

AuthApp com Firebase Authentication e Social Login

QXD0102 - Desenvolvimento de Software para Dispositivos Móveis

Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br

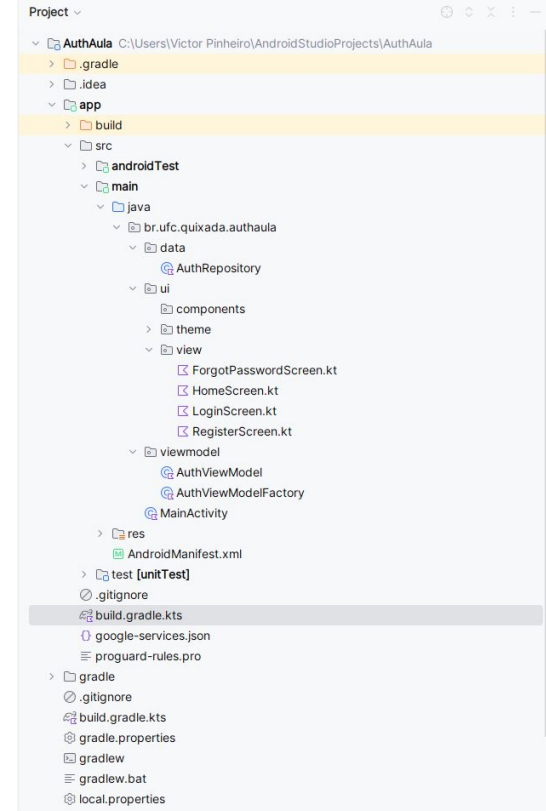


Agenda

- Estrutura do Projeto
- Funcionalidades do Projeto
- Tecnologias Utilizadas
- Fluxo de Navegação
- Criando o Projeto no Firebase
 - Adicionando SHA-1 no projeto
- Adicionar Dependências
- Criando o Repository (AuthRepository.kt)
- Criando o ViewModel (AuthViewModel.kt)
- Criando o ViewModelFactory (AuthViewModelFactory.kt)
- Testando o App

Estrutura do Projeto

- 1 Telas
 - **LoginScreen.kt**: Tela de login com Email/Senha e Login com Google.
 - **RegisterScreen.kt**: Tela de cadastro de novos usuários.
 - **ForgotPasswordScreen.kt**: Tela de recuperação de senha via email.
 - **HomeScreen.kt**: Tela de boas-vindas que exibe o nome do usuário e oferece a opção de logout.
- 2 Camada de Dados
 - **AuthRepository.kt**: Repositório que abstrai as interações com o Firebase Authentication e Firestore.
- 3 Camada de Lógica
 - **AuthViewModel.kt**: ViewModel para gerenciar os estados das telas e interagir com o repositório.



Funcionalidades do Projeto

- **Login com Email/Senha**
 - Usuários podem fazer login utilizando suas credenciais de email e senha cadastradas.
 - Tratamento de erros: Mensagem de "Usuário ou senha inválida" aparece se o login falhar.
- **Cadastro de Novo Usuário**
 - Os usuários podem criar uma conta informando:
 - Nome.
 - Email.
 - Senha.
 - O nome e email do usuário são salvos no Firestore e utilizados para exibição na tela inicial.

Funcionalidades do Projeto

- **Login com Google**
 - Integração com Firebase Authentication para login social com Google.
 - Após o login, os dados do usuário (nome, email) são salvos no Firestore, se ainda não existirem.
- **Recuperação de Senha**
 - Usuários podem solicitar um email para redefinir sua senha.
 - Tratamento de erros: Mensagens para casos de email inválido ou falhas no envio.

Funcionalidades do Projeto

- **Logout**
 - Botão na tela inicial que permite deslogar e voltar para a tela de login.
- **Exibição de Nome do Usuário**
 - O nome do usuário logado é exibido na HomeScreen, recuperado do Firestore.

Tecnologias Utilizadas

- **Firebase**
 - Firebase Authentication (Email/Senha, Google Login).
 - Firestore Database para armazenar dados dos usuários.
- **Jetpack Compose**
 - Criação de layouts modernos e responsivos.
 - Gerenciamento de estado com remember e LaunchedEffect.

Tecnologias Utilizadas

- **Arquitetura MVVM**
 - Separação de responsabilidades com Repository e ViewModel.
- **Kotlin Coroutines**
 - Chamadas assíncronas para interagir com o Firebase.

Fluxo de Navegação

- **LoginScreen:** Login ou navegação para registro/recuperação de senha.
- **RegisterScreen:** Cadastro de novo usuário e redirecionamento para login.
- **ForgotPasswordScreen:** Envio de email para redefinir senha.
- **HomeScreen:** Tela de boas-vindas com nome do usuário e opção de logout.

Criando o Projeto no Firebase

- Acesse o Firebase Console.
- Crie um novo projeto e adicione um App Android.
- Baixe o arquivo google-services.json e coloque na pasta app/.
- No Firebase Console, vá para Authentication > Sign-in method e:
 - Ative Email e Senha.
 - Ative Google Sign-In e configure um email de suporte.

Criando o Projeto no Firebase

AuthAula ▾

Authentication


Usuários Método de login Modelos Uso Configurações Extensions

Provedores de login


Adicionar novo provedor

Provedor	Status
E-mail/senha	ativado
Google	ativado

Avançado

 **Autenticação multifator por SMS**

Permita que usuários incluam uma camada extra de segurança nas contas deles. Assim que essa autenticação for ativada, integrada e configurada, os usuários poderão fazer login com um processo de duas etapas por SMS [Saiba mais](#)

 A MFA e outros recursos avançados estão disponíveis no Identity Platform, uma solução completa de identidade do cliente do Google Cloud, criada em parceria com o Firebase. Esse upgrade está disponível nos planos Spark e Blaze.

Fazer upgrade para ativar

Adicionando SHA-1 no projeto

- **Comando**
 - `./gradlew signingReport`
- **Aguarde a execução.** Se tudo estiver correto, você verá uma saída parecida com esta:
- Copie a SHA-1 e SHA-256 e adicione ao Firebase Console na seção de configurações do seu app Android.

```
Task :app:signingReport
Variant: debug
Config: debug
Store: /Users/seu_usuario/.android/debug.keystore
Alias: AndroidDebugKey
MD5:   A1:B2:C3:D4:E5:F6:G7:H8:I9:J0
SHA1:  11:22:33:44:55:66:77:88:99:AA:BB:CC:DD:EE:FF:00:11:22:33:44
SHA-256:  12:34:56:78:9A:BC:DE:F0:11:22:33:44:55:66:77:88:99:AA:BB:CC:DD:EE:FF:00:11:22:33:44:55:66
```

Adicionar Dependências

- build.gradle (Project: app)

```
plugins {
    alias(libs.plugins.android.application) apply false
    id("org.jetbrains.kotlin.android") version "1.9.10" apply false
    id("com.google.gms.google-services") version "4.3.15" apply false
    id("org.jetbrains.compose") version "1.5.3" apply false
}
```

Adicionar Dependências

- build.gradle (Module: app) - Parte 1

```
plugins {  
    alias(libs.plugins.android.application)  
    id("org.jetbrains.kotlin.android")  
    id("com.google.gms.google-services")  
    id("org.jetbrains.compose")  
}  
  
android {  
    namespace = "br.ufc.quixada.authaula"  
    compileSdk = 35  
  
    defaultConfig {  
        applicationId = "br.ufc.quixada.authaula"  
        minSdk = 26  
        targetSdk = 35  
        versionCode = 1  
        versionName = "1.0"  
  
        testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
        vectorDrawables {  
            useSupportLibrary = true  
        }  
    }  
}
```

Adicionar Dependências

- build.gradle (Module: app) - Parte 2

```
buildTypes {  
    release {  
        isMinifyEnabled = false  
        proguardFiles(  
            getDefaultProguardFile("proguard-android-optimize.txt"),  
            "proguard-rules.pro"  
        )  
    }  
}  
compileOptions {  
    sourceCompatibility = JavaVersion.VERSION_1_8  
    targetCompatibility = JavaVersion.VERSION_1_8  
}  
kotlinOptions {  
    jvmTarget = "1.8"  
}  
buildFeatures {  
    compose = true  
}  
composeOptions {  
    kotlinCompilerExtensionVersion = "1.5.3"  
}
```

Adicionar Dependências

- build.gradle (Module: app) - Parte 3

```
packaging {
    resources {
        excludes += "/META-INF/{AL2.0,LGPL2.1}"
    }
}

dependencies {
    // Firebase
    implementation(platform("com.google.firebase:firebase-bom:33.7.0"))
    implementation("com.google.firebase:firebase-analytics")
    implementation("com.google.firebase:firebase-auth-ktx") // 🔥 Adicionado para autenticação
    implementation("com.google.firebase:firebase-messaging-ktx:24.1.0")

    implementation("com.google.firebase:firebase-firestore-ktx")

    // Google Sign-In (Para Social Login com Google)
    implementation("com.google.android.gms:play-services-auth:20.7.0") // 🆕 Adicionado
```


Adicionar Dependências

- build.gradle (Module: app) - Parte 4

```
// Jetpack Compose
implementation(platform("androidx.compose:compose-bom:2023.09.01"))
implementation("androidx.compose.ui:ui")
implementation("androidx.compose.material3:material3")
implementation("androidx.compose.ui:ui-tooling-preview")
implementation("com.google.firebase:firebase-database-kt")
debugImplementation("androidx.compose.ui:ui-tooling")

// Navigation Compose
implementation("androidx.navigation:navigation-compose:2.7.2")

// Coroutines
implementation("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.7.3")

// Lifecycle
implementation("androidx.lifecycle:lifecycle-runtime-compose:2.6.2")
implementation("androidx.lifecycle:lifecycle-viewmodel-compose:2.6.2")

// AndroidX Core
implementation(libs.androidx.core.ktx)
implementation(libs.androidx.lifecycle.runtime.ktx)
implementation(libs.androidx.activity.compose)
```

Adicionar Dependências

- build.gradle (Module: app) - Parte 5

```
// Testes
testImplementation(libs.junit)
androidTestImplementation(libs.androidx.junit)
androidTestImplementation(libs.androidx.espresso.core)
androidTestImplementation(platform(libs.androidx.compose.bom))
androidTestImplementation(libs.androidx.ui.test.junit4)
debugImplementation(libs.androidx.ui.tooling)
debugImplementation(libs.androidx.ui.test.manifest)
}
```

Criando o Repository (AuthRepository.kt)

```
class AuthRepository {
    private val auth: FirebaseAuth = FirebaseAuth.getInstance()
    private val firestore: FirebaseFirestore = FirebaseFirestore.getInstance()

    // Registro de usuário
    suspend fun registerUser(email: String, password: String, name: String): Boolean {
        return try {
            val result = auth.createUserWithEmailAndPassword(email, password).await()
            val uid = result.user?.uid

            if (uid != null) {
                val user = hashMapOf(
                    "uid" to uid,
                    "name" to name,
                    "email" to email,
                    "created_at" to System.currentTimeMillis()
                )
                firestore.collection("users").document(uid).set(user).await()
            }
            true
        } catch (e: Exception) {
            Log.e("AuthRepository", "Erro no cadastro: ${e.message}")
            false
        }
    }
}
```

Criando o Repository (AuthRepository.kt)

```
// Login com email e senha
suspend fun loginUser(email: String, password: String): Boolean {
    return try {
        auth.signInWithEmailAndPassword(email, password).await()
        true
    } catch (e: Exception) {
        Log.e("AuthRepository", "Erro no login: ${e.message}")
        false
    }
}

suspend fun resetPassword(email: String): Boolean {
    return try {
        auth.sendPasswordResetEmail(email).await()
        true
    } catch (e: Exception) {
        Log.e("AuthRepository", "Erro ao enviar email de recuperação: ${e.message}")
        false
    }
}
```

Criando o Repository (AuthRepository.kt)

```
suspend fun getUsername(): String? {
    return try {
        val uid = auth.currentUser?.uid
        if (uid != null) {
            val snapshot = firestore.collection("users").document(uid).get().await()
            snapshot.getString("name") // Retorna o nome salvo no Firestore
        } else {
            null
        }
    } catch (e: Exception) {
        Log.e("AuthRepository", "Erro ao buscar nome do usuário: ${e.message}")
        null
    }
}

// Login com Google
fun getGoogleSignInClient(context: Context): GoogleSignInClient {
    val gso = GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestIdToken(context.getString(br.ufc.quixada.authaula.R.string.default_web_client_id))
        .requestEmail()
        .build()
    return GoogleSignIn.getClient(context, gso)
}
```

Criando o Repository (AuthRepository.kt)

```
suspend fun loginWithGoogle(idToken: String): Boolean {
    return try {
        val credential = GoogleAuthProvider.getCredential(idToken, null)
        val result = auth.signInWithCredential(credential).await()
        val user = result.user

        user?.let {
            val uid = it.uid
            val name = it.displayName ?: "Usuário"
            val email = it.email ?: ""

            // Verifica se o usuário já existe no Firestore antes de salvar
            val userRef = firestore.collection("users").document(uid)
            val snapshot = userRef.get().await()

            if (!snapshot.exists()) {
                val userData = hashMapOf(
                    "uid" to uid,
                    "name" to name,
                    "email" to email,
                    "created_at" to System.currentTimeMillis()
                )
                userRef.set(userData).await()
            }
        }
        true
    } catch (e: Exception) {
        Log.e("AuthRepository", "Erro no login Google: ${e.message}")
        false
    }
}
```

Criando o Repository (AuthRepository.kt)

```
// Logout
fun logout () {
    auth.signOut()
}

// Verifica se o usuário está logado
fun isUserLogged (): Boolean {
    return auth.currentUser != null
}
}
```

AuthViewModel.kt

```
class AuthViewModel (private val repository: AuthRepository) : ViewModel() {
    var loginResult: ((Boolean) -> Unit)? = null
    var registerResult: ((Boolean) -> Unit)? = null

    fun login(email: String, password: String, onResult: (Boolean) -> Unit) {
        viewModelScope.launch {
            val success = repository.loginUser(email, password)
            onResult(success) // Retorna true ou false para a tela de login
        }
    }

    fun resetPassword(email: String, onResult: (Boolean) -> Unit) {
        viewModelScope.launch {
            val success = repository.resetPassword(email)
            onResult(success)
        }
    }

    fun getUserName (onResult: (String?) -> Unit) {
        viewModelScope.launch {
            val name = repository.getUserName()
            onResult(name)
        }
    }
}
```


AuthViewModel.kt

```
fun loginWithGoogle(idToken: String, onResult: (Boolean) -> Unit) {
    viewModelScope.launch {
        val success = repository.loginWithGoogle(idToken)
        onResult(success)
    }
}

fun getGoogleSignInClient(context: Context): GoogleSignInClient {
    return repository.getGoogleSignInClient(context)
}

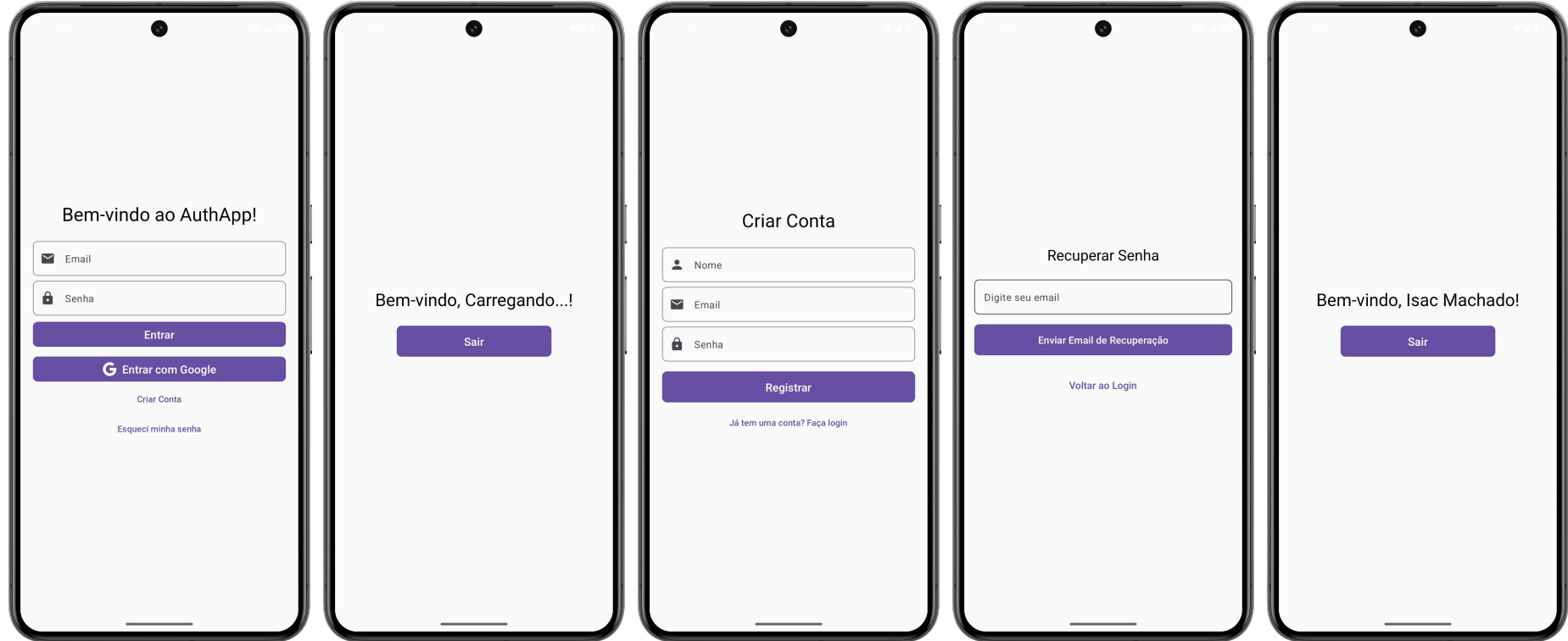
fun logout() {
    repository.logout()
}

fun register(email: String, password: String, name: String, onResult: (Boolean) -> Unit) {
    viewModelScope.launch {
        val success = repository.registerUser(email, password, name)
        onResult(success)
    }
}
}
```

AuthViewModelFactory.kt

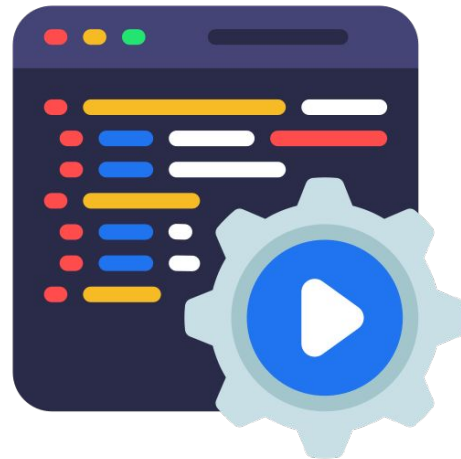
```
class AuthViewModelFactory(private val repository: AuthRepository) :
    ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        if (modelClass.isAssignableFrom(AuthViewModel::class.java)) {
            @Suppress("UNCHECKED_CAST")
            return AuthViewModel(repository) as T
        }
        throw IllegalArgumentException("Unknown ViewModel class")
    }
}
```

Testando o App



Referências

- https://firebase.google.com/docs/android/setup?hl=pt&authuser=0&_gl=1*6r8x2r*_ga*NDg2OTkxMjQxLjE3MzMwODA3OTM.*_ga_CW55HF8NVT*MTczODA2NzUzNC41LjE1MTczODA2Nzg1MC42MC4wLjA.
- <https://developer.android.com/reference/kotlin/android/app/Service>
- <https://developer.android.com/develop/background-work/services?hl=pt-br>



Obrigado!

Dúvidas?



Universidade Federal do Ceará - *Campus* Quixadá

Prof. Francisco Victor da Silva Pinheiro
victorpinheiro@ufc.br

