Plano de Implementação para Claude: App FisioFlow Profissionais (MVP)

Para: Claude Code

De: Manus AI

Projeto: FisioFlow - App Mobile para Profissionais

Fase: 2 (MVP)

Data: 24 de Janeiro de 2026

---

1. Objetivo

Implementar o **MVP (Minimum Viable Product)** do aplicativo **FisioFlow Profissionais** em React Native e Expo, utilizando Firebase como backend. O foco principal é criar uma ferramenta de produtividade móvel para fisioterapeutas, com ênfase na gestão de agendamentos e prontuários.

2. Decisões Técnicas (Já Aprovadas)

- **Framework**: React Native com Expo (SDK 54+)
- **Linguagem**: TypeScript
- **Backend**: Firebase (Auth, Firestore, Storage, Functions)
- **Estilização**: NativeWind (Tailwind CSS para React Native)
- **UI Library**: React Native Paper (para componentes complexos como Modals, Inputs)
- **Navegação**: React Navigation (Bottom Tabs + Native Stack)
- **State Management**: Zustand
- **Data Fetching**: TanStack React Query v5
- **Ambiente**: Desenvolvimento em Ubuntu, build iOS via Expo EAS.

3. Estrutura do Projeto (Monorepo)

Você trabalhará dentro da pasta `apps/mobile-pro/`. A estrutura inicial deve ser:

Plain Text

```
/fisioflow-monorepo
├── apps/
```

```
│   └── mobile-pro/             # <== VOCÊ ESTÁ AQUI
│       ├── src/
│       │   ├── api/              # Lógica de comunicação com Firebase
│       │   ├── assets/           # Imagens, fontes
│       │   ├── components/       # Componentes reutilizáveis (UI, etc)
│       │   ├── hooks/            # Hooks customizados
│       │   ├── navigation/       # Configuração do React Navigation
│       │   ├── screens/          # Telas do aplicativo
│       │   └── store/            # Stores do Zustand
│       ├── app.json              # Configuração do App Expo
│       ├── eas.json              # Configuração do EAS Build
│       ├── index.js              # Ponto de entrada
│       └── package.json
└── packages/
    ├── api/                    # (Opcional) Mover api/ para cá se for comparti
    └── types/                  # Tipos TypeScript compartilhados
```

4. Passo a Passo da Implementação

Tarefa 0: Setup Inicial e Firebase

1. **Inicialize o projeto Expo**: Dentro de `apps/mobile-pro`, execute `npx expo prebuild --clean` para gerar as pastas `ios` e `android` se necessário para configurar libs nativas.
2. **Instale as dependências principais** (ver `package.json` abaixo).
3. **Configure o Firebase**: Crie um arquivo `src/api/firebase.ts`. As credenciais do Firebase (google-services.json, etc.) serão fornecidas.

TypeScript

```
// src/api/firebase.ts
import { initializeApp } from 'firebase/app';
import { getAuth } from 'firebase/auth';
import { getFirestore } from 'firebase/firestore';
import { getStorage } from 'firebase/storage';

const firebaseConfig = {
  // ... (credenciais serão injetadas aqui)
};

const app = initializeApp(firebaseConfig);
export const auth = getAuth(app);
export const db = getFirestore(app);
export const storage = getStorage(app);
```

4. **Configure NativeWind**: Siga a documentação do NativeWind para configurar o `tailwind.config.js`.

Tarefa 1: Autenticação (Firebase Auth)

1. **Crie as telas de Autenticação** em `src/screens/auth/`:
   - `LoginScreen.tsx`: Campos para email e senha, botão "Entrar", link para "Esqueci minha senha".
   - `ForgotPasswordScreen.tsx`: Campo para email, botão para enviar link de recuperação.
2. **Implemente a lógica de autenticação** em `src/hooks/useAuth.ts`:
   - `signInWithEmail(email, password)`
   - `signOut()`
   - `sendPasswordReset(email)`
3. **Gerencie o estado do usuário** com Zustand em `src/store/authStore.ts`. O store deve conter `user`, `isLoggedIn`, etc.
4. **Configure a Navegação**: Crie um Stack Navigator que mostra as telas de Auth se `isLoggedIn` for `false`, e o App principal (Bottom Tab Navigator) se for `true`.

Tarefa 2: Navegação Principal (Bottom Tab)

Crie um Bottom Tab Navigator em `src/navigation/AppNavigator.tsx` com 4 abas:

1. **Dashboard**: Ícone de "home"
2. **Agenda**: Ícone de "calendar"
3. **Pacientes**: Ícone de "users"
4. **Perfil**: Ícone de "user"

Tarefa 3: Tela de Agenda (PRIORIDADE MÁXIMA)

Esta é a tela mais importante. A experiência do usuário deve ser fluida e rápida.

1. **Estrutura da Tela** (`src/screens/app/AgendaScreen.tsx`):
   - **Header**: Um seletor de data (ex: `react-native-calendars`) e botões para navegar entre dias/semanas.
   - **Corpo**: Uma `ScrollView` vertical com a grade de horários.
2. **Componente de Grade de Horários** (`src/components/agenda/ScheduleGrid.tsx`):
   - Renderize as horas do dia (ex: 07:00, 07:30, 08:00...).
   - Use linhas para separar as horas cheias e meias-horas.

- Sobreponha os agendamentos buscados do Firestore nesta grade.

3. **Componente de Agendamento** (`src/components/agenda/AppointmentCard.tsx`):
   - Card que representa um agendamento.
   - Deve exibir: Nome do paciente, tipo de atendimento, status (confirmado, pendente).
   - Ao ser pressionado, abre o Modal de Edição.
4. **Modal de Agendamento/Edição** (`src/components/agenda/AppointmentModal.tsx`):
   - Use `Modal` do React Native Paper.
   - Ao clicar em um horário vago na grade, o modal abre para **criar** um novo agendamento.
   - Ao clicar em um `AppointmentCard` , o modal abre para **editar** o agendamento existente.
   - **Campos**: Busca de paciente, seletor de data/hora, tipo de atendimento, status, botão de salvar/atualizar e botão de excluir.
5. **Lógica de Dados**:
   - Use TanStack Query para buscar (`useQuery`) e atualizar (`useMutation`) os agendamentos.
   - A query deve ser refetchada ao mudar de data.
   - Use os listeners real-time do Firestore para atualizações automáticas na tela.

Tarefa 4: Telas de Pacientes

1. **Tela de Lista de Pacientes** (`src/screens/app/PatientsListScreen.tsx`):
   - **Header**: Barra de busca para filtrar pacientes por nome.
   - **Corpo**: Uma `FlatList` renderizando os pacientes.
   - **Componente de Item** (`src/components/patients/PatientListItem.tsx`): Exibe foto, nome e contato do paciente. Ao clicar, navega para o perfil do paciente.
2. **Tela de Perfil do Paciente** (`src/screens/app/PatientProfileScreen.tsx`):
   - Use um Native Stack Navigator para esta seção.
   - Exibe informações detalhadas do paciente.
   - Lista o histórico de agendamentos.
   - Lista o histórico de prontuários (SOAP).
   - Botão para adicionar um novo prontuário (navega para a tela de SOAP).

Tarefa 5: Tela de Prontuário (SOAP)

1. **Tela de Criação/Edição de Prontuário** (`src/screens/app/SoapNoteScreen.tsx`):
    - Um formulário com 4 campos de texto grandes (`TextInput` do React Native Paper):
        - **S** (Subjetivo): O que o paciente relata.
        - **O** (Objetivo): O que o fisioterapeuta observa/mede.
        - **A** (Avaliação): Análise e diagnóstico do profissional.
        - **P** (Plano): Plano de tratamento para as próximas sessões.
    - Botão "Salvar Prontuário".

Tarefa 6: Tela de Dashboard

1. **Tela de Dashboard** (`src/screens/app/DashboardScreen.tsx`):
    - **Componente de Resumo do Dia** (`src/components/dashboard/SummaryWidget.tsx`): Cards mostrando: Nº de pacientes hoje, Receita estimada do dia, Novos pacientes.
    - **Componente de Próximos Agendamentos** (`src/components/dashboard/UpcomingAppointments.tsx`): Uma lista simples dos próximos 3-4 agendamentos.

5. Estrutura de Dados (Firestore)

Sugestão de coleções:

- `professionals/` : (ID do profissional) -> informações do profissional
- `patients/` : (ID do paciente) -> informações do paciente
- `appointments/` : (ID do agendamento) -> { professionalId, patientId, date, startTime, endTime, status, ... }
- `soapNotes/` : (ID da nota) -> { patientId, appointmentId, subjective, objective, assessment, plan, ... }

6. `package.json` Sugerido

JSON

```
{
  "name": "mobile-pro",
  "version": "1.0.0",
  "private": true,
  "main": "index.js",
  "scripts": {
    "start": "expo start",
```

```
    "android": "expo start --android",
    "ios": "expo start --ios",
    "web": "expo start --web"
  },
  "dependencies": {
    "expo": "~54.0.0",
    "react-native": "0.76.0",
    "react": "19.0.0-rc.0",
    "typescript": "^5.3.3",
    "@react-navigation/native": "^6.1.17",
    "@react-navigation/bottom-tabs": "^6.5.20",
    "@react-navigation/native-stack": "^6.9.26",
    "react-native-screens": "~3.31.1",
    "react-native-safe-area-context": "4.10.1",
    "firebase": "^10.12.2",
    "nativewind": "^4.0.1",
    "react-native-paper": "^5.12.3",
    "@tanstack/react-query": "^5.45.1",
    "zustand": "^4.5.2",
    "react-native-calendars": "^1.1305.0"
  },
  "devDependencies": {
    "@types/react": "~18.2.79",
    "tailwindcss": "^3.4.4"
  }
}
```

7. Checklist de Implementação

- [ ] **Tarefa 0**: Setup do Projeto e Firebase.
- [ ] **Tarefa 1**: Implementar fluxo de Autenticação (Login, Logout, Reset de Senha).
- [ ] **Tarefa 2**: Criar Navegação Principal (Bottom Tab).
- [ ] **Tarefa 3**: Implementar Tela de Agenda (Grid, Cards, Modal).
- [ ] **Tarefa 4**: Implementar Telas de Pacientes (Lista e Perfil).
- [ ] **Tarefa 5**: Implementar Tela de Prontuário SOAP.
- [ ] **Tarefa 6**: Implementar Tela de Dashboard.
- [ ] **Final**: Garantir que o app está funcional e pronto para testes.

Observação Final: A prioridade máxima é a **Tela de Agenda**. Ela deve ser robusta, rápida e intuitiva. Comece por ela após a autenticação estar funcional. Boa implementação!