРАБОТЧИКА**

Студент(ы) группы БИВ224:

* Воробьев Антон Вадимович, avvorobev\_5@edu.hse.ru, +79652039942;
* Саблин Арсений Михайлович, amsablin@edu.hse.ru, +79170284688.

Тема работы:

**Приложение для анализа музыкальных композиций из Spotify**

Руководитель:

Полякова Марина Васильевна

**Оглавление**

[**Введение** 3](#_Toc135473525)

[**Характеристики программы** 4](#_Toc135473526)

[**Описание необходимого окружения** 4](#_Toc135473527)

[**Установка и запуск** 4](#_Toc135473528)

[**Обращение к базе данных** 4](#_Toc135473529)

[**Структура приложения** 4](#_Toc135473530)

[**Описание методов** 6](#_Toc135473531)

[**average** 6](#_Toc135473532)

[**describe\_artist** 7](#_Toc135473533)

[**corr\_matrix** 8](#_Toc135473534)

[**top\_tracks** 8](#_Toc135473535)

[**artist\_evolution** 9](#_Toc135473536)

[**genre\_evolution** 10](#_Toc135473537)

[**average\_artists\_popularity** 11](#_Toc135473538)

[**count\_of\_tracks** 11](#_Toc135473539)

[**polar\_graph\_for\_all** 12](#_Toc135473540)

[**polar\_graph\_for\_track** 13](#_Toc135473541)

[**Обращение к программе** 13](#_Toc135473542)

# **Введение**

Данное руководство предназначено для программистов редактирующих исходный код приложения для анализа музыкальных композиций из Spotify.

Руководство разработчика для Приложения для анализа музыкальных композиций из Spotify содержит информацию о том, как использовать приложение для получения аналитических данных о музыкальных композициях, хранящихся в Spotify. В руководстве описываются требования к окружению и установке приложения, а также инструкции по запуску и использованию приложения.

Далее в руководстве предоставляются инструкции по использованию функций приложения, включая поиск композиций, отображение аналитических данных и визуализацию полученных результатов. Руководство также включает описание готовой базы данных, хранящейся в Spotify, и инструкции по взаимодействию с ней.

# **Характеристики программы**

## **Описание необходимого окружения**

Для работы приложения необходим дистрибутив Anaconda и встроенные в нее библиотеки.

Anaconda **—** это свободно распространяемый дистрибутив программного обеспечения для языков программирования Python и R, который включает в себя большое количество научных и аналитических библиотек, инструментов для работы с данными и визуализации.

## **Установка и запуск**

1. Установить Anaconda или Miniconda на свой компьютер, если они еще не установлены.
2. Открыть командную строку или терминал, в зависимости от операционной системы.
3. Открыть поиск и ввести “Anaconda Prompt”, нажать на результат поиска.
4. С помощью команды *cd* войти в директорию /work/scripts.
5. Запустить приложение, используя команду  
   *python main.py.*

## **Обращение к базе данных**

Обращение к базе данных происходит при помощи библиотеки pandas.

Чтение базы данных из .csv файла происходит при помощи функции pd.read\_csv(“Путь файла”) (pd – общепринятое сокращение от pandas)

Обращение к элементам базы данных может происходить по индексам (как в стандартном массиве) либо при помощи функции loc, если необходимо получить треки с определенными параметрами.

## **Структура приложения**

├───data

│ │ SpotifyFeaturesLast.csv – база данных

│ │

│ ├───deprecated

│ │ SpotifyFeaturesWithYears.csv – устаревшая база данных

│ │

│ ├───generator

│ │ generator.py – парсер годов для база данных

│ │ result.csv – вывод парсера

│ │

│ └───source

│ SpotifyFeatures.csv – исходная база данных (обрабатывалась пасрером)

│

├───graphics

│ main.ui – qt designer интерфейс

│ ui\_make.py – скрипт, переводящий .ui в .py

│

├───images

│ ic\_launcher.png – иконка приложения

│

├───library

│ ├───analysis

│ │ │ analysis\_methods.py – файл с функциями анализа

│ │ │

│ │ └───\_\_pycache\_\_

│ │ analysis\_methods.cpython-310.pyc

│ │

│ ├───ui

│ │ │ MainWindow.py – главное окно приложения

│ │ │ PlotDialog.py - диалог

│ │ │

│ │ └───\_\_pycache\_\_

│ │ MainWindow.cpython-310.pyc

│ │ PlotDialog.cpython-310.pyc

│ │

│ └───\_\_pycache\_\_

├───notes

│ README.md

│ requirements.txt – файл с зависимостями

│ Руководство пользователя.docx

│ Руководство разработчика.docx

│ ТЗ.docx

│

├───output – сохраняемые графики

│ .png

│ artist\_parameter\_evolution.png

│ average.png

│ correlation.png

│ genre\_evolution.png

│ hist.png

│ polar\_graph.png

│ polar\_graph\_for\_all.png

│

└───scripts

│ main.py – главный файл

│

│

└───\_\_pycache\_\_

## **Описание методов**

В данном разделе приведено описание методов анализа, которые реализованы в файле analysis\_methods.py.

### **average**

Обращение:

**average** (*param\_key*, *param\_number*, *data\_frame*, *graph\_type*, *color*=” r”)

Описание:

Данная функция строит график зависимости ключевого параметра от среднего значения количественного параметра.

Параметры:

* *param\_key* –
  + **тип**: строка;
  + **описание**: качественный параметр, по которому строится зависимость, откладывается по оси x.
* *param\_number* –
  + **тип**: строка;
  + **описание**: количественный параметр, по которому строится график, откладывается по оси y.
* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.
* *graph\_type* –
  + **тип**: строка;
  + **описание**: тип графика, который должен быть построен.
* *color* –
  + **тип**: строка;
  + **описание**: цвет графика;
  + **значение по умолчанию**: r (red – красный).

Выходные данные:

График, заданного типа, построенный по заданным пользователем параметрам, который выводится на экран и сохраняется в папку output.

### **describe\_artist**

Обращение:

**describe\_artist**(*name*, *param*, *data\_frame*)

Описание:

Функция, которая описывает параметр артиста.

Параметры:

* *name* –
  + **тип**: строка;
  + **описание**: имя артиста.
* *param* –
  + **тип**: строка;
  + **описание**: параметр, по которому описывается артист.
* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.

Выходные данные:

Словарь с описанием параметра артиста.

### **corr\_matrix**

Обращение:

**corr\_matrix**(*data\_frame*)

Описание:

Функция, которая строит матрицу корреляций параметров базы данных.

Параметры:

* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.

Выходные данные:

Матрица корреляций, которая выводится на экран и сохраняется в папку output.

### **top\_tracks**

Обращение:

**top\_tracks**(*data\_frame*, *key*, *count*, *year\_from*= -1, *year\_to*= -1, *reverse*= False)

Описание:

Функция для получения топа заданного количества треков по определенному параметру и за определенный промежуток времени.

Параметры:

* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.
* key –
  + **тип**: строка;
  + **описание**: параметр, по которому составляется топ.
* count –
  + **тип**: целочисленный;
  + **описание**: количество треков, которые войдут в топ.
* year\_from –
  + **тип**: целочисленный;
  + **описание**: год, с которого составляется топ;
  + **значение по умолчанию**: -1.
* year\_to –
  + **тип**: целочисленный;
  + **описание**: год, до которого составляется топ;
  + **значение по умолчанию**: -1.
* reverse –
  + **тип**: булевый;
  + **описание**: отвечает, нужно ли делать топ по возрастанию или убыванию;
  + **значение по умолчанию**: False (по убыванию).

Выходные данные:

Массив из словарей, ключами которых являются “artist\_name”, “track\_name”, “genre” и параметр key.

### **artist\_evolution**

Обращение:

**artist\_evolution**(*artsist*, *date\_frame*, *param*)

Описание:

Функция, которая строит график какого-либо параметра заданного артиста во времени (по годам либо по датам выхода песен, если их небольшое количество).

Параметры:

* *artist* –
  + **тип**: строка;
  + **описание**: имя артиста.
* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.
* *param* –
  + **тип**: строка;
  + **описание**: параметр, по которому строится график.

Выходные данные:

График эволюции артиста по параметру. По оси X откладывается дата (год либо дата выпуска), по оси Y среднее значение параметра в эту дату (год либо дату выпуска).

### **genre\_evolution**

Обращение:

**genre\_evolution**(*genre*, *data\_frame*)

Описание:

Функция, которая строит график средней популярности жанра за время выпуска музыкальных композиций в этом жанре.

Параметры:

* *genre* –
  + **тип**: строка
  + **описание**: жанр, по которому строится график
* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.

Выходные данные:

График эволюции жанра. По оси X откладывается год, по оси Y – средняя популярность жанра за этот год.

### **average\_artists\_popularity**

Обращение:

**average\_artists\_popularity**(*data\_frame*)

Описание:

Функция для получения средних значений популярности артистов

Параметры:

* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.

Выходные данные:

Словарь, ключом которого является имя артиста, значением – среднее значение популярности

### **count\_of\_tracks**

Обращение:

**count\_of\_tracks**(*param*, *bins*, *data\_frame*, *color*=”r”)

Описание:

Функция для построения гистограммы распределения количества треков по определенному параметру.

Параметры:

* *param* –
  + **тип**: строка;
  + **описание**: параметр, по которому строится распределение.
* *bins* –
  + **тип**: целочисленный;
  + **описание**: количество колонок на гистограмме.
* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.
* *color* –
  + **тип**: строка;
  + **описание**: цвет столбцов гистограммы;
  + **значение по умолчанию**: r (red – красный).

Выходные данные:

Гистограмма распределения по параметру. По оси X откладывается параметр, заданный пользователем, по оси Y – количество музыкальных композиций, соответствующих данному значению параметра.

### **polar\_graph\_for\_all**

Обращение:

**polar\_graph\_for\_all**(*data\_frame*)

Описание:

Функция для построения полярного графика для 100 самых популярных треков и для всех треков.

Параметры:

* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.

Выходные данные:

Полярный график. По окружности откладываются названия параметров, по радиусам к параметрам – среднее значения этих параметров, соединенных между собой линиями. Синим цветом выводится полярный график для 100 самых популярных музыкальных композиций, красным – для всех музыкальных композиций.

### **polar\_graph\_for\_track**

Обращение:

**polar\_graph\_for\_track**(*track\_id*, *data\_frame*, *track\_name*)

Описание:

Функция для построения полярного графика для определенного трека

Параметры:

* *track\_id* –
  + **тип**: строка;
  + **описание**: id музыкальной композиции.
* *data\_frame* –
  + **тип**: DataFrame;
  + **описание**: база данных, прочитанная при помощи библиотеки pandas.
* *track\_name* –
  + **тип**: строка;
  + **описание**: имя музыкальной композиции.

Выходные данные:

Полярный график. По окружности откладываются названия параметров, по радиусам к параметрам – значения этих параметров, соединенных между собой линиями.

# **Обращение к программе**

Обращение к программе происходит при помощи интерфейса (его использование описано в Руководстве пользователя). Интерфейс реализован при помощи библиотеки PyQT. Файл “main.ui” представляет собой раскладку интерфейса, выполненную в QT designer. Выполняя “ui\_make.py”, файл с раскладкой преобразуется в “MainWindow.py”, который, в свою очередь, вызывается в запускном файле “main.py”.