

# Playing Around Before The Party Starts



## РАЗРАБОТКА НА ПРОЕКТ

STRYPESLAB COURSE 2023

### Тема:

*Проект за Управление, Прехвърляне и Споделяне на Плейлисти между  
Spotify и Deezer*

Изготвил:

*Мартин Димитров Христов*

Версия на проекта:

*0.1*

Септември  
2023

# Съдържание

|   |           |
|---|-----------|
| <b>Съдържание.....</b>  | <b>2</b>  |
| <b>Увод.....</b>  | <b>3</b>  |
| <b>Първа глава - Изисквания към програмния продукт и избор на средства за проектиране и реализация.....</b> | <b>3</b>  |
| 1.1 Функционални изисквания.....  | 3         |
| 1.1.2 Прехвърляне на Плейлисти.....   | 3         |
| 1.1.3 Споделяне на Плейлисти.....   | 3         |
| 1.1.4 Преглед на Всички Плейлисти.....  | 4         |
| 1.1.5 Добавяне на Песни към Плейлисти.....  | 4         |
| 1.1.6 Премахване на Песни от Плейлисти.....   | 4         |
| 1.1.7 Създаване на Плейлисти.....   | 4         |
| 1.1.8 Премахване на Плейлисти.....  | 4         |
| 1.1.9 Анализ на Плейлисти.....  | 4         |
| 1.1.10 Препоръки за Нови Песни.....   | 4         |
| 1.2 Избор на средства за изграждане на приложението.....  | 4         |
| 1.2.1 Потребителски интерфейс.....  | 4         |
| 1.2.3 Бази данни.....   | 5         |
| 1.2.4 Програмен език.....   | 5         |
| 1.3 Основни структури в приложението.....   | 5         |
| 1.3.1 Обекти.....   | 5         |
| 1.3.2 Диаграма на взаимодействие между обектите.....  | 6         |
| <b>Втора глава - Реализация на програмното задание.....</b>   | <b>6</b>  |
| 2.1 Архитектура на приложението.....  | 6         |
| 2.2 Основни моменти при реализацията.....   | 7         |
| 2.2.1 Система за автентикация.....  | 7         |
| 2.2.3 Автентикация с Deezer и Spotify.....  | 8         |
| 2.2.5 Проверка за изтекли токени на музикалните платформи.....  | 10        |
| 2.2.5 Прехвърляне на плейлисти от една платформа на друга.....  | 11        |
| 2.2.2 Система за създаване на Token за споделяне на Playlist.....   | 12        |
| 2.2.5 Търсене в дълбочина на песни.....   | 14        |
| 2.2.5.1 Spotify.....  | 14        |
| 2.2.5.2 Deezer.....   | 14        |
| <b>Трета глава - Ръководство на потребителя.....</b>  | <b>15</b> |
| 3.1 Инсталация.....   | 15        |
| 3.2 Изисквания.....   | 15        |
| 3.3 Основни екрани.....   | 15        |
| 3.3.1 Начален екран (/).....  | 15        |
| 3.3.2 Автентикация (/login).....  | 16        |
| 3.3.3 Всички плейлисти (/playlists).....  | 16        |
| 3.3.4 Единичен плейлист (/view).....  | 16        |
| 3.3.5 Профил (/profile).....  | 17        |
| 3.3.6 Препоръки за нови песни (/recommendations).....   | 17        |
| <b>Заклучение и бъдещо развитие.....</b>  | <b>18</b> |
| <b>Използвани термини.....</b>  | <b>19</b> |

# Увод

21 век е епоха в която услугите за стрийминг на музика са се превърнали в неразделна част от ежедневието на всеки човек. Управлението и споделянето на плейлисти между различни платформи е обичайно желание сред музикалните ентусиасти.

Услугите за стрийминг на музика като Spotify и Deezer преобразяват начина, по който човекът има достъп и се наслаждава на музиката. Тези платформи предлагат огромни библиотеки от песни, персонализирани плейлисти и препоръки, които отговарят на индивидуалните предпочитания. Въпреки това, едно ограничение, с което се сблъскват потребителите, е трудността да управляват своите плейлисти и да ги споделят в различни услуги.

Проектът има за цел да предостави на потребителите инструмент за контрол над тяхната музикална колекция, който да преодолее границите между различните музикални платформи. Този уеб базиран софтуер предоставя редица ключови функционалности, като управление на всички плейлисти на едно място, изтриване, добавяне, търсене, откриване на нови песни и дори персонална статистика за любими изпълнители и жанрове.

## Първа глава - Изисквания към програмния продукт и избор на средства за проектиране и реализация

### 1.1 Функционални изисквания

#### 1.1.2 Прехвърляне на Плейлисти

Една от ключовите функционалности на проекта е възможността за прехвърляне на плейлисти между различни платформи като Spotify и Deezer. Това позволява на потребителите да си запазят своята любима музика и при преход от една услуга към друга.

#### 1.1.3 Споделяне на Плейлисти

Проектът прави музикалното споделяне много по-лесно. Потребителите могат да споделят своите плейлисти с приятели и семейство, независимо от това на коя платформа са създадени. Това създава възможност за обмен на музикални вкусове и откриване на нова музика.

#### 1.1.4 Преглед на Всички Плейлисти

Уеб приложението предоставя удобен и централизиран начин за преглед на всички плейлисти от различни музикални платформи.

#### 1.1.5 Добавяне на Песни към Плейлисти

За допълнително удобство, приложението позволява да се добавят нови песни към настоящи плейлисти. Това става чрез интуитивен интерфейс, който позволява да се търси и добавя музика в съответния плейлист.

#### 1.1.6 Премахване на Песни от Плейлисти

Предоставена е възможност за премахване на песни от съществуващи плейлисти от различните платформи.

#### 1.1.7 Създаване на Плейлисти

Потребителят може да създава колекция от песни в избрана от него музикална платформа, като избира сам името на колекцията при създаване.

#### 1.1.8 Премахване на Плейлисти

Потребителят може да изтрива колекция от песни в избрана от него музикална платформа.

#### 1.1.9 Анализ на Плейлисти

Приложение предоставя възможност за анализ на съществуващи колекции с музика. Това включва извличане на информация за жанровете, които потребителят най-често слуша, и статистика за най-често слушаните артисти.

#### 1.1.10 Препоръки за Нови Песни

Приложението предоставя възможност за откриване на нова музика, според вкуса музика, която потребителят предпочита да слуша.

### 1.2 Избор на средства за изграждане на приложението

#### 1.2.1 Потребителски интерфейс

За потребителския интерфейс са използвани HTML, CSS и библиотеката Bootstrap v4.5.2. Използван е JavaScript за обработване

на данни изпратени от сървъра и опресняването на потребителския интерфейс, без да има нужда да се презарежда страницата.

### 1.2.3 Бази данни

За съхранение и управление на данните е използван SQLAlchemy. Това е мощен ORM (Обектно-релацион мапер), който улеснява взаимодействието между приложението и базата данни. Този инструмент е силно препоръчван и използван при разработването на уеб приложения с Flask.

### 1.2.4 Програмен език

Програмният език, използван за разработката на бекенд частта на приложението, е Python и Flask.

Flask, предоставя минималистичен и гъвкав начин за създаване на уеб приложения, което позволява фокусирането върху основните функционалности и бързото развитие на проекта.

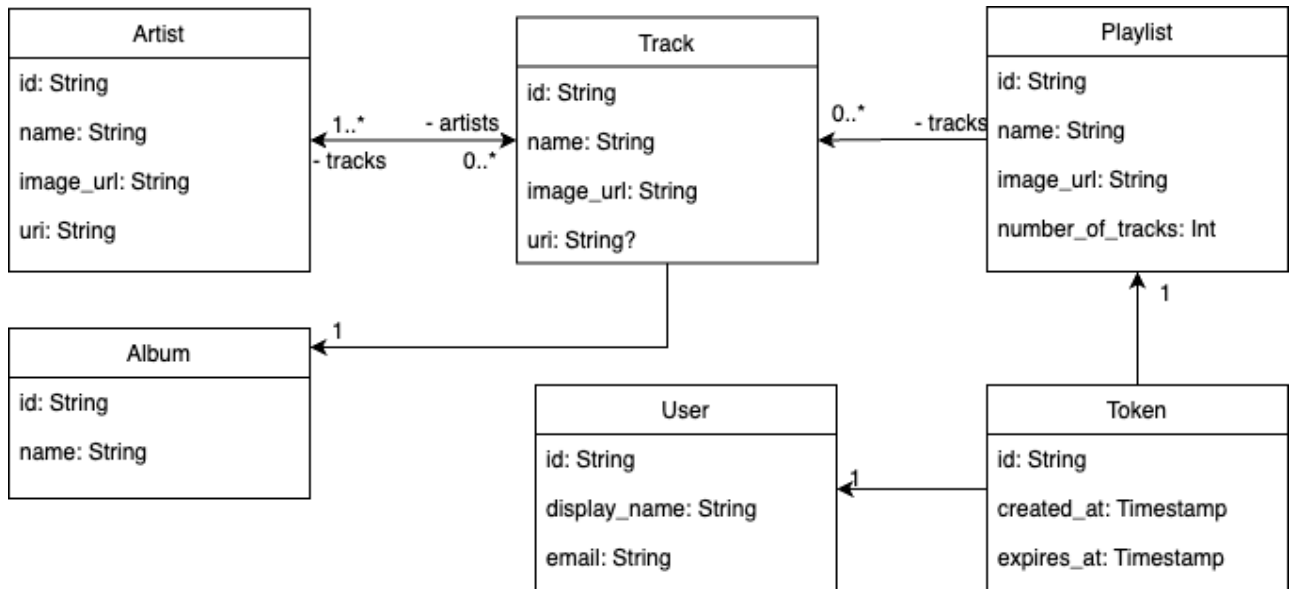
## 1.3 Основни структури в приложението

### 1.3.1 Обекти

Основните обекти, които ще се използват за представяне на информация са:

- **User** - който ще преглежда и управлява своите плейлисти
- **Artist** - който ще е част от изпълнителите на определен *Track*
- **Track** - който ще съхранява кои са изпълнителите му и на кой албум принадлежи
- **Album** - който ще е принадлежащия албум на даден *Track*
- **Playlist** - ще съхранява списък от различни обекти на *Track*
- **Token** - ще съхранява, кой го е създал, кой *Playlist* е бил споделен и кога изтича годността му

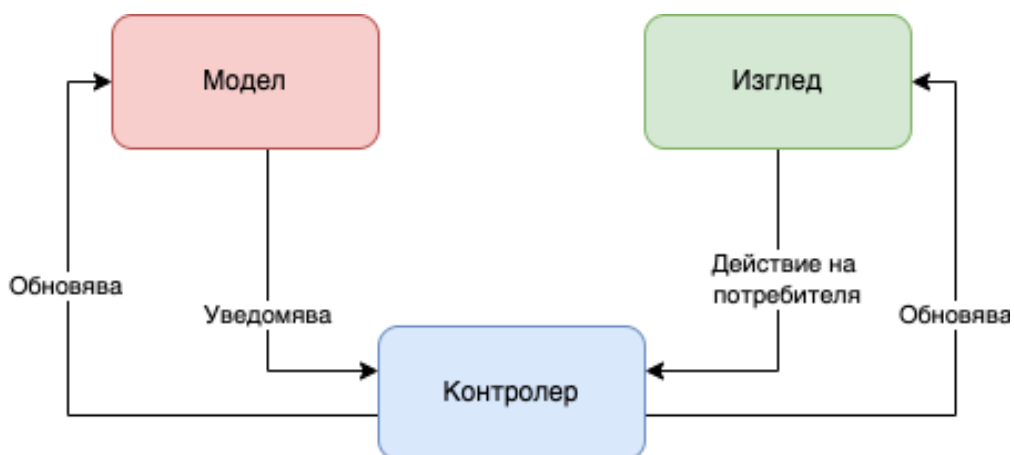
### 1.3.2 Диаграма на взаимодействие между обектите



## Втора глава - Реализация на програмното задание

### 2.1 Архитектура на приложението

Flask приложението е създадено съгласно архитектурния модел Model-View-Controller (Модел-Изглед-Контролер). Тази архитектура е изключително популярна при реализацията на уеб приложения и Flask не остава по-назад в тази статистика.




Основни елементи:

- **Модел** - централен компонент в шаблона. С прости думи - съдържа данните в приложението. Тук е посочена цялата информация, която е от съществено значение за показване на отделните обекти по отношение на достъпа или други валидации.
- **Изглед** - изходящият поток от информация (това, което приложението изпраща като отговор до дисплея, както и до потребителя, в следствие на неговата заявка). Възможни са няколко изгледа на една и съща информация.
- **Контролер** - третата част от този шаблон. Той е отговорен за предоставянето на данните, присъстващи в Изгледа и интерпретирането на потребителските действия.

## 2.2 Основни моменти при реализацията

### 2.2.1 Система за автентикация



```
1 from flask_login import LoginManager
2 from flask_login import login_required, logout_user, current_user
3
4 login_manager = LoginManager(app)
5
6 @login_manager.user_loader
7 def get_user(id):
8     return User.query.filter_by(id=id).first()
9
10 @login_manager.unauthorized_handler
11 def unauthorized():
12     return redirect(url_for('login'))
13
14 @app.route('/profile')
15 @login_required
16 def profile():
17
18 @app.route('/logout')
19 @login_required
20 def logout():
21     session.pop(TOKEN_INFO_SPOTIFY, None)
22     session.pop(TOKEN_INFO_DEEZER, None)
23     logout_user()
24     return redirect(url_for('index'))
```

- **LoginManager** се инициализира с Flask приложението, за да се справи със задачи, свързани с автентикацията на потребителя.

- **@login\_manager.user\_loader:** Това е декоратор, който регистрира функция (get\_user) с Flask-Login. Тази функция отговаря за зареждането на потребителски обект въз основа на уникалния идентификатор (id) на потребителя. Когато потребител влезе, Flask-Login ще извика тази функция, за да извлече потребителския обект, свързан с предоставения идентификатор.
- **@login\_manager.unauthorized\_handler:** Този декоратор регистрира неоторизираната функция като неоторизиран манипулатор. В случай, че потребител се опита да получи достъп до защитен ресурс, без да бъде удостоверен, Flask-Login ще го пренасочи към определената страница.
- **@login\_required:** Това е декоратор, който се използва за защита на маршрути в уеб приложението. Когато се прилага към маршрут, той гарантира, че само удостоверени потребители имат достъп до този него. Ако потребителят не се е автентикирал, той ще бъде пренасочен към страницата за автентикиране в приложението.
- **logout\_user():** Тази функция се предоставя от Flask-Login и се използва за излизане на потребителя от приложението. Той прекратява сесията на потребителя и го маркира като неавтентикиран.

Този код настройва необходимите компоненти за автентикиране и оторизиране на потребителите в приложението. Той позволява на потребителите да се автентикат, защитава маршрутите от неоторизиран достъп и осигурява сигурност в софтуера.

### 2.2.3 Автентикация с Deezer и Spotify

```

1 @app.route("/callback", methods=["GET"])
2 def callback():
3     code = request.args.get('code')
4     deezer_client = get_deezer_client(session_token_info=None)
5     access_token = deezer_client.fetch_token(code)
6     if access_token:
7         deezer_client = get_deezer_client(session_token_info=access_token)
8         user = deezer_client.get_current_user()
9         user_db = User.query.filter_by(email=user.email).first()
10        if user_db is None:
11            user_db = User(email=user.email, display_name=user.display_name, id=user.id)
12            db.session.add(user_db)
13            db.session.commit()
14        login_user(user_db)
15        expires_in_seconds = int(access_token.split("expires=")[1])
16        expiration_time = datetime.now(pytz.utc) + timedelta(seconds=expires_in_seconds)
17
18        session[TOKEN_DEEZER_EXPIRES_ON] = expiration_time
19        session[TOKEN_INFO_DEEZER] = access_token
20        return redirect(url_for('playlists'))
21    else:
22        return "Failed to retrieve access token."

```



```

1 @app.route('/redirect')
2 def redirect_page():
3     code = request.args.get('code')
4     spotify_client = get_spotify_client(session_token_info=None)
5     token_info = spotify_client.exchange_code_for_token(code)
6     if token_info:
7         spotify_client = get_spotify_client(session_token_info=token_info)
8         user = spotify_client.get_current_user()
9
10        user_db = User.query.filter_by(email=user.email).first()
11
12        if user_db is None:
13            user_db = User(email=user.email, display_name=user.display_name, id=user.id)
14            db.session.add(user_db)
15            db.session.commit()
16
17            login_user(user_db)
18            session[TOKEN_INFO_SPOTIFY] = token_info
19            return redirect(url_for('playlists'))
20        else:
21            return "Failed to retrieve access token."

```

- Получава се код за удостоверяване от платформата за автентикация.
- Този код се обменя за токен за достъп, който позволява на приложението да прави удостоверени заявки към Deezer и/или Spotify от името на потребителя.
- Извличат се подробности за потребителя от Deezer и/или Spotify, като неговия имейл и екранно име.
  - Ако потребителят не съществува в локалната база данни, създава се нов потребителски запис.
- Потребителят е автентикиран в системата и се задава времето за изтичане на токена за достъп от Deezer, който се съхранява заедно с токена за достъп в сесията.
- При Spotify токена, няма нужда да се смята и прави това, защото форматът, в който е получен токена, има поле, което казва кога токенът ще стане невалиден.

## 2.2.5 Проверка за изтекли токени на музикалните платформи

```
1 def check_for_expired_tokens():
2     token_spotify = session.get(TOKEN_INFO_SPOTIFY)
3     token_deezer = session.get(TOKEN_INFO_DEEZER)
4     print("Checking for expired tokens")
5
6     if token_spotify:
7         spotify_client = get_spotify_client(session_token_info=token_spotify)
8         session[TOKEN_INFO_SPOTIFY] = spotify_client.refresh_token_if_needed()
9     if token_deezer:
10        current_time = datetime.now()
11        d = str(session.get(TOKEN_DEEZER_EXPIRES_ON))
12        given_datetime = datetime.fromisoformat(d)
13        current_time = datetime.now(pytz.utc)
14        if current_time >= given_datetime:
15            print("Expired token")
16            session[TOKEN_INFO_DEEZER] = None
17        else:
18            print(given_datetime)
19            print("Token not expired")
```

Функцията, *check\_for\_expired\_tokens()*, отговаря за проверката дали токени за достъп до Spotify и Deezer са изтекли.

За Spotify:

- Ако токен за достъп на Spotify съществува в сесията, създава се екземпляр на клиент на Spotify, използвайки този токен.
- След това се проверява дали токена трябва да бъде обновен и го актуализира съответно в сесията.

За Deezer:

- Извлича се времето на изтичане на токена за достъп на Deezer, съхранен в сесията.
- Сравнява текущото време с времето на изтичане, за да определи дали токена е изтекъл.
- Ако токена е изтекъл, се отпечата съобщение и задава токена за достъп на Deezer в сесията на None, за да покаже, че вече не е валиден.
- Ако токена не е изтекъл, се отпечата времето на изтичане и съобщение, което показва, че токена все още е валиден.

## 2.2.5 Прехвърляне на плейлисти от една платформа на друга

```
1 @app.route('/transfer_playlist/<from_platform>/<to_platform>', methods=['POST'])
2 @login_required
3 def transfer_playlist_deezer_to_spotify(from_platform, to_platform):
4     check_for_expired_tokens()
5     playlist_id = request.json.get("playlist_id")
6     name = request.json.get("name")
7
8     token_info_spotify = session.get(TOKEN_INFO_SPOTIFY)
9     token_info_deezer = session.get(TOKEN_INFO_DEEZER)
10
11     playlist = None
12     tracks = None
13
14     if from_platform.lower() == 'spotify':
15         spotify_client = get_spotify_client(session_token_info=token_info_spotify)
16         tracks = spotify_client.get_tracks_in_playlist(playlist_id)
17     elif from_platform.lower() == 'deezer':
18         deezer_client = get_deezer_client(session_token_info=token_info_deezer)
19         tracks = deezer_client.get_tracks_in_playlist(playlist_id)
20
21     if to_platform.lower() == 'spotify':
22         spotify_client = get_spotify_client(session_token_info=token_info_spotify)
23         new_playlist_id = spotify_client.create_playlist(name)
24         spotify_client.add_tracks_to_playlist(tracks=tracks, playlist_id=new_playlist_id)
25         playlist = spotify_client.get_playlist(new_playlist_id)
26     elif to_platform.lower() == 'deezer':
27         deezer_client = get_deezer_client(session_token_info=token_info_deezer)
28         new_playlist_id = deezer_client.create_playlist(name)
29         deezer_client.add_tracks_to_playlist(new_playlist_id, tracks)
30         playlist = deezer_client.get_playlist_info(new_playlist_id)
31
32     data = \
33     {
34         'playlist_id': playlist.id,
35         'playlist_name': playlist.name,
36         'number_of_tracks': playlist.number_of_tracks,
37         'image_url': playlist.image_url,
38         'view_url': url_for('view', playlist_id=playlist.id, platform=to_platform.lower(), playlist_name=playlist.name),
39         'delete_url': url_for('delete', platform=to_platform.lower(), playlist_id=playlist.id),
40         'platform': to_platform.lower()
41     }
42     return jsonify(data=data), 200
```

**/transfer\_playlist/<from\_platform>/<to\_platform>**, позволява на потребителите да прехвърлят плейлисти между Spotify и Deezer, като извличат песни от една платформа и създават нов плейлист на друга.

- **check\_for\_expired\_tokens()**: Тази функция се извиква, за да провери дали токените за достъп за Spotify и Deezer са изтекли и да ги опресни, ако е необходимо.
- **playlist\_id** и **името** се извличат от JSON данните, изпратени в POST заявката. Тези параметри са от значение за идентифициране на изходния плейлист и именуване на новия плейлист на целевата платформа.
- В зависимост от платформата източник (**from\_platform**), се извличат песните от посочения плейлист с помощта на подходящия клиент (Spotify или Deezer).

- В зависимост от целевата платформа (*to\_platform*), се създава нов плейлист с посоченото име на целевата платформа и добавя извлечените песни към него.
- Накрая се връща JSON отговор, съдържащ подробности за прехвърления плейлист, включително неговия идентификатор, име, брой песни, URL адрес на изображение, URL адрес за преглед, URL адрес за изтриване и целевата платформа.

### 2.2.2 Система за създаване на *Token* за споделяне на *Playlist*

Функцията *generate\_token()* улеснява създаването на споделени плейлисти чрез извличане и организиране на песни от Spotify или Deezer, създаване на плейлисти, токени и свързани записи в базата данни, и предоставяне на потребителя URL за достъп до споделения плейлист.

- Функцията извлича параметрите: „playlist\_id“, „playlist\_name“ и „platform“. Те показват, кой плейлист ще бъде споделян, от коя платформа е и какво е неговото име.
- Работа с извличането на плейлист:
  - Ако 'playlist\_id' и 'playlist\_name' не са предоставени, функцията пренасочва потребителя към страницата 'playlists', което показва, че липсва важна информация.
  - В зависимост от параметъра „платформа“ („spotify“ или „deezer“), се извличат песните в посочения плейлист от Spotify или Deezer, като използва съответните клиентски функции.
- Създаване на плейлист:
  - Създава се нов обект „Плейлист“ с уникален идентификатор.
  - След това кодът преминава през извлечените песни и проверява дали те вече съществуват в базата данни. Ако песен съществува, тя добавя изпълнителя, албума и песента към списъка за изпълнение.
  - Ако дадена песен не съществува в базата данни, тя създава нова песен, албум и записи на изпълнител, ако е необходимо, и ги добавя към списъка за изпълнение.
- Създаване на токен:
  - Създава се нов обект „Token“ с уникален идентификатор, кога е бил създаде и кога ще изтече (обикновено 1 ден след създаването), препратка към създадения плейлист и идентификационен номер на текущия потребител.
- Извършване на промени: Промените, направени в базата данни, включително новия списък за изпълнение, токен и свързаните записи се потвърждават след извикване на *commit()*

- Генериране на URL за споделяне:
  - URL за споделяне се изгражда въз основа на URL адреса на домейна и новосъздадения идентификатор на токена.

```

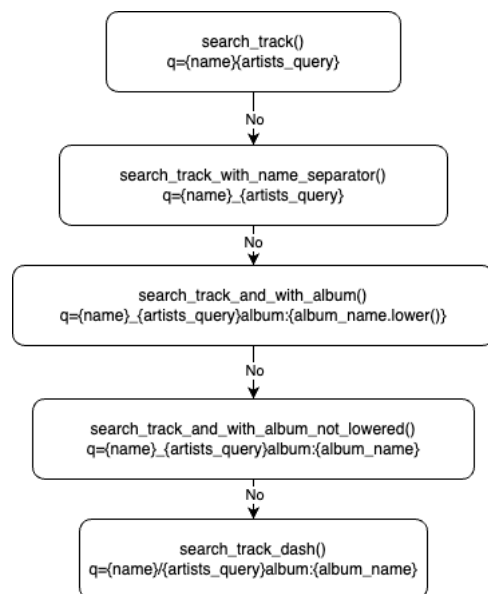
1  @app.route('/generate_token', methods=['GET', 'POST'])
2  @login_required
3  def generate_token():
4      check_for_expired_tokens()
5      playlist_id = request.args.get('playlist_id')
6      playlist_name = request.args.get('playlist_name')
7      platform = request.args.get('platform')
8      token_info_deezer = session.get(TOKEN_INFO_DEEZER)
9      token_info_spotify = session.get(TOKEN_INFO_SPOTIFY)
10
11     if not playlist_id or not playlist_name:
12         return redirect(url_for('playlists'))
13     tracks = []
14     if token_info_spotify and platform.lower() == "spotify":
15         spotify_client = get_spotify_client(token_info_spotify)
16         tracks = spotify_client.get_tracks_in_playlist(playlist_id=playlist_id)
17     elif token_info_deezer and platform.lower() == "deezer":
18         deezer_client = get_deezer_client(token_info_deezer)
19         tracks = deezer_client.get_tracks_in_playlist(playlist_id=playlist_id)
20
21     playlist = Playlist(name=playlist_name, id=str(uuid.uuid4()))
22
23     for track in tracks:
24         existing_track = Track.query.filter_by(name=track.name).first()
25         if existing_track:
26             for artist in existing_track.artists:
27                 artist = Artist.query.filter_by(name=artist.name).first()
28                 if not artist:
29                     artist = Artist(id=str(uuid.uuid4()), name=artist.name)
30                     db.session.add(artist)
31
32                 if artist not in existing_track.artists:
33                     existing_track.artists.append(artist)
34
35                 existing_track.album_id = track.album.id
36
37                 if existing_track not in playlist.tracks:
38                     playlist.tracks.append(existing_track)
39         else:
40             album = Album.query.get(track.album.id)
41             if not album:
42                 album = Album(id=track.album.id, name=track.album.name)
43                 db.session.add(album)
44
45             new_track = Track(id=track.id, name=track.name, album=album, uri=track.uri, image_url=track.image_url)
46             for a in track.artists:
47                 artist = Artist.query.filter_by(name=a.name).first()
48                 if not artist:
49                     artist = Artist(id=str(uuid.uuid4()), name=a.name)
50                     db.session.add(artist)
51
52                 new_track.artists.append(artist)
53                 if new_track not in playlist.tracks:
54                     playlist.tracks.append(new_track)
55                 db.session.add(new_track)
56
57     db.session.add(playlist)
58
59     token_id = str(uuid.uuid4())
60     created_at = datetime.utcnow()
61     expires_at = created_at + timedelta(days=1)
62     new_token = Token(
63         id=token_id,
64         created_at=created_at,
65         expires_at=expires_at,
66         playlist=playlist,
67         creator_id=current_user.id
68     )
69     db.session.add(new_token)
70
71     db.session.commit()
72     url = DOMAIN_URL + "shared/" + token_id
73
74     return render_template("shared_playlist_preview.html", token=new_token, url=url)

```

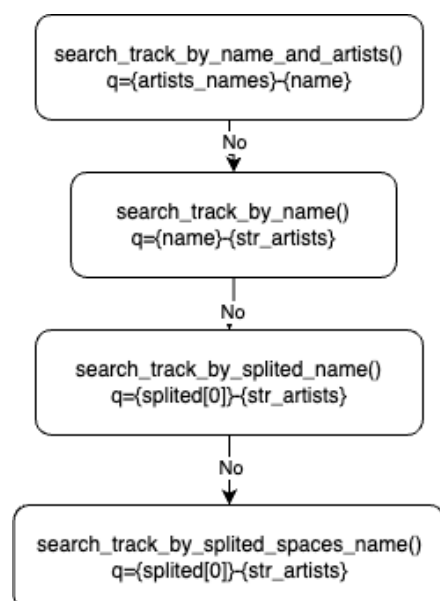
### 2.2.5 Търсене в дълбочина на песни

Поради различия в алгоритмите за търсене на различните музикални платформи, в този проект трябваше да се осъществят по-дълбоки търсения, когато потребителят търси точно определена песен от точно определени изпълнители. Различните показани заявки дават различни резултати при изпращане. Вероятно в бъдеще някои може да не работят, а дори може да не се стига до дъното при подобрене в API на платформите за музика

#### 2.2.5.1 Spotify



#### 2.2.5.2 Deezer



# Трета глава - Ръководство на потребителя

Самият проект и повече информация за него, може да намери [тук](#).

## 3.1 Инсталация

Командия за инсталация и пускане на кода (въведени в Terminal):

```
$ virtualenv env
$ source env/bin/activate
$ pip install -r requirements.txt
$ python3 app.py
```

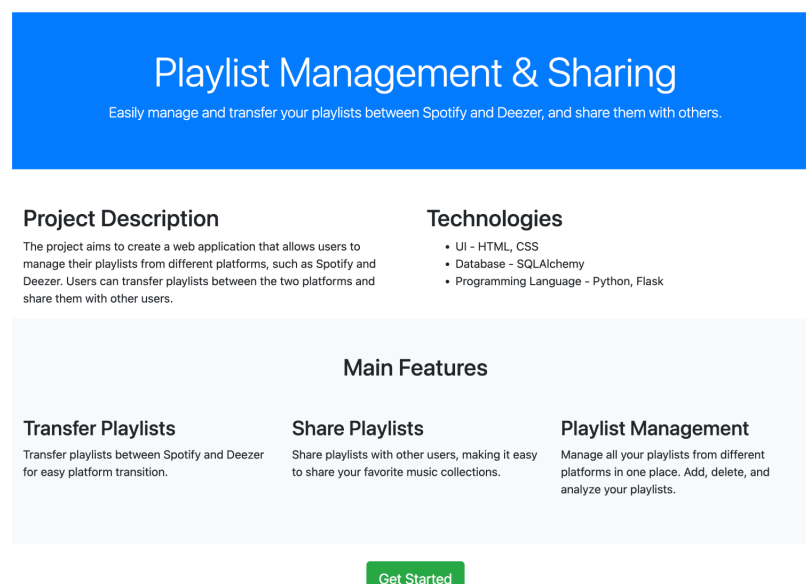
## 3.2 Изисквания

Python 3.x

Повече информация за изисквания и библиотеки, които използва проектът, можете да видите в [requirements.txt](#).

## 3.3 Основни екрани

### 3.3.1 Начален екран (/)



### 3.3.2 Автентикация (/login)

Login

Sign in with Spotify

Sign in with Deezer


### 3.3.3 Всички плейлисти (/playlists)

App Playlists Profile Recommendations Logout

Search playlists

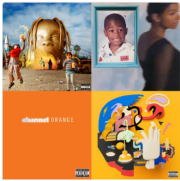
#### Playlists Spotify

New Playlist Name Create



Test Playlist  
Tracks: 6

Unfollow



Good life  
Tracks: 4

Unfollow

#### Playlists Deezer

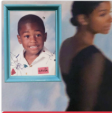
No Deezer playlists found.

Authorize Deezer

### 3.3.4 Единичен плейлист (/view)


App Playlists Profile Recommendations Logout

#### Good life




Crush  
Album: Crush  
Duckwrth

Remove




Colors and Shapes  
Album: Colors and Shapes  
Mac Miller

Remove



Pink Matter  
Album: channel ORANGE  
Frank Ocean , André 3000


Remove



STOP TRYING TO BE GOD


#### Add new songs

It was a good day Search




It Was A Good Day  
Album: The Predator  
Ice Cube

Add to Playlist



It Was A Good Day  
Album: Greatest Hits  
Ice Cube

Add to Playlist




It Was A Good Day - Remix  
Album: Bootlegs And B-Sides  
Ice Cube

Add to Playlist




### 3.3.5 Профил (/profile)

[App](#) [Playlists](#) [Profile](#) [Recommendations](#) [Logout](#)




## Hey, Martin Hristev



### Top 20 Artists Leaderboard

See who's rocking your charts!

|    |               |
|----|---------------|
| 1  | Mac Miller    |
| 2  | Kanye West    |
| 3  | JPEGMAFIA     |
| 4  | JAY-Z         |
| 5  | A\$AP Rocky   |
| 6  | Joey Bada\$\$ |
| 7  | Denzel Curry  |
| 8  | Doja Cat      |
| 9  | Travis Scott  |
| 10 | Metro Boomin  |
| 11 | Freddie Gibbs |
| 12 | Baby Keem     |
| 13 | BROCKHAMPTON  |




### Top 5 Genres Leaderboard

See which genres are your favorites!


|   |                     |    |
|---|---------------------|----|
| 1 | rap                 | 16 |
| 2 | hip hop             | 13 |
| 3 | pop rap             | 5  |
| 4 | underground hip hop | 3  |
| 5 | trap                | 3  |


### 3.3.6 Препоръки за нови песни (/recommendations)

[App](#) [Playlists](#) [Profile](#) [Recommendations](#) [Logout](#)




## Melodies Just for You







**The Hillbillies**  
Baby Keem , Kendrick Lamar  
The Hillbillies




**Think Fast (feat. Weezer)**  
Dominic Fike , Weezer  
Sunburn




**Fueled Up**  
Quavo  
Rocket Power



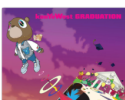
**EARFQUAKE**  
Tyler, The Creator  
IGOR




**Nosetalgia**  
Pusha T , Kendrick Lamar  
My Name Is My Name



**ICE COLD ZEL FREESTYLE [Feat. ICECOLDBISHOP]**  
Denzel Curry , ICECOLD BISHOP  
BLOOD ON MY NIKEZ



**Good Morning**  
Kanye West  
Graduation



**OK**  
Teezo Touchdown  
How Do You Sleep At Night?

## Заклучение и бъдещо развитие

Всяко едно от зададените функционални изисквания е изпълнено. Това съвсем не значи обаче, че приложението е напълно готово. Има елементи, в които програмният кодът може да бъде подобрен, както и потребителският интерфейс.

Една от целите, към които се стреми проектът, е той да бъде в помощ на повече любители на музиката. Функционалните изисквания припокриват едни солидни основи, а именно - интеграцията и управлението на плейлисти в две от най-известните платформи за стриймване на музика - Deezer и Spotify. Визуализацията, която е изключително важна за едно успешно приложение, позволява създаването на плейлисти, както и добавянето и отстраняването на песни от тях.

Бъдещата реализация включва функционалност, която интегрира още известни платформи като Apple Music, YouTube Music и други. Това би привлекло още повече музикални ентусиасти към приложението. Намирането на песни в различните платформи чрез търсене също може да бъде подобро за да може абсолютно всяка търсена песен да бъде намерена при търсене и успешно споделена от потребителите в различните платформи.

Голяма част от допълнителните функционалности, които не са част от предадените функционални изисквания, биха допринесли за довършването на поставените основи на услугата и създаването на един стабилен, завършен и полезен продукт, който да се използва от много хора.

## Използвани термини

1. Стрийминг - (от англ. streaming) метод за предаване на данни в реално време
2. Плейлист - (от англ. playlist) колекция, най-често с музика
3. Интерфейс - от англ. interface
4. Бекенд - от англ. back-end
5. Токен - от англ. token