Roadmap 3.1 Detalhado: Plataforma AppFitness (Estado Atual e Próximos Passos)

Data da Atualização: 18 de Outubro de 2025

Este documento é o nosso mapa atualizado, detalhando minuciosamente a jornada do projeto AppFitness desde a sua concepção até o estado atual e delineando os próximos passos para o lançamento.

PARTE 1: O QUE FOI FEITO (A JORNADA ATÉ AGORA)

Nós transformamos uma ideia em uma aplicação robusta e, agora, em uma plataforma com uma interface de nível profissional.

✅ FASE 1: A FUNDAÇÃO (BACKEND E ESTRUTURA)

- **Milestone 1: Setup do Ambiente e Banco de Dados - CONCLUÍDO**
  - Ambiente de desenvolvimento configurado com Docker e Docker Compose.
  - Schema de banco de dados robusto e multi-tenant implementado em PostgreSQL.
  - Biblioteca inicial com 50+ exercícios pré-cadastrados.

✅ FASE 2: CONSTRUÇÃO DO MVP (BACKEND E FRONTEND INICIAL)

- **Milestone 2: API Core em Go - CONCLUÍDO**
  - API segura e performática construída.
  - Sistema de autenticação completo com JWT para Treinadores e Alunos.
  - Endpoints de gestão de perfil do treinador (/api/trainers/me).
- **Milestone 3: Ferramentas de Gestão (Backend) - CONCLUÍDO**
  - CRUD completo e seguro para Alunos (/api/students).
  - CRUD completo para Fichas de Treino (/api/workouts).
  - CRUD completo para Exercícios dentro de um treino (/api/workouts/{id}/exercises).
- **Milestone 4 & 5: Frontend Inicial - CONCLUÍDO e ARQUIVADO**
  - Um frontend funcional em React foi criado, validando todo o fluxo da API, desde o login do treinador, gestão de alunos, criação de treinos, até a visualização pelo aluno. **Esta versão cumpriu seu propósito de validar o backend, mas foi estrategicamente descartada em favor de uma arquitetura superior.**

✅ FASE 3: REENGENHARIA DA INTERFACE (PROFISSIONALIZAÇÃO)

Esta fase representa um pivô estratégico, descartando o frontend inicial para adotar uma arquitetura e um sistema de design de nível profissional, baseados na sua visão e no protótipo

white-label-coach.

- **Milestone 6: Transplante de Arquitetura Frontend - CONCLUÍDO**
  - **Decisão Estratégica:** O frontend antigo foi completamente removido do projeto appfitness.
  - **Integração do Novo Design:** O projeto white-label-coach, com seu design superior, foi transplantado para a pasta frontend do appfitness, tornando-se a nova base de código oficial.
  - **Atualização Tecnológica:** A base do frontend foi atualizada para **TypeScript** e **Tailwind CSS**, adotando as mesmas ferramentas do protótipo para garantir fidelidade visual e manutenibilidade.
- **Milestone** 7: Configuração **e Estabilização da Base - CONCLUÍDO**
  - **Resolução de Dependências:** O arquivo package.json foi completamente reconfigurado, resolvendo conflitos de versão e adicionando todas as dependências necessárias (radix-ui, lucide-react, tailwind-merge, etc.).
  - **Configuração do Ambiente:** Os arquivos de configuração (vite.config.ts, tsconfig.json, tailwind.config.ts, postcss.config.js) foram criados e ajustados, garantindo que o ambiente de desenvolvimento (npm run dev) funcione perfeitamente sem erros.
  - **Limpeza do Git:** O problema de repositório Git aninhado foi resolvido, integrando o histórico do novo frontend ao repositório principal do appfitness.
- **Milestone** 8: Conexão da "Casa" (Integração Core **Backend-Frontend) - EM ANDAMENTO**
  - **Ponte de Dados (api.ts):** Camada de serviço centralizada criada para gerenciar a comunicação com o backend Go. **CONCLUÍDO.**
  - **Painel** de Controle (AuthContext.tsx): Sistema de gerenciamento de estado global para autenticação e branding (cores e logo) implementado. **CONCLUÍDO.**
  - **Estrutura de Layouts:** Os layouts principais com navegação lateral (TrainerLayout, StudentLayout) e proteção de rotas (ProtectedRoute) foram criados, estabelecendo a estrutura visual para as áreas logadas. **CONCLUÍDO.**
  - **Conexão da Página de Login:** A página TrainerLogin.tsx foi a primeira a ser totalmente integrada, usando o api.ts e o AuthContext para se comunicar com o endpoint /api/login e autenticar o usuário com sucesso. **CONCLUÍDO.**
  - **Conexão do Dashboard do Treinador:** A página Dashboard.tsx foi integrada com sucesso aos endpoints /api/trainers/me e /api/students, exibindo a saudação personalizada e a lista de alunos reais do banco de dados. **CONCLUÍDO.**

PARTE 2: PRÓXIMOS PASSOS (O CAMINHO PARA O LANÇAMENTO)

Com a nova arquitetura do frontend estabilizada e as primeiras páginas conectadas, nosso foco é continuar a "ligar os fios" do resto da casa e preparar para o deploy.

📋 FASE 4: INTEGRAÇÃO COMPLETA DO APLICATIVO

- **[ ] Conectar o Dashboard do Aluno:**
  - Integrar a página student/Dashboard.tsx para buscar e exibir os treinos reais do aluno logado via endpoint /api/students/me/workouts.
- **[ ] Conectar a Visualização de Treinos (Aluno):**
  - Criar a página de detalhes do treino do aluno, seguindo o novo design.
  - Integrá-la ao endpoint /api/students/me/workouts/{id} para exibir a lista de exercícios com imagens, séries, repetições, etc.
- **[ ] Conectar a Gestão de Alunos (Treinador):**
  - Implementar a funcionalidade completa no StudentsView.tsx: adicionar, editar e apagar alunos, conectando aos respectivos endpoints (POST, PUT, DELETE em /api/students).
- **[ ] Conectar a Gestão de Treinos (Treinador):**
  - Integrar as telas de WorkoutsView.tsx e ExercisesView.tsx para permitir que o treinador crie e edite as fichas de treino e os exercícios de seus alunos.
- **[ ]** Conectar **a Página de Configurações (WhiteLabelSettings.tsx):**
  - Integrar a página de configurações de personalização ao endpoint PUT /api/trainers/me para salvar o logo e a cor da marca.

📋 FASE 5: DEPLOY E LANÇAMENTO BETA

- **[ ] Preparar o Frontend para Produção (PWA):**
  - Configurar o Vite para transformar o projeto em um Progressive Web App instalável.
- **[ ] Preparar o Backend para Produção (Docker):**
  - Criar um Dockerfile para a aplicação Go.
- **[ ] Orquestrar a Produção (docker-compose.yml):**
  - Criar um arquivo docker-compose.yml de produção para orquestrar os contêineres do Backend, Frontend (servido via Caddy/Nginx) e