# Documento de Estrutura: MedControl MVP

## 1. Visão Geral

Um aplicativo Android para gestão de medicamentos com foco em **interconectividade**. O app permite que um responsável monitore a adesão ao tratamento de dependentes e receba alertas caso uma dose seja esquecida.

## 2. Stack Tecnológica

* **Linguagem:** Kotlin
* **Interface:** Jetpack Compose (Modern UI)
* **Arquitetura:** MVVM (Model-View-ViewModel) - *Padrão recomendado pelo Google.*
* **Banco de Dados & Auth:** Firebase (Firestore, Auth e Cloud Messaging para notificações).
* **Gerenciamento de Dependências:** Gradle (Kotlin DSL).

## 3. Arquitetura de Dados (O Cérebro)

Conforme definimos, utilizaremos um modelo NoSQL estruturado para permitir escalabilidade:

* **Coleção users:** Perfis, tipos de conta e vínculos.
* **Coleção medications:** Detalhes dos remédios e níveis de estoque.
* **Coleção schedules:** Histórico de doses (passadas e futuras).
* **Coleção connections:** Tabela de permissões entre Master, Dependente e Controle.

## 4. Plano de Execução (Passo a Passo)

Para construir com IA de forma eficiente, seguiremos esta ordem:

### Fase 1: Fundação (Configuração)

1. **Configuração do Projeto:** Criar o projeto no Android Studio e conectar ao console do Firebase.
2. **Modelagem:** Criar as Data Classes em Kotlin (que definimos anteriormente).
3. **Autenticação:** Implementar Login/Cadastro (E-mail/Senha) e criação de Perfil inicial.

### Fase 2: Funcionalidade Core (O Dependente)

1. **Cadastro de Medicamento:** Tela para inserir nome, dose e estoque.
2. **Lógica de Agendamento:** Criar a função que gera os horários (ex: de 8h em 8h) no banco de dados.
3. **Sistema de Alarmes Local:** Configurar o AlarmManager do Android para tocar no celular do dependente mesmo offline.

### Fase 3: Conectividade (O Diferencial)

1. **Sistema de Convites:** Gerar um ID único e permitir que um "Master" adicione um "Dependente" ou "Controle".
2. **Sincronização Cloud:** Garantir que quando o dependente marcar "Tomei", o status mude no Firebase em tempo real.
3. **Cloud Functions (Opcional) ou Alertas:** Lógica para disparar notificação para o "Controle" se o status não mudar para "TAKEN" após 15 minutos do horário previsto.

### Fase 4: Monetização e Ajustes

1. **Paywall:** Trava lógica que impede adicionar mais de 1 contato se o plano for "FREE".
2. **Gestão de Estoque:** Alerta visual quando o medicamento estiver acabando.

## 5. Fluxo de Funcionamento (User Logic)

1. **O Dependente** recebe uma notificação local: *"Hora do Dipirona"*.
2. Ele abre o app e clica em **"Confirmar Dose"**.
3. **O Sistema** decrementa 1 unidade do estoque e marca como concluído no banco.
4. **SE** ele não clicar em 15 minutos:
   * O Firebase detecta o atraso.
   * Uma notificação Push é enviada para o **Contato de Controle**: *"João ainda não tomou o remédio X"*.

## 1. Fluxo de Telas (Opção B)

**Imagine o caminho que o usuário faz dentro do app. Para o MVP, precisamos de 5 núcleos de telas:**

### A. Onboarding e Login

* **Tela de Boas-vindas: Explicação rápida (3 slides) sobre o controle e a rede de apoio.**
* **Login/Cadastro: E-mail/Senha ou Google.**
* **Escolha de Perfil: "Sou um Dependente" ou "Sou um Responsável/Master".**

### B. Dashboard (Home)

* **Visão do Dependente: Lista dos próximos remédios do dia com botão de "Check" gigante.**
* **Visão do Master: Lista de pessoas que ele cuida e o status de cada uma (Verde: Tudo ok | Vermelho: Atraso).**

### C. Gestão de Medicamentos (Cadastro)

* **Formulário: Nome, Foto (opcional), Dosagem, Frequência (horas) e Quantidade em Estoque.**
* **Configuração de Alerta de Estoque Baixo (ex: "Avisar quando restar 5 unidades").**

### D. Rede de Cuidado (Conexões)

* **Tela de Gerenciamento: Listar Dependentes e Contatos de Controle.**
* **Adicionar Novo: Campo para digitar o ID do outro usuário ou ler QR Code.**
* **Status de Convite: "Pendente" ou "Aceito".**

### E. Consultas Médicas

* **Calendário simples para marcar data/hora, especialidade e qual dependente irá à consulta.**

## 

## 

## 

## 2. Lógica dos Planos Pagos (Opção C)

**Para o seu modelo de negócio, a trava lógica deve ser baseada na Coleção de Conexões.**

| **Recurso** | **Plano Free (Gratuito)** | **Plano Premium (Pago)** |
| --- | --- | --- |
| **Dependentes** | **Máximo 1** | **Ilimitado (ou conforme o nível)** |
| **Contatos de Controle** | **Máximo 1** | **Até 5 por dependente** |
| **Histórico** | **Últimos 7 dias** | **Vitalício** |
| **Alertas de Estoque** | **Notificação Simples** | **Notificação + SMS/WhatsApp (opcional)** |

**Regra de Implementação: Quando o usuário Master clicar em "Adicionar Dependente", o app verifica no Firebase: if (user.plan == "FREE" && user.dependentCount >= 1) { Mostrar Paywall }.**

3. Estrutura de Pastas e Data Classes (Opção A)

**Agora, a base técnica para você levar para a IA de código. No Android (Kotlin), usamos o padrão Clean Architecture simplificado:**

### Estrutura de Pastas:

**Plaintext**

**com.seuapp.medcontrol/**

**├── data/ # Modelos de dados e Firebase**

**│ ├── model/ # Data Classes (User, Med, Schedule)**

**│ └── repository/ # Funções de salvar/ler do Firebase**

**├── ui/ # Telas (Jetpack Compose)**

**│ ├── components/ # Botões e cards reutilizáveis**

**│ ├── dashboard/ # Tela principal**

**│ ├── auth/ # Login e Cadastro**

**│ └── theme/ # Cores e fontes**

**├── viewmodel/ # Lógica por trás das telas**

**└── util/ # Calculadora de horários e notificações**

### O "Cérebro" em Código (Data Classes):

**Aqui estão as classes prontas para o seu banco de dados:**

**Kotlin**

**// Perfil do Usuário**

**data class UserProfile(**

**val uid: String = "",**

**val name: String = "",**

**val role: String = "MASTER", // MASTER, DEPENDENT, CONTROLLER**

**val planType: String = "FREE",**

**val myMasterId: String? = null // Preenchido se for dependente**

**)**

**// O Remédio**

**data class Medication(**

**val id: String = "",**

**val name: String = "",**

**val dosage: String = "",**

**val currentStock: Int = 0,**

**val alertThreshold: Int = 5,**

**val intervalInHours: Int = 8,**

**val ownerId: String = "" // UID do dependente**

**)**

**// O Alerta de Horário**

**data class DoseSchedule(**

**val id: String = "",**

**val medId: String = "",**

**val timeMillis: Long = 0L,**

**val status: String = "PENDING", // PENDING, TAKEN, MISSED**

**val confirmedAt: Long? = null**

**)**