PROJETO – TuneCatch

API do TuneCatch –.

A API do TuneCatch – *Maestro* – serve como ponte entre a interface do usuário (frontend) e sua lógica de negócios JavaScript com a camada de informações persistentes (banco de dados), sendo responsável por processar requisições e garantir a integridade das informações trafegadas.

A arquitetura da API segue princípios RPC e RESTful, com separação clara de responsabilidades por meio de verbos HTTP (**GET**, **POST**, **PATCH**, **PUT**, **DELETE**) e estruturação de rotas semânticas que representam os recursos principais do sistema — como usuários, playlists, autenticação e reprodução de músicas.

É possível segmentá-la em endpoints de esferas distintas, classificando-os de acordo com o recurso que ela está manipulando e/ou acessando.

1. Microsserviço de Usuários (/maestro/usuario)

Este serviço lida com todas as operações relacionadas a perfis de utilizador, autenticação, seguidores e banimento.

1.1 Criar um novo Usuário

- **Rota:** POST /maestro/usuario
- **Tipo:** RESTful
- **Objetivo:** Cadastrar um novo usuário no sistema.
- **Corpo da Requisição (JSON):**

```
{
  "username": "novo_usuario",
  "email": "email@dominio.com",
  "password": "senha_forte_123",
  "displayName": "Nome de Exibição (Opcional)",
  "avatarURL": "https://url.da.imagem/avatar.png (Opcional)"
}
```

- **Respostas Possíveis:**
  - **201 Created**: Usuário criado com sucesso.

```
{
  "message": "Usuário criado com sucesso.",
  "id": 1,
  "username": "novo_usuario",
  "email": "email@dominio.com",
  "dateCreated": "2025-06-14T12:00:00Z"
}
```

  - **400 Bad Request**: Campos em falta ou inválidos.
  - **409 Conflict**: username ou email já em uso.

1.2 Autenticar um Usuário

- **Rota:** POST /maestro/usuario/fazer-login
- **Tipo:** RPC-Style
- **Objetivo:** Autenticar um usuário e obter acesso.
- **Corpo da Requisição (JSON):**

```
{
  "credential": "novo_usuario_ou_email@dominio.com",
  "password": "senha_forte_123"
}
```

- **Respostas Possíveis:**
  - **200 OK**: Login bem-sucedido.

```
{
  "message": "Login realizado com sucesso.",
  "idUserLogged": 1
}
```

  - **401 Unauthorized**: Credenciais incorretas.
  - **403 Forbidden**: Conta banida.

1.3 Obter Usuário por ID

- **Rota:** GET /maestro/usuario/{id}
- **Tipo:** RESTful
- **Objetivo:** Retornar os dados completos (exceto senha) de um usuário pelo seu ID.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "id": 1,
    "username": "stargazer",
    "displayName": "Marco",
    "email": "marco.tune@catch.com",
    "avatarURL": "https://i.pinimg.com/avatar.jpg",
    "isBanned": false,
    "isAdmin": true,
    "date_created": "2025-06-12T14:10:00Z",
    "date_updated": "2025-06-13T09:30:00Z",
    "createdPlaylists": [],
    "likedPlaylists": [],
    "following": [],
    "followers": [],
    "userBadges": []
}
```

  - **404 Not Found**: Usuário não encontrado.

1.4 Obter Usuário por Username

- **Rota:** GET /maestro/usuario/nome-usuario/{username}
- **Tipo:** RESTful
- **Objetivo:** Obter os detalhes de um usuário pelo seu nome de usuário.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "id": 1,
    "username": "stargazer",
    "displayName": "Marco",
    "email": "marco.tune@catch.com",
    "avatarURL": "https://i.pinimg.com/avatar.jpg",
    "createdPlaylists": [
        { "id": 101, "name": "late night drive" }
    ]
}
```

  - **404 Not Found**: Usuário não encontrado.

1.5 Obter todos os Usuário

- **Rota:** GET /maestro/usuario/
- **Tipo:** RESTful
- **Objetivo:** Obter todos os usuários do sistema.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.


```
{
    "users": [
        {
            "id": 1,
            "username": "Stargazer",
            "displayName": "marcola",
            "avatarURL":
"https://i.pinimg.com/736x/d9/db/14/d9db14b5f7fc1718eda42eff94
1339f2.jpg",
            "date_created": "2025-06-11T21:45:32.742Z"
        },
        {
            "id": 3,
            "username": "opiastri",
            "displayName": "leti",
            "avatarURL":
"https://pbs.twimg.com/profile_images/1922421206323519488/6ZBO
X96x_400x400.jpg",
            "date_created": "2025-06-12T23:59:34.598Z"
        }
    ]
}
```

1.6 Pesquisar Usuários por Nome de Exibição

- **Rota:** GET /maestro/usuario/nome-exibicao?q={query}
- **Tipo:** RESTful
- **Objetivo:** Buscar usuários por nome de exibição.
- **Parâmetros de Query:**
  - q: O termo a ser pesquisado no nome de exibição.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso. Retorna um objeto com uma lista de usuários.

```
{
  "users": [
    {
      "id": 1,
      "username": "stargazer",
      "displayName": "Marco",
      "avatarURL": "https://i.pinimg.com/avatar.jpg"
    }
  ]
}
```

1.7 Atualizar Perfil do Usuário

- **Rota:** PUT /maestro/usuario/{id}/detalhes
- **Tipo:** RESTful
- **Objetivo:** Atualizar os detalhes do perfil de um usuário.
- **Corpo da Requisição (JSON):**

```
{
    "currentPassword": "senha_atual_obrigatoria",
    "username": "novo_username (Opcional)",
    "email": "novo.email@dominio.com (Opcional)",
    "displayName": "Novo Nome (Opcional)",
    "avatarURL": "https://nova.url/avatar.png (Opcional)"
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "message": "Perfil atualizado com sucesso.",
  "user": {
    "id": 1,
    "username": "novo_username",
    "displayName": "Novo Nome",
    "email": "novo.email@dominio.com"
  }
}
```

  - **403 Forbidden**: Senha atual incorreta.
  - **404 Not Found**: Usuário não encontrado.
  - **409 Conflict**: Novo username ou email já em uso.

1.8 Atualizar Senha do Usuário

- **Rota:** PUT /maestro/usuario/{id}/senha
- **Tipo:** RPC-Style
- **Objetivo:** Atualizar a senha de um usuário.
- **Corpo da Requisição (JSON):**

```
{
    "currentPassword": "senha_atual_obrigatoria",
    "newPassword": "nova_senha_forte"
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "message": "Senha atualizada com sucesso."
}
```

  - **403 Forbidden**: Senha atual incorreta.

1.9 Seguir / Deixar de Seguir um Usuário

- **Rota:** POST /maestro/usuario/seguir/{id_usuario}
- **Tipo:** RPC-Style
- **Objetivo:** Adicionar ou remover um usuário da lista de "seguindo" (toggle).
- **Parâmetros da Rota:**
  - id_usuario: ID do usuário que está a realizar a ação.
- **Corpo da Requisição (JSON):**

```
{
  "idToBeFollowed": 2
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "message": "Lista de usuários seguidos atualizada!",
    "user": {
        "id": 1,
        "username": "stargazer",
        "following": [
            { "id": 2, "username": "rocketman" }
        ]
    }
}
```

  - **403 Forbidden**: Usuário a tentar seguir-se a si mesmo.
  - **404 Not Found**: Um dos utilizadores não foi encontrado.

1.10 Banir um Usuário

- **Rota:** POST /maestro/usuario/banir-perfil/{id_usuario}
- **Tipo:** RPC-Style
- **Objetivo:** Banir um usuário do sistema (requer privilégios de administrador).
- **Parâmetros da Rota:**
  - id_usuario: ID do administrador a realizar a ação.
- **Corpo da Requisição (JSON):**

```
{
  "idToBeBanned": 3
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "message": "Usuário banido com sucesso!",
  "userBanned": {
    "id": 3,
    "username": "usuario_banido"
  }
}
```

  - **401 Unauthorized**: O requisitante não é um administrador.
  - **404 Not Found**: A conta a ser banida não existe.
  - **409 Conflict**: A conta já está banida.

1.11 Deletar um Usuário

- **Rota:** DELETE /maestro/usuario/{id_usuario}
- **Tipo:** RESTful
- **Objetivo:** Excluir permanentemente um perfil de usuário.
- **Respostas Possíveis:**
    - **204 No Content**: Sucesso. (Nenhum corpo de resposta).
    - **404 Not Found**: A conta a ser excluída não existe.

2. Endpoints para Playlists (/maestro/playlist)

Este serviço lida com a criação, gestão e interação com as playlists do sistema.

2.1 Criar uma nova Playlist

- **Rota:** POST /maestro/playlist/{id_usuario}
- **Tipo:** RESTful
- **Objetivo:** Criar uma nova playlist para um usuário.
- **Corpo da Requisição (JSON):**

```
{
  "name": "Minha Nova Playlist",
  "description": "Uma breve descrição da playlist (Opcional)",
  "isVisible": true,
  "coverImageURL": "https://url.da.imagem/capa.png (Opcional)"
}
```

- **Respostas Possíveis:**
  - **201 Created**: Playlist criada com sucesso.

```
{
  "message": "Playlist criada com sucesso!",
  "playlist": {
    "id": 101,
    "name": "Minha Nova Playlist",
    "creatorId": 1
  }
}
```

  - **404 Not Found**: Usuário criador não encontrado.

2.2 Atualizar uma Playlist

- **Rota:** PUT /maestro/playlist/{id_playlist}
- **Tipo:** RESTful
- **Objetivo:** Atualizar os detalhes de uma playlist existente.
- **Corpo da Requisição (JSON):**

```
{
  "name": "Novo Nome da Playlist (Opcional)",
  "description": "Nova descrição (Opcional)",
  "isVisible": false,
  "coverImageURL": "https://nova.url/capa.png (Opcional)"
}
```

- **Respostas Possíveis:**
  - **200 OK**: Playlist atualizada com sucesso.

```
{
  "message": "Playlist atualizada com sucesso!",
  "playlist": {
    "id": 101,
    "name": "Novo Nome da Playlist",
    "isVisible": false
  }
}
```

  - **404 Not Found**: Playlist não encontrada.

2.3 Deletar uma Playlist

- **Rota:** DELETE /maestro/playlist/{id_playlist}
- **Tipo:** RESTful
- **Objetivo:** Excluir uma playlist existente.
- **Respostas Possíveis:**
  - **204 No Content**: Sucesso. (Nenhum corpo de resposta).
  - **404 Not Found**: Playlist não encontrada.

2.4 Obter todas as Playlists

- **Rota:** GET /maestro/playlist
- **Tipo:** RESTful
- **Objetivo:** Buscar todas as playlists do sistema.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "playlists": [
    {
      "id": 101,
      "name": "Minha Nova Playlist",
      "creator": { "id": 1, "username": "stargazer" }
    }
  ]
}
```

2.5 Obter Playlist por ID

- **Rota:** GET /maestro/playlist/{id_playlist}
- **Tipo:** RESTful
- **Objetivo:** Buscar uma playlist específica pelo seu ID.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

    ```
    {
      "id": 101,
      "name": "Minha Nova Playlist",
      "description": "Uma breve descrição.",
      "isVisible": true,
      "likes": 50,
      "creator": { "id": 1, "username": "stargazer" }
    }
    ```

  - **404 Not Found**: Playlist não encontrada.

2.6 Pesquisar Playlists Públicas por Nome

- **Rota:** GET /maestro/playlist/publica/{nome}
- **Tipo:** RESTful
- **Objetivo:** Buscar playlists públicas por nome (busca parcial).
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "playlists": [
    {
      "id": 101,
      "name": "Minha Nova Playlist",
      "creator": { "id": 1, "username": "stargazer" }
    }
  ]
}
```

2.7 Obter todas as Playlists de um Usuário

- **Rota:** GET /maestro/playlist/usuario/{id_usuario}
- **Tipo:** RESTful
- **Objetivo:** Buscar todas as playlists (públicas e privadas) criadas por um usuário.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "playlists": [
    { "id": 101, "name": "Minha Nova Playlist", "isVisible":
true }
  ]
}
```

  - **404 Not Found**: Usuário não encontrado.

2.8 Obter Playlists Públicas de um Usuário

- **Rota:** GET /maestro/playlist/usuario-publicas/{id_usuario}
- **Tipo:** RESTful
- **Objetivo:** Buscar apenas as playlists públicas criadas por um usuário.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "playlists": [
    { "id": 101, "name": "Minha Nova Playlist", "isVisible": true }
  ]
}
```

  - **404 Not Found**: Usuário não encontrado.

2.9 Curtir / Descurtir uma Playlist

- **Rota:** POST /maestro/playlist/curtir/{id_usuario}
- **Tipo:** RPC-Style
- **Objetivo:** Adicionar ou remover um "like" de um usuário em uma playlist.
- **Corpo da Requisição (JSON):**

```
{
  "playlistId": 101
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "message": "Playlist curtida com sucesso.",
  "newLikeCount": 51
}
```

  - **404 Not Found**: Usuário ou playlist não encontrados.
  - **409 Conflict**: Usuário tentando curtir a própria playlist.

2.10 Alterar Visibilidade da Curtida

- **Rota:** POST /maestro/playlist/curtir/alterar-visibilidade/{id_usuario}
- **Tipo:** RPC-Style
- **Objetivo:** Alterar a visibilidade de uma curtida no perfil do usuário.
- **Corpo da Requisição (JSON):**

```
{
  "playlistId": 101,
  "isVisible": false
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "message": "Visibilidade da curtida atualizada com
sucesso!"
}
```

  - **404 Not Found**: A curtida especificada não foi encontrada para este usuário.

2.11 Obter todas as Playlists Curtidas por um Usuário

- **Rota:** GET /maestro/playlist/curtidas/{id_usuario}
- **Tipo:** RESTful
- **Objetivo:** Buscar todas as playlists que um usuário curtiu.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "likedPlaylists": [
    {
      "id": 102,
      "name": "Outra Playlist",
      "creator": { "id": 2, "username": "rocketman" },
      "UserPlaylistLiked": {
        "likeVisibleOnProfile": true,
        "liked_at": "2025-06-14T10:00:00Z"
      }
    }
  ]
}
```

  - **404 Not Found**: Usuário não encontrado.

2.12 Obter Playlists Curtidas Públicas de um Usuário

- **Rota:** GET /maestro/playlist/curtidas-publicas/{id_usuario}
- **Tipo:** RESTful
- **Objetivo:** Buscar apenas as playlists curtidas por um usuário que ele marcou como visíveis.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "publicLikedPlaylists": [
    {
      "id": 102,
      "name": "Outra Playlist",
      "creator": { "id": 2, "username": "rocketman" }
    }
  ]
}
```

  - **404 Not Found**: Usuário não encontrado.

3. Endpoints para Badges (/maestro/badges)

Este serviço lida com a criação, gestão e atribuição de conquistas (badges) no sistema.

3.1 Criar uma nova Badge

- **Rota:** POST /maestro/badges
- **Tipo:** RESTful
- **Objetivo:** Criar uma nova badge no sistema.
- **Corpo da Requisição (JSON):**

```
{
  "name": "Explorador",
  "description": "Visitou 10 perfis de outros usuários.",
  "iconURL": "https://example.com/badge/explorer.png"
}
```

- **Respostas Possíveis:**
  - **201 Created**: Badge criada com sucesso.

```
{
    "message": "Badge criada com sucesso",
    "badge": {
        "id": 1,
        "name": "Explorador",
        "description": "Visitou 10 perfis de outros
usuários.",
        "iconURL": "https://example.com/badge/explorer.png"
    }
}
```

  - **409 Conflict**: Já existe uma badge com este nome.

3.2 Obter todas as Badges

- **Rota:** GET /maestro/badges
- **Tipo:** RESTful
- **Objetivo:** Retornar todas as badges cadastradas no sistema.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "badges": [
        {
            "id": 1,
            "name": "Explorador",
            "description": "Visitou 10 perfis diferentes.",
            "iconURL":
"https://example.com/badge/explorer.png"
        }
    ]
}
```

3.3 Obter Badge por ID

- **Rota:** GET /maestro/badges/{id}
- **Tipo:** RESTful
- **Objetivo:** Buscar uma badge específica pelo seu ID.
- **Respostas Possíveis:**
  - **200 OK**:
  - **404 Not Found**: Badge não encontrada.

3.4 Obter Badge por Nome

- **Rota:** GET /maestro/badges/nome/{name}
- **Tipo:** RESTful
- **Objetivo:** Buscar uma badge específica pelo seu nome.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.
  - **404 Not Found**: Badge não encontrada.

3.5 Atualizar uma Badge

- **Rota:** PUT /maestro/badges/{id}
- **Tipo:** RESTful
- **Objetivo:** Atualizar os detalhes de uma badge existente.
- **Corpo da Requisição (JSON):**

```
{
  "name": "Explorador Mestre",
  "description": "Visitou 50 perfis de outros usuários.",
  "iconURL": "https://example.com/badge/master_explorer.png"
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.
  - **404 Not Found**: Badge não encontrada.
  - **409 Conflict**: O novo nome já está em uso por outra badge.

3.6 Deletar uma Badge

- **Rota:** DELETE /maestro/badges/{id}
- **Tipo:** RESTful
- **Objetivo:** Excluir uma badge do sistema.
- **Respostas Possíveis:**
  - **204 No Content**: Sucesso.
  - **404 Not Found**: Badge não encontrada.

3.7 Conceder uma Badge a um Usuário

- **Rota:** POST /maestro/badges/conceder/{id_usuario}
- **Tipo:** RPC-Style
- **Objetivo:** Atribuir uma badge a um usuário.
- **Corpo da Requisição (JSON):**

```
{
  "badgeId": 1,
  "isVisibleOnProfile": true
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.
    { "message": "Badge concedida ao usuário com sucesso!" }
  - **404 Not Found**: Usuário ou badge não encontrados.
  - **409 Conflict**: O usuário já possui esta badge.

3.8 Alterar Visibilidade da Badge

- **Rota:** POST /maestro/badges/alterar-visibilidade/{id_usuario}
- **Tipo:** RPC-Style
- **Objetivo:** Alterar a visibilidade de uma badge que um usuário possui.
- **Corpo da Requisição (JSON):**

```
{
  "badgeId": 1,
  "isVisible": false
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

    ```
    { "message": "Visibilidade da badge atualizada com sucesso!" }
    ```

  - **404 Not Found**: Associação usuário-badge não encontrada.

3.9 Obter Badges de um Usuário

- **Rota:** GET /maestro/badges/usuario/{id_usuario}
- **Tipo:** RESTful
- **Objetivo:** Buscar todas as badges que um usuário possui.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "userId": 1,
  "username": "stargazer",
  "badges": [
    {
      "id": 1,
      "name": "Explorador",
      "description": "...",
      "iconURL": "...",
      "UserBadge": {
        "isVisibleOnProfile": true,
        "granted_at": "2025-06-15T13:00:00Z"
      }
    }
  ]
}
```

  - **404 Not Found**: Usuário não encontrado.

4. Endpoints para Tags (/maestro/tag)

Este serviço lida com a criação e gestão de tags, e a sua associação com playlists.

4.1 Criar uma nova Tag

- **Rota:** POST /maestro/tag
- **Tipo:** RESTful
- **Objetivo:** Criar uma nova tag no sistema.
- **Corpo da Requisição (JSON):**

```
{
  "name": "Road Trip",
  "category": "Ocasião",
  "iconEmoji": "🚗"
}
```

- **Respostas Possíveis:**
  - **201 Created**: Sucesso.

```
{
    "message": "Tag criada com sucesso!",
    "tag": {
        "id": 1,
        "name": "Viagem de Carro",
        "category": "Ocasião",
        "iconEmoji": "🚗"
    }
}
```

  - **400 Bad Request**: Campos em falta ou categoria inválida.
  - **409 Conflict**: Nome da tag já em uso.

4.2 Obter todas as Tags

- **Rota:** GET /maestro/tag
- **Tipo:** RESTful
- **Objetivo:** Buscar todas as tags do sistema.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "tags": [
        {
            "id": 1,
            "name": "Viagem de Carro",
            "category": "Ocasião",
            "iconEmoji": "🚗"
        }
    ]
}
```

4.3 Obter Tag por ID

- **Rota:** GET /maestro/tag/{id}
- **Tipo:** RESTful
- **Objetivo:** Buscar uma tag específica pelo seu ID.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.
  - **404 Not Found**: Tag não encontrada.

4.4 Pesquisar Tags por Nome

- **Rota:** GET /maestro/tag/nome/{name}
- **Tipo:** RESTful
- **Objetivo:** Buscar tags por nome (busca parcial e case-insensitive).
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "tags": [
        { "id": 1, "name": "Viagem de Carro" },
        { "id": 2, "name": "Viagem para a Praia" }
    ]
}
```

4.5 Atualizar uma Tag

- **Rota:** PUT /maestro/tag/{id}
- **Tipo:** RESTful
- **Objetivo:** Atualizar os detalhes de uma tag existente.
- **Corpo da Requisição (JSON):**

```
{
  "name": "Roadtrip Noturna",
  "category": "Ocasião",
  "iconEmoji": "🌃"
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.
  - **404 Not Found**: Tag não encontrada.
  - **409 Conflict**: O novo nome já está em uso.

4.6 Deletar uma Tag

- **Rota:** DELETE /maestro/tag/{id}
- **Tipo:** RESTful
- **Objetivo:** Excluir uma tag do sistema.
- **Respostas Possíveis:**
    - **204 No Content**: Sucesso.
    - **404 Not Found**: Tag não encontrada.

4.7 Adicionar / Remover Tag de uma Playlist

- **Rota:** POST /maestro/tag/adicionar/{id_playlist}
- **Tipo:** RPC-Style
- **Objetivo:** Adicionar ou remover uma tag de uma playlist (toggle).
- **Corpo da Requisição (JSON):**

```
{
  "tagId": 5
}
```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "message": "A tag 'Viagem de Carro' foi adicionada à
playlist 'Estrada'.",
    "playlist": { "...objeto da playlist atualizada..." }
}
```

  - **404 Not Found**: Playlist ou tag não encontradas.

4.8 Buscar Playlists por Tags

- **Rota:** GET /maestro/tag/playlists-marcadas?q={tags}
- **Tipo:** RESTful
- **Objetivo:** Buscar playlists que contenham uma ou mais tags específicas.
- **Parâmetros de Query:**
  - q: Nomes de tags separados por vírgula (ex: Rock,80s).
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "playlists": [
        { "id": 105, "name": "Rock Classics" }
    ]
}
```

4.9 Obter Tags de uma Playlist

- **Rota:** GET /maestro/tag/playlists-tags/{id_playlist}
- **Tipo:** RESTful
- **Objetivo:** Buscar todas as tags associadas a uma playlist específica.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
    "tags": [
      {
          "id": 1,
          "name": "RoadTrip",
          "category": "Ocasião"
      }
      ]
}
```

  - **404 Not Found**: Playlist não encontrada.

5. Endpoints para Músicas (Tracks) (/maestro/track)

Este serviço lida com a busca de músicas no Spotify e a sua gestão nas playlists.

5.1 Pesquisar Músicas no Spotify

- **Rota:** GET /maestro/track/pesquisa/{nome}
- **Tipo:** RESTful
- **Objetivo:** Retornar uma lista de músicas do Spotify com base num termo de pesquisa.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "tracks": [
    {
      "id": "spotify_track_id",
      "name": "Daylight",
      "artists": "David Kushner",
      "duration": "3:32",
      "album": {
        "name": "Daylight",
        "coverImageUrl": "https://url.to/image.jpg"
      }
    }
  ]
}
```

  - **500 Internal Server Error**: Falha na comunicação com a API do Spotify.

5.2 Obter Playlists que Contêm uma Música

- **Rota:** GET /maestro/track/playlists-com-musica/{nome_musica}
- **Tipo:** RESTful
- **Objetivo:** Obter playlists que contêm uma música com o nome específico (ignorando espaços e maiúsculas/minúsculas).
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "playlists": [
    {
      "id": 101,
      "name": "sad songs",
      "creator": { "id": 1, "username": "stargazer" }
    }
  ]
}
```

5.3 Adicionar uma Música a uma Playlist

- **Rota:** POST /maestro/track/adicionar/{id_track}
- **Tipo:** RPC-Style
- **Objetivo:** Adicionar uma música do Spotify a uma playlist.
- **Parâmetros da Rota:**
  - id_track: ID da música no **Spotify**.
- **Corpo da Requisição (JSON):**
  ```
  { "playlistId": 101 }
  ```

- **Respostas Possíveis:**
  - **200 OK**: Sucesso.
    ```
    { "message": "A música 'Daylight' foi adicionada à playlist 'sad songs' com sucesso!" }
    ```

  - **400 Bad Request**: ID da playlist em falta.
  - **404 Not Found**: Música não encontrada no Spotify ou playlist não encontrada no sistema.

5.4 Obter Músicas de uma Playlist

- **Rota:** GET /maestro/track/playlist-musicas/{id_playlist}
- **Tipo:** RESTful
- **Objetivo:** Obter a lista de músicas de uma playlist específica, ordenadas pela posição.
- **Respostas Possíveis:**
  - **200 OK**: Sucesso.

```
{
  "tracks": [
    {
      "id": 205,
      "spotifyID": "spotify_id_123",
      "trackName": "Daylight",
      "artistName": "David Kushner",
      "albumName": "Daylight",
      "albumCoverURL": "https://url.to/image.jpg",
      "durationMs": 212933,
      "PlaylistTrack": {
          "position": 1
      }
    }
  ]
}
```

  - **404 Not Found**: Playlist não encontrada.

5.5 Remover uma Música de uma Playlist

- **Rota:** DELETE /maestro/track/{id_playlist}/{id_musica}
- **Tipo:** RESTful
- **Objetivo:** Remover uma música de uma playlist. Se a música ficar "órfã", é excluída da base de dados.
- **Parâmetros da Rota:**
  - id_playlist: ID da playlist no seu sistema.
  - id_musica: ID da música no seu sistema.
- **Respostas Possíveis:**
  - **204 No Content**: Sucesso.
  - **404 Not Found**: Playlist ou música não encontradas.