



Ministério da Educação
Universidade Federal do Cariri



UNIVERSIDADE FEDERAL DO CARIRI - UFCA
PRÓ-REITORIA DE GRADUAÇÃO - PROGRAD
CENTRO DE CIÊNCIA E TECNOLOGIA - CCT
CURSO DE BACHARELADO EM ENGENHARIA DE SOFTWARE

ESPECIFICAÇÃO PROJETO 1 - POO

TEMA 10 – CATÁLOGO DE FILMES E SÉRIES

1. Visão Geral

Desenvolver um sistema de **linha de comando (CLI)** ou **API mínima** (FastAPI/Flask, opcional) para gerenciar um **catálogo pessoal de filmes e séries**, com **avaliações**, **status de visualização**, **temporadas/episódios**, **histórico** e **relatórios de consumo de mídia**. O sistema deve permitir acompanhar o progresso de séries e comparar avaliações entre mídias. A persistência deve ser simples (em **JSON** ou **SQLite**) e a modelagem orientada a objetos deve enfatizar **herança**, **encapsulamento**, **validações** e **composição**.

2. Requisitos Funcionais

1. Cadastro de mídias

- ✓ Campos: título, tipo (**FILME** ou **SERIE**), gênero, ano, duração (minutos), classificação indicativa, elenco (lista), e status (**NÃO ASSISTIDO**, **ASSISTINDO**, **ASSISTIDO**).
- ✓ Impedir duplicidade de títulos com mesmo tipo e ano.

2. Séries e episódios

- ✓ Para **SERIE**, registrar temporadas e episódios:
 - Campos: nº temporada, nº episódio, título, duração, data de lançamento, status de visualização, nota (opcional).
- ✓ Marcar série como “ASSISTIDA” quando **todos os episódios** estiverem concluídos.

3. Avaliações

- ✓ Permitir ao usuário avaliar filmes e episódios (nota de 0 a 10).
- ✓ Calcular automaticamente a **nota média da série** e **nota geral do catálogo**.

4. Histórico de visualização



Ministério da Educação
Universidade Federal do Cariri

- ✓ Registrar data/hora de conclusão de cada mídia.
- ✓ Permitir relatório de tempo total assistido no período (por semana/mês).
- 5. **Listas e favoritos**
 - ✓ Criar listas personalizadas (“Para assistir”, “Favoritos”, “Melhores de 2025”, etc.).
 - ✓ Adicionar/remover mídias dessas listas.
- 6. **Relatórios**
 - ✓ Média de notas por gênero.
 - ✓ Tempo total assistido por tipo (filme/série).
 - ✓ Top 10 filmes/séries mais bem avaliados.
 - ✓ Séries com maior número de episódios assistidos.
- 7. **Configurações (settings.json)**
 - ✓ Nota mínima para “recomendado”.
 - ✓ Limite de listas personalizadas por usuário.
 - ✓ Multiplicador de duração para converter minutos → horas totais.

3. Requisitos Técnicos de POO (RT)

- **Modelagem e herança:**
 - ✓ **Midia** (classe base) → **Filme** e **Serie**.
 - ✓ **Serie** agrega **Temporada**, que contém **Episodio**.
 - ✓ **Usuario** possui listas (**ListaPersonalizada**) e histórico.
- **Encapsulamento e validações:**
 - ✓ **@property** para validar duração (>0), notas (0–10), título não vazio, número de temporada/episódio positivo.
 - ✓ Atualizações automáticas de status (**Serie** → “ASSISTIDA” ao completar episódios).
- **Métodos especiais (≥ 4):**
 - ✓ **__str__**/**__repr__**: exibição formatada de mídias.
 - ✓ **__eq__**: comparar mídias por título + tipo.
 - ✓ **__len__**: número total de episódios em uma série.
 - ✓ **__lt__**: ordenar mídias por nota média.
- **Cálculos e agregações:**
 - ✓ Funções para calcular nota média por tipo/gênero e total de minutos assistidos.
 - ✓ Conversão de tempo total em horas com arredondamento configurável.
- **Persistência:**
 - ✓ Módulo **dados.py** com funções para salvar/carregar mídias, episódios, usuários e listas em JSON ou SQLite.
 - Rotina de **seed** com filmes e séries pré-cadastradas.
- **Testes (pytest):**



Ministério da Educação
Universidade Federal do Cariri

- ✓ Criação e atualização de séries/episódios; cálculo de nota média; total de tempo assistido; duplicidade; relatórios.

- **Interface:**

- ✓ CLI com subcomandos:
 - `midia adicionar, midia avaliar, midia listar, midia relatorio top,`
 - `serie adicionar-episodio, serie atualizar-status,`
 - `usuario criar-lista, usuario adicionar-favorito.`
- Ou API mínima equivalente.

4. Regras de negócio (essenciais)

- **Nota:** deve ser numérica (0–10).
- **Série assistida:** todos os episódios com status “ASSISTIDO”.
- **Duplicidade:** não permitir mesmo título + tipo + ano.
- **Favoritos:** máximo de listas definido em `settings.json`.
- **Tempo assistido:** soma de durações de mídias com status “ASSISTIDO”.
- **Relatórios:** considerar apenas mídias finalizadas (status “ASSISTIDO”).

5. Critérios de aceite

- ✓ **POO:** classes bem modeladas, encapsulamento com `@property`, **≥ 4 métodos especiais** aplicados.
- ✓ **Regras:** integridade de status, notas e histórico.
- ✓ **Testes:** **≥ 15** cobrindo casos felizes e de erro (estoque, cupom, frete, pagamento, cancelamento).
- ✓ **Relatórios:** **≥ 3** implementados (notas por gênero, top 10, tempo total).
- ✓ **Documentação:** **README** com instruções, diagrama simples e decisões de implementação.

6. Cronograma

Semana 1 — 18/11/2025

Tema: Modelagem OO e definição do projeto

Entregas:

- UML textual (classes, atributos, métodos principais, relacionamentos).
- Arquivo **README.md** inicial com:
 - Descrição do projeto e objetivo.
 - Estrutura planejada de classes.
- Código inicial com classes vazias e docstrings de propósito.



Semana 2 — 25/11/2025

Tema: Implementação das classes base e encapsulamento

Entregas:

- Classes **Midia**, **Filme**, **Serie**, **Episodio** com validações (**@property**) e métodos especiais.
- Métodos especiais principais (**__len__**, **__eq__**/**__repr__** onde couber).
- Testes básicos (**pytest**) para criação e manipulação de objetos.

Semana 3 — 02/12/2025

Tema: Herança, relacionamentos e persistência básica

Entregas:

- Relacionamento **Serie–Temporada–Episodio** implementado.
- Persistência simples (JSON ou SQLite).
- Relatório inicial: **ocupação** por período.

Semana 4 — 09/12/2025

Tema: Regras de negócio e integração

Entregas:

- Regras de status automático e cálculo de médias.
- Criação de listas personalizadas e favoritos.
- CLI ou API mínima funcional.
- Testes cobrindo fluxos principais.

Semana 5 — 16/12/2025 (Entrega final)

Tema: Padrões de projeto, refinamento e documentação final

Entregas:

- Relatórios consolidados (notas, top 10, tempo total, por gênero).
- **README** completo (execução, testes, exemplos) + diagrama final.
- Todos os testes passando (**pytest**).



Ministério da Educação
Universidade Federal do Cariri

- Repositório GitHub com tag **v1.0**.

7. Entrega

A entrega de cada etapa será feita via Classroom. Deve ser enviado o link da tag do github com a entrega até as 18:00h da data da entrega.

- [Git: criando tags](#)

8. Avaliação

A avaliação será feita seguindo os seguintes critérios: [Roteiro de Avaliação](#)