**DOCUMENTAÇÃO TÉCNICA DO PROJETO – API FILMES**

**Título do Projeto:** Sistema de Catálogo de Filmes

Grupo: Luana de Brito Rodrigues, João Henrique Weigel Falce,

Salomão Konjunski da Silva e Eloi Cesar Gonçalves.

1. **Objetivo do Projeto**

**O objetivo do projeto visa criar uma API com o tema Catálogo de Filmes executando todo o código em linguagem C# incluindo um banco de dados em SQlite sendo criado com o comando (**db.Database.EnsureCreated();)**. O código tem a visão clara de um catálogo disponibilizando os principais métodos do HTTP garantindo uma API feita com mais qualidade.**

1. **Estrutura da Resolução**
   1. **Inicialmente foi feito um padrão de roteiro de como seria apresentado os filmes e suas descrições, (** título, sinopse, data de lançamento, gênero e diretor). Após serem criados seguiu-se com as configurações necessárias para a inclusão do banco de dados utilizando o mapeador “EntityFrameworkCore”, que gerou automaticamente as tabelas no banco SQLite, sem ser necessário a criação manual.
2. **Endpoints da API**

Configuramos a conexão com o banco no arquivo FilmesContext.cs. Em seguida, separamos as rotas em arquivos diferentes, de acordo com cada função:

- ROTA\_GET.cs: Para buscar todos os filmes, por ID, por gênero ou por diretor.

- ROTA\_POST.cs: Pra cadastrar novos filmes.

- ROTA\_PUT.cs: Pra atualizar informações de um filme.

- ROTA\_DELETE.cs: Pra excluir filmes.

Toda a configuração da API, ativação das rotas e a conexão com o banco ficaram no Program.cs. Também usamos o Swagger pra testar a visualização e clareza das rotas em execução.

Para ser feito a execução do código foi necessário a aplicação dos comandos codificados pelo terminal: “dotnet restore”, “dotnet build” e “dotnet run”.

1. **Organização do Código**

Houve a escolha de utilizar o Entity Framework Core, pois possui mais facilidade na gestão de dados em APIs .NET. Ao fornecer uma camada de abstração entre o código da aplicação e o banco de dados, permitiu-se programar com dados, como objetos, sem se preocupar com a complexidade das tabelas e colunas. Optou-se por separar as rotas em arquivos pra deixar o código organizado contendo transparência. O banco SQLite foi selecionado por ser leve, fácil de configurar e funcionar direto no próprio projeto, sem precisar instalar servidor. Swagger foi escolhido com o intuito de obter resultados nítidos para as correções necessárias.

1. **Justificativa Técnica**

Durante o desenvolvimento da API, houve resultados inesperados, ocorrendo um erro no terminal “duplicata (CS0579)”, que travava o projeto. Reservou-se um certo tempo para a análise e identificação do erro, contudo, depois de reconhece-lo, foi realizada a correção e entendimento do ocorrido. Com o desenvolvimento do trabalho proposto a construir uma API do zero, utilizando um banco de dados automatizado, Entity Framework, Swagger para testar e o reforço de comandos do terminal como dotnet restore, dotnet build e dotnet run, adquiriu-se amplos conhecimentos, tanto na compilação do código quanto na importância do trabalho em equipe.