Musiclist: Um Sistema de Recomendação de Músicas baseado em Gênero e Subgênero utilizando o K-means

Aline Rose Alencar Santos[1]

[1] *Universidade Federal do Tocantins, Ciência da Computação, Tocantins, Brasil*

Reception date of the manuscript:
Acceptance date of the manuscript:
Publication date:

Abstract— A disponibilidade e o avanço da tecnologia na indústria da música têm revolucionado a forma de consumir música. As plataformas de streaming, como o Spotify, proporcionaram acesso a uma vasta biblioteca musical, desafiando os usuários a encontrarem músicas que correspondam aos seus gostos pessoais. Para solucionar esse desafio, os sistemas de recomendação desempenham um papel fundamental. Neste trabalho, foi criado um sistema de recomendação utilizando o algoritmo de agrupamento K-means, levando em consideração os critérios de gênero e subgênero. A implementação ocorre por meio de uma API com o Django Rest Framework, que é consumida pelo frontend em Vue.js. Essa abordagem visa oferecer aos usuários uma experiência personalizada e agradável ao descobrirem novas músicas de acordo com suas preferências musicais.

Keywords—Sistema de Recomendação, K-means, Django Rest, API

I. Introdução

A evolução da tecnologia e a disponibilidade de música digital têm transformado significativamente a indústria da música. Anteriormente, as pessoas costumavam adquirir música por meio de formatos físicos, como CDs ou vinis, e tinham acesso a um número limitado de opções. No entanto, com o surgimento da internet, a música se tornou amplamente acessível e disponível em grande escala. Isso levou ao surgimento de plataformas de streaming, como o Spotify, que permitiram organizar e gerenciar essa imensa quantidade de músicas.

As plataformas de streaming de música atuais possuem vastas bibliotecas de músicas disponíveis para os usuários. Com milhões de faixas disponíveis, encontrar músicas que sejam do interesse pessoal de cada usuário pode ser um desafio. É nesse contexto que entram os sistemas de recomendação. De acordo com [1], sistemas de recomendação são filtros de informações que apresentam itens ou objetos que podem ser do interesse dos usuários.

No caso do Spotify, o sistema de recomendação funciona de forma a compreender as preferências musicais individuais e fornecer recomendações personalizadas com base nessas preferências. Para isso, são considerados vários fatores, como as músicas que o usuário já ouviu, as playlists que ele criou, artistas relacionados, gêneros musicais e até mesmo o comportamento de outros usuários com preferências semelhantes.

Nesse contexto, este trabalho tem como objetivo criar um sistema de recomendação utilizando o algoritmo de agrupamento Kmeans e levando em consideração dois critérios de entrada: o gênero e o subgênero. O trabalho é dividido em duas partes principais: a busca e o tratamento dos dados e a implementação do modelo em uma API utilizando o Django Rest Framework. Essa API é consumida para mostrar as recomendações no utulizando o framework Vue.js, assim, foi construido uma interface para proporcionar aos usuários do sistema uma experiência personalizada e agradável ao descobrir novas músicas de acordo com seus gostos musicais.

II. Metodologia

O objetivo deste artigo é desenvolver um sistema de recomendação de músicas chamado "Musiclist" que utilize o algoritmo de agrupamento K-means para recomendar músicas aos usuários com base em suas preferências de gênero e subgênero musical. Nesse sentido, o trabalho foi desenvolvido em duas principais etapas e entre essas subetapas.

a. Busca e treinamento dos dados

A primeira etapa do trabalho consistiu na coleta e tratamento dos dados. Para isso, foram utilizados dois datasets diferentes. O primeiro dataset foi obtido a partir da plataforma Kaggle, contendo informações de músicas, artistas, gêneros, subgêneros e outros atributos relacionados (disponível em: [https://www.kaggle.com/datasets/imuhammad/audio-features-and-lyrics-of-spotify-songs]). O segundo dataset continha dados de artistas, incluindo suas imagens e o gênero

Contact data: Aline Rose A. Santos, aline.rose@mail.uft.edu.br

musical ao qual pertencem.

A escolha de utilizar esses dois datasets se deu pela necessidade de ter informações visuais dos artistas, como suas imagens, para posteriormente exibi-las na parte do front-end do sistema. Dessa forma, o primeiro passo foi realizar a junção (merge) dos dois datasets, considerando que o nome do artista era o critério de combinação. Essa abordagem permitiu criar um único dataset final que reunia as informações de ambos os datasets.

```
Data columns (total 8 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   track_name          32860 non-null  object
 1   artistas            32865 non-null  object
 2   track_popularity    32865 non-null  int64
 3   track_album_name    32860 non-null  object
 4   playlist_name       32865 non-null  object
 5   playlist_genre      32865 non-null  object
 6   playlist_subgenre   32865 non-null  object
 7   imagem_artista_url  32865 non-null  object
dtypes: int64(1), object(7)
```

Fig. 1: Informações do dataset final

A partir do dataset final resultante dessa junção, foram realizadas várias etapas para preparar os dados antes de utilizar o modelo de treinamento K-means. Algumas dessas etapas incluíram:

- Deleção de colunas irrelevantes: Foram identificadas colunas no dataset que foram consideradas irrelevantes para o modelo e, portanto, foram removidas do conjunto de dados.
- Preenchimento da coluna de imagens faltantes: Se algumas linhas no dataset não possuíam imagens para os artistas correspondentes, foi inserida uma imagem padrão ou uma imagem que indica que o artista não possui imagem disponível.
- Preenchimento de artistas desconhecidos: Caso houvesse artistas com informações ausentes ou desconhecidas, esses registros foram preenchidos para garantir a integridade dos dados.
- Remoção de músicas duplicadas: Identificaram-se músicas duplicadas no dataset e essas duplicatas foram removidas, mantendo apenas uma instância de cada música.

Depois de realizar essas etapas, o dataset foi finalmente preparado para ser usado no modelo de treinamento. No entanto, ainda era necessário realizar algumas partes adicionais antes de aplicar o K-means:

- Verificação e remoção de linhas duplicadas: Foi feita uma verificação adicional para garantir que não houvesse linhas duplicadas restantes no dataset, a fim de evitar informações redundantes.
- Preenchimento de linhas faltantes com a média: Caso houvesse valores faltantes em algumas linhas do dataset, esses valores foram preenchidos com a média dos valores correspondentes, de forma a manter a integridade dos dados.
- Verificação de outliers: A verificação de outliers foi feita usando histogramas. Um histograma é uma representação gráfica da distribuição dos dados em intervalos. Ao plotar um histograma dos valores do conjunto de dados, é possível visualizar a frequência de ocorrência dos diferentes valores e identificar regiões onde há valores incomuns ou extremos.

Após a preparação dos dados, foram gerados histogramas para visualização das distribuições de gêneros e subgêneros. Também foram plotados gráficos para analisar a distribuição de músicas por gêneros e subgêneros.

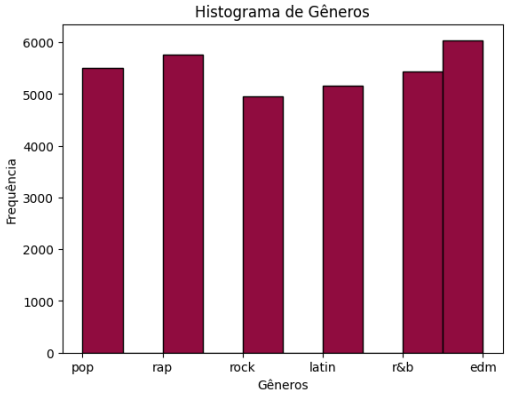

Fig. 2: Histograma dos Gêneros

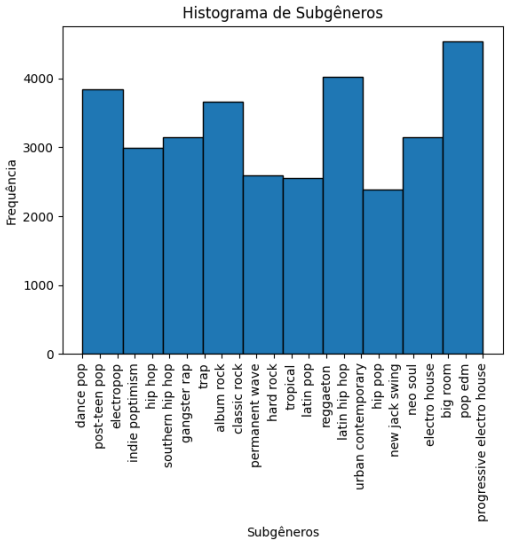

Fig. 3: Histograma dos Subgêneros

b. Implementação do Kmeans em um endpoint da API em Django Rest Framework

A implementação do algoritmo K-means em um endpoint da API utilizando o Django Rest Framework segue os seguintes passos:

- Leitura dos dados: Utiliza-se a biblioteca pandas para ler o arquivo CSV gerado e carregar os dados em um DataFrame.
- Pré-processamento dos dados: Realiza-se o pré-processamento dos dados para prepará-los para o algoritmo K-means. Utiliza-se a classe LabelEncoder do scikit-learn para transformar as variáveis categóricas "playlistgenre" e "playlistsubgenre" em valores numéricos. Isso é necessário porque o K-means trabalha ape-

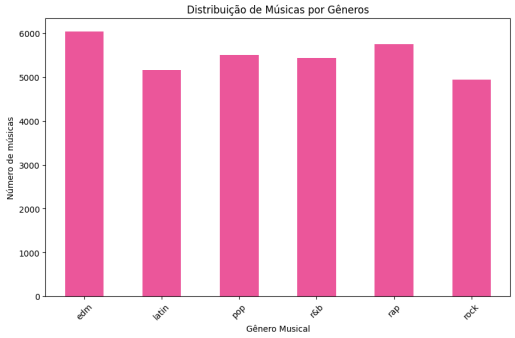

Fig. 4: Distribuição das músicas por Gêneros

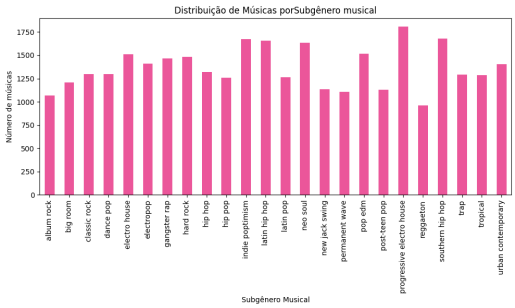

Fig. 5: Distribuição das músicas por Subgêneros

nas com dados numéricos. Selecionam-se as colunas relevantes e as codificações são aplicadas.

- Criação do endpoint: É criado um endpoint na API que recebe as informações de gênero e subgênero fornecidas pelos usuários. O endpoint utiliza a classe de visualização do Django Rest Framework para lidar com as solicitações HTTP, especialmente o método POST para receber os dados de entrada.

- Aplicação do algoritmo K-means: Aplica-se o algoritmo K-means ao conjunto de dados pré-processado. Define-se o número de clusters como 30, com base em experimentações anteriores. Realiza-se a clusterização dos dados, agrupando as músicas com base em suas características (gênero e subgênero) em clusters distintos.

- Geração de recomendações: A partir do cluster filtrado, seleciona-se aleatoriamente um número pré-definido de músicas para gerar as recomendações. Para cada música selecionada, verifica-se se o artista associado já existe no banco de dados. Em caso negativo, cria-se um novo objeto "Artist" no banco de dados. Criam-se objetos "Music" no banco de dados com base nas recomendações selecionadas, associando-os ao artista correspondente.

- Preparação da resposta: Preparam-se os dados relevantes para a resposta da API, incluindo o ID da recomendação, gênero, subgênero e detalhes da playlist gerada. Utilizam-se as classes e funções fornecidas pelo Django Rest Framework para formatar os dados de resposta em um formato adequado para a API.

- Retorno da resposta: A resposta contendo as recomendações de músicas é retornada ao usuário por meio da API.

A implementação do algoritmo K-means em um endpoint da API, utilizando o Django Rest Framework, permite aos usuários obter recomendações de músicas com base nos gêneros e subgêneros fornecidos. Essa abordagem segue uma estrutura escalável e orientada a recursos.

Para consumir a API, foi desenvolvido um projeto utilizando o Vue.js, um framework JavaScript.

III. Resultados

Para compreender o funcionamento dos agrupamentos, o modelo foi treinado separadamente em um ambiente com Jupyter Notebook. As entradas foram atribuídas a variáveis e gráficos foram plotados para visualizar os agrupamentos.

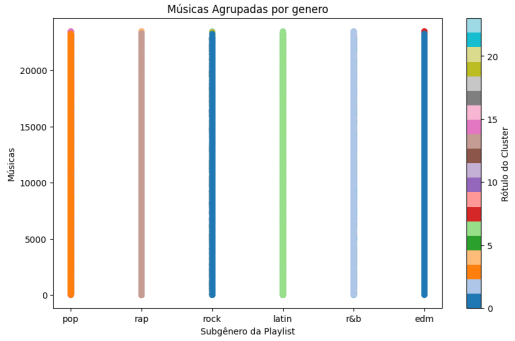

Fig. 6: Distribuição das músicas por Gêneros

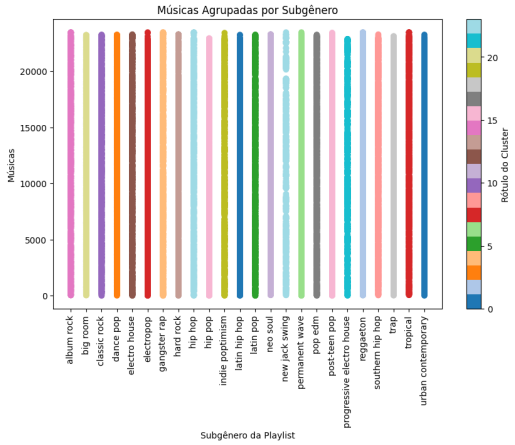

Fig. 7: Distribuição das músicas por Subgêneros

Além disso, no frontend, utilizando o Vue.js, é possível visualizar as playlists recomendadas.

A imagem 8 ilustra a tela na qual o usuário pode inserir seus gêneros e subgêneros de preferência.

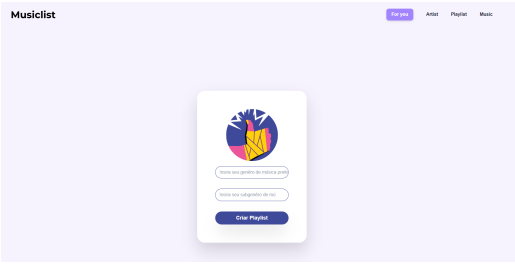

Fig. 8: Entrada de dados no front-end

IV. Playlists recomendadas

1. pop e dance pop

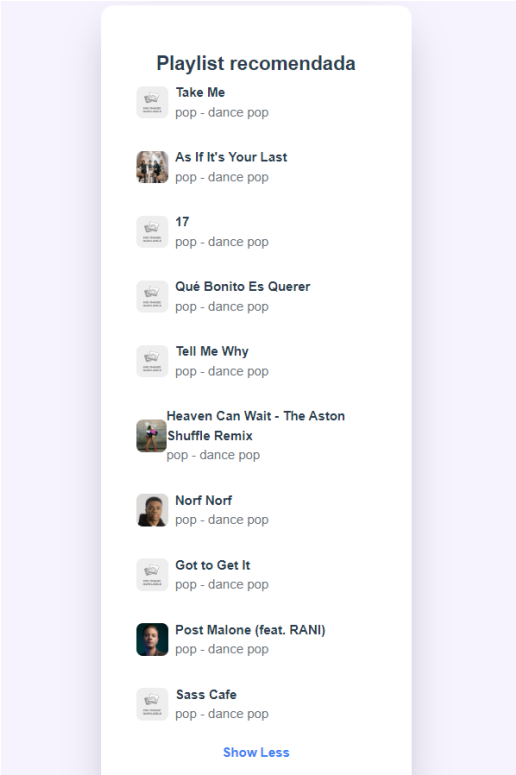

Fig. 9: Playlist recomendada para pop e dance pop

2. rock e classic rock

3. edm e pop edm

4. rap e gangster rap

5. rap e trap

6. rb e urban contemporary

Em suma, o algoritmo oferece recomendações de playlists de forma bastante precisa. No entanto, é perceptível no gráfico a presença de alguns pontos atípicos (outliers), embora não afetem significativamente os resultados das recomendações.

References

[1] C. L. R. da Motta, A. C. B. Garcia, A. S. Vivacqua, and F. M. Santoro, "Sistemas de Recomendação," 2018. [Online]. Available: https://www.researchgate.net/publication/328228374

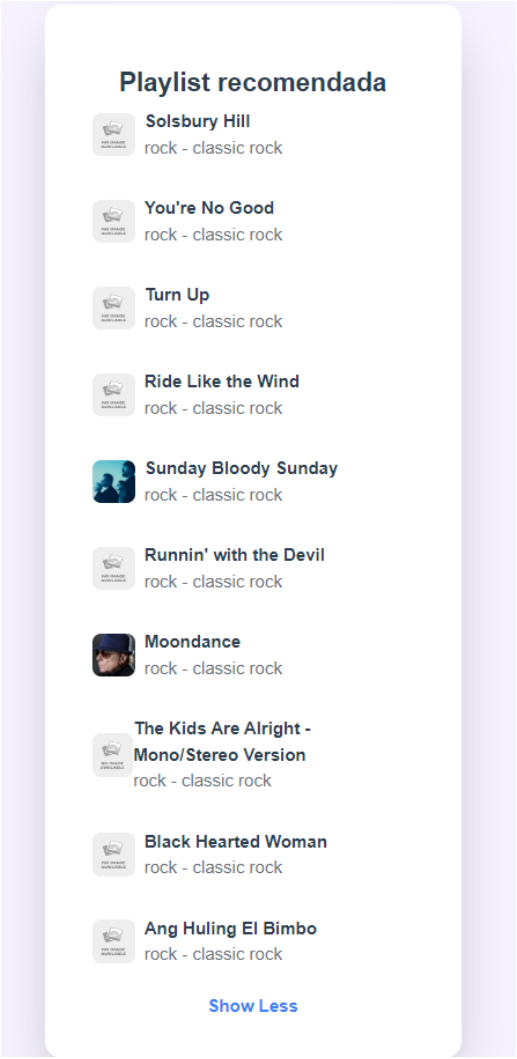

Fig. 10: Playlist recomendada para rock e classic rock

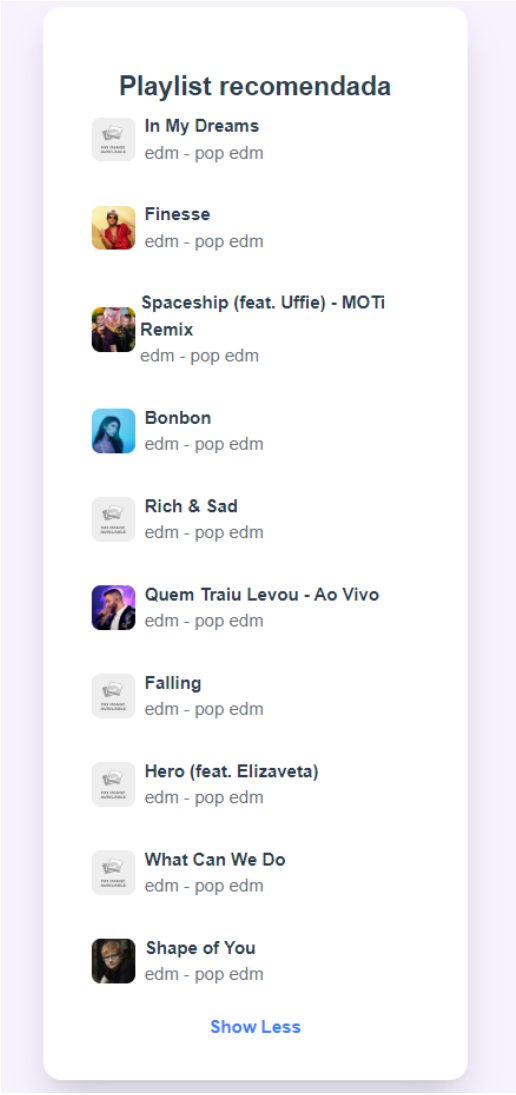

Fig. 11: Playlist recomendada para edm e edm pop

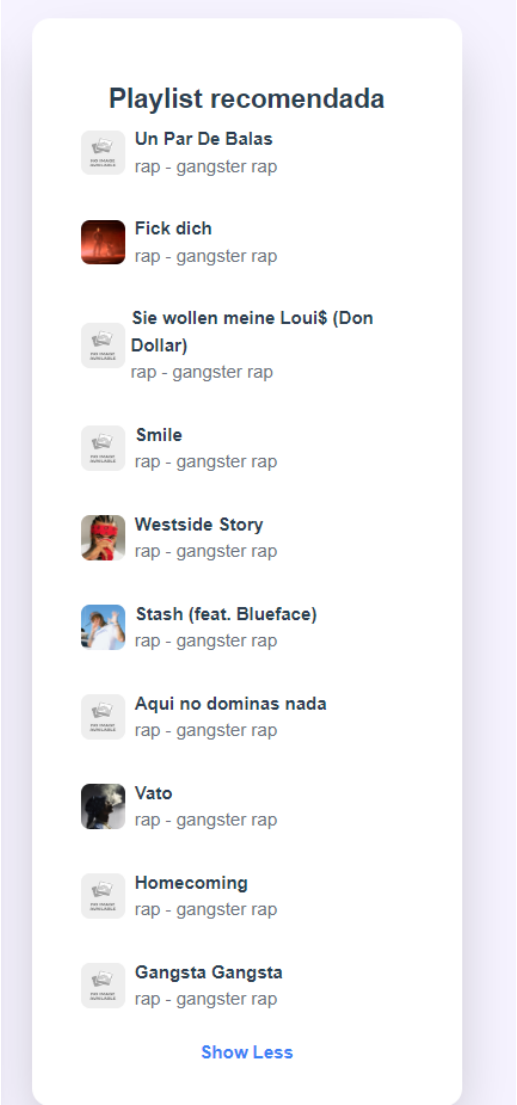

Fig. 12: Playlist recomendada para rap e gangster rap

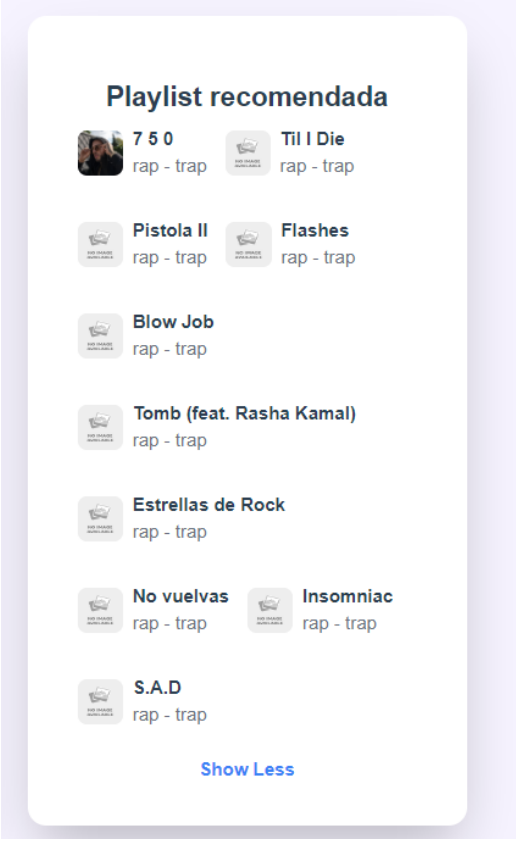

Fig. 13: Playlist recomendada para rap e gangster rap

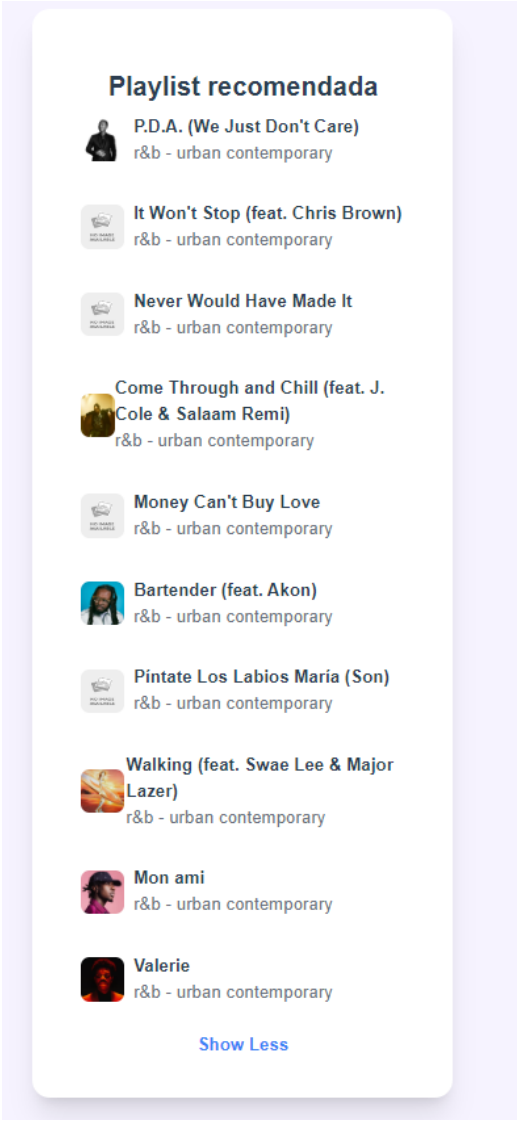

Fig. 14: Playlist recomendada rb e urban contemporary