Planejamento e Estrutura do Sistema

# 1. Introdução

Este documento descreve a arquitetura, funcionalidades, validações e operações necessárias para o desenvolvimento de um sistema de recomendação de cursos, com autenticação de usuários e integração com URLs externas para redirecionamento.

# 2. Arquitetura Geral

- Front-end: React.js (Hospedado no Vercel ou Netlify)  
- Back-end: Spring Boot (Hospedado no Heroku ou Render.com)  
- Banco de Dados: PostgreSQL (Heroku Postgres ou ElephantSQL)  
- Autenticação: Firebase Authentication  
- Recomendação: Machine Learning com Python (Integrado via AWS Lambda ou diretamente no back-end)

# 3. Estrutura das Tabelas

## 1. Usuários (users)

Atributos: id, name, email, password, created\_at  
Operações:  
- GET: Listar todos os usuários ou buscar por ID.  
- POST: Criar novo usuário.  
- PUT: Atualizar informações do usuário.  
- DELETE: Excluir usuário.

## 2. Cursos (courses)

Atributos: id, title, description, url, created\_at  
Operações:  
- GET: Listar todos os cursos ou buscar por ID.  
- POST: Criar novo curso.  
- PUT: Atualizar informações do curso.  
- DELETE: Excluir curso.

## 3. Avaliações (reviews)

Atributos: id, user\_id, course\_id, rating, comment, created\_at  
Operações:  
- GET: Listar todas as avaliações, buscar por ID, ou listar avaliações de um curso específico.  
- POST: Criar nova avaliação.  
- PUT: Atualizar uma avaliação existente.  
- DELETE: Excluir uma avaliação.

## 4. Recomendações (recommendations)

Atributos: id, user\_id, course\_id, score  
Operações:  
- GET: Listar recomendações para um usuário específico.  
- POST: Gerar novas recomendações.  
- DELETE: Excluir uma recomendação.

# 4. Validações Necessárias

## 1. Usuários

- Validação de e-mail único e válido, senha segura.  
- Verificação de existência antes de operações de atualização ou exclusão.

## 2. Cursos

- Validação de URL válida.  
- Verificação de existência antes de operações de atualização ou exclusão.

## 3. Avaliações

- Validação de rating dentro do intervalo permitido.  
- Verificação de duplicidade de avaliações para o mesmo curso pelo mesmo usuário.

## 4. Recomendações

- Verificação de existência de usuário antes de gerar recomendações.  
- Prevenção de recomendações duplicadas.

# 5. Estrutura de Pastas

- /frontend: React.js para a interface do usuário.  
- /backend: Spring Boot para a lógica de negócios e APIs.  
- /ml-scripts: Scripts em Python para machine learning e recomendações.

# 6. Fluxo de Desenvolvimento

- Configuração do Banco de Dados: Criar tabelas utilizando Hibernate a partir das entidades Java.  
- Desenvolvimento do Back-end: Implementar APIs RESTful no Spring Boot.  
- Desenvolvimento do Front-end: Criar interfaces e conectar com as APIs.  
- Sistema de Recomendação: Integrar scripts de ML no back-end.  
- Deploy: Subir front-end no Vercel/Netlify e back-end no Heroku/Render.com.