MyMusic

Dinâmica de trabalho:

- **30 minutos**: tempo inicial para explicar o desafio e para o grupo estruturar as tarefas entre si (planning).
- **1,5 a 2 horas**: grupo se divide para execução e usa esse tempo para desenvolver as tarefas planejadas.
- **30 minutos**: retrospectiva onde cada participante pode expor sua solução e dificuldades encontradas.

Introdução:

Um cliente possui uma aplicação em código legado para controle de músicas favoritas de seus usuários. É uma aplicação onde o usuário pode interagir com sua lista de músicas favoritas, com um banco de dados para armazenar as informações e um serviço de APIs que realizam a integração da interface cliente com o banco de dados.

Após análises de desempenho foi identificado que existem gargalos de performance na aplicação legada, principalmente na camada de APIs, hospedada em um servidor monolítico que não comporta a demanda atual e não permite redimensionamento de servidores de maneira fácil e rápida para atender maior demanda.

Foi então solicitado o desenvolvimento de novos serviços para substituir a camada de APIs, utilizando o mesmo banco de dados existente. O desenvolvimento dos serviços da aplicação deve seguir alguns conceitos básicos (requisitos), listados abaixo.

Obs: a aplicação cliente não será disponibilizada para desenvolvimento dos novos serviços, somente o banco de dados (ver tópico repositório ao fim do documento).

Requisitos:

- **Utilizar técnicas de micro-serviços**, permitindo um dimensionamento independente de APIs, agrupadas de acordo com os domínios de negócios identificados mais abaixo
  - Planejar como o deploy da aplicação será feito (12factor)
- **Documentar API**, disponibilizar página de documentação utilizando algum framework consolidado
  - Utilizar boas práticas de APIs REST e organização/nomenclatura do código.
- **Manter compatibilidade legado**, utilizar o banco de dados disponibilizado que contém todas as informações do legado.
- **Qualidade de código**, pensar numa maneira de garantir a qualidade do código sendo que a intenção é evoluir as rotas existentes com certa frequência.

Narrativa de Negócio:

As funcionalidades básicas da aplicação que acessam as APIs compreendem nas 4 ações principais descritas abaixo

- Permitir ao usuário buscar novas músicas no banco de dados:
  1. Serviço deve validar se usuário informou ao menos 3 caracteres, retornando um HTTP 400 caso a consulta tenha menos de 3 caracteres
  2. Busca deve ser realizada tanto na coluna de nome de artista quanto nome da música
  3. Busca por música não deve ser *case sensitive*
  4. Busca deve retornar valores **contendo** o filtro, não necessitando ser informado o nome completo de música ou artista
  5. Retorno deve estar ordenado pelo nome do artista e depois pelo nome da música
- Permitir ao usuário informar o 'nome do usuário' para buscar sua playlist:
  1. Usuário deve informar um nome de usuário válido, retornando um 404 caso não encontre
  2. Busca por usuário deve ser *case sensitive*
- Permitir ao usuário escolher as músicas do resultado da busca que deseja adicionar na sua playlist:

1. Deve receber um request contendo o identificador da música e o identificador da playlist
2. Deve validar se o identificador da música e o identificador da playlist existem

- Permitir ao usuário remover músicas de sua playlist:
    3. Deve receber um request contendo o identificador da música e o identificador da playlist
    4. Deve validar se o identificador da música e o identificador da playlist existem

Narrativa Técnica:

Multi-plataforma:

- Utilizar **.NET Core**. Escolha da **IDE** é livre, fica a cargo de cada desenvolvedor

Micro-serviços:

- Existem dois domínios de negócios que foram identificados e suas APIs devem ser construídas permitindo a possibilidade de serem "levantadas" em servidores de forma independente.
    1. **Músicas - Busca de Músicas**: domínio deve conter API de busca de Músicas
    2. **Playlist - Controle de Playlist**: domínio deve conter APIs de busca e alterações de playlist.

- API de listagem de Musicas:
    - Método: **GET**
    - Rota: **`/api/musicas?filtro='Bruno+Mars'`**
    - Parâmetros: **`QueryString: {filtro} string - Opcional`**
    - Retorno: **`Array do objeto "Musica" do modelo Json`**
    - Erros tratados: **204** com array vazio quando não houver dados e **400** quando caracteres de busca menor que 3
- API de Playlist de Usuário
    - Método: **GET**
    - Rota: **`/api/playlists?user='Robson'`**
    - Parâmetros: **`QueryString: {user} string - Obrigatorio`**
    - Retorno: **`Objeto "Playlist" do modelo Json`**
    - Erros tratados: **204** quando não encontrar usuário
- API de Adicionar relação de Musicas na Playlist
    - Método: **PUT**
    - Rota: **`/api/playlists/{playlistId}/musicas`**
    - Parâmetros:
        - **`Url: Path param: {playlistId} string`**
        - **`Body: Array do objeto "Musica" do modelo Json`**
    - Retorno: **200 OK**
    - Erros tratados: **400** quando identificadores não forem encontrados no banco
- API de Remover relação de Músicas da Playlist
    - Método: **DELETE**
    - Rota: **`/api/playlists/{playlistId}/musicas/{musicaId}`**
    - Parâmetros:
        - **`Url: {playlistId}`**
    - Retorno: **200 OK**
    - Erros: **400** quando identificadores não forem encontrados no banco

Banco de dados:

- O banco de dados é um SQLite e está localizado no diretório: **`"./database/MyMusic.db"`**

Documentação:

- As APIs devem estar documentadas na home de cada dominio/endpoint de API

Desempenho:

- Na API de GET de músicas, pensar numa maneira de guardar pesquisas pelo mesmo termo por 10min

Modelo de Dados

JSON:

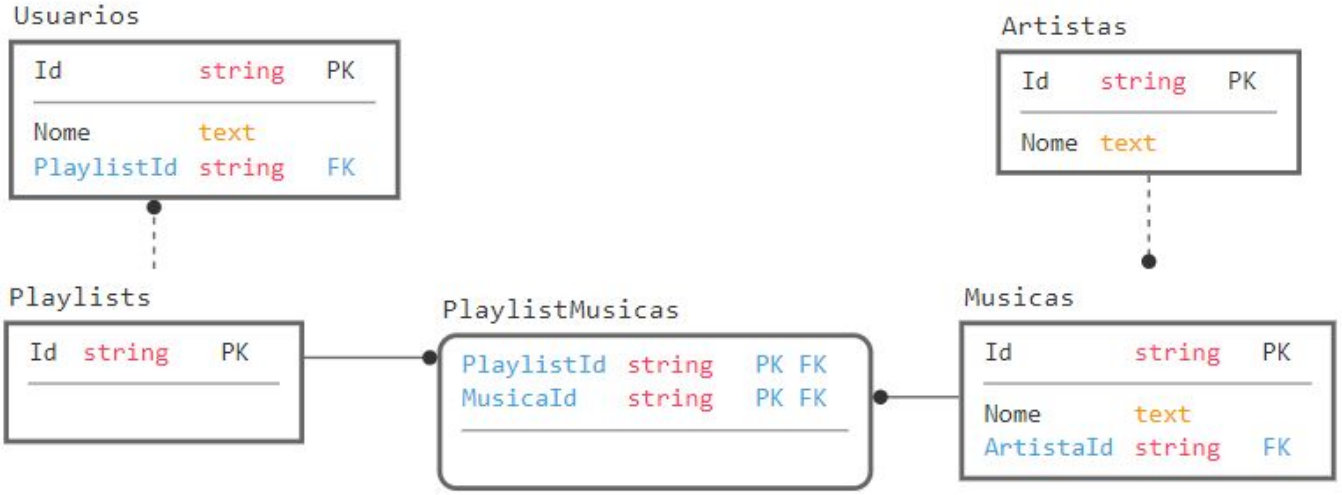
- Objeto **“Musica”**

```
{
  "id": "string",
  "nome": "string",
  "artistaId": "string",
  "artista": {
    "id": "string",
    "nome": "string"
  }
}
```

- Objeto **“Playlist”**

```
{
  "id": "string",
  "playlistMusicas": [
    {
      "playlistId": "string",
      "musicaId": "string",
      "musica": {
        "id": "string",
        "nome": "string",
        "artistaId": "string",
        "artista": {
          "id": "string",
          "nome": "string"
        }
      }
    }
  ],
  "usuario": {
    "id": "string",
    "nome": "string",
    "playlistId": "string"
  }
}
```

BANCO:



Observação:
Não é necessário, porém é possível utilizar uma ferramenta para abrir e visualizar o arquivo **MyMusic.db** de maneira mais fácil, como: https://sqlitestudio.pl/index.rvt