TRABALHO 1 – DESENVOLVIMENTO BACKEND COM SPRING BOOT

Objetivo

Desenvolver uma API REST completa, em equipe, aplicando os conceitos estudados durante o semestre: **arquitetura em camadas**, **DTOs e validações**, **tratamento de exceções**, e **autenticação/autorização com JWT**.

O grupo deve criar um **sistema inspirado em um grande case real** (ex.: Netflix, Spotify, MercadoLivre, Steam, iFood, Jogo online, Sistema de agendamento, etc.), porém com **nome e identidade próprios** (ex.: *UnescFlix*, *SoundHub*, *UniMarket*, *JogaUnesc*, etc.).

Formação das Equipes

- Até **5 integrantes**.
- Todos os membros devem contribuir no **GitHub** do projeto.
- Cada aluno deve ser responsável por ao menos **uma entidade principal** e **suas operações (CRUD + lógica de negócio)**.

Requisitos Mínimos do Sistema

Arquitetura

- Aplicar **arquitetura em camadas**: Controller, Service, Repository, Entity, DTO, Validation, ExceptionHandler.
- Uso de **Spring Boot 3+**, **Spring Data JPA**, e **Spring Security com JWT**.
- Banco de dados **PostgreSQL**.
- Utilizar **Flyway** para versionamento do banco.

Regras e Entidades (mínimo exigido)

O sistema deve conter **no mínimo 5 entidades relacionadas** (1:N e N:N), por exemplo:

Exemplo Netflix-like:
Usuário → Perfil → Filme → Categoria → Avaliação

Exemplo Jogo:
Jogador → Partida → Personagem → Item → Conquista

Requisitos funcionais mínimos:

1. CRUD completo de pelo menos 5 entidades.
2. Relacionamentos entre entidades (1:N, N:N).
3. Autenticação e autorização via JWT.
4. Validações com anotações (@NotBlank, @Email, @Size, etc.).

5. Tratamento centralizado de exceções (@ControllerAdvice).
6. DTOs para entrada e saída de dados (evitar expor entidades diretas).
7. Logs básicos de operações importantes (opcional, para bônus).

Tecnologias

- **Backend:** Spring Boot 3, Spring Data JPA, Spring Security, JWT, Lombok
- **Banco de dados:** PostgreSQL 12 a 17
- **Versionamento:** Git + GitHub (repositório por equipe)
- **Documentação:** Swagger/OpenAPI ou Readme no Git

Entregas Obrigatórias

1. Documento de Análise

- Nome do sistema e propósito.
- Responsabilidades de cada um da equipe
- Requisitos funcionais e não-funcionais.
- Diagrama de classes e de relacionamento (UML).

2. Protótipo das Telas

Não teremos FRONTEND (a não ser que a equipe entregue algo em conjunto – não obrigatório). Porém para entender o funcionamento do sistema, deve-se criar esboços de tela, no Figma ou outro software a escolha da equipe.

- Fluxo principal (login, CRUDs, listagens).
- Interface de referência (não precisa ser funcional).

3. Banco de Dados

- Modelo relacional (DER).
- Scripts ou migrations (Flyway).

4. Código-Fonte

- Projeto completo com README.md documentando:
    - Instruções para rodar o projeto.
    - Estrutura de pastas.
    - Exemplo de requests e responses.
    - Credenciais.

5. Apresentação

- Pitch de **5–10 minutos** apresentando:
    - Contexto do sistema.
    - Arquitetura e camadas.
    - Demonstração das APIs funcionando (via Postman, Swagger ou front).

Critérios de Avaliação

Critério	Descrição	Peso
1. Estrutura e Arquitetura	Uso correto das camadas, DTOs, validações e boas práticas.	2,0
2. Segurança (JWT)	Implementação correta de autenticação e autorização.	1,5
3. Funcionalidades e Entidades	Quantidade, coerência e relação entre as entidades.	2,0
4. Qualidade Técnica	Código limpo, padronização, tratamento de erros, README.	1,5
5. Criatividade e Escopo	Originalidade e complexidade da solução.	1,0
6. Apresentação e Documentação	Clareza, comunicação e organização dos artefatos.	2,0
Total		**10,0**

Prazos de Entrega

Etapa	Entrega	Aula	Data
1 **Documento + DER**	**Planejamento e modelagem**	**Aula 1**	**08/10/25**
2 **CRUDs básicos**	**Entidades e relacionamentos**	**Aula 2**	**22/10/25**
3 **JWT + validações**	**Segurança e refino**	**Aula 3**	**29/10/25**
4 **Aplicação das Camadas**	**Controller, Services, Repository**	**Aula 4**	**05/11/25**
5 **Refinamento da Solução**	**Swagger + Ajustes**	**Aula 5**	**12/11/25**
	Projeto Final + Apresentação	**Aula 6**	**19/11/25**