ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE (ERS)

1. Introdução

Este documento detalha os requisitos descritos anteriormente, especificando comportamentos esperados, fluxos de uso, validações, estrutura de dados e modelos UML (descritos em texto). Serve como base técnica para desenvolvedores e testadores.

2. Arquitetura Geral do Sistema

2.1 Arquitetura de Camadas

O sistema será dividido em:

- Camada de Apresentação (Front-end): Interface para interação com usuário (React, Angular, etc.)
- Camada de Aplicação (Back-end): Regras de negócio e controle de fluxos (Spring Boot)
- Camada de Persistência: Banco de dados relacional (MySQL)
- Camada de Segurança: JWT, autenticação e autorização por perfil

3. Casos de Uso Detalhados

UC01 – Cadastrar Aluno

- Ator Primário: Administrador
- Pré-condição: Usuário autenticado com permissão de administrador
- Fluxo Principal:
    1. Administrador acessa tela de cadastro
    2. Informa: nome, CPF, data de nascimento, telefone, e-mail
    3. Clica em “Salvar”
    4. Sistema valida os dados e salva no banco
    5. Sistema exibe mensagem de sucesso
- Fluxo Alternativo (CPF já cadastrado):
    - Sistema exibe erro: “Aluno já cadastrado”
- Pós-condição: Aluno registrado com status "Ativo"

UC02 – Registrar Pagamento

- Ator: Administrador

- Fluxo Principal:
    1. Acessa ficha do aluno
    2. Clica em “Registrar pagamento”
    3. Informa valor, data, forma de pagamento
    4. Confirma
    5. Sistema registra e atualiza status de matrícula

UC05 – Cadastrar Treino

- Ator: Instrutor
- Fluxo Principal:
    1. Acessa “Gerenciar Treinos”
    2. Cria novo treino: nome, objetivo, duração
    3. Associa exercícios existentes
    4. Salva

UC09 – Executar Treino (Aluno)

- Ator: Aluno
- Pré-condição: Aluno autenticado e com treino ativo
- Fluxo Principal:
    1. Acessa seu painel
    2. Visualiza treino do dia
    3. Marca exercícios como realizados
    4. Salva progresso
    5. Pós-condição: Execução registrada no histórico

4. Modelos UML (descritos em texto)

4.1 Diagrama de Casos de Uso (simplificado – textual)

Ator: Administrador

→ Cadastrar Aluno

→ Gerenciar Planos

→ Registrar Pagamentos

→ Emitir Relatórios

Ator: Instrutor

→ Cadastrar Exercícios

→ Criar Treinos

→ Associar Treinos a Alunos

Ator: Aluno

→ Visualizar Treinos

→ Executar Treino

4.2 Diagrama de Classes (descrição)

Classe: Aluno

id: Long

nome: String

cpf: String

email: String

telefone: String

status: Enum (ATIVO, INATIVO)

planoAtual: Plano

listaTreinos: List<Treino>

Classe: Plano

id: Long

nome: String

preco: Double

duracaoDias: int

Classe: Pagamento

id: Long

dataPagamento: Date

valor: Double

status: Enum (PAGO, PENDENTE)

Classe: Treino

id: Long

nome: String

objetivo: String

exercicios: List<Exercicio>

Classe: Exercicio

id: Long

nome: String

series: int

repeticoes: int

carga: Double

5. Estrutura do Banco de Dados (modelo lógico simplificado)

TABELA ALUNO

- id (PK)
- nome
- cpf
- email
- telefone
- plano_id (FK)

TABELA PLANO

- id (PK)
- nome

- preco
- duracao_dias

TABELA PAGAMENTO
- id (PK)
- aluno_id (FK)
- valor
- data_pagamento
- status

TABELA TREINO
- id (PK)
- nome
- objetivo

TABELA EXERCICIO
- id (PK)
- nome
- series
- repeticoes
- carga

TABELA TREINO_EXERCICIO (relacional N:N)
- treino_id (FK)
- exercicio_id (FK)

TABELA ALUNO_TREINO (para associação de treino com aluno)
- aluno_id (FK)

- treino_id (FK)

- data_inicio

- data_fim

6. Regras de Validação

- CPF deve ser único e válido (formato e dígito verificador)
- Aluno só pode ser ativo se tiver pagamento válido nos últimos 30 dias
- Exercício deve ter pelo menos 1 série e 1 repetição
- Login e senha devem conter autenticação segura com JWT
- Data de treino não pode ser retroativa

7. Requisitos de Interface

- Tema claro e moderno, com feedback visual para cada ação
- Ícones intuitivos para ações como salvar, editar, excluir
- Dashboard com cards de alunos ativos, inadimplentes e frequências
- Tela de treino com visual semelhante a um check-list para o aluno

8. Requisitos de API REST

POST /api/alunos

```
{
  "nome": "Carlos",
  "cpf": "123.456.789-00",
  "email": "carlos@email.com",
  "telefone": "11999999999",
  "planoId": 1
}
```

Autenticação: via token JWT no header

Documentação automática: Swagger em /swagger-ui.html