ToDo App Firebase

Documentação Técnica

Victor Hugo Monteiro Murilo de Melo

9 de fevereiro de 2026

1 Visão Geral

Aplicativo Android (Kotlin + Jetpack Compose) para gerenciamento de tarefas com autenticação de usuários. Utiliza Firebase Authentication para login/cadastro e Cloud Firestore para persistência em tempo real, com arquitetura MVVM e injeção de dependências via Hilt.

2 Arquitetura e Estrutura

2.1 Estrutura de Diretórios

```
app/src/main/java/com/example/todoapp_firebase/
|-- data/
|   |-- model/ (Task)
|   |-- repository/ (AuthRepository, TaskRepository)
|   |-- Response.kt (Sealed class para estados)
|-- di/ (AppModule - Hilt)
|-- ui/
|   |-- auth/ (LoginScreen, SignUpScreen, AuthViewModel)
|   |-- task/ (ListScreen, TaskViewModel, AddTaskDialog)
|   |-- components/ (CustomButton, CustomTextField, TaskItem)
|   |-- navigation/ (Routes, AppNavHost)
|   |-- theme/ (Theme, Color, Type)
|-- util/ (Constants)
|-- MainActivity.kt
|-- TodoApplication.kt
```

2.2 Padrão MVVM

Fluxo: View (Compose) ↔ ViewModel (StateFlow) ↔ Repository ↔ Firebase.

- **View**: Interação do usuário (Screens Composable).
- **ViewModel**: Gerenciamento de estado e lógica de negócio.
- **Repository**: Mediação de dados e abstração do Firebase.
- **Firebase**: Persistência cloud (Authentication + Firestore).

3 Modelo de Dados e Navegação

3.1 Entidade Principal (Firestore)

```
data class Task(
    @DocumentId val id: String = "",
    val title: String = "",
    val description: String = "",
    val isCompleted: Boolean = false,
    val userId: String = ""
)
```

3.2 Gerenciamento de Estado

```
sealed class Response<out T> {
    object Loading : Response<Nothing>()
    data class Success<out T>(val data: T) : Response<T>()
    data class Error(val message: String) : Response<Nothing>()
}
```

3.3 Navegação (NavHost)

```
NavHost(startDestination = "login") {
    composable("login") { LoginScreen(...) }
    composable("signup") { SignUpScreen(...) }
    composable("home") { ListScreen(...) }
}
```

4 Implementação da Persistência

4.1 Repository Pattern

AuthRepository: Gerencia autenticação via Firebase Auth.

```
interface AuthRepository {
    val currentUser: FirebaseUser?
    suspend fun login(email: String, password: String): Response<Boolean>
    suspend fun signUp(email: String, password: String): Response<Boolean>
    fun signOut()
}
```

TaskRepository: Gerencia CRUD de tarefas no Firestore.

```
interface TaskRepository {
    fun getTasks(): Flow<Response<List<Task>>>
    suspend fun addTask(title: String, description: String): Response<Boolean>
    suspend fun updateTask(task: Task): Response<Boolean>
    suspend fun deleteTask(taskId: String): Response<Boolean>
}
```

4.2 Sincronização em Tempo Real

Utiliza Kotlin Flow com `callbackFlow` para escutar mudanças no Firestore:

```
override fun getTasks(): Flow<Response<List<Task>>> = callbackFlow {
    val userId = auth.currentUser?.uid ?: ""
    val subscription = tasksCollection.whereEqualTo("userId", userId)
        .addSnapshotListener { snapshot, error ->
            if (snapshot != null) {
                val tasks = snapshot.documents.map { doc -> /*...*/ }
                trySend(Response.Success(tasks))
            }
        }
    awaitClose { subscription.remove() }
}
```

Vantagens: Atualizações automáticas, sincronização multi-dispositivo.

4.3 Correção Crítica

Problema: Checkbox não atualizava UI. **Causa:** `toObjects()` reutilizava referências. **Solução:** Mapeamento manual força novas instâncias a cada snapshot.

5 Injeção de Dependências (Hilt)

```
@Module @InstallIn(SingletonComponent::class)
object AppModule {
    @Provides @Singleton
    fun provideFirebaseAuth() = FirebaseAuth.getInstance()
    @Provides @Singleton
    fun provideFirebaseFirestore() = FirebaseFirestore.getInstance()
    @Provides @Singleton
    fun provideAuthRepository(impl: AuthRepositoryImpl): AuthRepository = impl
    @Provides @Singleton
    fun provideTaskRepository(impl: TaskRepositoryImpl): TaskRepository = impl
}
```

6 Design System e Interface

Cores: Purple80 (#D0BCFF), Purple40 (#6650a4), PurpleGrey40 (#625b71), Pink40 (#7D5260).

Tipografia: Sistema padrão Android (Roboto).

Componentes: CustomButton (loading), CustomTextField (senha), TaskItem (checkbox/delete), AddTaskDialog.

7 Validação e Dependências

Validação: Login/Cadastro (e-mail e senha obrigatórios, senhas correspondentes), Nova Tarefa (título obrigatório).

Dependências: Firebase BOM 32.7.0 (auth, firestore), Hilt 2.48, Compose UI 1.6.0, Navigation 2.7.7, Coroutines 1.7.3.

8 Desenvolvimento e Futuro

IAs usadas: Google Gemini (arquitetura MVVM, componentes UI, validações), Claude (documentação, debugging).

Melhorias: Edição de tarefas, datas, prioridades, tags; Cache Room (offline); Paginação; Testes; Animações; Widget; Segurança granular.