Documentação Detalhada do Projeto MedicamentosApp

8 de Junho de 2025

Versão 1.0
Documentação para Desenvolvedores e Stakeholders

Contents

1 Introdução **2**

2 Objetivos do Aplicativo **2**

3 Estrutura do Projeto **2**

4 Configuração do Firebase **2**
4.1 Pré-requisitos . . . 2
4.2 Passos de Configuração . . . 3
4.3 Testando a Configuração . . . 4

5 Pacotes e Classes **4**
5.1 Pacote `com.dosecerta.medicamentosapp` . . . 4
5.1.1 `MainActivity` . . . 4
5.1.2 `LoginActivity` . . . 5
5.1.3 `CadastroUsuarioActivity` . . . 6
5.1.4 `MedicamentosActivity` . . . 6
5.1.5 `CadastroMedicamentoActivity` . . . 6
5.1.6 `DetalhesMedicamentoActivity` . . . 7
5.1.7 `PerfilActivity` . . . 7
5.1.8 `RepositorioDeMedicamentos` . . . 7
5.2 Pacote `com.dosecerta.medicamentosapp.data` . . . 7
5.2.1 `AuthRepository` . . . 7
5.2.2 `MedicamentosRepository` . . . 7
5.2.3 `StorageRepository` . . . 7
5.3 Pacote `com.dosecerta.medicamentosapp.model` . . . 7
5.3.1 `Medicamento` . . . 7
5.4 Pacote `com.dosecerta.medicamentosapp.ui.theme` . . . 8

6 Fluxos de Funcionalidades **8**

7 Dependências **8**

8 Melhorias Sugeridas **8**

1 Introdução

O **MedicamentosApp** é um aplicativo Android desenvolvido em Kotlin para gerenciamento de medicamentos. Ele permite que usuários autenticados cadastrem, listem, visualizem detalhes e associem imagens a medicamentos, além de configurarem lembretes. O aplicativo utiliza o **Firebase** para autenticação (`Firebase Authentication`), armazenamento de dados (`Firestore`), e upload de arquivos (`Storage`). A interface combina layouts XML tradicionais com temas modernos implementados via **Jetpack Compose**, suportando modos claro e escuro.

Esta documentação detalha a arquitetura, funcionamento, configuração do Firebase, fluxos de usuário, e sugestões de melhorias. O objetivo é fornecer um guia completo para desenvolvedores que desejam entender, manter ou expandir o projeto.

2 Objetivos do Aplicativo

- Permitir que usuários autenticados gerenciem medicamentos com informações como nome, dosagem, frequência e quantidade.
- Oferecer integração em tempo real com o Firebase para armazenamento e sincronização de dados.
- Suportar captura e upload de imagens de medicamentos.
- Proporcionar uma interface intuitiva com suporte a temas dinâmicos.
- Facilitar a configuração de lembretes (atualmente com feedback visual via `Toast`).

3 Estrutura do Projeto

O projeto segue uma arquitetura modular, com pacotes que separam responsabilidades:

- `com.dosecerta.medicamentosapp`: Contém as atividades (`Activities`) responsáveis pela interface e navegação.
- `com.dosecerta.medicamentosapp.data`: Inclui repositórios para interação com Firebase.
- `com.dosecerta.medicamentosapp.model`: Define o modelo de dados para medicamentos.
- `com.dosecerta.medicamentosapp.ui.theme`: Configura temas e tipografia com Jetpack Compose.

Os layouts XML estão em `res/layout`, e os recursos de strings em `res/values`. A arquitetura utiliza o padrão de repositórios para isolar a lógica de acesso a dados, promovendo manutenibilidade e escalabilidade.

4 Configuração do Firebase

O aplicativo depende do Firebase para autenticação, banco de dados e armazenamento. Abaixo estão os passos detalhados para configurar o Firebase no projeto.

4.1 Pré-requisitos

- Conta no Firebase Console.
- Android Studio com suporte a Gradle.
- Aplicativo registrado no Firebase com o pacote `com.dosecerta.medicamentosapp`.

4.2 Passos de Configuração

1. **Criar Projeto no Firebase**:
   - Acesse o Firebase Console.
   - Clique em "Adicionar projeto" e insira um nome (ex.: "MedicamentosApp").
   - Desative o Google Analytics, se não for necessário.
2. **Registrar o Aplicativo**:
   - No console, clique em "Adicionar aplicativo" e selecione Android.
   - Insira o nome do pacote (`com.dosecerta.medicamentosapp`).
   - Baixe o arquivo `google-services.json` e coloque-o na pasta `app/` do projeto.
3. **Configurar Gradle**:
   - No arquivo `build.gradle` (nível do projeto), adicione:

```
buildscript {
    dependencies {
        classpath 'com.google.gms:google-services:4.4.2'
    }
}
```

   - No arquivo `build.gradle` (nível do módulo `app`), adicione:

```
plugins {
    id 'com.google.gms.google-services'
}
dependencies {
    implementation 'com.google.firebase:firebase-auth:23.0.0'
    implementation 'com.google.firebase:firebase-firestore
        :25.0.0'
    implementation 'com.google.firebase:firebase-storage
        :21.0.0'
    implementation 'androidx.activity:activity-ktx:1.9.0'
}
```

   - Sincronize o projeto no Android Studio.
4. **Habilitar Serviços no Firebase**:
   - No Firebase Console, habilite:
     - **Authentication**: Ative o provedor "E-mail/Senha".
     - **Firestore**: Crie um banco de dados em modo de produção.
     - **Storage**: Configure regras de armazenamento (exemplo abaixo).
5. **Regras de Segurança do Firestore**:
   - Configure as regras para permitir acesso apenas a usuários autenticados:

```
rules_version = '2';
service cloud.firestore {
    match /databases/{database}/documents {
        match /usuarios/{userId}/{document=**} {
            allow read, write: if request.auth != null &&
                request.auth.uid == userId;
        }
    }
```

```
}
```

6. **Regras de Segurança do Storage**:
   - Configure as regras para upload de imagens:

```
rules_version = '2';
service firebase.storage {
    match /b/{bucket}/o {
        match /imagens_medicamentos/{allPaths=**} {
            allow read, write: if request.auth != null;
        }
    }
}
```

7. **Inicializar o Firebase no Aplicativo**:
   - O plugin `google-services` inicializa automaticamente o Firebase ao compilar o projeto.
   - As classes `AuthRepository`, `MedicamentosRepository` e `StorageRepository` já utilizam as instâncias do Firebase.

4.3 Testando a Configuração

- Execute o aplicativo em um emulador ou dispositivo.
- Tente cadastrar um usuário via `CadastroUsuarioActivity`.
- Verifique no Firebase Console se o usuário aparece na seção Authentication.
- Adicione um medicamento e confirme se ele é salvo na coleção `usuarios/{userId}/medicamentos` no Firestore.
- Faça upload de uma imagem e verifique se ela aparece em `imagens_medicamentos/` no Storage.

5 Pacotes e Classes

5.1 Pacote `com.dosecerta.medicamentosapp`

5.1.1 `MainActivity`

- **Descrição**: Ponto de entrada que verifica o estado de autenticação e redireciona.
- **Funcionamento**:
  - Instancia `AuthRepository` e verifica `currentUser`.
  - Usa `Intent` para navegar para `MedicamentosActivity` (usuário autenticado) ou `LoginActivity` (não autenticado).
  - Chama `finish()` para evitar retorno à tela inicial.
- **Código principal**:

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    val auth = AuthRepository()
    if (auth.currentUser != null) {
        startActivity(Intent(this, MedicamentosActivity::class.java
            ))
```

```
    } else {
        startActivity(Intent(this, LoginActivity::class.java))
    }
    finish()
}
```

- **Navegação**:
  - Autenticado → `MedicamentosActivity`.
  - Não autenticado → `LoginActivity`.

5.1.2 `LoginActivity`

- **Descrição**: Tela de login com autenticação via Firebase.
- **Componentes de UI**:
  - `etEmail`: Campo para e-mail.
  - `etSenha`: Campo para senha (máscara de texto).
  - `btnLogin`: Botão para iniciar login.
  - `tvCadastrar`: Link para cadastro.
- **Funcionamento**:
  - Valida se os campos estão preenchidos.
  - Chama `AuthRepository.login()` com callbacks:
    - `onSuccess`: Exibe `Toast` e navega para `MedicamentosActivity`.
    - `onFailure`: Exibe erro via `Toast`.
  - O clique em `tvCadastrar` inicia `CadastroUsuarioActivity`.
- **Código principal**:

```
btnLogin.setOnClickListener {
    val email = etEmail.text.toString().trim()
    val senha = etSenha.text.toString().trim()
    if (email.isEmpty() || senha.isEmpty()) {
        Toast.makeText(this, "Preencha todos os campos", Toast.
           LENGTH_SHORT).show()
    } else {
        authRepository.login(email, senha,
            onSuccess = {
                Toast.makeText(this, "Login realizado com sucesso
                   !", Toast.LENGTH_SHORT).show()
                startActivity(Intent(this, MedicamentosActivity::
                   class.java))
                finish()
            },
            onFailure = { e ->
                Toast.makeText(this, "Erro ao fazer login: ${e.
                   localizedMessage}", Toast.LENGTH_LONG).show()
            }
        )
    }
}
```

5.1.3 `CadastroUsuarioActivity`

- **Descrição**: Tela para cadastro de novos usuários.
- **Componentes de UI**:
  - `etNome`: Campo para nome (não salvo no Firebase).
  - `etEmail`: Campo para e-mail.
  - `etSenha`: Campo para senha.
  - `btnCadastrar`: Botão para cadastro.
  - `tvVoltarLogin`: Link para login.
- **Funcionamento**:
  - Valida campos de e-mail e senha.
  - Chama `AuthRepository.cadastrar()` com callbacks.
  - Navega para `LoginActivity` após sucesso ou clique em `tvVoltarLogin`.

5.1.4 `MedicamentosActivity`

- **Descrição**: Lista medicamentos e permite adicionar novos.
- **Componentes de UI**:
  - `listViewMedicamentos`: Exibe nomes de medicamentos.
  - `btnNovoMedicamento`: Botão para cadastro.
- **Funcionamento**:
  - Usa `ArrayAdapter` para popular `ListView`.
  - Escuta mudanças em tempo real com `MedicamentosRepository.listarMedicamentos()`.
  - Atualiza a lista dinamicamente.
- **Código principal**:

```
medicamentosRepo.listarMedicamentos(
    onMedicamentosChanged = { lista ->
        medicamentos.clear()
        medicamentos.addAll(lista)
        adapter.clear()
        adapter.addAll(lista.map { it.nome })
        adapter.notifyDataSetChanged()
    },
    onError = { e ->
        Toast.makeText(this, "Erro ao carregar medicamentos: ${e.
           message}", Toast.LENGTH_SHORT).show()
    }
)
```

5.1.5 `CadastroMedicamentoActivity`

- **Descrição**: Tela para cadastrar medicamentos.
- **Componentes de UI**:
  - `etNome`, `etDosagem`, `etFrequencia`, `etQuantidade`: Campos de entrada.
  - `btnSalvar`: Botão para salvar.

- **Funcionamento**:
  - Valida campos obrigatórios.
  - Cria objeto `Medicamento` e salva via `MedicamentosRepository`.

5.1.6 `DetalhesMedicamentoActivity`

- **Descrição**: Exibe detalhes do medicamento com suporte a imagens e lembretes.
- **Componentes de UI**:
  - `tvTituloMedicamento`, `tvQuantidade`: Exibem informações.
  - `imgIlustracao`: Imagem capturada.
  - `timePicker`, `switchLembrete`: Configuram lembretes.
- **Funcionamento**:
  - Recebe dados via `Intent`.
  - Usa `MediaStore.ACTION_IMAGE_CAPTURE` para capturar imagens.
  - Exibe `Toast` para lembretes.

5.1.7 `PerfilActivity`

- **Descrição**: Exibe informações do usuário e permite logout.
- **Funcionamento**: Usa dados mockados e chama `AuthRepository.logout()`.

5.1.8 `RepositorioDeMedicamentos`

- **Descrição**: Singleton para lista local (não utilizado).
- **Funcionamento**: Armazena nomes em `mutableListOf<String>()`.

5.2 Pacote `com.dosecerta.medicamentosapp.data`

5.2.1 `AuthRepository`

- **Funcionamento**: Gerencia autenticação com `FirebaseAuth`.
- **Métodos**: `cadastrar()`, `login()`, `logout()`.

5.2.2 `MedicamentosRepository`

- **Funcionamento**: Gerencia dados no Firestore, com salvamento e escuta em tempo real.
- **Estrutura**: `usuarios/{userId}/medicamentos`.

5.2.3 `StorageRepository`

- **Funcionamento**: Faz upload de imagens com nome único baseado em timestamp.

5.3 Pacote `com.dosecerta.medicamentosapp.model`

5.3.1 `Medicamento`

- **Funcionamento**: Modelo de dados para serialização no Firestore.

5.4 Pacote `com.dosecerta.medicamentosapp.ui.theme`

- **Funcionamento**: Define temas claro/escuro e tipografia com Jetpack Compose.

6 Fluxos de Funcionalidades

1. **Autenticação**:
   - Usuário acessa `MainActivity`, que redireciona para `LoginActivity` se não autenticado.
   - Usuário insere e-mail/senha e autentica via Firebase.
   - Após login, navega para `MedicamentosActivity`.
2. **Cadastro de Medicamentos**:
   - Usuário clica em "Cadastrar novo medicamento" em `MedicamentosActivity`.
   - Preenche campos em `CadastroMedicamentoActivity` e salva no Firestore.
3. **Listagem**: Dados do Firestore são exibidos em tempo real em `MedicamentosActivity`.
4. **Detalhes e Imagens**: Usuário visualiza detalhes e captura imagens em `DetalhesMedicamentoActivity`.

7 Dependências

- `com.google.firebase:firebase-auth:23.0.0`
- `com.google.firebase:firebase-firestore:25.0.0`
- `com.google.firebase:firebase-storage:21.0.0`
- `androidx.activity:activity-ktx:1.9.0`
- `androidx.compose.material3`

8 Melhorias Sugeridas

- **Imagens**: Integrar `StorageRepository` ao cadastro de medicamentos.
- **Lembretes**: Usar `AlarmManager` para notificações reais.
- **Validações**: Implementar regex para e-mail e limites para campos numéricos.
- **Navegação**: Adotar `Navigation Component`.
- **Perfil**: Persistir nome do usuário no Firebase.
- **UI**: Substituir `ListView` por `RecyclerView`.
- **Testes**: Adicionar testes com JUnit e Espresso.