**FlixFilmes**

**Funcionalidades**

**Autenticação de Usuário**

* Os usuários podem se cadastrar e fazer login para acessar o aplicativo.
* O sistema de autenticação é baseado em JWT.
* Os usuários podem atualizar suas informações pessoias, como senha e email, por exemplo.

**Cartazes de Filmes**

* Os usuários podem ver os filmes em destaques a partir da API The movie Database (TMDb)
* A aplicação consome a API pública do TMDb para obter dados como sinopse, elenco, avaliações e mais.

**História**

**Registro Usuário e Login**

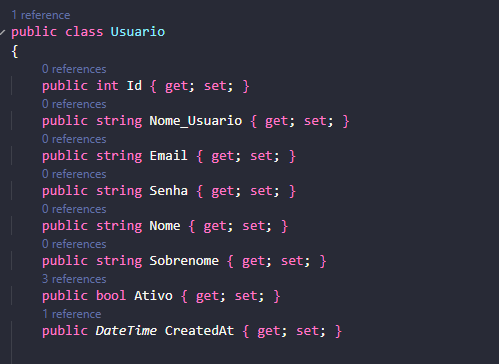
* Como usuário, posso me registrar na plataforma fornecendo meu nome, e-mail e senha para acessar as funcionalidades do FlixFilmes.
* Como usuário, posso fazer login usando minhas credenciais para acessar minha conta existente.

• **Filmes em Cartazes**

* Como usuário autenticado, posso visualizar os filmes em destaque.

**Back - end**

**Domínio - Usuario**

****

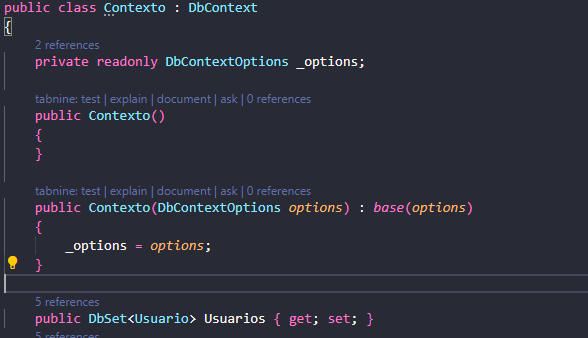
- O usuário deve possuir um construtor que instância a hora da criação e seta a propriedade ‘Ativo’ como true;

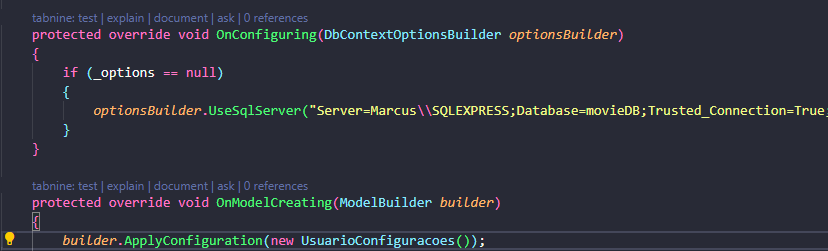
- O usuário deve possuir os métodos de Restaurar e Deletar;

**Repositorio - Usuario**

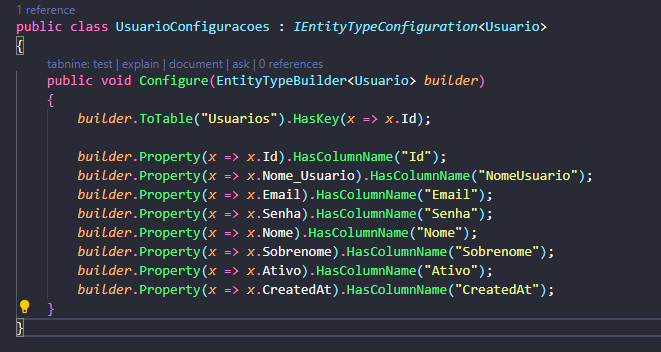
- O Repositorio deve conter as configurações do banco e o Contexto que possui a string de conexão com o banco.

**- Contexto**





**- Configurações EntityFramework**



**Repositorio – Usuario (Métodos)**

• **GetByIdAsync**

* Descrição**:** Obtém um usuário pelo seu ID.
* Parâmetros**:**
  + id: Inteiro representando o ID do usuário a ser obtido.
  + ativo (opcional): Booleano indicando se o usuário deve estar ativo (true) ou não. Padrão é true.
* Retorno**:** Uma tarefa (Task) que retorna um objeto Usuario correspondente ao ID fornecido.

• **GetAllAsync**

* Descrição**:** Obtém todos os usuários do sistema.
* Parâmetros**:**
  + ativo: Booleano indicando se devem ser retornados apenas usuários ativos (true) ou todos os usuários (false).
* Retorno**:** Uma tarefa (Task) que retorna uma coleção (IEnumerable) contendo todos os usuários correspondentes ao filtro especificado.

• **FindAsync**

* Descrição**:** Busca usuários com base em um predicado fornecido.
* Parâmetros**:**
  + predicate: Expressão lambda que especifica a condição de busca dos usuários.
* Retorno**:** Uma tarefa (Task) que retorna uma coleção (IEnumerable) de usuários que satisfazem o predicado.

• **FindEmailAsync**

* Descrição**:** Obtém um usuário pelo seu endereço de e-mail.
* Parâmetros**:**
  + email: String contendo o endereço de e-mail do usuário a ser obtido.
* Retorno**:** Uma tarefa (Task) que retorna um objeto Usuario correspondente ao endereço de e-mail fornecido.

• **AddAsync**

* Descrição**:** Adiciona um novo usuário ao sistema.
* Parâmetros**:**
  + entity: Objeto do tipo Usuario contendo as informações do usuário a ser adicionado.
* Retorno**:** Uma tarefa (Task) que retorna o ID do usuário adicionado.

• **UpdateAsync**

* Descrição**:** Atualiza as informações de um usuário existente no sistema.
* Parâmetros**:**
  + entity: Objeto do tipo Usuario contendo as novas informações do usuário.
* Retorno**:** Uma tarefa (Task) que indica se a operação de atualização foi bem-sucedida (true) ou não (false).

• **DeleteAsync**

* Descrição**:** Deleta um usuário do sistema.
* Parâmetros**:**
  + entity: Objeto do tipo Usuario a ser deletado.
* Retorno**:** Uma tarefa (Task) que indica a conclusão da operação de deleção.

**Aplicação - Usuario**

• **SalvarUsuario**

* Descrição**:** Salva um novo usuário no sistema.
* Parâmetros**:**
  + usuario: Objeto do tipo Usuario contendo as informações do usuário a ser salvo.
* Retorno**:** Uma tarefa (Task) que representa a conclusão da operação, retornando o ID do usuário salvo.

• **Atualizar**

* Descrição**:** Atualiza as informações de um usuário existente no sistema.
* Parâmetros**:**
  + usuario: Objeto do tipo Usuario contendo as novas informações do usuário.
* Retorno**:** Uma tarefa (Task) que indica se a operação de atualização foi bem-sucedida (true) ou não (false).

• **AtualizarSenha**

* Descrição**:** Atualiza a senha de um usuário existente, verificando a senha antiga.
* Parâmetros**:**
  + usuario: Objeto do tipo Usuario contendo o ID do usuário e a nova senha.
  + senhaAntiga: String contendo a senha antiga do usuário para verificação.
* Retorno**:** Uma tarefa (Task) que indica se a operação de atualização de senha foi bem-sucedida (true) ou não (false).

• **ObterUsuario**

* Descrição**:** Obtém um usuário pelo seu ID.
* Parâmetros**:**
  + id: Inteiro representando o ID do usuário a ser obtido.
* Retorno**:** Uma tarefa (Task) que retorna um objeto Usuario correspondente ao ID fornecido.

• **ObterEmail**

* Descrição**:** Obtém um usuário pelo seu endereço de e-mail.
* Parâmetros**:**
  + email: String contendo o endereço de e-mail do usuário a ser obtido.
* Retorno**:** Uma tarefa (Task) que retorna um objeto Usuario correspondente ao endereço de e-mail fornecido.

• **Deletar**

* Descrição**:** Deleta um usuário do sistema pelo seu ID.
* Parâmetros**:**
  + id: Inteiro representando o ID do usuário a ser deletado.
* Retorno**:** Uma tarefa (Task) que indica se a operação de deleção foi bem-sucedida (true) ou não (false).

• **Restaurar**

* Descrição**:** Restaura um usuário previamente deletado, tornando-o ativo novamente.
* Parâmetros**:**
  + id: Inteiro representando o ID do usuário a ser restaurado.
* Retorno**:** Uma tarefa (Task) que indica se a operação de restauração foi bem-sucedida (true) ou não (false).

• **GetAll**

* Descrição**:** Obtém todos os usuários do sistema, opcionalmente filtrando por usuários ativos ou inativos.
* Parâmetros**:**
  + ativo: Booleano indicando se devem ser retornados apenas usuários ativos (true) ou todos os usuários (false).
* Retorno**:** Uma tarefa (Task) que retorna uma coleção (IEnumerable) contendo todos os usuários correspondentes ao filtro especificado.

**Api – Controller Usuario**

• **Obter Usuário**

* Descrição**:** Obtém um usuário pelo seu ID.
* Rota**:** GET /Buscar/{id}
* Parâmetros**:**
  + id (na rota): Inteiro representando o ID do usuário a ser obtido.
* Retorno**:** Um objeto UsuarioResposta contendo os detalhes do usuário ou uma mensagem de erro.

• **GetAll**

* Descrição**:** Obtém todos os usuários do sistema.
* Rota**:** GET /BuscarTodos
* Parâmetros**:**
  + ativo (na query): Booleano indicando se os usuários a serem retornados devem estar ativos.
* Retorno**:** Uma lista de objetos UsuarioResposta contendo todos os usuários ou uma mensagem de erro.

• **AddUsuario**

* Descrição**:** Adiciona um novo usuário ao sistema.
* Rota**:** POST /Add
* Parâmetros**:**
  + Corpo da requisição: Um objeto CriarUsuario contendo os detalhes do usuário a ser adicionado.
* Retorno**:** O ID do usuário adicionado ou uma mensagem de erro.

• **Logar**

* Descrição**:** Autentica um usuário no sistema.
* Rota**:** POST /Logar
* Parâmetros**:**
  + Corpo da requisição: Um objeto AutenticarUsuario contendo os dados de login do usuário.
* Retorno**:** Um objeto contendo uma mensagem de sucesso e o usuário autenticado ou uma mensagem de erro.

• **UpdateUsuario**

* Descrição**:** Atualiza as informações de um usuário existente no sistema.
* Rota**:** PUT /Update
* Parâmetros**:**
  + Corpo da requisição: Um objeto AtualizarUsuario contendo as novas informações do usuário.
* Retorno**:** Uma confirmação da atualização ou uma mensagem de erro.

• **UpdateSenha**

* Descrição**:** Atualiza a senha de um usuário existente no sistema.
* Rota**:** PUT /UpdateSenha
* Parâmetros**:**
  + Corpo da requisição: Um objeto AtualizarSenhaUsuario contendo o ID do usuário, a nova senha e a senha antiga.
* Retorno**:** Uma confirmação da atualização ou uma mensagem de erro.

• **Desativar**

* Descrição**:** Desativa um usuário do sistema.
* Rota**:** DELETE /Desativar/{id}
* Parâmetros**:**
  + id (na rota): Inteiro representando o ID do usuário a ser desativado.
* Retorno**:** Uma confirmação da desativação ou uma mensagem de erro.

• **Ativar**

* Descrição**:** Ativa um usuário previamente desativado no sistema.
* Rota**:** PUT /Ativar/{id}
* Parâmetros**:**
  + id (na rota): Inteiro representando o ID do usuário a ser ativado.
* Retorno**:** Uma confirmação da ativação ou uma mensagem de erro.

**Utilização da API do The Movie Database (TMDb) em React**

**- Requisitos**

• Conta no [TMDb](https://www.themoviedb.org/) para obter uma chave de API.

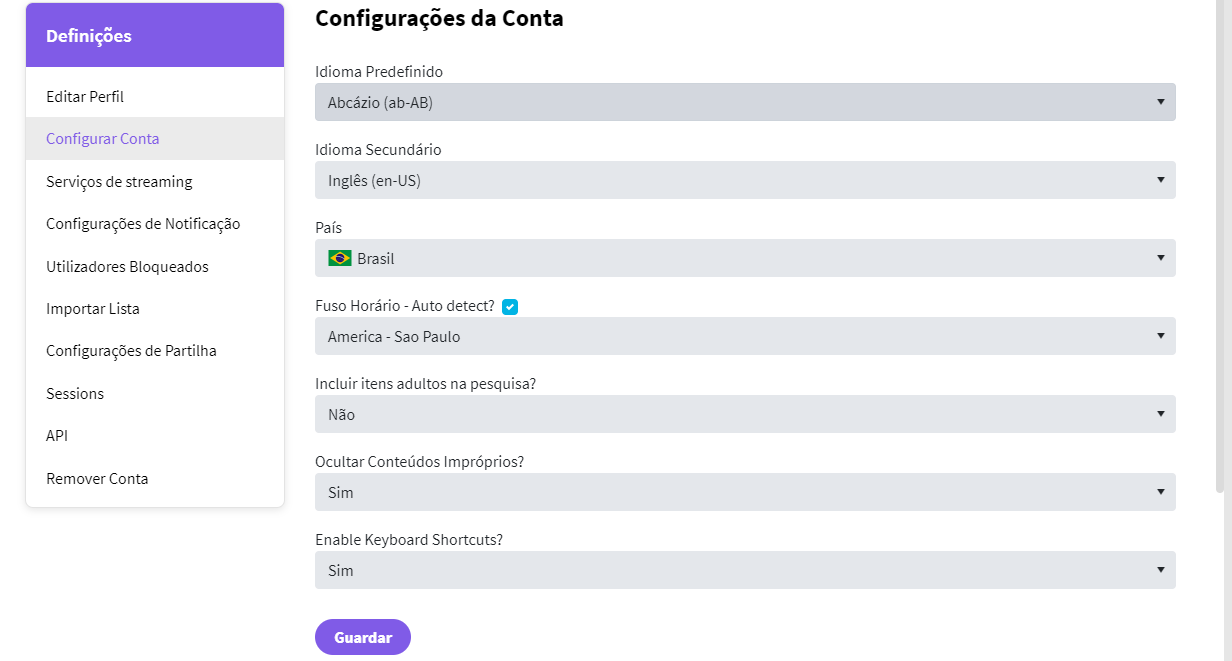
• Ambiente de desenvolvimento React configurado (create-react-app ou qualquer outro setup).

• Crie uma conta no TMDb.

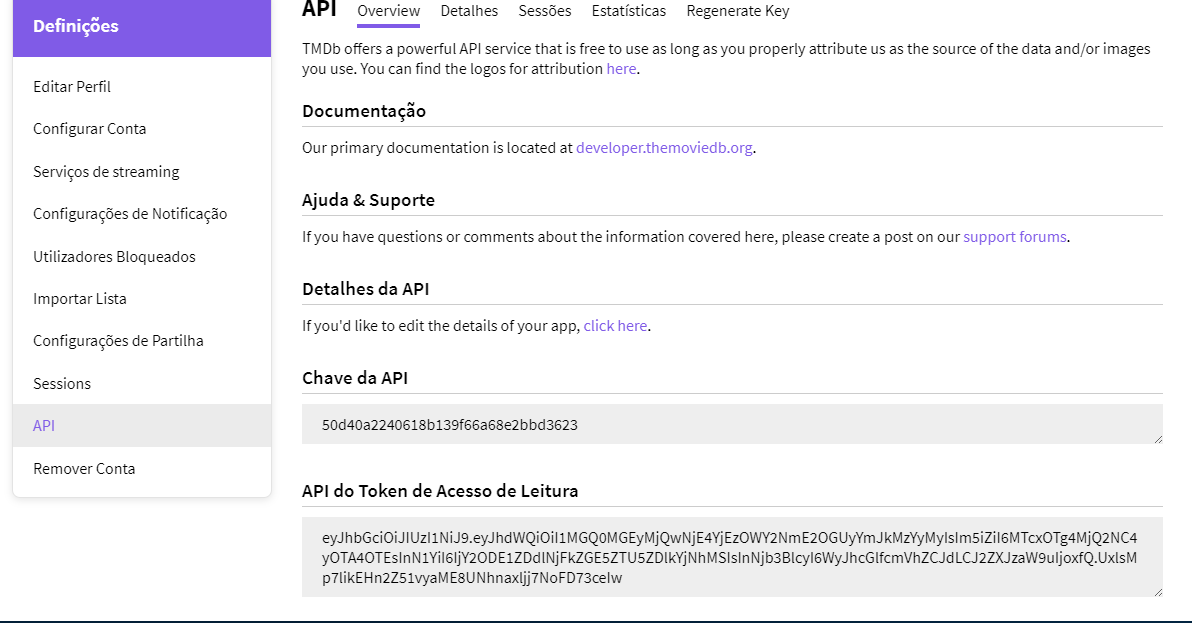
• Navegue até a seção de configurações da conta e obtenha sua chave de API.

<https://developer.themoviedb.org/docs/getting-started>

• Configure a conta o obtenha a chave.



**- API**



- Instale as bibliotecas necessárias

**npm install axios**

- Axios é uma biblioteca de cliente HTTP baseada em Promises para fazer requisições HTTP de forma simplificada. Ela é usada principalmente em aplicativos front-end para se comunicar com servidores back-end, mas também pode ser usada no Node.js.

**npm install swiper**

- Swiper é uma das mais populares para criar carrosséis e slides touch-friendly em aplicações web. Desenvolvida principalmente para dispositivos móveis, o Swiper oferece uma rica experiência de navegação com suporte para gestos touch e uma grande variedade de funcionalidades.

**- Consumindo TMDb no React**

