

# Super Gerenciador Musical

## Sumário

<b>Sumário</b>	<b>1</b>
<b>1- Pré-configurações e execução da aplicação</b>	<b>3</b>
1.1 - Pré-configurações para o frontend	3
1.2 - Passos para a execução	3
<b>2 - Detalhando a Jornada do Usuário</b>	<b>4</b>
2.1 - Inicializando a aplicação	4
2.2 - Buscando e Adicionando Músicas	5
2.3 - Buscando Álbum, Playlists Públicas, Artistas e listando suas músicas	7
2.3.1 - Buscando Álbum e listando suas músicas	7
2.3.2 - Buscando Playlists Públicas e listando suas músicas	8
2.3.3 - Buscando Artista e listando suas músicas	9
2.4 - Playlists do usuário	10
2.4.1 - Listando Playlists do Usuário	10
2.4.2 - Criando Playlists	11
2.5 - Parâmetros das músicas em uma playlist	12
2.6 - Filtrando a busca de músicas	13
2.6.1 - Filtrando por tonalidade	13
2.6.2 - Filtrando por modo	14
2.6.3 - Filtrando por compasso	16
2.7 - Filtrando busca de músicas numa playlist	17
2.8 - Detalhamento de músicas	18
2.9 - Preview de música numa playlist do usuário	20
<b>3 - Arquitetura e Padrão de Projeto</b>	<b>21</b>
3.1 - Padrão de Projeto	21
3.2 - Arquitetura do sistema	21
3.2.1 - Diagrama de Componentes	21
3.2.2 - Diagrama de Classes	22
3.3 - Pacotes	22
3.3.1 Controllers	23
3.3.2 Models	23
<b>4 - Autenticação</b>	<b>24</b>
<b>5 - Organização do frontend</b>	<b>26</b>
<b>6 - Controllers</b>	<b>28</b>

<b>7 - Models e Services</b>	<b>30</b>
7.1 - Entities	30
7.2 - Services	30
Classe abstrata - Serviço Spotify	31
7.2.1 - Busca	31
Interface Serviço de Busca	31
7.2.1.1 Busca por ID	31
7.2.1.2 Busca por Tag	32
7.2.1.3 Listagem Músicas	33
7.2.2 - Filtragem	34
7.2.3 - Parâmetros	36
7.2.4 - Playlist Usuários	36
Adicionador de Músicas numa Playlist	37
Removedor de Músicas numa Playlist	37
Procurador de Playlists do Usuário Atual	38

# 1- Pré-configurações e execução da aplicação

Nossa aplicação é dividida em frontend e backend, o nosso backend é implementado utilizando o framework Spring Boot, com a linguagem Java, já o nosso frontend é implementado utilizando a biblioteca React com a linguagem Javascript.

## 1.1 - Pré-configurações para o frontend

Para executar o frontend, é necessária a utilização do npm, a versão utilizada é a 8.5.5, além disso, também é necessário a utilização do node cuja versão utilizada foi a 16.15.0, recomendamos que ambos sejam instalados em suas versões mais recentes. Com estes passos feitos, dentro da pasta supergerenciadormusical/front rode o comando *npm install*.

## 1.2 - Passos para a execução

Execute o backend, executando o arquivo SuperGerenciadorMusicalApplication.java como Spring Boot App, feito isso, rode o seguinte comando *npm run dev* na pasta supergerenciadormusical/front para executar o frontend.

Feito isso, abra o navegador e siga para “<http://localhost:3000/login>”. Nessa aba ocorre a autenticação, terminada a autenticação, por fim, abra uma nova aba e vá para <http://localhost:3000/>, onde todos os serviços da aplicação podem ser realizados.

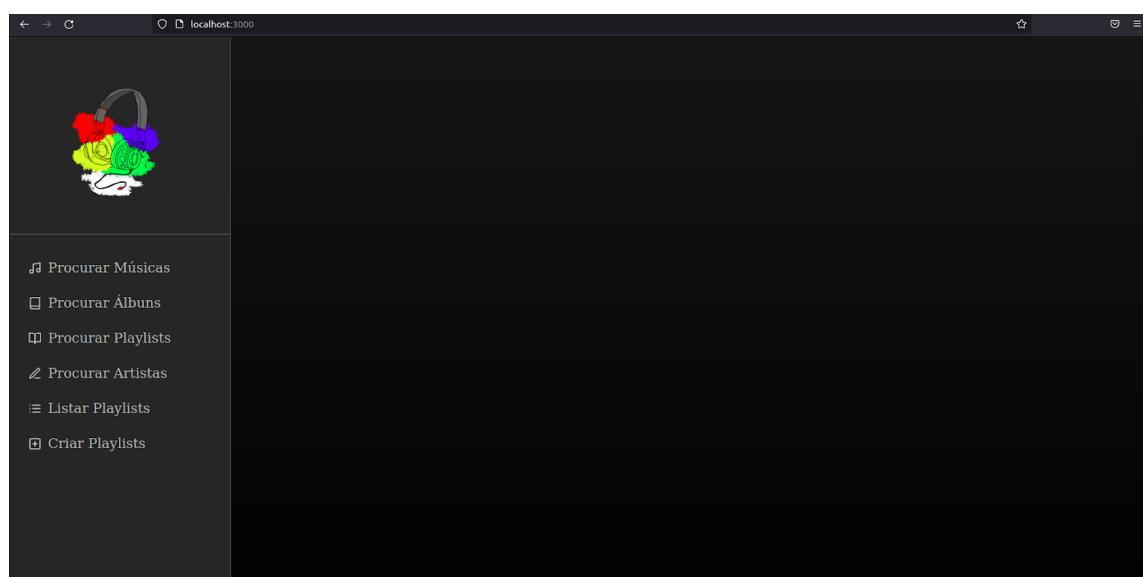
## 2 - Detalhando a Jornada do Usuário

### 2.1 - Inicializando a aplicação

Para rodar a aplicação, é necessário seguir até “<http://localhost:3000/login>”, assim como explicitado anteriormente. É importante ressaltar que o projeto do back já deve ter sido executado e estar funcionando.

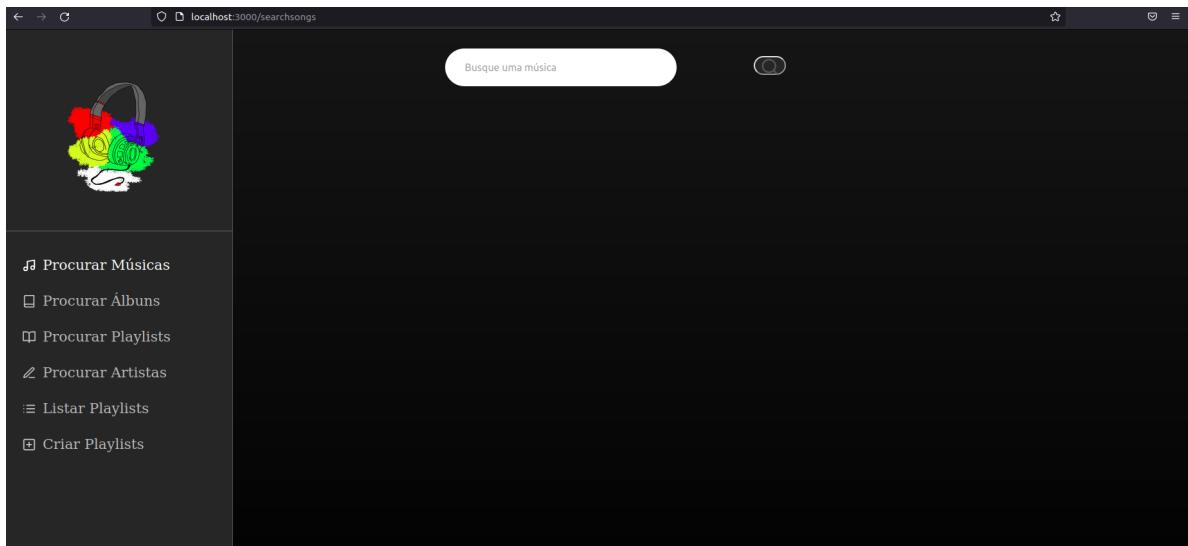


Em seguida, ao clicar em entrar, o usuário é redirecionado à tela do Spotify, e quando ele aceita, é direcionado a uma tela preta. Assim, ele deve abrir uma nova aba e seguir para “<http://localhost:3000/>”, a qual é a home da nossa aplicação.

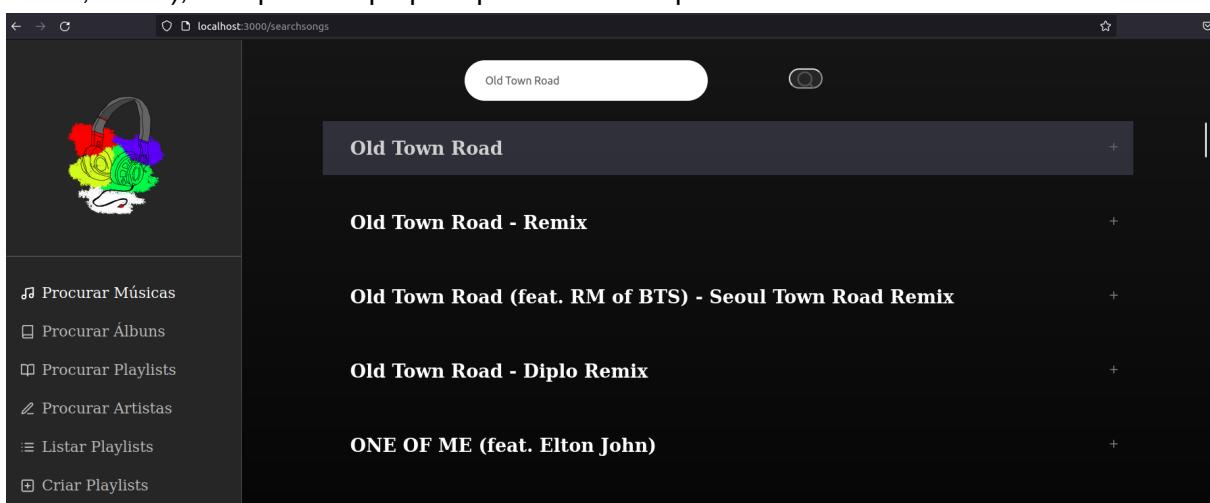


## 2.2 - Buscando e Adicionando Músicas

Clique em Procurar Músicas.



Digite o título da música que deseja procurar (você também pode informar seu artista, álbum, e etc.), e clique na lupa para procurar. Exemplo:



Para cada música, o usuário tem duas opções: clicar no nome e receber os parâmetros da música, ou clicar no ícone na lateral direita e eventualmente selecionar uma playlist para adicionar a música. No final da lista de parâmetros também há uma opção de adicionar a música em uma playlist.

Por exemplo, clicando no nome da música:

The screenshot shows a dark-themed user interface for a music application. On the left, there's a sidebar with a colorful headphones icon and a list of navigation options: 'Procurar Músicas', 'Procurar Álbuns', 'Procurar Playlists', 'Procurar Artistas', 'Listar Playlists', and 'Criar Playlists'. The main content area displays the title 'Old Town Road' above a small thumbnail image of a person riding a horse. Below the title, various song metrics are listed with their values: Danceability (0.907), DurationMs (113000), Energy (0.53), Instrumentalness (0.00000223), Key (1), and Liveness (0.101).

Navegando para o fim da lista de parâmetros:

This screenshot shows the same application interface after navigating to the end of the parameter list. The song title 'Old Town Road' and its metrics remain the same. However, new parameters are now visible: Speechiness (0.127) and TimeSignature (4). The sidebar and overall layout are identical to the previous screenshot.

Tanto ao clicar em “Adicionar música”, quanto ao clicar no ícone à direita em uma música buscada, levarão o usuário a tela onde é mostrado suas playlists, para ele selecionar a playlist em que a música será adicionada.

This screenshot shows a modal or separate window titled 'Selecione a playlist'. It lists several playlists available for selection: 'Private', 'Helloo,World', 'QualquerNome', and 'My Playlist #1'. The sidebar on the left is identical to the previous screenshots, showing the music application's navigation menu.

## 2.3 - Buscando Álbum, Playlists Públicas, Artistas e listando suas músicas

### 2.3.1 - Buscando Álbum e listando suas músicas

O procedimento de busca é análogo, clique em “Procurar Álbuns”, pesquise um título de álbum e clique na lupa para procurar, será listado os resultados da pesquisa, e selecione o álbum desejado clicando em seu nome.

The screenshot shows a dark-themed web application interface. On the left is a sidebar with a colorful headphones icon and a list of navigation items: Procurar Músicas, Procurar Álbuns, Procurar Playlists, Procurar Artistas, Listar Playlists, and Criar Playlists. The main content area has a search bar at the top with the placeholder "Abey Road". Below the search bar is a list of search results, each with a small thumbnail image and the album title: "Abbey Road (Remastered)", "Abbey Road (Super Deluxe Edition)", "Abbey Road", "Bob Baldwin Presents: Abbey Road and ...", "Love Goes: Live at Abbey Road Studios", and "At Abbey Road".

Após isso, suas músicas serão listadas:

The screenshot shows the same dark-themed application. The sidebar remains the same. The main content area now displays the songs from the selected album, "Abbey Road (Remastered)". It shows five songs with their titles and a plus sign (+) to the right of each: "Come Together - Remastered 2009", "Something - Remastered 2009", "Maxwell's Silver Hammer - Remastered 2009", "Oh! Darling - Remastered 2009", and "Octopus's Garden - Remastered 2009". Above the song list, there is a small thumbnail image of the Abbey Road cover and the album title "Abbey Road (Remastered)".

Você pode adicionar essas músicas em uma de suas playlists, fazendo o procedimento já explicado em 2.2.

### 2.3.2 - Buscando Playlists Públicas e listando suas músicas

O procedimento de busca é análogo, clique em “Procurar Playlists”, pesquise um título de playlist pública e clique na lupa para procurar, será listado os resultados da pesquisa, e selecione a playlist desejada clicando em seu nome

A screenshot of a web browser window titled "localhost:3000/searchplaylists". The search bar contains the query "Playlist chico buarque". Below the search bar, there is a list of search results:

- O Melhor de Chico Buarque** (with a small profile picture)
- This Is Chico Buarque** (highlighted in a dark blue box)
- Chico Buarque playlist** (with a small profile picture)
- Playlist: Chico Buarque** (with a small profile picture)
- Chico Buarque** (with a small profile picture)
- Chico Buarque** (with a small profile picture)

The left sidebar of the application shows navigation links:

- Procurar Músicas
- Procurar Álbuns
- Procurar Playlists
- Procurar Artistas
- Listar Playlists
- Criar Playlists

Após isso, suas músicas serão listadas

A screenshot of a web browser window showing the contents of the "This Is Chico Buarque" playlist. The title "This Is Chico Buarque" is displayed prominently at the top.

Below the title, there is a thumbnail image of Chico Buarque and a link labeled "Mostrar parâmetros".

The list of songs includes:

- Que Tal um Samba?**
- As Caravanas**
- Tua Cantiga**
- Paratodos**

The left sidebar of the application shows navigation links:

- Procurar Músicas
- Procurar Álbuns
- Procurar Playlists
- Procurar Artistas
- Listar Playlists
- Criar Playlists

Você pode adicionar essas músicas em uma de suas playlists, fazendo o procedimento já explicado em 2.2.

### 2.3.3 - Buscando Artista e listando suas músicas

O procedimento de busca é análogo, clique em “Procurar Artista”, pesquise um nome de artista e clique na lupa para procurar, será listado os resultados da pesquisa, e selecione o artista desejado clicando em seu nome

A screenshot of a web browser window titled "localhost:3000/searchartists". In the search bar, the text "Harry Styles" is entered. Below the search bar, a list of search results is displayed:

- Harry Styles** (highlighted)
- Harry Younger Styles**
- One Direction**
- Harry S Music**
- styles kelvin harry**
- Harry Stockwell**

The left sidebar contains navigation links:

- Procurar Músicas
- Procurar Álbuns
- Procurar Playlists
- Procurar Artistas
- Listar Playlists
- Criar Playlists

Após isso, suas músicas serão listadas

A screenshot of a web browser window showing the results for "Harry Styles". At the top, there is a profile picture of Harry Styles and the text "Harry Styles". Below this, a list of songs is displayed:

- As It Was** +
- Late Night Talking** +
- Matilda** +
- Music For a Sushi Restaurant** +
- Daylight** +

The left sidebar contains navigation links:

- Procurar Músicas
- Procurar Álbuns
- Procurar Playlists
- Procurar Artistas
- Listar Playlists
- Criar Playlists

Você pode adicionar essas músicas em uma de suas playlists, fazendo o procedimento já explicado em 2.2.

## 2.4 - Playlists do usuário

### 2.4.1 - Listando Playlists do Usuário

Ao clicar em “Listar Playlists”, é fornecido uma listagem com as playlists criadas pelo usuário; ele pode selecionar uma playlist para visualizar (clicando no nome) ou para remover (clicando no ícone na lateral direita); o usuário pode também clicar na opção de criar uma playlist e será direcionado para a tela de criação de playlists.

The image consists of two vertically stacked screenshots of a web-based music application interface, likely a Django project as indicated by the URL in the top screenshot.

**Top Screenshot (localhost:3000/listplaylists):**

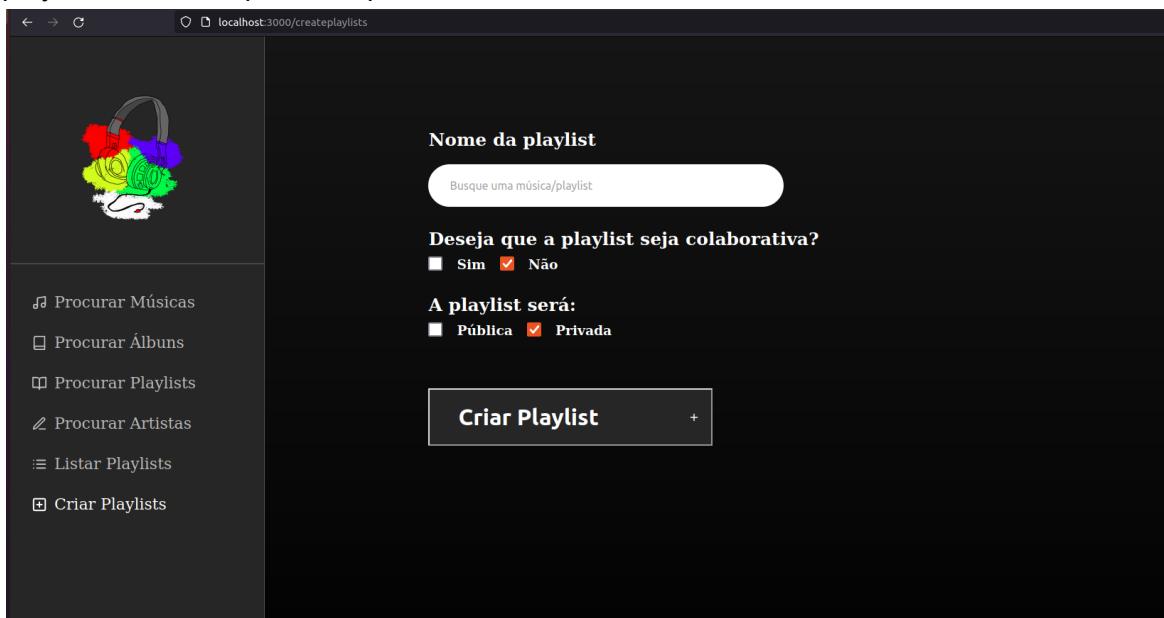
- Left Sidebar:** Includes icons for headphones, a microphone, and a search bar, followed by a navigation menu with links: Procurar Músicas, Procurar Álbuns, Procurar Playlists, Procurar Artistas, Listar Playlists, and Criar Playlists.
- Main Content:** A table titled "Playlists" showing four entries:
  - Private**: Associated with a user icon and a delete icon.
  - Helloo,World**: Associated with a user icon and a delete icon.
  - QualquerNome**: Associated with a user icon and a delete icon.
  - My Playlist #1**: Associated with a user icon and a delete icon.
- Bottom:** A button labeled "Criar Playlist" with a plus sign.

**Bottom Screenshot (localhost:3000/my\_playlist/1):**

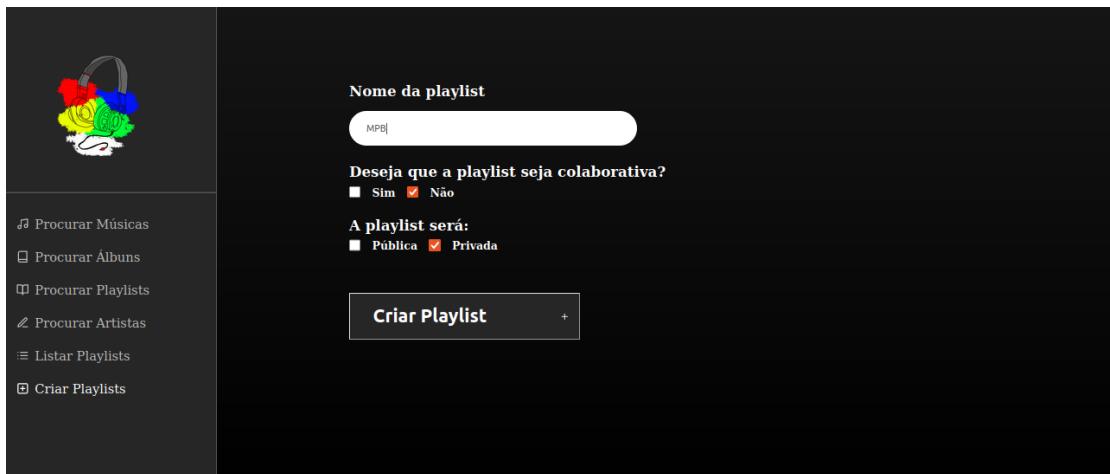
- Left Sidebar:** Same as the top screenshot.
- Main Content:** Details for "My Playlist #1":
  - AC/DC**: Associated with a user icon and a delete icon.
  - BACK IN BLACK**: Song title with a delete icon.
  - Mostrar parâmetros**: A link.
  - Back In Black**: Song title with a delete icon.
  - Highway to Hell**: Song title with a delete icon.
  - THATS WHAT I WANT**: Song title with a delete icon.

## 2.4.2 - Criando Playlists

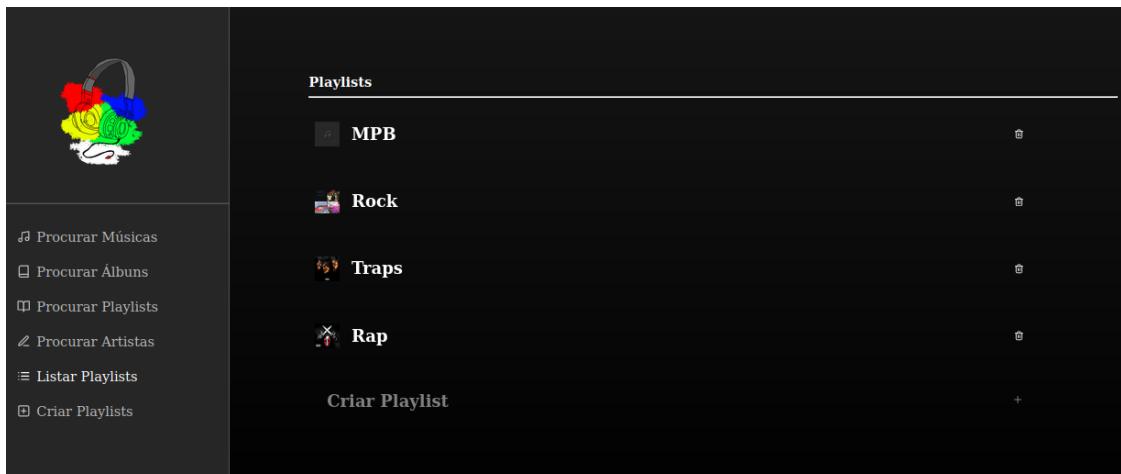
A criação de uma playlist ocorre com a passagem de três parâmetros: uma string para o nome e dois booleanos que indicam se ela será colaborativa ou não, pública ou privada. A playlist criada não pode ser pública e colaborativa simultaneamente.



Por exemplo, criando a Playlist MPB, privada e não colaborativa:



Agora, essa nova playlist foi criada:



## 2.5 - Parâmetros das músicas em uma playlist

Para uma playlist pública ou privada, há a opção “Mostrar parâmetros”:

The screenshot shows a Spotify interface. At the top, there is a small thumbnail of a colorful house and the playlist title 'Chilled Cow's Favorites'. Below the title, there is a button labeled 'Mostrar parâmetros' (Show parameters). Underneath this button, several song titles are listed: 'eternal.summer', 'emeralds/shower', '2020', and 'Tropical Storm'. Each song title has a '+' sign to its right, likely indicating more options.

Ao clicar nela, mostram-se os parâmetros ao quais podemos ordenar as músicas dessa playlist:

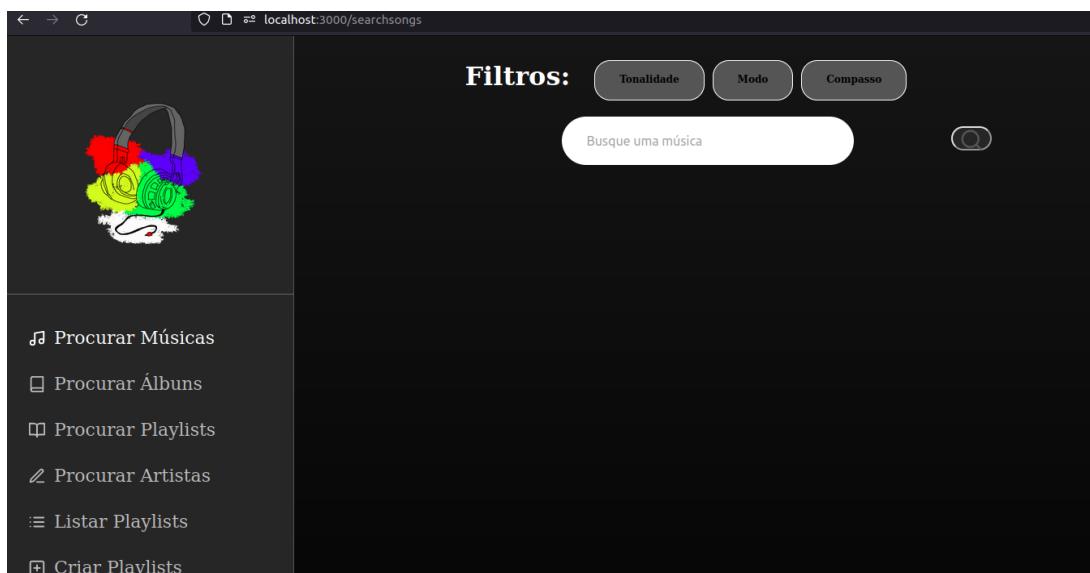
This screenshot shows the same Spotify playlist after the 'Show parameters' button was clicked. The interface now includes a row of ten buttons at the bottom, each representing a different audio feature: Acousticness, Danceability, DurationMs, Energy, Instrumentalness, Key, Liveness, Loudness, Speechiness, TimeSignature, Tempo, and Valence. The playlist title 'Chilled Cow's Favorites' and the song list are visible above these buttons.

Clicando em um desses parâmetros, as músicas serão ordenadas:

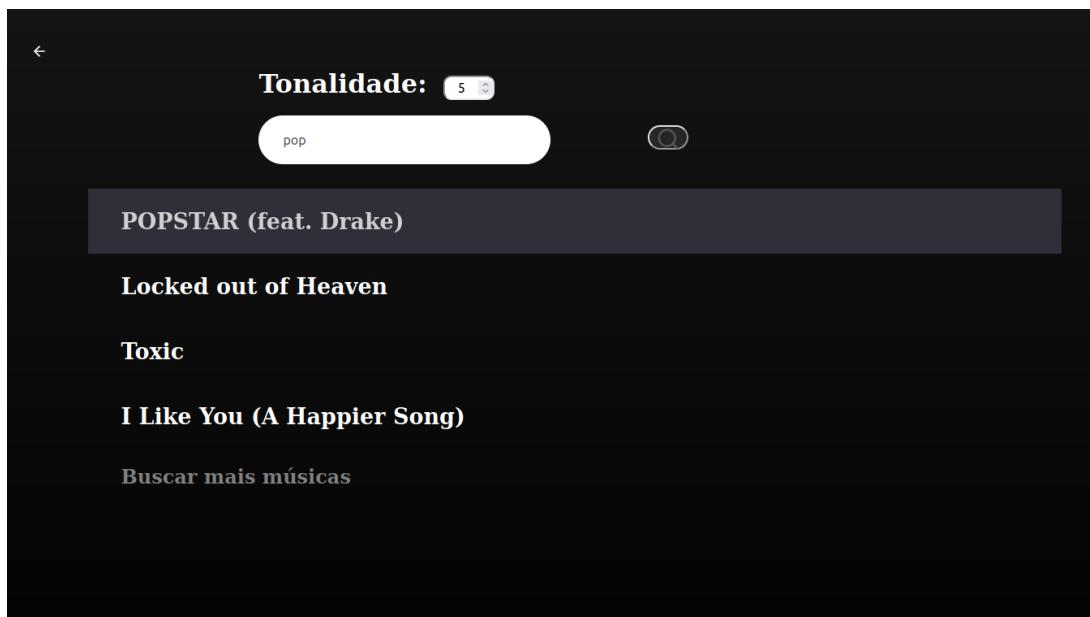
This screenshot shows the Spotify interface with the 'Energy' parameter selected for sorting. On the left, there is a sidebar with various navigation options. The main area displays the 'Chilled Cow's Favorites' playlist with four songs listed: 'eternal.summer' (Energy: 0.711), 'hishitsu' (Energy: 0.663), 'what once was a dream' (Energy: 0.64), and 'Herewego' (Energy: 0.589). The songs are ordered from highest energy to lowest energy.

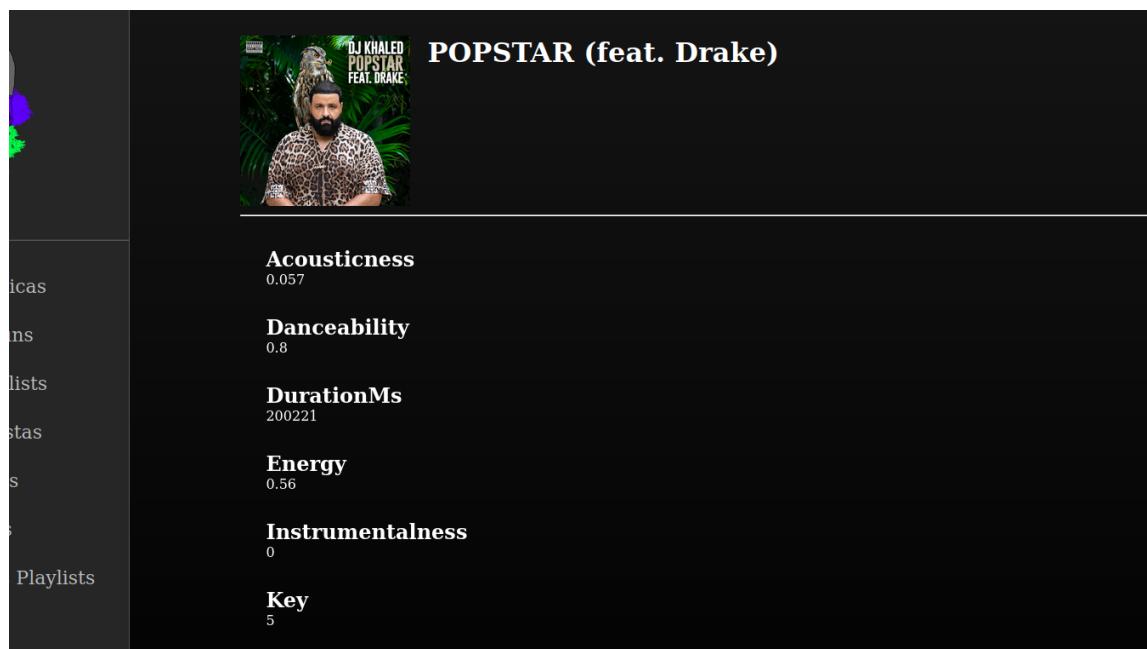
## 2.6 - Filtrando a busca de músicas

Na tela da busca de músicas, foram adicionados botões que permitem a filtragem da busca. Sendo possível filtrar a busca de acordo com os seguintes parâmetros: compasso, tonalidade e modo.



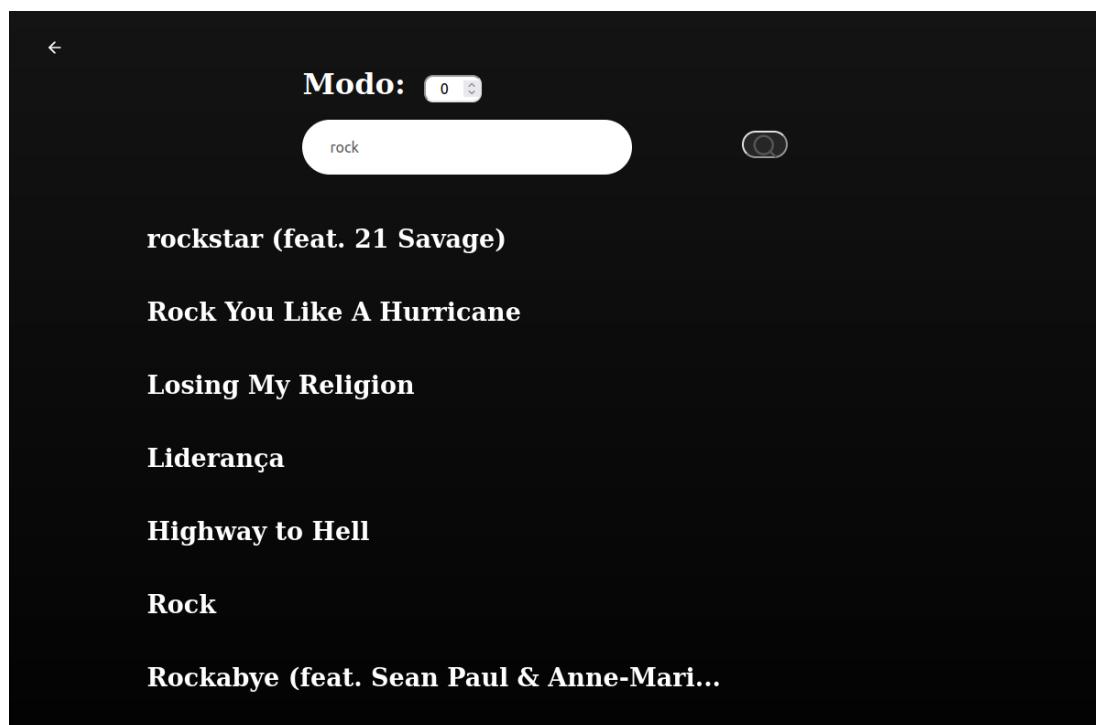
### 2.6.1 - Filtrando por tonalidade

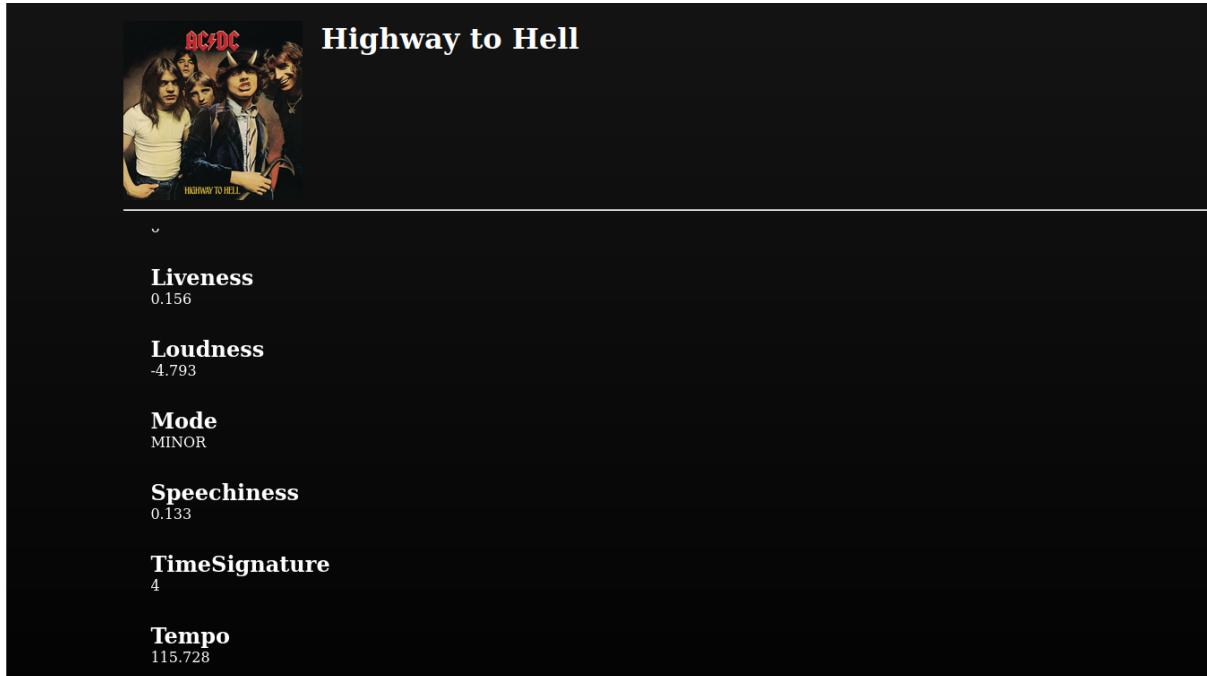




(Escolhendo a primeira música, POPSTAR, da busca; a tonalidade é representada pelo parâmetro Key)

## 2.6.2 - Filtrando por modo





(Escolhendo a quinta música da busca; o modo (mode) tem as opções MINOR e MAJOR, que são determinadas na aplicação por suas formas numéricas - 0 e 1, respectivamente)

## 2.6.3 - Filtrando por compasso

Compasso: 4

beatles

Beatles e Apostolos

Here Comes The Sun - Remastered 2009

Era um Garoto, Que Como Eu, Amava os ...

Eleanor Rigby

Come Together - Remastered 2009

Yellow Submarine

Let It Be - Remastered 2009



**Come Together - Remastered 2009**

0.0920

**Loudness**  
-11.913

**Mode**  
MINOR

**Speechiness**  
0.0393

**TimeSignature**  
4

**Tempo**  
165.007

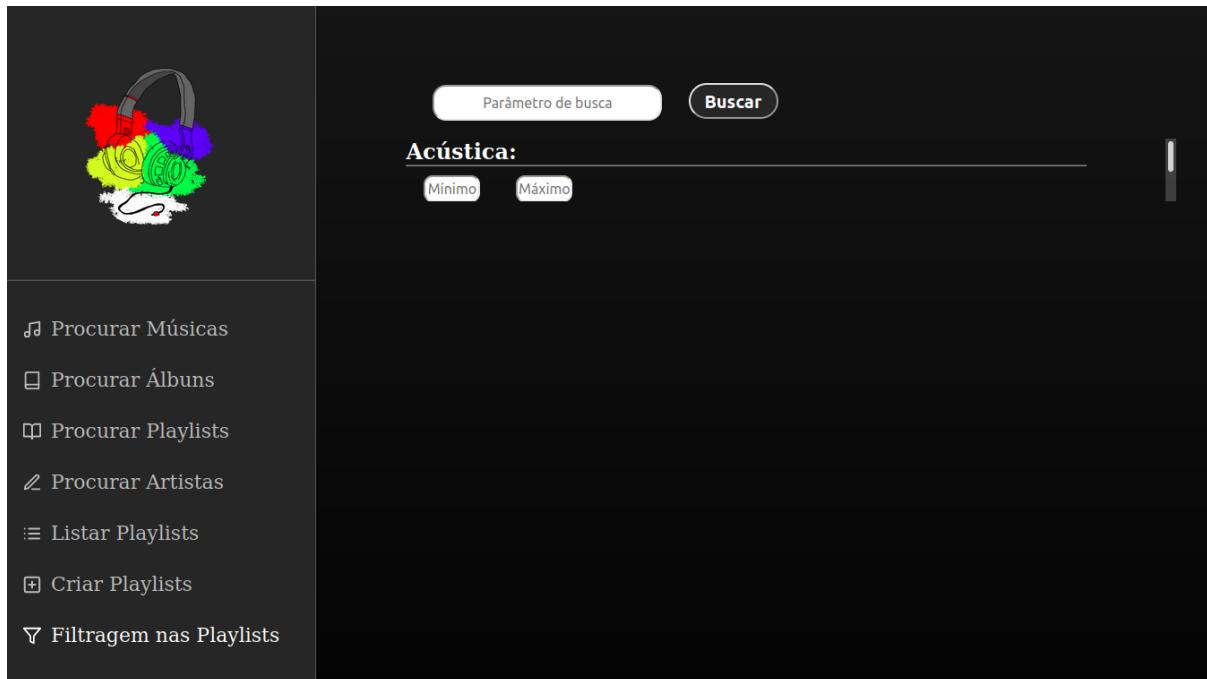
**Valence**  
0.187

(Escolhendo a quinta música retornada; o compasso é representada pelo parâmetro TimeSignature)

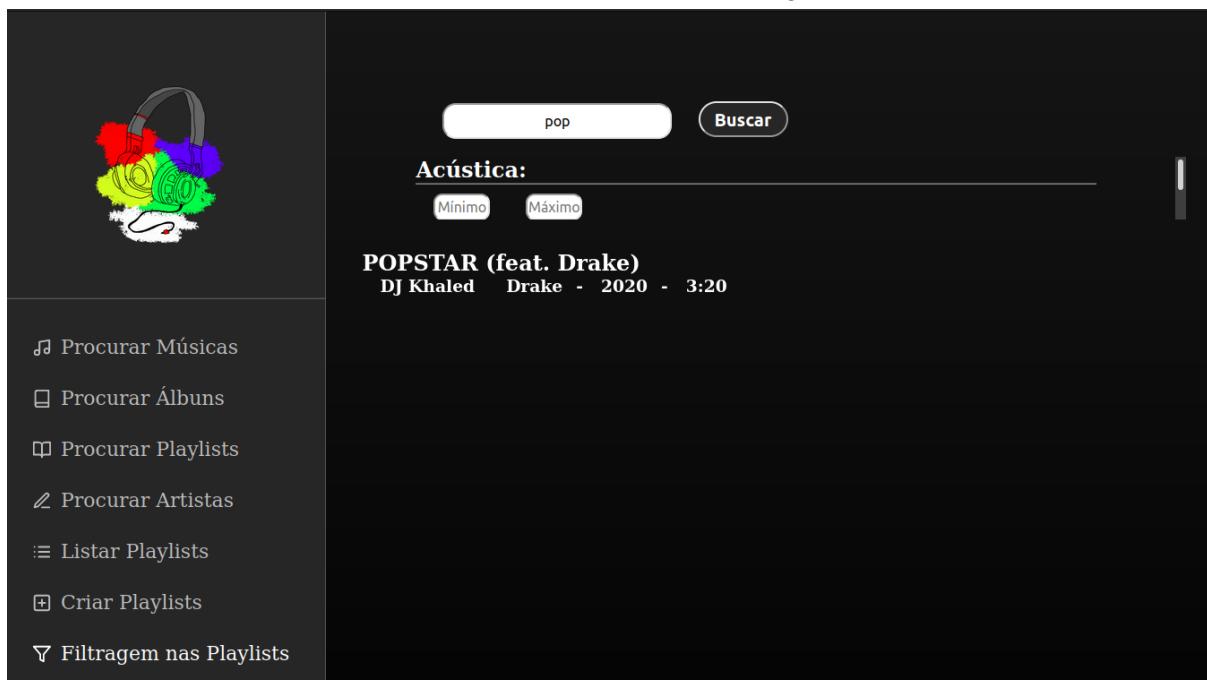
## 2.7 - Filtrando busca de músicas numa playlist

A tela da filtragem nas playlists permite que sejam determinados intervalos para cada parâmetro, sendo que, assim como dito anteriormente, há um intervalo inicial para cada parâmetro, e quando os intervalos não são alterados de uma busca para a outra, eles permanecem da forma que ocorre na busca anterior.

Os parâmetros apresentados de filtragem são acústica, danceabilidade e energia, visualizados por uma rolagem vertical.



Por exemplo, determinando para acústica, danceabilidade e energia, intervalos respectivos  $[0.0 ; 0.1]$ ,  $[0.75 ; 1.0]$ ,  $[0.45 ; 0.65]$ , e especificando a string “pop”:



**POPSTAR (feat. Drake)**

**Acousticness**  
0.057

**Danceability**  
0.8

**DurationMs**  
200221

**Energy**  
0.56

**Instrumentalness**  
0

**Key**  
5

(música obtida na filtragem dada anteriormente, com os valores acousticness, danceability e energy dentro dos intervalos estabelecidos)

## 2.8 - Detalhamento de músicas

Para as músicas nas playlists aparecem as informações desejadas: artistas, duração e ano. Além disso, para os álbuns, aparecem o ano de lançamento, e para as suas músicas, aparecem os artistas e a duração.

**Iron Man 2**  
2010

**Shoot to Thrill**  
AC/DC - 5:18

**Rock 'N' Roll Damnation**  
AC/DC - 3:37

**Guns for Hire**  
AC/DC - 3:25

**Cold Hearted Man**  
AC/DC - 3:34

**Back In Black**  
AC/DC - 4:16

**Thunderstruck**  
AC/DC - 4:52

**Let There Be Rock**  
AC/DC - 4:52

(Exemplo de álbum)

**HIP HOP 2022 🔥 New Rap & Trap Hits**

---

Mostrar parâmetros

- ▷ **Jimmy Cooks (feat. 21 Savage)**  
Drake 21 Savage - 2022 - 3:38
- ▷ **Hot Shit (feat. Ye & Lil Durk)**  
Cardi B Kanye West Lil Durk - 2022 - 3:32
- ▷ **Dua Lipa**  
Jack Harlow - 2022 - 2:15
- ▷ **Cooped Up (with Roddy Ricch)**  
Post Malone Roddy Ricch - 2022 - 3:06
- ▷ **Distraction**  
Polo G - 2022 - 2:51

(Exemplo de playlist pública)

**Private**

---

Mostrar parâmetros

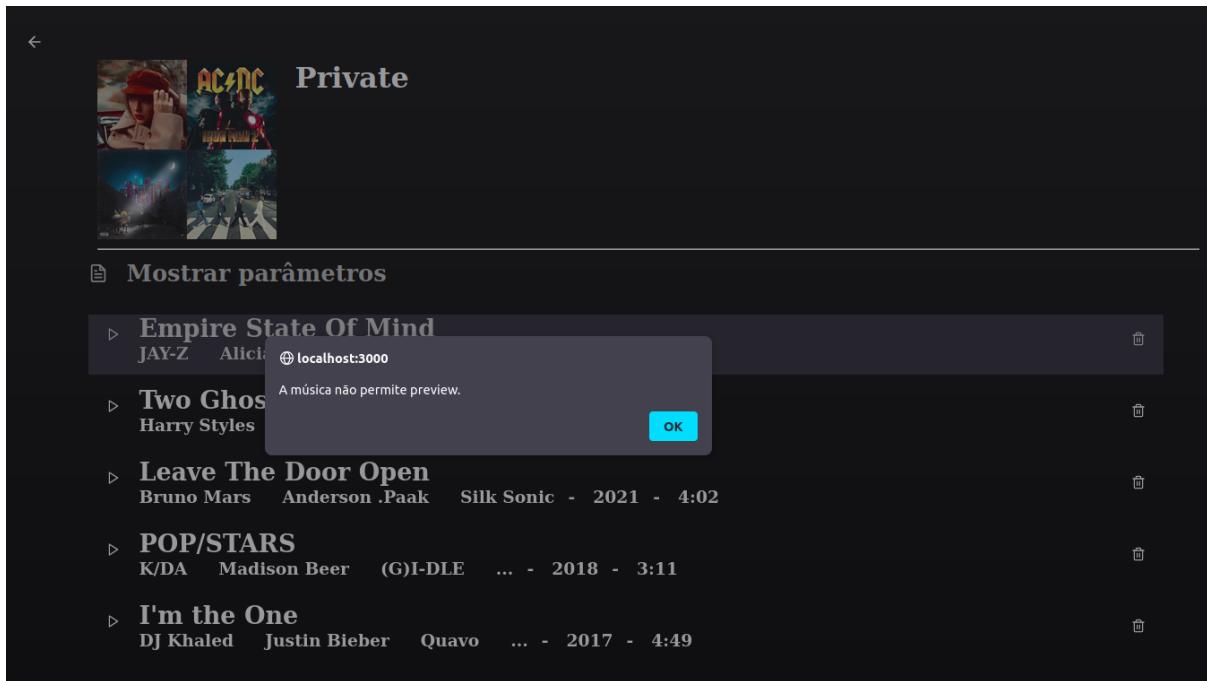
- ▷ **Empire State Of Mind**  
JAY-Z Alicia Keys - 2009 - 4:37
- ▷ **Two Ghosts - Recorded at Metropolis S**  
Harry Styles - 2017 - 3:44
- ▷ **Leave The Door Open**  
Bruno Mars Anderson .Paak Silk Sonic - 2021 - 4:02
- ▷ **POP/STARS**  
K/DA Madison Beer (G)I-DLE ... - 2018 - 3:11
- ▷ **I'm the One**  
DJ Khaled Justin Bieber Quavo ... - 2017 - 4:49

(Exemplo de playlist do usuário)

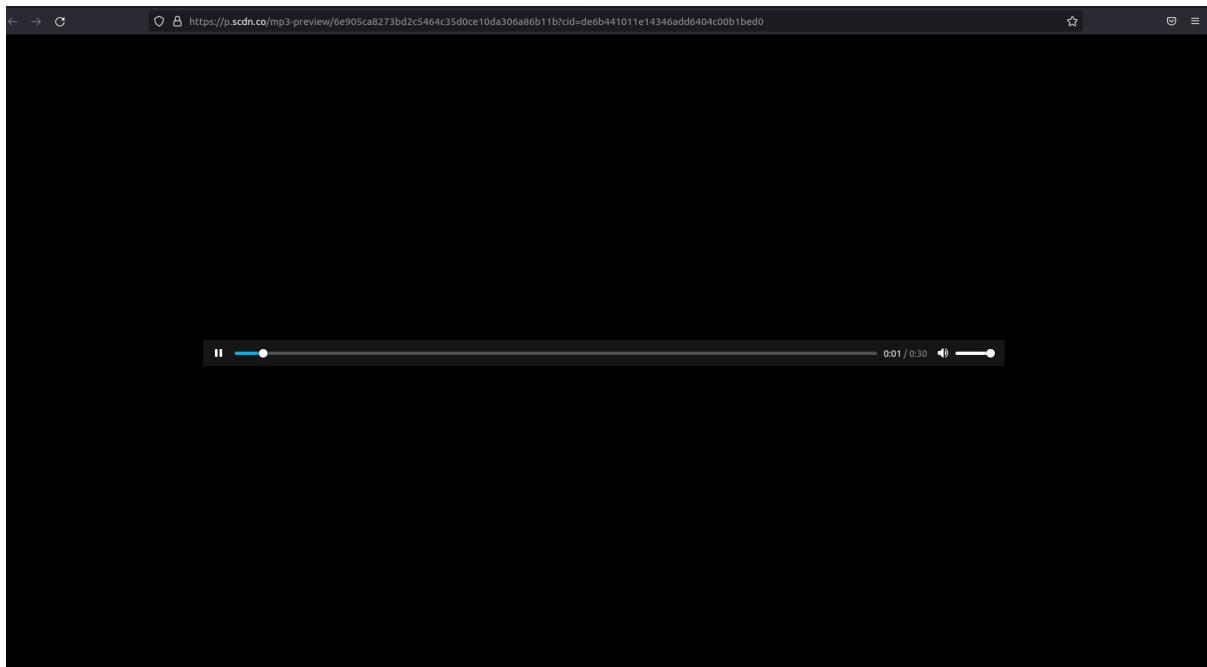
Para músicas com nomes longos, há um limite de caracteres que aparecerão na tela, seguidos de reticências. O mesmo acontece com a quantidade de artistas.

## 2.9 - Preview de música numa playlist do usuário

Cada música nas playlists apresenta ao lado de seu título um botão que, caso a música permita preview, possibilitará que um trecho da música seja escutado.



Quando a música não permite a preview, isso é indicado ao usuário quando ele tenta acessar o trecho da música.

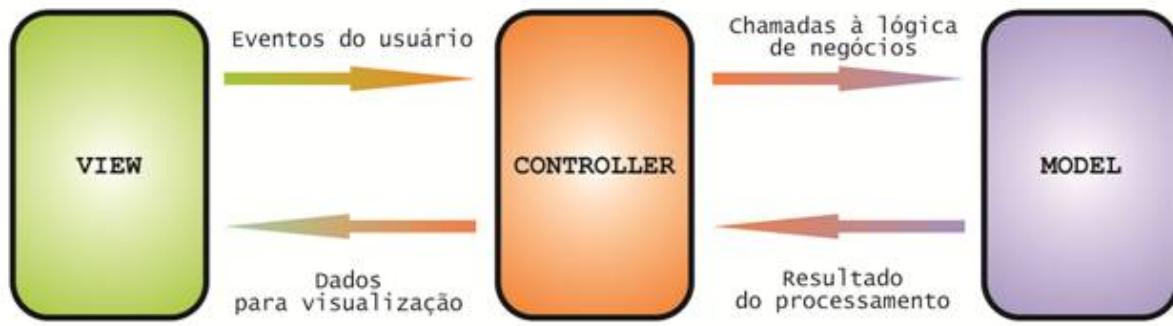


Quando a música permite a preview, por sua vez, uma nova aba é aberta com a url de preview, contendo um trecho da música.

# 3 - Arquitetura e Padrão de Projeto

## 3.1 - Padrão de Projeto

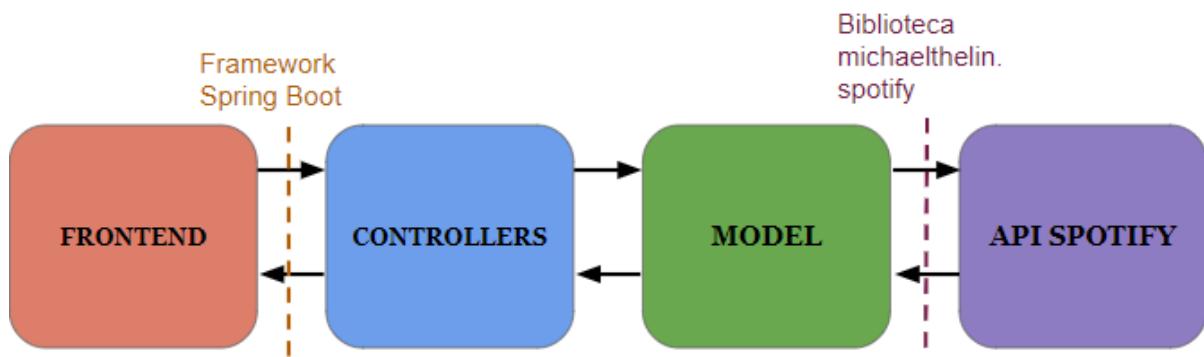
Para este sistema seguimos o padrão MVC, no qual possui camada de frontend (interface com o usuário), controllers (realizam a intermediação do frontend com o backend) e o models onde se encontra a lógica de negócios do sistema e os services que realizam a comunicação com os dados da API do Spotify.



## 3.2 - Arquitetura do sistema

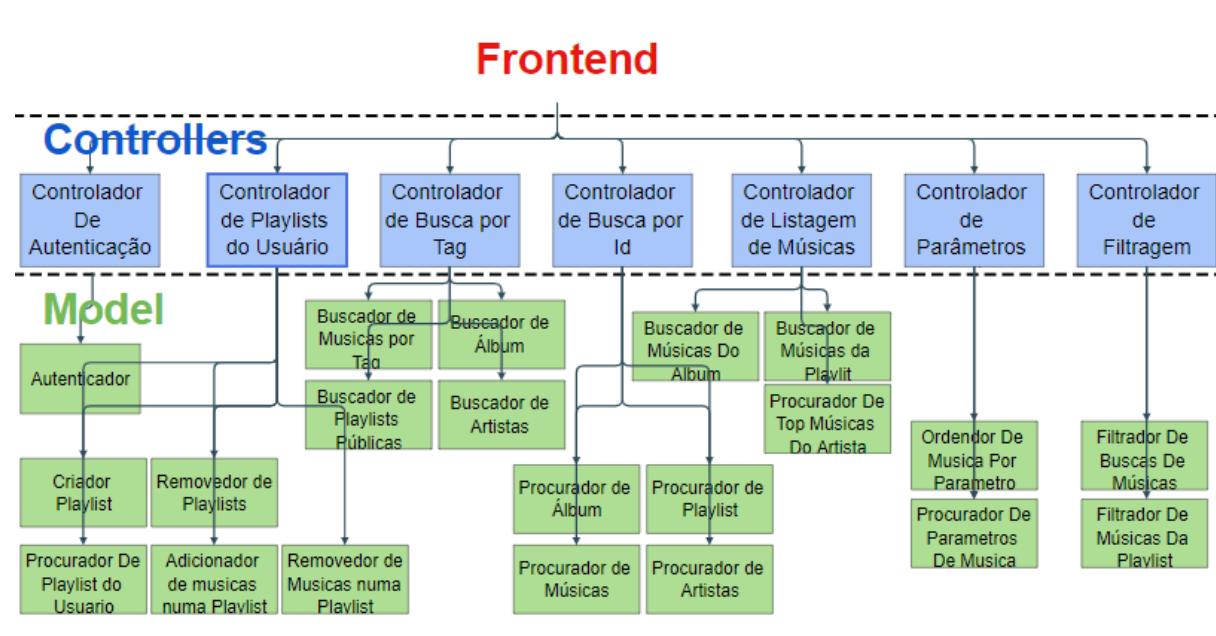
### 3.2.1 - Diagrama de Componentes

Os componentes do nosso sistema se comunicam segundo o seguinte diagrama:



### 3.2.2 - Diagrama de Classes

As classes do nosso sistema se comunicam segundo o seguinte diagrama:  
(Por simplificação omitimos seus atributos e métodos)



### 3.3 - Pacotes

Os pacotes definem a organização das nossas classes, com base nas funções executadas ou participação em termos de arquitetura.

```
▼ 📁 src/main/java
  ▶ 📁 com.mac0321.SuperGerenciadorMusical
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.controllers
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.models.entities
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.models.services
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.models.services.busca
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.models.services.busca.busca_por_id
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.models.services.busca.busca_por_tag
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.models.services.busca.listagem_musicas
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.models.services.filtragem
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.models.services.parametros
    ▶ 📁 com.mac0321.SuperGerenciadorMusical.models.services.playlist_usuarios
```

### 3.3.1 Controllers

Essas classes são responsáveis por orquestrar as rotas HTTP da aplicação. Define-se aqui que, cada rota, ao ser requisitada de forma pré-definida e com parâmetros estabelecidos, retorna uma resposta definida, em geral, um objeto.

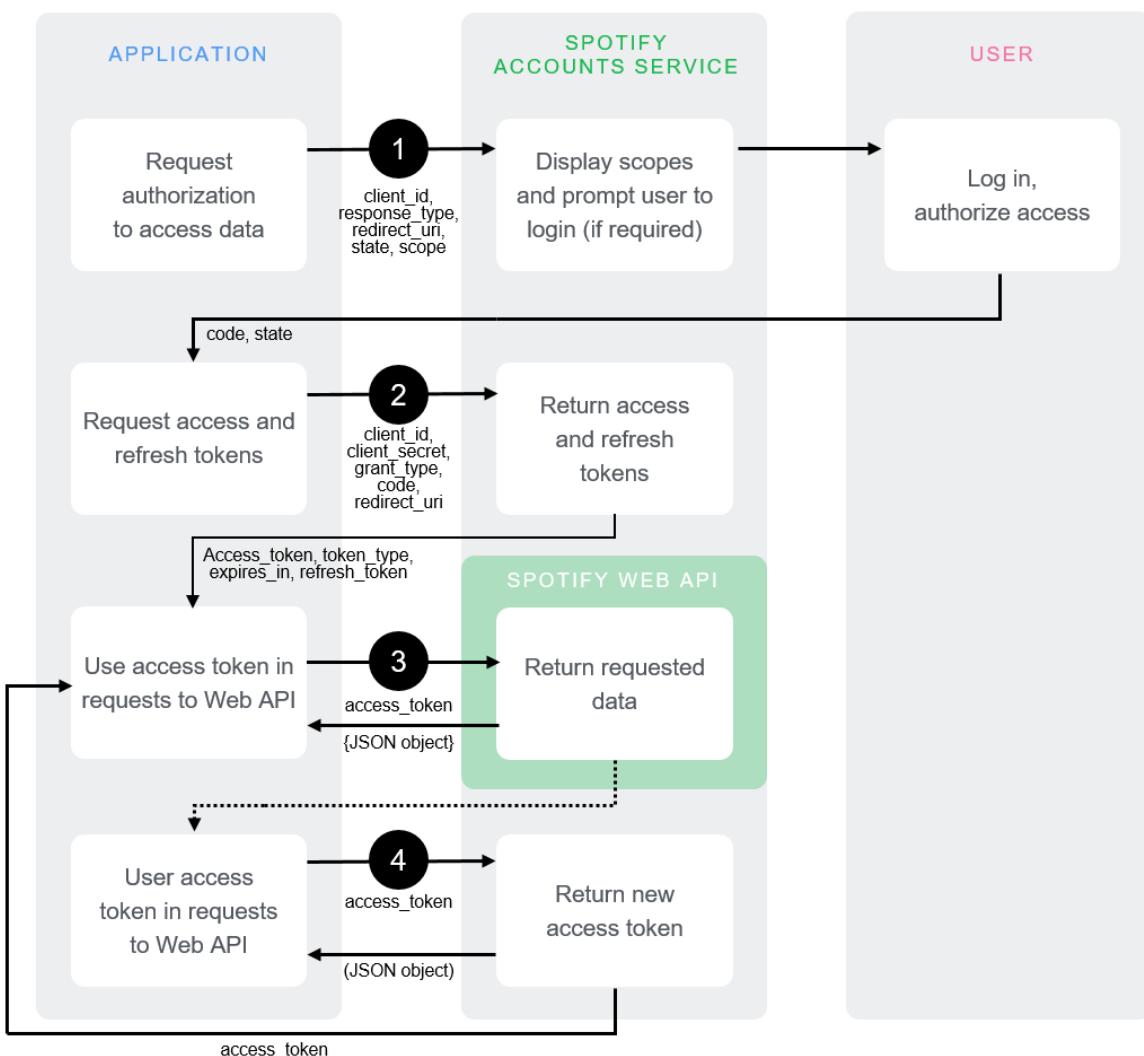
### 3.3.2 Models

Nesse pacote, as classes estão divididas em dois subpacotes: *entities* e *services*.

- **Entities** - classes auxiliares, que são usadas em mais de um lugar do código que, no geral definem como alguns dados estão dispostos
- **Services** - classes que executam diretamente alguma funcionalidade da API. É através dessas classes que os parâmetros definidos são passados para a API, e retornam o objeto desejado. Nesse pacote, existem 4 subpacotes (*busca*, *filtragem*, *parametros* e *playlist\_usuarios*) que executam funções diferentes, sendo elas: busca no Spotify, filtragem de resultados, ordenação de resultados e operações em playlists, respectivamente para os pacotes citados.

## 4 - Autenticação

Para utilizar os serviços da API do Spotify, é necessário realizar uma etapa de autenticação, pois estaremos operando com dados dos usuários da plataforma. Para realizar essa etapa, decidimos utilizar a autenticação via Code Flow, nela, nosso aplicativo realiza um pedido de autorização para o usuário para realizar algumas ações em seu nome, se o pedido é aceito pelo usuário, o Spotify retorna um código via URL que pode ser trocado futuramente por um token de Acesso. Posteriormente, o token de acesso é passado como parâmetro em algumas requisições na API. Abaixo, segue-se um diagrama que representa a autenticação via Code Flow. Implementamos utilizando ferramentas da biblioteca: <https://github.com/spotify-web-api-java/spotify-web-api-java>



A requisição é feita através de uma requisição HTTP do tipo GET no endpoint `/authorize` da API, que retorna uma URI para uma página de login do Spotify, onde ocorrerá efetivamente a solicitação. Essa requisição é construída através de uma classe da biblioteca auxiliar utilizada, e o método é invocado através de requisição do tipo GET na rota `/autenticacao/inicia`. Segue abaixo a imagem da tela de autenticação do Spotify.



## Projetão 2022 - LabPOO

You agree that Projetão 2022 - LabPOO will be able to:

### View your Spotify account data

Your name and username, your profile picture, how many followers you have on Spotify and your public playlists

### View your activity on Spotify

The content you are playing  
The content you are playing and Spotify Connect devices information  
Playlists you've made and playlists you follow  
Your collaborative playlists

### Take actions in Spotify on your behalf

Control Spotify on your devices  
Create, edit, and follow private playlists  
Create, edit, and follow playlists  
Stream and control Spotify on your other devices

You can remove this access at any time at [spotify.com/account](https://www.spotify.com/account).

For more information about how Projetão 2022 - LabPOO can use your personal data, please see Projetão 2022 - LabPOO's privacy policy.



Logged in as Henrique Paes de Souza.  
[Not you?](#)

Ao conceder as permissões para a aplicação, o usuário é redirecionado para a página inicial do aplicativo. Simultaneamente ao redirecionamento, ocorre a troca do código pelo access token, bem como armazenamento do dado obtido, através de uma requisição na rota /autenticacao/finaliza. Essa rotina está definida no capítulo 7.1, que trata do autenticador. Note que, na etapa 1 deste diagrama, a aplicação determina o escopo de permissões que irá pedir, que substancialmente define quais ações a aplicação toma em nome do usuário.

Os escopos definidos foram:

playlist-modify-private, playlist-modify-public, playlist-read-private, playlist-read-collaborative, user-read-playback-state, user-modify-playback-state, streaming user-read-currently-playing"

## 5 - Organização do frontend

O frontend se encontra na pasta **supergerenciadormusical/front/** sendo dividido em:

### Página:

A pasta “Página” apresenta uma tela que passa suas propriedades para cada child; representa a barra lateral da aplicação, na qual está o logo e o link para as principais telas, e o background escuro no qual cada informação é trazida em cada tela.

### Componentes:

A pasta componentes apresenta componentes que são reutilizados em múltiplas telas:

- Apresentação do Elemento: apresenta um header para todas as páginas associadas à exposição de informações sobre um determinado elemento - que pode ser uma música, álbum ou playlist.
- Botão do Parâmetro: é o modelo utilizado nos botões dos parâmetros apresentados nas páginas de ordenação das músicas em uma playlist de acordo com um determinado parâmetro.
- Busca Dados Da Playlist: componente utilizado tanto pelas playlists do usuário como pelas playlists públicas, para obtenção dos dados expostos na página da respectiva playlist.
- Buscar Dados: a busca de álbuns, playlists e artistas ocorre de modo muito semelhante, logo esse componente é utilizado na realização de tais buscas.
- “Buscar Músicas Filtradas” e “Filtro na Busca de Músicas”: a busca das músicas filtrada por um dos três parâmetros - compasso, tonalidade ou modo - ocorre utilizando esse componente. São impostos valores de entrada válidos para cada parâmetro.
- “Informações da Música”, “Mapear Artistas” e “Mapear Música”: para cada música encontrada em um álbum, playlist ou para um dado artista, informações como duração, artistas e ano são expostas utilizando esses componentes.
- Parâmetro: componente utilizado na tela da exposição das informações de uma música, na qual cada parâmetro e seu respectivo valor é passado.
- Tabela (e pastas internas): utilizados para determinar as músicas de acordo com a ordem determinada por um dado parâmetro, para as playlists.
- Tratamento de Erro: empregado em todas as requisições realizadas; os possíveis erros comuns a todas chamadas estão associadas a problemas na requisição (a requisição foi realizada, mas nenhuma resposta foi obtida), na resposta (a requisição foi realizada e o servidor respondeu com um status code) ou na própria configuração da requisição (algo durante a configuração da requisição causou um erro).
- Voltar: componente representado pelo ícone da seta apontando para a esquerda, que indica o retorno para a página anterior.

### Imagens:

Apresenta a Logo utilizada e imagens substitutivas de imagens de playlists e artistas, utilizadas quando necessário.

### Páginas:

A pasta “Páginas” apresenta as páginas existentes no front:

- “Artista”, “Álbum”, “Playlist Pública”: apresentam músicas obtidas na busca de acordo com o respectivo elemento; as músicas podem ser adicionadas nas playlists do usuário ou podem ter suas informações paramétricas acessadas. Para o caso da playlist pública, o usuário tem a opção de mostrar a tabela de parâmetros.
- “Playlist”: representa cada playlist do usuário; o usuário tem a opção de remover músicas, acessar suas informações, e mostrar a tabela de parâmetros.
- Busca nas Playlists: representa a tela na qual ocorre a busca de músicas nas playlists do usuário de acordo com intervalos para os parâmetros.
- “Buscar Álbuns”, “Buscar Playlists”, “Buscar Artistas”, “Buscar Músicas”: associadas a busca de músicas de acordo com álbum, playlist, artista e título, respectivamente.
- Criar Playlist: página associada a criação da playlist; o usuário deve colocar um nome para a sua playlist e indicar se ela é pública ou privada, colaborativa ou não.
- Listar Playlists: lista as playlists do usuário; o usuário pode acessar cada playlist e também pode removê-las.
- Filtragem de busca (e pastas internas): apresentam as páginas associadas à busca de música de acordo com os parâmetros de compasso, tonalidade e modo.
- “Música” e “Música da Playlist”: são as telas que expõem informações, respectivamente, de qualquer música, e de uma música de uma das playlists do usuário. A diferença é que a primeira tem a opção de ser adicionada a uma playlist.
- Página Inicial: tela de login e autenticação.
- “Parâmetros” e “Tabela de Parâmetros”: associadas à exposição das músicas de uma playlist, ordenadas de acordo com um determinado parâmetro.
- Selecionar playlist: permite que o usuário escolha a playlist na qual irá adicionar uma música escolhida.

#### **Estilos:**

Apresenta algumas características de estilização: as cores utilizadas e estilizações que ocorrem múltiplas vezes nos componentes e páginas.

## 6 - Controllers

Os controllers, como já foi dito anteriormente, são responsáveis pela integração do frontend com o backend, essa integração ocorre por meio de requisições http enviadas pelo frontend (origins = "http://localhost:3000/") e recebidas pelo backend http://localhost:8080/. Cada Rest Controller, fica situado em uma rota, por exemplo, o ControladorDePlaylistsDoUsuário recebe as requisições enviadas na rota: http://localhost:8080/playlists.

### **Controlador De Autenticação**

Responsável pela intermediação do frontend com o Autenticador do model.

Rota: "/autenticacao"

Do ponto de vista do front-end, a autenticação consiste em duas etapas:

1- requisição do tipo GET na rota "/autenticacao/iniciar"

2- redirecionar o usuário para a uri retornada pela requisição

Todo o resto, que refere-se à parte burocrática da conexão com a API do Spotify, está implementado pelo Autenticador.

### **Controlador De Playlists Do Usuário**

Responsável pela intermediação do frontend com os serviços de criação de playlist, listagem playlist, remoção de playlist, adição de músicas numa playlist e remoção de músicas numa playlist.

Rota: "/playlists"

### **Controlador De Busca Por Tag**

Responsável pela intermediação do frontend com os serviços de busca de músicas, de álbuns, playlists públicas e de artistas.

Rota: "/buscar-por-tag"

### **Controlador De Busca Por Id**

Responsável pela intermediação do frontend com os serviços de obtenção de músicas, de álbuns, playlists públicas e de artistas pelo seu ID.

### **Controlador De Listagem De Músicas**

Responsável pela intermediação do frontend com os serviços de listagem de músicas em um álbum, playlist e artista.

Rota: "/listar-musicas"

### **Controlador De Parâmetros De Música**

Responsável pela intermediação do frontend com os serviços de obter os parâmetros de músicas pelos seus ids, e de ordenar músicas por um determinado parâmetro.

Rota: "/parametros"

### **Controlador de Filtragem**

Responsável pela intermediação do frontend com os serviços de filtrar músicas e filtrar músicas na playlist.

Rota: "/filtragem"

**Observação:**

Para a funcionalidade de ouvir o preview de uma música foi adicionado o método Obter música com Market, ele é responsável pela intermediação do frontend com o serviço de obter música com Market, opção que permite obter o parâmetro preview da classe Track.

# 7 - Models e Services

Irei dividir essa sessão em subcapítulos, um para cada subpacote desse pacote.

## 7.1 - Entities

### Autenticador

Como foi apresentado anteriormente, o Spotify exige que a nossa aplicação tenha permissão do usuário para executar funções em sua conta, via API. O método para obtenção do token de acesso é especificado por uma rotina, implementada pelo Spotify, que segue o padrão OAuth 2.0, como descrito no capítulo para autenticação. A classe Autenticador representa a forma como gerenciamos o código recebido pelo usuário, no caso, fazemos com que toda a aplicação possua apenas um autenticador, que executa as comunicações com a API e guarda o token de acesso definido. Garantimos a unicidade do elemento ao implementarmos o padrão Singleton para essa classe. O Autenticador conta com dois métodos, sendo eles:

- *criarAutenticador* - quando não há autenticadores, cria um novo, caso contrário, retorna referência para autenticador já existente.
- *autorizacaoUsuario* - retorna URI para a página de autorização do Spotify
- *requisitaTokenAcesso* - faz a chamada na API para trocar o código recebido por um token de acesso
- *setTokens* - seta os tokens recebidos como atributos
- *getToken* - retorna o token de acesso atual

### Usuário Atual

**Herda de:** Serviço Spotify

**Método:** public User executaServiço()

Essa classe tem como objetivo obter os dados do usuário atualmente logado para isso ela gera uma requisição do tipo GET na rota <https://api.spotify.com/v1/me> e obtém como resultado dessa requisição esses dados e os retorna por meio do objeto User.

## 7.2 - Services

### Classe abstrata - Serviços do Aplicativo

Essa classe abstrata abrange todas as classes que realizam algum tipo de serviço no nosso projeto.

## **Classe abstrata - Serviço Spotify**

**Herda de:** Serviço do aplicativo

Essa classe abstrata abrange de todos os serviços do nosso gerenciador musical que possuem como atributo um objeto do tipo spotifyApi (basicamente, um gerador de urls de requisição à api, definido pela biblioteca citada anteriormente), ou seja, todos os serviços que realizam requisições diretamente no banco de dados do Spotify.

### **7.2.1 - Busca**

#### **Interface Serviço de Busca**

**Método:** public AbstractModelObject executaServiço(String tagDeProcura, int offset)

Essa interface realiza a busca de algum tipo de estrutura de dados da API, como álbuns, músicas e playlists, de acordo com certos parâmetros(a tagDeProcura e o offset) e retorna um Paging<> de objetos que satisfazem a tagDeProcura dada. A tagDeProcura é uma string dada pelo usuário que servirá como base para o algoritmo de busca de dados da API. O offset deve ser incrementado de 50 em 50 a cada vez que o usuário faz uma requisição consecutiva usando a mesma tagDeProcura, para assim obter novos resultados, seu valor máximo é 1000. Nessa interface realizamos uma requisição do tipo GET na rota <https://api.spotify.com/v1/search>.

#### **7.2.1.1 Busca por ID**

#### **Interface Serviço de Procura Única**

**Método:** public AbstractModelObject executaServiço(String id)

A partir do id de um objeto do banco de dados, ele retorna o objeto em si, podendo ser um artista, um álbum, uma playlist, etc. Para isso, realiza-se uma requisição do tipo GET, a rota pode variar dependendo do objeto a ser obtido.

#### **Procurador de Álbum**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de procura única

**Método:** veja sua interface

Implementa a sua interface na busca de um álbum, realizando uma requisição do tipo GET na rota <https://api.spotify.com/v1/albums/id>.

### **Procurador de Artista**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de procura única

**Método:** veja sua interface

Implementa a sua interface na busca de um artista, realizando uma requisição do tipo GET na rota <https://api.spotify.com/v1/artists/id>.

### **Procurador de Músicas**

**Herda de:** Serviço Spotify

**Método:** public Track[] executaServiço(String [] ids)

Essa classe obtém as músicas referenciadas pelos seus ids no array ids. Para isso, realiza-se uma requisição do tipo GET na rota <https://api.spotify.com/v1/tracks>.

**Método:** public Track executaServiço(String id) (veja sua interface)

Implementa a sua interface na busca de uma música, realizando uma requisição do tipo GET na rota <https://api.spotify.com/v1/tracks/id>.

### **Procurador de Playlist**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de procura única

**Método:** veja sua interface

Implementa a sua interface na busca de uma playlist, realizando uma requisição do tipo GET na rota [https://api.spotify.com/v1/playlists/playlist\\_id](https://api.spotify.com/v1/playlists/playlist_id).

#### **7.2.1.2 Busca por Tag**

### **Buscador de Álbuns**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de busca

**Método:** veja sua interface

Essa classe implementa o ato de buscar álbuns por meio de sua interface, nesse caso a tagDeProcura é o nome do álbum buscado.

### **Buscador de Artistas**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de busca

**Método:** veja sua interface

Essa classe implementa o ato de buscar artistas por meio de sua interface, nesse caso a tagDeProcura é o nome do artista buscado.

### **Buscador de Músicas por Tag**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de busca

**Método:** veja sua interface

Essa classe implementa o ato de buscar as músicas do Spotify por meio de sua interface, nesse caso a tagDeProcura é o nome da música buscada.

### **Buscador de Playlists Públicas**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de busca

**Método:** veja sua interface

Essa classe implementa o ato de buscar as playlists públicas por meio de sua interface, nesse caso a tagDeProcura é o nome da playlist buscada.

#### **7.2.1.3 Listagem Músicas**

##### **Buscador de Músicas da Playlist**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de busca

**Método:** veja sua interface

Essa classe implementa o ato de buscar as músicas de uma playlist por meio de sua interface, nesse caso a tagDeProcura é o nome da playlist.

##### **Buscador de Músicas do Álbum**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de busca

**Método:** veja sua interface

Essa classe implementa o ato de buscar as músicas de um álbum por meio de sua interface, nesse caso a tagDeProcura é o nome do álbum.

##### **Procurador de Top Músicas do Artista**

**Herda de:** Serviço Spotify

**Método:** public Track[] executaServiço(String id)

Essa classe obtém as músicas mais tocadas de um artista - referenciado pelo seu id. Para isso, realiza-se uma requisição do tipo GET na rota <https://api.spotify.com/v1/artists/id/top-tracks>.

## 7.2.2 - Filtragem

### Filtrador de Buscas de Músicas

**Herda de:** Serviço do aplicativo

**Método:** public Track[] executaServiço(String tagDeProcura, int offset, int[] indicesDosFiltros, Float[] valoresMinMaxPorFiltro)

Esse método obtém 100 músicas pela classe Buscador de Músicas por Tag(usando a tagDeProcura e o offset como argumentos) usando certos parâmetros dados pelos vetores indiceDosFiltros e valoresMinMaxPorFiltro para filtrar essa músicas usando a classe filtrador de Músicas por Intervalo, veja que primeiro obtemos e filtramos 50 músicas, depois mais 50. Por fim, transforma-se a lista de Tracks obtidas na filtragem num vetor usando a classe Gerador De Array.

### Filtrador de Músicas das Playlists

**Herda de:** Serviço do aplicativo

**Método:** public Track[] executaServiço(String tagDeProcura, int[] indices\_dos\_intervals, Float[] intervalos\_de\_busca)

Esse método primeiramente obtém todos os ids das músicas das playlists do usuário, usando a classe Gerador de IDs das Músicas do Usuário Atual, a partir disso usa o Filtrador de Músicas por Nome para filtrar essas músicas das playlists por meio de uma tagDeProcura, depois as músicas remanescentes são filtradas a partir de certos parâmetros dados pelos vetores indices\_dos\_intervals e intervalos\_de\_busca usando a classe filtrador de Músicas por Intervalo.

### Filtrador De Músicas Por Intervalo

**Método:** public Track[] filtra(Float[] intervalos\_de\_busca, int[] índices\_dos\_intervals, String[] ids)

Primeiramente irei explicar a semântica dos vetores intervalos\_de\_busca e índices\_dos\_intervals, para cada dois índices consecutivos do primeiro vetor (i e i + 1) - sendo o primeiro par - representam respectivamente os valores mínimo e máximo da filtragem para um dado parâmetro que é identificado pelo índice i/2 do vetor índices\_dos\_intervals, para realizar essa identificação usamos a classe Mapeamento de Parâmetros. Esse método, filtra as músicas de acordo com os intervalos de parâmetros e usando o método filtrar por intervalo para realizar a filtragem.

**Método:** private String[] filtra\_por\_intervalo(String[] ids, Float mínimo, Float máximo, Float[] parâmetro\_da\_música)

Nesse método, os vetores ids e parâmetro da música , para um mesmo índice, se referem a uma mesma Track. Esse método adiciona em uma lista os ids das músicas que estão no

intervalo [mínimo, máximo] para um dado parâmetro e retorna esses ids por um array usando a classe Gerador de Array.

### **Filtrador de Músicas por Nome**

**Método:** public Track[] filtra(String ids, String nome)

A partir dos ids das músicas, obtêm as músicas usando a classe Procurador de Músicas, então filtra apenas as músicas que contêm em seus nomes uma parte de um certo nome, argumento da função, desconsiderando se as letras são maiúsculas ou minúsculas.

### **Gerador de Array**

**Método:** public String[] listStringParaArray(List<String> lista)

Converte uma lista de String em um array de String.

**Método:** public Track[] listTrackParaArray(List<Track> lista)

Converte uma lista de Track em um array de Track.

**Método:** Track[] pagingTrackParaArray(Paging<Track> paging)

Converte um paging de Track em um array de Track.

**Método:** Track[] listTrackParaArray(List<Track[]> lista, int tamanho)

Converte uma lista de Track em um array de Track, usando como parâmetro extra o tamanho do vetor devolvido.

### **Gerador de IDs das Músicas do Usuário Atual**

**Método:** String[] obtem\_ids()

Obtêm os ids de todas as músicas de todas as playlists do usuário, com repetição, para isso primeiramente usa a classe Procurador de Playlists do Usuário Atual para obter as playlists do usuário e a classe Procurador de Playlist para obter as músicas de uma playlist, por fim converte a lista de ids em um array usando a classe Gerador de Array.

### **Mapeamento dos Parâmetros**

**Método:** public void atualiza\_mapa(AudioFeatures parâmetros\_da\_música)

Delega para as chaves de 1 a 13 do HashMap os atributos dessa AudioFeatures, sendo que essas chaves de 1 a 13 que simbolizam os valores possíveis para um elemento do vetor indice\_dos\_intervalos do método da classe Filtrador de Músicas por Intervalo.

**Método:** public float valor\_do\_mapa(int índice)

Devolve o valor do atributo da última AudioFeature que foi usada como argumento na atualiza\_mapa armazenado na key índice.

## 7.2.3 - Parâmetros

### Procurador de Parâmetros de Músicas

**Herda de:** Serviço Spotify

**Método:** public AudioFeatures[] executaServiço(String [] ids\_das\_músicas)

Essa classe obtém os parâmetros de cada uma das músicas referenciadas pelos seus ids no array ids\_das\_músicas. Para isso, realiza-se uma requisição do tipo GET na rota <https://api.spotify.com/v1/audio-features>.

### Ordenador de Músicas por Parâmetro

**Herda de:** Serviço do aplicativo

**Método:** public <T extends Comparable<T>> String[] ordenaMúsicas(T[] parâmetro, String[] ids)

Dada dois vetores, um de parâmetros- obtido pelo AudioFeatures da música - e outro de ids, onde um mesmo índice dos dois vetores se referencia a mesma música, ordeno o vetor parâmetro de ordem decrescente e o vetor de ids é ordenado juntamente com ele, para que um mesmo índice dos dois vetores se remeta a mesma música, por fim devolvo o vetor de ids ordenado.

**Método:** private <T extends Comparable<T>> void quickSort(T[] parâmetro, int ini, int fim, String [] ids)

Ordeno o vetor de Comparable , e consequentemente o vetor de ids também, utilizando o algoritmo do quicksort.

**Método:** private <T extends Comparable<T>> int separaSedgewick(T[] parâmetro, int ini, int fim, String[] ids)

Utilizo o método proposto por Sedgewick para escolher o próximo pivô do QuickSort.

### Trocador de Elementos

**Método:** public <T extends Comparable<T>> void trocar(T[] vetor, int i, int j)

Troca os elementos T[i] e T[j] de lugar no vetor dado, sendo este um vetor de objetos que implementam a interface Comparable<T>.

## 7.2.4 - Playlist Usuários

### Interface Serviço de Modificação de Músicas de uma Playlist

**Método:** public AbstractModelObject executaServiço(String playlistID, String uris [])

Essa interface é responsável por modificar as músicas, referenciadas por seus uris, de uma playlist - especificada pelo playlistID - ou seja, remover ou acrescentar músicas nela, as funções que a implementam retornam o AbstractModelObject retornado pela api durante a requisição. As exceções sobre a privacidade da playlist (ser pública ou não) são administradas pela API. Os serviços dessa interface são performados ao se realizar uma requisição do tipo POST na rota [https://api.spotify.com/v1/playlists/{playlist\\_id}/tracks](https://api.spotify.com/v1/playlists/{playlist_id}/tracks).

## **Adicionador de Músicas numa Playlist**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de modificação de músicas de uma playlist

**Método:** veja sua interface.

Essa classe implementa o ato de adicionar músicas de uma playlist por meio da sua interface, retorna os snapshot id da playlist modificada.

## **Removedor de Músicas numa Playlist**

**Herda de:** Serviço Spotify

**Implementa:** Serviço de modificação de músicas de uma playlist

**Método:** veja sua interface.

Essa classe implementa o ato de remover músicas de uma playlist por meio da sua interface, retorna os snapshot id da playlist modificada.

## **Gerador de Json**

**Método:** public JSONArray urisParaJSONArray(String [] array)

Transforma um array de uris de músicas em um jsonArray e o devolve, essa classe é utilizada para remover músicas de uma playlist.

## **Criador de Playlist**

**Herda de:** Serviço Spotify

**Método:** public AbstractModelObject executaServiço(String userID, String nome\_da\_playlist, boolean serColaborativa, boolean serPública, String descrição)

A partir do userID, cria uma playlist vazia desse usuário, com nome dado por nome\_da\_playlist e suas características(ser colaborativa, ser pública e sua descrição) são dadas pelos outros argumentos da função, dados pelo usuário. Essa classe realiza uma requisição do tipo POST na rota [https://api.spotify.com/v1/users/{user\\_id}/playlists](https://api.spotify.com/v1/users/{user_id}/playlists) e retorna a playlist criada.

## **Removedor de Playlists**

**Herda de:** Serviço Spotify

**Método:** public int executaServiço(String userID, String playlistID)

A partir do id do usuário, ele remove a playlist desse usuário que contenha o determinado playlistID, a função dessa classe retorna 1 caso haja sucesso na remoção e 0 caso

contrário. Essa classe realiza uma requisição do tipo DELETE na rota [https://api.spotify.com/v1/playlists/playlist\\_id/followers](https://api.spotify.com/v1/playlists/playlist_id/followers).

### **Procurador de Playlists do Usuário Atual**

**Herda de:** Serviço Spotify

**Método:** public AbstractModelObject executaServiço(int offset)

Essa classe retorna as playlists de um usuário, contudo a cada busca será devolvido no máximo 50 playlists. Para permitir o acesso a mais playlists, usamos o argumento offset, na qual o offset deve ser incrementado de 50 em 50 a cada vez que o usuário faz uma requisição consecutiva desse tipo, para assim obter novos resultados, seu valor máximo é 1000. Esse serviço é executado através de uma requisição do tipo GET na rota <https://api.spotify.com/v1/me/playlists>.