# Documentação Funcional e Técnica - Treino & Dieta App

## 1. Visão Geral

### 1.1. Objetivo do Sistema  
O \*\*Treino & Dieta App\*\* é uma aplicação web e mobile projetada para permitir que usuários registrem e acompanhem sua evolução física, dieta e treinos. A aplicação será uma \*\*SPA (Single Page Application)\*\* desenvolvida com \*\*Angular 2 no front-end\*\* e \*\*FastAPI (Python) no back-end\*\*, utilizando \*\*AWS Lambda e DynamoDB\*\* para escalabilidade e eficiência.  
  
### 1.2. Público-Alvo  
- \*\*Usuários Finais:\*\* Indivíduos que desejam monitorar seu progresso de treino e dieta.  
- \*\*Gerentes de Projetos:\*\* Profissionais responsáveis pelo acompanhamento do desenvolvimento do sistema.  
- \*\*Analistas de Negócios:\*\* Especialistas em requisitos funcionais que precisam garantir que o sistema atenda às necessidades do usuário.  
- \*\*Desenvolvedores:\*\* Equipe técnica responsável pela implementação do sistema.  
  
### 1.3. Benefícios do Sistema  
✅ Monitoramento e visualização da evolução física e de treino.  
✅ Segurança reforçada utilizando \*\*AWS Cognito e AWS KMS\*\*.  
✅ Alta escalabilidade com \*\*AWS Lambda\*\* e \*\*DynamoDB\*\*.  
✅ Arquitetura modular e sustentável baseada em \*\*DDD, TDD, SOLID e Clean Architecture\*\*.  
✅ Implantação automatizada com \*\*GitHub Actions e AWS CDK\*\*.

## 2. Requisitos Funcionais

### 2.1. Requisitos Gerais  
- [RF01] O sistema deve permitir que o usuário crie e edite um perfil.  
- [RF02] O usuário deve ser capaz de registrar e visualizar seu progresso.  
- [RF03] O sistema deve fornecer gráficos de evolução do treino e dieta.  
- [RF04] O usuário pode inserir dados via \*\*web\*\* ou \*\*mobile\*\*.  
- [RF05] O sistema deve permitir login seguro via \*\*AWS Cognito\*\*.  
- [RF06] O sistema deve enviar \*\*notificações push e e-mails\*\* com lembretes de treino.  
- [RF07] O sistema deve permitir a exportação dos dados em \*\*PDF/CSV\*\*.  
  
### 2.2. Requisitos Não-Funcionais  
- [RNF01] O sistema deve operar com \*\*tempo de resposta inferior a 300ms\*\* por requisição.  
- [RNF02] O sistema deve possuir \*\*testes automatizados (TDD)\*\* garantindo 80%+ de cobertura.  
- [RNF03] O sistema deve armazenar informações sensíveis utilizando \*\*AWS KMS\*\*.

## 3. Arquitetura do Sistema

### 3.1. Padrões Utilizados  
✅ \*\*DDD (Domain-Driven Design)\*\*: Separação clara de \*\*domínio, aplicação, infraestrutura e interfaces\*\*.  
✅ \*\*TDD (Test-Driven Development)\*\*: Desenvolvimento guiado por testes unitários e de integração.  
✅ \*\*SOLID\*\*: Princípios de desenvolvimento para código escalável e sustentável.  
✅ \*\*Clean Architecture\*\*: Modularidade e desacoplamento para manutenção simplificada.  
  
### 3.2. Tecnologias Utilizadas  
- \*\*Front-end\*\*: Angular 2  
- \*\*Back-end\*\*: FastAPI (Python)  
- \*\*Banco de Dados\*\*: AWS DynamoDB  
- \*\*Armazenamento\*\*: Amazon S3  
- \*\*Autenticação\*\*: AWS Cognito  
- \*\*Segurança\*\*: AWS KMS, IAM  
- \*\*Infraestrutura\*\*: AWS CDK  
- \*\*CI/CD\*\*: GitHub Actions

## 4. Fluxo de Funcionalidades

### 4.1. Cadastro e Login  
1. Usuário cria conta via \*\*AWS Cognito\*\*.  
2. Backend gera \*\*token JWT\*\* para acesso seguro.  
3. Tokens são armazenados em \*\*AWS DynamoDB\*\* e protegidos via \*\*AWS KMS\*\*.  
  
### 4.2. Input de Dados (Treino e Dieta)  
1. Usuário registra treino e alimentação via \*\*web ou mobile\*\*.  
2. API FastAPI recebe os dados e os armazena no \*\*DynamoDB\*\*.  
3. Arquivos de progresso são enviados para \*\*Amazon S3\*\*.  
  
### 4.3. Relatórios e Visualização  
1. Backend processa os dados para exibir \*\*gráficos e estatísticas\*\*.  
2. Usuário pode visualizar o progresso via \*\*dashboard Angular\*\*.  
3. Relatórios podem ser exportados para \*\*PDF/CSV\*\*.

## 5. Segurança e Monitoramento

### 5.1. Segurança  
✅ \*\*AWS KMS\*\* para armazenamento seguro de credenciais e tokens.  
✅ \*\*AWS WAF\*\* para proteção contra ataques DDoS.  
✅ \*\*AWS IAM\*\* com permissões mínimas necessárias para cada serviço.  
  
### 5.2. Monitoramento e Logging  
✅ \*\*AWS CloudWatch\*\* para logs e métricas.  
✅ \*\*AWS X-Ray\*\* para rastreamento de chamadas API.  
✅ \*\*SonarQube\*\* para análise de código e cobertura de testes.

## 6. CI/CD e Deploy

### 6.1. Pipeline CI/CD (GitHub Actions)  
1. \*\*Build e Testes Automatizados\*\*  
2. \*\*Deploy Automático no AWS Lambda\*\*  
3. \*\*Infraestrutura provisionada via AWS CDK\*\*  
4. \*\*Monitoramento contínuo via AWS CloudWatch\*\*  
  
### 6.2. Exemplo de Workflow CI/CD  
```yaml  
on: push  
jobs:  
 deploy:  
 runs-on: ubuntu-latest  
 steps:  
 - uses: actions/checkout@v2  
 - run: cd frontend && npm install && npm test  
 - run: cd backend && pytest tests/  
 - run: cd cdk && cdk deploy  
```