ToDo List com Firebase

Autenticação segura, sincronização em tempo real e interface moderna com Jetpack Compose. Projeto Android desenvolvido com Clean Architecture, Kotlin Coroutines e Cloud Firestore.

Tecnologias & Funcionalidades

📱 Funcionalidades Principais

- **Autenticação segura**

  Email e senha com Firebase Authentication

- **Sincronização em tempo real**

  Dados atualizam instantaneamente entre dispositivos

- **CRUD completo**

  Criar, ler, atualizar e deletar tarefas

- **Interface moderna**

  Jetpack Compose com Material Design 3

🛠️ Stack Técnica

- **Kotlin 1.8+**

  Linguagem principal com Coroutines e Flow

- **Jetpack Compose**

  UI declarativa sem XML

- **Firebase Auth**

  Autenticação e gerenciamento de usuários

- **Cloud Firestore**

  BD NoSQL com Security Rules

- **MVVM Pattern**

  Arquitetura limpa e testável

Arquitetura Reativa com Flow

Implementação de `callbackFlow` para converter listeners do Firestore em Flows do Kotlin, garantindo gerenciamento automático de recursos e sincronização em tempo real.

```
fun getTasks(userId: String): Flow<List<Task>> = callbackFlow {
    Log.d("TaskRepository", "getTasks called")

    val listener = firestore
        .collection("tasks")
        .whereEqualTo("userId", userId)
        .addSnapshotListener { snapshot, error ->
            if (error != null) {
                close(error)
                return@addSnapshotListener
            }

            val tasks = snapshot?.documents?.mapNotNull {
                Task(
                    id = it.getString("id") ?: "",
                    title = it.getString("title") ?: "",
                    isCompleted = it.getBoolean("isCompleted") ?: false,
                    userId = it.getString("userId") ?: "",
                    timestamp = it.getLong("timestamp") ?: 0L
                )
            }?.sortedByDescending { it.timestamp } ?: emptyList()

            trySend(tasks)
        }

    awaitClose { listener.remove() }
}
```

Vantagens do Padrão

flatMapLatest

Cancela listeners antigos e ativa novos sem vazamento de memória

stateIn

Mantém estado ativo enquanto ViewModel existe

awaitClose

Limpeza automática de recursos ao sair da tela

Desafios Encontrados & Soluções

1

Sincronização & Ciclo de Vida

Problema: Tarefas desapareciam da tela após aparecer brevemente

Causa: ViewModel usava `collect()` manual que era cancelado quando a coroutine terminava

Solução: `flatMapLatest` + `stateIn` mantêm listener ativo enquanto ViewModel existe

2

Erro PERMISSION_DENIED

Problema: Após logout/login, recebia erro ~35 segundos depois

Causa: `getCurrentUser()` retornava apenas uma vez (usava `flowOf()` em vez de listener ativo)

Solução: `AuthStateListener` para reagir continuamente a mudanças de sessão do Firebase

3

Firestore Security Rules

Problema: Snapshot listeners falhavam com PERMISSION_DENIED mesmo com rules corretas

Causa: Validar `resource.data` dentro das rules causa falhas com snapshot listeners (edge cases)

Solução: Rules simples (apenas auth) + filtragem complexa feita na query Android com `whereEqualTo()`

Ferramentas de IA Utilizadas

- **Claude 3.5 Sonnet:** Estrutura reativa, debugging e padrões avançados
- Prompt:

> Como converter um Firebase Snapshot Listener em um Kotlin Flow?
> Preciso que getTasks() retorne um Flow<List<Task>> e que o listener
> seja automaticamente removido quando o Flow é cancelado usando awaitClose.

Funcionalidades Testadas

- Sign up com novo email/senha
- Login com credenciais corretas
- Criar, editar e deletar tarefas
- Marcar tarefa como concluída
- Sincronização em tempo real (2 dispositivos)