Ótimo, você está se preparando para um desafio técnico que envolve:

1. **Manipulação de dados com JavaScript (Node.js)**
2. **Integração com uma API externa (Spotify)**
3. **Persistência em banco de dados (PostgreSQL)**
4. **Exibição dos dados no terminal**
5. **Boas práticas e segurança da informação**

Vamos montar um **guia rápido**, com **exemplos de código e estrutura clara**, que você pode seguir durante a entrevista.

**🧠 Estrutura de Solução (Resumo Mental Rápido)**

1. Autenticar na API do Spotify

2. Buscar músicas mais ouvidas

3. Filtrar as 10 mais recentes

4. Buscar músicas dos artistas

5. Armazenar no PostgreSQL com segurança

6. Exibir no terminal

**🛠️ Tecnologias e Boas Práticas**

* **Node.js** com axios para requisições HTTP
* **pg** para PostgreSQL
* **dotenv** para variáveis de ambiente (⚠️ Segurança!)
* **SQL parametrizado** (❌ sem SQL injection)
* **Async/Await** (Código limpo)
* **Modularização** (Separar responsabilidades)

**📁 Estrutura de Projeto Recomendada**

spotify-script/

├── .env

├── index.js

├── db.js

├── spotify.js

├── utils.js

├── package.json

**1️⃣ Instalação Inicial**

npm init -y

npm install axios pg dotenv

**2️⃣ .env (⚠️ Segurança)**

SPOTIFY\_CLIENT\_ID=your\_client\_id

SPOTIFY\_CLIENT\_SECRET=your\_client\_secret

PGHOST=localhost

PGUSER=postgres

PGPASSWORD=your\_password

PGDATABASE=spotify\_db

PGPORT=5432

*Nunca* exponha isso em código. O uso de .env demonstra **conhecimento em segurança e boas práticas**.

**3️⃣ db.js – Conexão segura ao Postgres**

require('dotenv').config();

const { Pool } = require('pg');

const pool = new Pool({

host: process.env.PGHOST,

user: process.env.PGUSER,

password: process.env.PGPASSWORD,

database: process.env.PGDATABASE,

port: process.env.PGPORT,

});

module.exports = {

query: (text, params) => pool.query(text, params),

};

**4️⃣ spotify.js – Autenticação e busca de dados**

const axios = require('axios');

require('dotenv').config();

let token = '';

async function authenticate() {

const response = await axios.post('https://accounts.spotify.com/api/token',

'grant\_type=client\_credentials',

{

headers: {

'Content-Type': 'application/x-www-form-urlencoded',

'Authorization': 'Basic ' + Buffer.from(

process.env.SPOTIFY\_CLIENT\_ID + ':' + process.env.SPOTIFY\_CLIENT\_SECRET

).toString('base64'),

},

}

);

token = response.data.access\_token;

}

async function fetchTopTracks() {

if (!token) await authenticate();

const res = await axios.get(

'https://api.spotify.com/v1/playlists/37i9dQZEVXbMDoHDwVN2tF',

{ headers: { Authorization: `Bearer ${token}` } }

);

return res.data.tracks.items.map(item => ({

name: item.track.name,

artist: item.track.artists[0].name,

release\_date: item.track.album.release\_date,

artist\_id: item.track.artists[0].id,

}));

}

async function fetchArtistTracks(artistId) {

const res = await axios.get(

`https://api.spotify.com/v1/artists/${artistId}/top-tracks?market=US`,

{ headers: { Authorization: `Bearer ${token}` } }

);

return res.data.tracks.map(t => ({

name: t.name,

popularity: t.popularity,

}));

}

module.exports = {

fetchTopTracks,

fetchArtistTracks,

};

**5️⃣ index.js – Script principal**

const { fetchTopTracks, fetchArtistTracks } = require('./spotify');

const db = require('./db');

async function saveTrack(track) {

await db.query(

`INSERT INTO tracks (name, artist, release\_date)

VALUES ($1, $2, $3) ON CONFLICT DO NOTHING`,

[track.name, track.artist, track.release\_date]

);

}

async function main() {

console.log('Buscando músicas...');

const topTracks = await fetchTopTracks();

const sorted = topTracks

.sort((a, b) => new Date(b.release\_date) - new Date(a.release\_date))

.slice(0, 10);

for (const track of sorted) {

console.log(`🎵 ${track.name} - ${track.artist} (${track.release\_date})`);

await saveTrack(track);

const artistTracks = await fetchArtistTracks(track.artist\_id);

console.log(`📀 Mais faixas de ${track.artist}:`);

artistTracks.forEach(t => console.log(` • ${t.name} (popularity: ${t.popularity})`));

}

}

main().catch(err => console.error(err));

**6️⃣ SQL – Criação de Tabela no PostgreSQL**

Execute no **PgAdmin**:

CREATE TABLE IF NOT EXISTS tracks (

id SERIAL PRIMARY KEY,

name TEXT NOT NULL,

artist TEXT NOT NULL,

release\_date DATE,

UNIQUE(name, artist)

);

**💡 Dicas Rápidas para a Entrevista**

| **Tema** | **O que falar** |
| --- | --- |
| **Segurança** | Uso de .env para proteger credenciais, uso de queries parametrizadas (evita SQL injection). |
| **Boas práticas** | Código modular, async/await para legibilidade, tratamento de erros. |
| **Escalabilidade** | Pode comentar sobre cache/token refresh, persistência assíncrona com filas. |
| **Design** | Separação de responsabilidade: spotify.js cuida de API, db.js cuida de banco. |

**🧪 Teste rápido com Postman**

Você pode usar o Postman para testar as rotas da API do Spotify:

1. Faça um **POST** para https://accounts.spotify.com/api/token
2. Header: Authorization: Basic <base64(client\_id:client\_secret)>
3. Body: grant\_type=client\_credentials

Se quiser, posso criar um repositório base para você com todos esses arquivos prontos, ou até gerar um script bash para inicializar isso localmente.

Quer que eu prepare esse starter repo?