Documentação Técnica - ToDo List Firebase

Versão: 1.0 • Data: 09/02/2026
Autores: Luccas Asaphe, Matheus Nascimento
Plataforma: Android (API 24+) • Arquitetura: Clean Architecture + MVVM

1. Visão Geral

Este documento detalha as decisões arquiteturais, modelos de dados, fluxos técnicos e escolhas implementadas no projeto ToDo List com Firebase.

2. Modelo de Dados

2.1 Entidade: User

```
User (Domain Model)
├── id: String        (UID do Firebase Auth)
└── email: String     (Email da conta)
```

Armazenamento: Não persiste no Firestore

- Vem diretamente do Firebase Authentication
- Gerenciado por AuthStateListener
- Sincronizado automaticamente em tempo real

2.2 Entidade: Task

```
Task (Domain Model)
├── id: String              (UUID gerado pelo app)
├── title: String           (Título da tarefa)
├── description: String     (Descrição opcional)
├── isCompleted: Boolean    (Status concluído/não concluído)
├── userId: String          (UID do proprietário)
└── timestamp: Long         (Timestamp de criação em milissegundos)
```

Armazenamento: Firestore - coleção tasks

3. Arquitetura de Camadas

3.1 Presentation Layer (UI)

Responsabilidade: Exibir dados e capturar interação do usuário

- Activities:

1. MainActivity.kt: Activity principal que hospeda NavGraph

- Screens (Composables):

2. LoginScreen.kt: UI de autenticação
3. SignUpScreen.kt: UI de cadastro
4. TaskListScreen.kt: UI de lista de tarefas

3.2 Domain Layer (Lógica de Negócio)

Responsabilidade: Definir interfaces e modelos independentes de implementação

3.3 Data Layer (Persistência)

Responsabilidade: Implementar repositórios com Firebase

4. Padrões e Decisões Técnicas

4.1 callbackFlow para Listeners

Problema: Firebase listeners são callbacks, não Flows

Solução: Usar callbackFlow para converter

```
override fun getTasks(userId: String): Flow<List<Task>> = callbackFlow {
    val listener = firestore.collection("tasks")
        .whereEqualTo("userId", userId)
        .addSnapshotListener { snapshot, error ->
            if (error != null) {
                close(error)
            } else {
                val tasks = snapshot?.documents?.mapNotNull { ... } ?: emptyList()
                trySend(tasks)
            }
        }

    awaitClose { listener.remove() }  // Cleanup automático
}
```

Benefícios:

- Listeners são gerenciados automaticamente
- Cleanup (remove listener) quando Flow cancela
- Compatível com Coroutines

4.2 flatMapLatest para Reagir a Mudanças

Problema: Quando userId muda (logout/login), precisa manter listener ativo

Solução: Usar flatMapLatest no ViewModel

```
val taskState: StateFlow<TaskState> = _userId
    .flatMapLatest { userId ->
        if (userId != null) {
            taskRepository.getTasks(userId)
                .map { tasks -> TaskState.Success(tasks) as TaskState }
                .catch { error -> emit(TaskState.Error(...)) }
        } else {
```

```
            flowOf(TaskState.Loading as TaskState)
        }
    }
    .stateIn(
        scope = viewModelScope,
        started = SharingStarted.Lazily,
        initialValue = TaskState.Loading
    )
```

Benefícios:

- Cancela Flow anterior automaticamente quando userId muda
- Mantém listener ativo enquanto ViewModel existe
- Limpa recursos automaticamente

4.3 stateIn para Estado Cacheado

Problema: Listeners devem estar sempre ativos, não apenas quando UI observa

Solução: Converter Flow em StateFlow com stateIn

4.4 AuthStateListener para Sincronizar Sessão

Problema: Firebase não notifica mudanças de auth state automaticamente

Solução: Configurar listener no init do ViewModel

4.5 Ordenação em Memória (sem índices Firestore)

Problema: Firestore cobrava índices compostos para .orderBy()

Solução: Ordenar em memória no app

4.6 Filtragem na Query Android (não na Security Rule)

Problema: Snapshot listeners com resource.data em rules falhava

Solução: Filtrar na query com whereEqualTo

5. Configuração Firebase

5.1 Authentication

Habilitar: Email/Password sign in

Funcionalidades:

- Criar conta: createUserWithEmailAndPassword()
- Login: signInWithEmailAndPassword()
- Logout: signOut()
- Recuperar sessão: currentUser ou AuthStateListener

5.2 Security Rules

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /tasks/{document=**} {
      allow read: if request.auth != null;
      allow create: if request.auth != null &&
                       request.resource.data.userId == request.auth.uid;
      allow update: if request.auth != null &&
                       resource.data.userId == request.auth.uid;
      allow delete: if request.auth != null &&
                       resource.data.userId == request.auth.uid;
    }
  }
}
```

Explicação:

- read: Apenas usuários autenticados
- create: Apenas owner (userId == uid) pode criar
- update: Apenas owner pode editar
- delete: Apenas owner pode deletar

6. Tratamento de Erros

6.1 Erros de Autenticação

- FirebaseAuthWeakPasswordException: Senha < 6 caracteres → Validar antes de enviar
- FirebaseAuthInvalidCredentialsException: Email/senha incorretos → Mostrar mensagem
- FirebaseAuthUserCollisionException: Email já existe → Sugerir fazer login
- FirebaseNetworkException: Sem internet → Mostrar 'Sem conexão'

6.2 Erros de Firestore

- PERMISSION_DENIED: Sem autenticação ou rule nega
- DEADLINE_EXCEEDED: Timeout → Retry automático
- NOT_FOUND: Documento não existe → Ignorar
- INTERNAL: Erro do servidor → Retry automático

7. Performance

7.1 Otimizações

- callbackFlow: Listeners reutilizados
- stateIn: State cacheado
- flatMapLatest: Cancela listeners antigos automaticamente
- LazyColumn: Renderiza apenas items visíveis
- collectAsState(): Observação eficiente

7.2 Limitações

- Tarefas em listas grande (>100): Considerar paginação
- Queries complexas: Usar índices Firestore se necessário
- Imagens: Não suportadas (dados apenas texto)

8. Segurança

8.1 Autenticação

- Senhas são hasheadas pelo Firebase
- Session management automático
- Tokens renovados automaticamente
- Logout limpa currentUser e cancela listeners

8.2 Dados

- Firestore rules garantem acesso apenas de owner
- userId vinculado ao Firebase UID
- Sem dados sensíveis armazenados localmente

8.3 Comunicação

- HTTPS obrigatório do Firebase
- TLS 1.2+ para comunicação
- Sem secrets hardcoded no código

9. Testes

9.1 Emulador Android

- Sign up com email/senha válido
- Login com credenciais corretas/incorretas
- CRUD de tarefas (criar, editar, deletar, marcar concluído)
- Logout e login novamente (persistência)
- Sincronização em tempo real (2 emuladores)

9.2 Dispositivo Físico

- Mesmas verificações que emulador

9.3 Casos Edge

- Logout/login rápido
- Adicionar tarefa e logout antes de sincronizar
- Deletar tarefa não existente
- Sem conexão de internet
- Mudar usuário no meio de operação

10. Debugging

10.1 Logcat Filters

- TaskRepository: Firestore operations
- TaskViewModel: Task state management
- AuthViewModel: Auth state management
- AuthRepository: Login/signup operations

10.2 Firebase Console Monitoring

- Authentication: Ver usuários criados, logins
- Firestore: Ver documentos, tráfego em tempo real
- Performance: Latência de operations

11. Próximos Passos / Melhorias

- Paginação para listas grandes
- Imagens em tarefas (Cloud Storage)
- Categorias/tags de tarefas
- Dark mode
- Busca/filtro de tarefas

Versão do Firebase SDK: Última compatível com API 24