

1. Modelos (Models)

As classes de domínio representam as entidades principais do sistema e a lógica de negócio intrínseca.

- **Usuario**: Classe base para utilizadores. Armazena `Id`, `NomeUsuario`, `Email` e `SenhaHash`. Gere listas de `Avaliacoes` e `Favoritos`. Possui lógica para validação e definição de senha (hashing simples).
- **Administrador**: Herda de `Usuario`. Representa utilizadores com privilégios elevados. Contém métodos de negócio (`CadastrarMidia`, `EditarMidia`) que aplicam validações específicas, como garantir que o título e o género não sejam vazios e a duração seja positiva.
- **Midia (Abstrata)**: Classe base para obras audiovisuais. Define propriedades comuns: `Titulo`, `Genero`, `Sinopse`, `Duracao`, `ClassificacaoIndicativa`, `DataLancamento` e lista de `Equipe`. Define propriedades abstratas para médias de notas que as subclasses devem implementar.
- **Filme**: Herda de `Midia` e implementa `IAvaliavel`. Possui uma lista própria de `Avaliacoes`. Calcula as suas médias (Geral, Narrativa, etc.) com base direta nas avaliações que recebe.
- **Serie**: Herda de `Midia`. Diferente do filme, não recebe avaliações diretas. A sua `NotaMediaGeral` (e outras médias) é calculada pela média aritmética das notas das suas `Temporadas`.
- **Temporada**: Implementa `IAvaliavel`. Contém `NumeroTemporada`, `Titulo`, `Sinopse`, lista de `Episodios` e lista de `Avaliacoes`. Calcula as suas médias baseada nas avaliações dos utilizadores.
- **Episodio**: Entidade simples contendo `NumeroEpisodio`, `Titulo`, `Sinopse` e `Duracao`. Validada para impedir números de episódios negativos.
- **Avaliacao**: Classe complexa que armazena a data, o utilizador e 16 notas individuais divididas em 4 categorias (Narrativa, Execução, Visual, Auditiva). Possui métodos (`GetMediaNarrativa`, `GetMediaGeral`) para calcular as médias ponderadas dessa avaliação específica.
- **Equipe (Abstrata)**, **Ator**, **EquipeTecnica**: Representam os profissionais. `Ator` possui a propriedade extra `Papel`. `EquipeTecnica` é genérica para outras funções.

2. Controllers

Gerem as requisições HTTP e conectam-se aos serviços. Todos herdam de `ControllerBase` e usam o atributo `[ApiController]`.

- **UsuarioController**: Pontos de entrada para cadastro, login, atualização e consulta de utilizadores comuns.
- **AdministradorController**: Similar ao de utilizador, mas focado na gestão da entidade `Administrador` (endpoints separados para segurança lógica).
- **FilmeController**: CRUD completo de filmes.
- **SerieController**: CRUD de séries.
- **TemporadaController**: Gere o cadastro de temporadas (vinculadas a séries) e adição de episódios.
- **AvaliacaoController**: Recebe submissões de avaliações e lista avaliações por mídia.
- **EquipeController**: Adiciona atores/técnicos e lista a equipa de uma mídia ou temporada.

3. Services: Interface e Service

A camada de serviço contém toda a regra de negócio.

- **IUsuarioService / UsuarioService:**
 - Verifica duplicidade de e-mail no cadastro.
 - Realiza login verificando hash de senha.
 - Diferencia o retorno de **Tipo** ("Administrador" ou "Comum").
- **IAdministradorService / AdministradorService:**
 - Garante que operações sejam feitas na tabela de Usuários mas filtradas pelo tipo Administrador.
- **IFilmeService / FilmeService:**
 - Impede cadastro de filmes com títulos já existentes.
 - Mapeia entidade **Filme** para **FilmeRespostaDTO**.
- **ISerieService / SerieService:**
 - Calcula dinamicamente a **NotaMediaGeral** (média das temporadas) ao listar.
 - Carrega a árvore de dependências (**Temporadas** -> **Avaliacoes**) para o cálculo correto.
- **ITemporadaService / TemporadaService:**
 - Valida se o número da temporada já existe naquela série.
 - Vincula episódios à temporada correta e impede números de episódios duplicados.
- **IAvaliacaoService / AvaliacaoService:**
 - Verifica se o **Avaliadold** corresponde a uma Mídia ou Temporada existente.
 - **Regra de Ouro:** Impede que o mesmo utilizador avalie a mesma obra duas vezes.
 - Cria a instância complexa de **Avaliacao** com todas as 16 notas.
- **IEquipeService / EquipeService:**
 - Controla a lógica de adicionar membro: verifica se o ID alvo é Mídia ou Temporada e instancia a subclasse correta (**Ator** ou **EquipeTecnica**).

4. DTOs (Data Transfer Objects)

Usados para validar a entrada de dados e formatar a saída.

- **Usuario/Administrador:** **CriarUsuarioDTO** e **CriarAdministradorDTO** exigem Nome, Email (validação de formato) e Senha (mínimo 6 caracteres). **LoginUsuarioDTO** exige apenas email e senha.
- **Filme/Serie:** **CriarFilmeDTO** e **CriarSerieDTO** validam campos obrigatórios como Título, Gênero e Classificação Indicativa.
- **Temporada:** **CriarTemporadaDTO** exige o **SerieId** para vínculo.
- **Avaliacao:** **CriarAvaliacaoDTO** contém todas as 16 notas com anotação **[Range(0, 10)]** para garantir que o utilizador envie valores válidos.
- **Equipe:** **CriarAtorDTO** (inclui Papel) e **CriarTecnicoDTO**. Exigem **NomeCompleto** e lista de **Funcoes**.

5. DataContext

Configuração do Entity Framework Core.

- Define os **DbSet** para cada entidade: **Usuarios**, **Midias**, **Temporadas**, **Episodios**, **Equipes**, **Avaliacoes**.
- **Configuração de Herança (TPH - Table Per Hierarchy):**
 - **Usuario** usa discriminador "TipoUsuario" ("Comum" vs "Admin").
 - **Midia** usa discriminador "TipoMidia" ("Filme" vs "Serie").
 - **Equipe** usa discriminador "TipoMembro" ("Ator" vs "Tecnico").
- **Relacionamentos:** Configura Cascade Delete para Séries -> Temporadas -> Episódios (se apagar a série, apaga tudo).

6. Explicação das Rotas e o que cada uma pede

UsuarioController (api/Usuario)

- **POST /cadastrar:** Pede JSON com **NomeUsuario**, **Email**, **Senha**.
- **POST /login:** Pede JSON com **Email**, **Senha**.
- **GET /:** Não pede nada, retorna lista de todos os utilizadores.
- **GET /{id}:** Pede ID (Guid) na URL.
- **PUT /{id}:** Pede ID na URL e JSON com **NomeUsuario**, **Email**.
- **DELETE /{id}:** Pede ID na URL.

AdministradorController (api/Administrador)

- **POST /cadastrar:** Pede JSON com **NomeUsuario**, **Email**, **Senha**.
- **POST /login:** Pede JSON com **Email**, **Senha**.
- *Outras rotas (GET, PUT, DELETE) seguem o padrão do Usuario, mas afetam admins.*

FilmeController (api/Filme)

- **POST /:** Pede JSON com **Titulo**, **Genero**, **Sinopse**, **Duracao**, **Classificacao**, **DataLancamento**.
- **GET /:** Retorna todos os filmes.
- **GET /{id}:** Retorna detalhes do filme e nota média.
- **PUT /{id}:** Pede ID e JSON completo de atualização.
- **DELETE /{id}:** Pede ID.

SerieController (api/Serie)

- **POST /:** Pede JSON com dados da série (**Titulo**, **Genero**...).
- **GET /:** Retorna séries e qtd de temporadas.
- **GET /{id}:** Retorna detalhes da série.

TemporadaController (api/Temporada)

- **POST /:** Pede JSON com **Serield**, **NumeroTemporada**, **Titulo**...
- **POST /episodio:** Pede JSON com **Temporadald**, **NumeroEpisodio**, **Titulo**, **Duracao**.
- **GET /serie/{serield}:** Pede ID da série, retorna todas as temporadas dela.

AvaliacaoController (api/Avaliacao)

- **POST /:** Pede JSON com **Usuarioid**, **Avaliadold** e as 16 notas (0-10).
- **GET /midia/{id}:** Pede ID da mídia, retorna lista de avaliações.

EquipeController (api/Equipe)

- **POST /ator:** Pede JSON com **Midiald** OU **Temporadald**, **NomeCompleto**, **Funcoes**, **Papel**.
- **POST /tecnico:** Pede JSON com **Midiald** OU **Temporadald**, **NomeCompleto**, **Funcoes**.
- **GET /midia/{midiald}:** Retorna elenco do filme/série.
- **GET /temporada/{temporadald}:** Retorna elenco da temporada.

7. Dependências Instaladas

Listadas no ficheiro `.csproj`:

- `Microsoft.EntityFrameworkCore` (v9.0.11): ORM principal.
- `Microsoft.EntityFrameworkCore.SqlServer` (v9.0.11): Conector para SQL Server.
- `Microsoft.EntityFrameworkCore.Tools` (v9.0.11): Ferramentas para Migrations.
- `Microsoft.EntityFrameworkCore.Design` (v9.0.11): Suporte a design-time.
- `Swashbuckle.AspNetCore` (v6.6.2): Swagger para documentação da API.

8. appsettings.json

Configurações da aplicação.

- **ConnectionStrings:** Define a conexão com o banco local (`Server=(localdb)\MSSQLLocalDB;Database=CineReviewDB`).
- **Logging:** Define o nível de log padrão como "Information".

9. Program.cs

Ponto de entrada da aplicação.

1. **Configuração de Banco:** Adiciona o `DataContext` usando SQL Server.
2. **Injeção de Dependência:** Regista todos os serviços com `AddScoped` (`IUsuarioService`, `IFilmeService`, etc.), garantindo que uma nova instância seja criada por requisição HTTP.
3. **Swagger:** Configura a geração da documentação da API.
4. **Pipeline:** Configura `UseSwagger` e `UseSwaggerUI` (apenas em desenvolvimento), redirecionamento HTTPS, autorização e mapeamento de controllers.