

Compilado Geral da Análise e Planejamento - FisioFlow Mobile

Data: 24 de Janeiro de 2026

Autor: Manus AI

Seção 1: Análise do Sistema FisioFlow Atual

Informações Gerais

- URL Produção:** <https://www.moocafisio.com.br/>
- Stack Tecnológico:** React 19.1.0 + TypeScript + Vite + Supabase
- UI Framework:** shadcn/ui + Radix UI + Tailwind CSS
- Gerenciador de Pacotes:** pnpm 9.15.0
- Node Version:** 24.x

Estrutura do Projeto

Frontend

- Framework:** React 19.1.0 com TypeScript
- Build Tool:** Vite 5.4.19
- Routing:** React Router DOM 6.30.1
- State Management:** Zustand 4.5.5
- Data Fetching:** TanStack Query 5.90.17
- Forms:** React Hook Form 7.61.1 + Zod 3.25.76

Backend/Database

- Database:** Supabase (PostgreSQL)
- ORM:** Drizzle ORM 0.45.1
- Auth:** Supabase Auth com RLS
- Real-time:** Supabase Real-time subscriptions
- Storage:** Vercel Blob + Vercel KV

Mobile (Preparado)

- **Framework:** Expo ~54.0.32
- **React Native:** 0.81.5
- **Navigation:** React Navigation 7.x
- **Styling:** NativeWind 4.2.1 (Tailwind para RN)
- **Capacitor:** @capacitor/core 8.0.1 (híbrido)

Funcionalidades Especiais

- **IA/ML:**
 - TensorFlow.js + MediaPipe para análise de movimento
 - OpenAI + Google Generative AI integrados
 - Vercel AI SDK
- **Automação:** Inngest 3.49.1 para background jobs
- **Análise de Imagens Médicas:** Cornerstone.js 4.15.5 (DICOM viewer)
- **PDF Generation:** jsPDF, React PDF Renderer
- **Monitoramento:** Sentry React 10.32.1
- **Analytics:** Vercel Analytics + Speed Insights

Interface Atual (Observações da Tela de Agenda)

Sidebar (Navegação Principal)

NÚCLEO

- Agenda ✓ (página atual)
- Dashboard
- Pacientes

CLÍNICA

- Prontuário
- Exercícios
- Protocolos

GESTÃO

- Comunicação
- Lista de Espera
- Tarefas

MÓDULOS (Expandíveis)

- Avaliações
- Cadastros
- CRM
- Operacionais
- Financeiro
- Relatórios
- Dashboard IA
- Administração
- Gamificação

SISTEMA

- Sair

Página de Agenda (Visualização Atual)

- **Layout:** Calendário semanal (19-25 jan 2026)
- **Horários:** 07:00 - 20:00 (intervalos de 30 min)
- **Visualizações:** Dia / Semana / Mês
- **Funcionalidades:**
 - Botão "Novo Agendamento" (destaque azul)
 - Filtros
 - Configurações da Agenda
 - Modo de Seleção (atalho: A)
 - Navegação entre semanas/dias
 - Contador: "219 agendamentos"
 - Status lista de espera
 - Indicador de usuários online (1 online)
 - Busca rápida (⌘ K)
- **Design Visual:**
 - Slots de horário com botões "+" para adicionar agendamentos
 - Cards de agendamento com nome do paciente e horário
 - Cores diferenciadas para slots

- Grid bem estruturado com linhas divisórias

Observações de UX/UI

1. **Sidebar bem organizada** com hierarquia clara (Núcleo > Clínica > Gestão > Módulos)
2. **Agenda é a página principal** (primeira no menu, foco do sistema)
3. **Interface limpa e moderna** com uso de Radix UI + Tailwind
4. **Muitos módulos disponíveis** - sistema robusto e completo
5. **Preparado para mobile** com Expo e React Native já configurados
6. **Sistema profissional** com autenticação, perfis de usuário, notificações

Dependências Relevantes para Mobile

Já Instaladas

- expo : ~54.0.32
- react-native : 0.81.5
- @react-navigation/native : 7.1.28
- @react-navigation/native-stack : 7.10.1
- @react-navigation/bottom-tabs : 7.10.1
- react-native-safe-area-context : 5.6.2
- react-native-screens : ~4.16.0
- react-native-gesture-handler : ~2.28.0
- expo-dev-client : ~6.0.20
- expo-splash-screen : ~31.0.0
- expo-status-bar : ~3.0.0
- nativewind : 4.2.1

Capacidades Mobile Preparadas

- Camera: react-native-vision-camera 4.7.3
- Biometria: expo-local-authentication ~17.0.0
- Haptics: expo-haptics ~15.0.0
- WebView: react-native-webview 13.15.0
- TensorFlow Lite: react-native-fast-tflite 2.0.0

Scripts Expo Disponíveis

JSON

```
"expo:start": "npx expo start",
"expo:android": "npx expo start --android",
"expo:ios": "npx expo start --ios",
"expo:web": "npx expo start --web",
"expo:build:dev": "eas build --profile development --platform ios",
"expo:build:preview": "eas build --profile preview --platform ios",
"expo:build:prod": "eas build --profile production --platform ios",
"expo:submit": "eas submit --platform ios"
```

Infraestrutura Atual

- **Hosting:** Vercel (configurado)
- **Database:** Supabase Pro
- **Edge Functions:** Supabase Functions
- **Cron Jobs:** Inngest para automações
- **Email:** SendGrid + Resend
- **Storage:** Vercel Blob

Escala Atual

- **600 atendimentos/mês**
- **15 profissionais**
- **219 agendamentos** (visível na interface)

Pontos Fortes Identificados

1. Stack moderna e escalável
2. Já possui estrutura Expo/React Native configurada
3. UI/UX profissional com shadcn/ui
4. Autenticação e segurança robustas (RLS)
5. Integração com IA (TensorFlow, OpenAI, Google AI)
6. Sistema de análise de movimento (MediaPipe)
7. Infraestrutura cloud preparada (Vercel + Supabase Pro)
8. Monitoramento e analytics configurados

Áreas de Atenção

1. ! Muitos módulos - pode ser complexo para usuários iniciantes
 2. ! Mobile ainda não está em produção (apenas preparado)
 3. ! Necessita separação clara entre app paciente e profissional
 4. ! React Native 0.81.5 está desatualizado (atual é 0.76.x)
 5. ! Expo 54 é recente, mas precisa validar compatibilidade
-

Seção 2: Insights de UX/UI para Apps de Saúde e Fisioterapia

Fonte Principal

Artigo: "How to Design a Fitness App: UX/UI Best Practices for Engagement and Retention"

URL: <https://www.zfort.com/blog/How-to-Design-a-Fitness-App-UX-UI-Best-Practices-for-Engagement-and-Retention>

Data: Abril 2025

Princípios Fundamentais

O design não é apenas sobre aparência, mas sobre como o app **se sente ao usar**. Em um mercado competitivo onde usuários abandonam apps após uma experiência ruim, o design é uma ferramenta essencial de retenção.

Equação Fundamental:

- **UX** = Consistência → fluxo suave, redução de fricção, jornadas que formam hábitos
- **UI** = Confiança → visuais claros, estrutura intuitiva, forte primeira impressão

9 Melhores Práticas de UX/UI

1. Onboarding Rápido e Sem Fricção

A primeira sessão é **tudo**. Se os usuários não entendem o valor ou não conseguem começar rapidamente, provavelmente nunca voltarão.

Princípios-chave:

- Comunicar valor cedo - mostrar o que o app fará por eles
- Minimizar passos - objetivo → plano → iniciar (idealmente em menos de 60 segundos)
- Modo convidado - deixar usuários explorarem sem criar conta

- Padrões inteligentes - pré-selecionar objetivos baseados em escolhas simples

Dicas de Interface:

- Barra de progresso para configuração (4 passos ou menos)
- Microcopy motivacional ("Vamos lá 💪")
- Prévias visuais do plano antes do signup
- Opções "Pular" / "Lembrar depois" para reduzir abandono

2. UI Simples e Focada

Desordem mata momentum. Um app de fitness deve ajudar usuários a iniciar, treinar e acompanhar - não distraí-los com complexidade.

Princípios de Design:

- Uma ação por tela - um botão, uma decisão, um objetivo
- Botões grandes e tocáveis - perfeito para uso em movimento
- Navegação clara - barra de abas com 3-5 áreas principais no máximo
- Dark mode e fontes escaláveis - melhor usabilidade em diferentes contextos

Dicas de Interface:

- CTAs em negrito ("Iniciar Treino", "Acompanhar Progresso")
- Contraste para ações importantes, tons suaves em outros lugares
- Evitar ícones sem rótulos - clareza > elegância
- Manter telas limpas durante treino (timer + próximo movimento apenas)

3. Modo de Treino Amigável ao Fluxo

Apps de fitness devem ajudar usuários a entrar e permanecer em **estado de fluxo**.

O que suporta o fluxo:

- Tela de treino sem distrações - sem pop-ups, sem barra de navegação, sem desordem
- Timer grande, movimento atual + próximo apenas
- Dicas de áudio e vibração - não precisa olhar para a tela
- Auto-carregamento do próximo exercício ou série

Melhorias Opcionais:

- Toggle "Não Perturbe" durante sessões
- Customização de tela: modo claro/escuro, apenas áudio, contador de repetições vs. cronômetro

- Saída rápida + auto-salvamento se usuário parar no meio do treino

4. Personalização que Parece Pessoal

Usuários não querem "um app de fitness". Eles querem **seu** app de fitness - um que se adapta aos seus objetivos, hábitos e progresso.

O que personalizar:

- Objetivos e preferências no onboarding
- Planos de treino adaptativos - ajustados baseados em performance e feedback
- Conteúdo recomendado - "Sugerido para você", "Retomar seu plano"
- Progresso em destaque - tornar streaks, badges e marcos visíveis na tela inicial

Detalhes de Design:

- Mostrar conclusão de meta semanal/mensal em forma visual
- Usar nome ou pronome do usuário ("Bom trabalho, Ana!")
- Tom adaptativo - gentil vs. insistente, dependendo do comportamento do usuário

5. Visualização de Progresso

Motivação prospera em resultados visíveis. Usuários precisam **ver** que seus esforços importam.

Elementos principais de feedback visual:

- Gráficos e linhas de tendência - comparações semanais/mensais
- Badges e conquistas - desbloqueio de marcos, recompensas de consistência
- Contadores de streak - "Você treinou 5 dias seguidos"
- Insights de performance - "+15% resistência desde o mês passado"

Dashboard e Visualização Diária:

- Tela "Hoje" - sessão planejada, check-in de humor, status de streak
- Dashboard de resultados - visão clara de metas atingidas, estatísticas e vitórias
- Comparação com o eu passado, não com outros - foco em crescimento pessoal

6. Microinterações e Gamificação Leve

Fitness é difícil. O design deve torná-lo **divertido, recompensador e vivo**.

Microinterações:

- Meta alcançada → animação + som/vibração

- Badge desbloqueado → confete, efeito de pulso, brilho
- Deslizar para completar → feedback tátil ou movimento de checkmark
- Streak de dias → pop-up com nudge "Continue assim!"

Camada de Gamificação:

- Pontos XP, níveis, streaks
- Desafios amigáveis - "Treine 3x esta semana"
- Recompensas semanais por consistência
- Barras de progresso para metas pessoais

Equilíbrio importa:

- Manter leve - evitar sobrecarga de jogo
- Deixar usuários optarem por elementos gamificados
- Vincular recompensas ao comportamento, não apenas cliques

7. Recursos Sociais que Motivam, Não distraem

Fitness se torna mais poderoso quando é **compartilhado**.

Interações Sociais Principais:

- Likes, comentários, emojis em treinos completados
- Progresso de amigos no feed ou dashboard
- Convidar amigos para participar de um desafio
- Celebrar juntos: "Você e Alex completaram 5 treinos esta semana!"

Objetivos Compartilhados e Desafios em Grupo:

- "Queimar 5000 kcal juntos esta semana"
- Leaderboards (visibilidade opcional)
- Streaks ou missões baseadas em equipe

Leve e Opcional:

- Permitir que usuários desativem recursos sociais
- Manter modos privados para individualistas
- Notificações apenas quando relevantes ou acionadas

8. Adaptação de Dispositivo e Idioma

Seu app não é usado apenas em uma tela - e seus usuários não pensam todos da mesma forma.

UX Multi-Dispositivo:

- Design mobile-first - amigável ao polegar, fluxo vertical
- Visualização em tablet - mais conteúdo por tela, layouts divididos para treinos
- Apps de TV (ex: Apple TV) - foco em visuais, controle por voz, timers grandes
- Wearables - estatísticas rápidas, controles rápidos, visualização minimalista de treino

Localização e Cultura:

- Tradução completa de texto - treinos, rótulos, onboarding
- Ajustar conteúdo (ex: estilos de treinamento, modelos) para normas regionais
- Ser sensível a tom, gestos e visuais
- Sistemas de data, hora, unidade: auto-adaptar (lbs/kg, milhas/km)

9. Melhoria Contínua de UX

Mesmo o melhor design nunca está "pronto". Hábitos de usuário evoluem, expectativas crescem e competidores se adaptam.

Testar o que Importa:

- Teste A/B: botões, layouts, copy, CTAs
- Testar: "Começar Agora" vs. "Treinar Hoje", pequenas mudanças de UI
- Testes de usabilidade com usuários reais - o que confunde, atrasa, frustra

Loops de Feedback:

- Prompts no app: "Isso foi fácil de usar?"
- Monitoramento e resposta a avaliações na App Store
- Enquetes rápidas após treinos ("Como foi isso?")

Análise de Comportamento:

- Rastrear mapas de calor de toques, telas de abandono, cliques de raiva
- Ferramentas: Hotjar, UXCam, Firebase Analytics
- Priorizar mudanças por impacto, não por opinião

Evoluir Sem Perder Familiaridade:

- Manter padrões principais estáveis (ex: barra de abas, fluxos-chave)
- Destacar mudanças: "Melhoramos seu dashboard"

- Evitar redesign por estética

Aplicação ao FisioFlow

Para App de Pacientes

1. **Onboarding simplificado** - cadastro com apenas nome, explorar exercícios prescritos imediatamente
2. **Visualização clara de progresso** - gráficos de evolução, badges de consistência
3. **Modo de exercício focado** - tela limpa com vídeo, timer e instruções
4. **Gamificação leve** - streaks de dias consecutivos, conquistas de progresso
5. **Comunicação com fisioterapeuta** - chat integrado, feedback rápido

Para App de Profissionais

1. **Dashboard poderoso** - visão geral de pacientes, agendamentos, métricas
2. **Prescrição rápida de exercícios** - biblioteca filtrada, templates de protocolos
3. **Registro de evolução eficiente** - notas SOAP, assinaturas digitais
4. **Gestão de agenda otimizada** - drag-and-drop, detecção de conflitos
5. **Análise de dados** - relatórios de adesão, progresso de pacientes, insights com IA

Tendências de Healthcare UX/UI 2026

- **Acessibilidade em primeiro lugar** - WCAG 2.1 AA compliance
- **Dark mode como padrão** - melhor para uso prolongado
- **Micro-animações** - feedback visual sutil para todas as ações
- **Personalização com IA** - conteúdo adaptado ao comportamento do usuário
- **Design inclusivo** - suporte a múltiplos idiomas, culturas e necessidades especiais

Seção 3: Comparativo de Tecnologias - React Native vs. Swift

Fonte

Artigo: "React Native vs Swift: Best Way to Build iOS Apps in 2026?"

URL: <https://www.mobiloud.com/blog/react-native-vs-swift>

Data: Dezembro 2025

Pontos-Chave Essenciais

- **Swift:** Linguagem de programação nativa para iOS, mantida pela Apple. Oferece a melhor performance possível e acesso total às APIs do sistema.
- **React Native:** Framework cross-platform que usa JavaScript/TypeScript para construir apps para iOS e Android a partir de uma única base de código, mantido pela Meta.

Tabela Comparativa para o Cenário FisioFlow

Fator	React Native + Expo	Swift (Nativo)	Vencedor para FisioFlow
Velocidade de Desenvolvimento	30-50% mais rápido (código único)	Mais lento (código separado por plataforma)	React Native
Custo de Desenvolvimento	40-50% menor	Mais caro (mais tempo, desenvolvedores mais caros)	React Native
Reaproveitamento de Código	~80-90% com o sistema web e entre iOS/Android	0%	React Native
Necessidade de Hardware (Mac)	Não (graças ao Expo EAS Build na nuvem)	Obrigatório	React Native
Performance	Excelente para este tipo de app	Ligeiramente superior, mas imperceptível aqui	Empate
Ecossistema e Time	Usa o stack atual (React, TS, Firebase)	Requer aprendizado de um novo ecossistema	React Native
Flexibilidade Futura (Android)	Quase 90% do app já está pronto	Requer desenvolver um novo app do zero	React Native

Análise de Performance para Fisioterapia

Para um aplicativo de fisioterapia com funcionalidades como visualização de vídeos de exercícios, agendamentos, prontuários, chat e notificações, a performance do **React Native**

é mais do que suficiente. Aplicativos de grande escala como Instagram, Facebook, Discord e Airbnb utilizam React Native com milhões de usuários, provando sua viabilidade e robustez para casos de uso complexos.

Desenvolvimento sem um Computador Mac

- **Com React Native + Expo:** É 100% possível desenvolver e publicar no Ubuntu. O desenvolvimento é feito localmente, e a compilação (build) e submissão para a App Store são feitas na nuvem através dos serviços EAS (Expo Application Services).
- **Com Swift:** É impossível sem um Mac. O Xcode, ferramenta essencial para o desenvolvimento nativo da Apple, só funciona no macOS. Alternativas como VMs são instáveis e violam os termos de serviço da Apple.

Conclusão da Análise Técnica

Para o projeto FisioFlow, **React Native com Expo é a escolha estratégica correta e definitiva.** A decisão se baseia em:

1. **Desenvolvimento mais rápido** (2-3 meses vs. 4-6 meses).
2. **Custo significativamente menor** (economia de 40-50%).
3. **Reaproveitamento massivo de código** e conhecimento do sistema web atual.
4. **Independência de hardware da Apple**, permitindo o desenvolvimento no seu ambiente atual.
5. **Flexibilidade para expansão** para o Android no futuro com esforço mínimo.
6. **Performance adequada** para todas as funcionalidades planejadas.
7. **Ecossistema maduro** e bem suportado pela comunidade e por empresas como a Meta.

Seção 4: Guia de Desenvolvimento iOS sem Mac com Expo EAS

O que é EAS (Expo Application Services)?

EAS é um conjunto de ferramentas da Expo que permite construir e distribuir aplicativos React Native de forma profissional. Seu principal benefício para o seu cenário é a capacidade de **compilar e publicar aplicativos iOS sem a necessidade de um computador Mac.**

Componentes Principais

1. **EAS Build:** Um serviço na nuvem que compila seu código em um ambiente nativo (incluindo macOS para iOS) e gera o binário do aplicativo (`.ipa` para iOS).
2. **EAS Submit:** Um serviço que pega o binário gerado pelo EAS Build e o envia automaticamente para a App Store Connect (para iOS) ou Google Play Store (para Android).
3. **EAS Update:** Permite enviar atualizações de JavaScript e assets (imagens, etc.) diretamente para os usuários sem precisar passar por uma nova revisão da loja de aplicativos, ideal para correções rápidas.

Workflow Completo de Desenvolvimento e Publicação no Ubuntu

1. **Desenvolvimento:** Você escreve o código do seu aplicativo em TypeScript/React Native no seu ambiente Ubuntu, usando o VS Code ou seu editor de preferência.
2. **Teste Local:** Você pode testar o aplicativo em um emulador Android no seu Ubuntu ou em um dispositivo iOS físico usando o app Expo Go.
3. **Build na Nuvem:** Quando estiver pronto para criar uma versão para a App Store, você executa o comando `eas build --profile production --platform ios` no seu terminal.
4. **Compilação:** O EAS CLI envia seu código para os servidores da Expo, que o compilam em uma máquina Mac virtual, gerando o arquivo `.ipa`.
5. **Submissão Automática:** Com o comando `eas submit`, o EAS envia esse arquivo `.ipa` diretamente para a sua conta na App Store Connect, pronto para ser distribuído via TestFlight (para testes) ou para revisão final.

Pré-requisitos

- Conta Expo (gratuita para começar).
- Conta Apple Developer (\$99/ano), que você já possui.
- EAS CLI instalado globalmente (`npm install -g eas-cli`).

Custos

- **Plano Gratuito:** Oferece 30 builds por mês, o que é suficiente para a fase inicial de desenvolvimento do MVP.
- **Plano Production:** Custa \$29/mês e oferece builds ilimitados, sendo uma opção muito mais econômica do que comprar e manter um Mac.

Conclusão sobre o EAS

Para o FisioFlow, **EAS Build** é a solução ideal e recomendada. Ela remove o principal obstáculo para o desenvolvimento iOS em ambientes não-Apple, oferecendo um fluxo de trabalho profissional, automatizado e com excelente custo-benefício.

Seção 5: Relatório Estratégico Revisado (com Firebase)

Data: 24 de Janeiro de 2026

Autor: Manus AI

1. Sumário Executivo

Este relatório, revisado de acordo com as novas decisões técnicas, apresenta um plano estratégico para o desenvolvimento de dois aplicativos móveis para iOS (Pacientes e Profissionais) para o ecossistema FisioFlow. A recomendação central é a adoção de **React Native com Expo**, utilizando **Firebase como backend** (Auth, Firestore, Cloud SQL, etc.). Esta abordagem alavanca a expertise em JavaScript/TypeScript, reduz drasticamente o tempo e o custo de desenvolvimento e viabiliza o ciclo completo de desenvolvimento e publicação em um ambiente Ubuntu, sem a necessidade de um Mac.

2. Estratégia de Desenvolvimento Mobile

2.1. Dois Aplicativos Separados: A Decisão Correta

A decisão de criar dois aplicativos distintos — **FisioFlow Pacientes** e **FisioFlow Profissionais** — está mantida e é a melhor abordagem estratégica. As necessidades e jornadas de cada perfil são fundamentalmente diferentes, e esta separação permitirá criar experiências otimizadas para cada público.

- **App do Paciente:** Foco em simplicidade, engajamento, visualização de exercícios e acompanhamento de progresso.
- **App do Profissional:** Ferramenta de produtividade focada em gestão de agenda, prontuários (SOAP) e análise de dados.

2.2. Stack Tecnológica (Revisada)

Componente	Tecnologia Escolhida	Observações
Frontend Mobile	React Native + Expo	Aproveita o conhecimento existente em React/TS.
Backend & DB	Firebase (Google Cloud)	Substitui o Supabase, alinhado com a nova decisão.

Autenticação	Firebase Authentication	Para login com Google, Apple, Email/Senha.
Banco de Dados	Cloud Firestore + Cloud SQL	Firestore para dados real-time e Cloud SQL para dados relacionais.
Build & Deploy	Expo EAS Build & Submit	Permite o desenvolvimento e publicação a partir do Ubuntu.

3. Planejamento e Estrutura dos Reppositórios

A recomendação de manter um **monorepo** (um único repositório Git) continua sendo a ideal para maximizar o reuso de código e simplificar a manutenção.

Estrutura de Pastas Sugerida (Revisada com Firebase):

Plain Text

```
/fisioflow-monorepo
├── apps
│   ├── web           # Código do sistema web atual
│   ├── mobile-patient # App do Paciente
│   └── mobile-pro    # App do Profissional
└── packages
    ├── ui            # Componentes React compartilhados
    ├── api           # Lógica de comunicação com Firebase
    └── types          # Tipos TypeScript compartilhados
    ... (outras configurações)
```

4. Roadmap de Desenvolvimento (Ajustado)

O desenvolvimento será iterativo, com foco na entrega de valor rápido. **Conforme solicitado, a Fase 2 (App do Profissional) será priorizada.**

Fase 1: Planejamento e Análise (Concluída)

Fase 2: App do Profissional (MVP - 3 a 5 semanas) - INICIANDO AGORA

O objetivo é criar uma ferramenta de produtividade para gestão em movimento.

- **Autenticação:** Login com Firebase Auth.
- **Dashboard Mobile:** Visão rápida do dia (pacientes, receita, etc.).

- **Agenda Mobile:** Visualização e gestão de agendamentos (criar, editar, cancelar).
- **Lista de Pacientes:** Acesso rápido à lista de pacientes com busca.
- **Perfil do Paciente (Simplificado):** Histórico de agendamentos e último prontuário.
- **Prontuário Rápido (SOAP):** Adicionar notas de evolução pelo celular.

Fase 3: App do Paciente (MVP - 2 a 4 semanas)

O objetivo é engajamento e adesão ao tratamento.

- **Onboarding Simplificado:** Login com Firebase Auth.
- **Tela Inicial (Hoje):** Visualização do plano do dia.
- **Meus Exercícios:** Lista de exercícios com vídeos e contador.
- **Tela de Execução de Exercício:** Modo focado com timer e feedback.
- **Acompanhamento de Progresso:** Gráficos de consistência e evolução.
- **Notificações Push:** Lembretes via Firebase Cloud Messaging.

Fase 4: Melhorias e Novas Funcionalidades (Contínuo)

- **Engajamento do Paciente:** Gamificação, mapa da dor interativo, chat com o profissional.
- **Produtividade do Profissional:** Prescrição de exercícios mobile, análise de movimento com IA, assinatura digital.

5. Próximos Passos e Recomendações

1. **Setup do Firebase:** Criar o projeto no Firebase, configurar Auth, Firestore, Cloud SQL e obter as credenciais.
2. **Estruturação do Monorepo:** Criar as pastas `apps/mobile-pro` e `packages/api` (para Firebase).
3. **Desenvolvimento do MVP do App Profissional:** Iniciar o desenvolvimento seguindo o plano detalhado para o Claude Code.

Com a mudança para Firebase, a base do projeto continua sólida e a estratégia de desenvolvimento com React Native e Expo permanece a mais eficiente e econômica.
