Sumário

[**1.CURSO DE FIREBASE (CANAL STACK MOBILE)** 2](#_heading=h.gjdgxs)

[**1.1.VÍDEO: Curso de Firebase – Introdução | Recursos| Preços| Criando uma conta do zero #1** 2](#_heading=h.30j0zll)

[**1.2.VÍDEO: Curso de Firebase – Configurando Firebase no Android Studio #2** 3](#_heading=h.1fob9te)

[**1.3.VÍDEO : Configurando Firebase no Android Studio – Nova Forma (2022)** 8](#_heading=h.3znysh7)

[**1.4.VÍDEO: Curso de Firebase – cadastro de usuários com e-mail e senha #3** 12](#_heading=h.2et92p0)

[**1.5.VÍDEO: Curso de Firebase – Tratando exceptions no Firebase #4** 20](#_heading=h.tyjcwt)

[**1.6.VÍDEO: Curso de Firebase – Autenticando usuários com e-mail e senha #5** 20](#_heading=h.3dy6vkm)

[**1.7.VIDEO: Curso de Firebase – Verificando usuário logado e deslogando usuário do sistema #6** 27](#_heading=h.1t3h5sf)

[**1.8.VÍDEO: Curso de Firebase – Iniciando com o Cloud Firestore – Guia absolutamente completo #7** 29](#_heading=h.4d34og8)

[**1.9.VÍDEO: Curso de Firebase – Como gravar dados no banco de dados #8** 33](#_heading=h.2s8eyo1)

[**1.10 VÍDEO: Curso de Firebase – Como fazer a leitura dos dados no banco de dados #9** 36](#_heading=h.17dp8vu)

[**1.11.VÍDEO: Curso Firebase – Como atualizar os dados no banco de dados #10** 41](#_heading=h.3rdcrjn)

[**1.12.VÍDEO: Curso Firebase – Como deletar os dados no banco de dados #11** 46](#_heading=h.26in1rg)

[**2.CURSO DE FIREBASE , CANAL ÁREA TECH BRASIL** 52](#_heading=h.lnxbz9)

[**2.1.VÍDEO: CURSO COMPLETO DE ANDROID + FIREBASE** 53](#_heading=h.35nkun2)

[**2.1.1. Criação do projeto no Android Studio** 53](#_heading=h.1ksv4uv)

[**2.1.2. Criação do design da tela de login** 54](#_heading=h.44sinio)

[**2.1.3.Integração do projeto no Android Studio com o Firebase Authentication** 59](#_heading=h.2jxsxqh)

[**2.1.4. Serviços que a plataforma Firebase oferece** 61](#_heading=h.z337ya)

[**2.1.5. Mudando o Android Manifest e Main Activity** 62](#_heading=h.3j2qqm3)

[**2.1.6. Colocando ação no botão de Login** 64](#_heading=h.1y810tw)

[**2.1.7. Colocando ação no botão de logout e ativando o checkbox** 67](#_heading=h.4i7ojhp)

[**2.1.8.Criando a tela de registro e botão registrar ativado** 72](#_heading=h.2xcytpi)

[**2.1.9 Criação de perfil** 79](#_heading=h.1ci93xb)

[**2.1.10 Tratativa de erros de cadastro** 90](#_heading=h.3whwml4)

[**3. CURSO DE FIREBASE (CANAL EXTRA MATCH 007)** 94](#_heading=h.2bn6wsx)

[**3.1. PLAYLIST FIREBASE** 94](#_heading=h.qsh70q)

[**3.1.1. VIDEO: Aula 56 – CRUD: Android e Firebase – Insert** 94](#_heading=h.3as4poj)

[**3.1.2.VÍDEO: Aula 57 – CRUD: Android e Firebase – Select** 101](#_heading=h.1pxezwc)

[**3.1.3.VÍDEO: Aula 58 – CRUD: Android e Firebase – Update e Delete** 104](#_heading=h.49x2ik5)

[**4. CURSO DE FIREBASE (CANAL JONE ARCE FRANCO)** 107](#_heading=h.2p2csry)

[**4.1.1. VÍDEO: Firebase Firestore – Regras de Segurança – Parte 1 – Criando regras de segurança** 107](#_heading=h.147n2zr)

[**4.1.2. VÍDEO: Firebase Firestore Regras de Segurança – Parte 2 – Regras para cada coleção** 108](#_heading=h.3o7alnk)

[**4.1.3. VÍDEO: Firebase Firestore Regras de Segurança – Parte 3 – Regras para cada usuário** 110](#_heading=h.23ckvvd)

**1.CURSO DE FIREBASE (CANAL STACK MOBILE)**

**FONTE:** [**https://www.youtube.com/watch?v=fSfiCIwpx6E&list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**](https://www.youtube.com/watch?v=fSfiCIwpx6E&list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9)**, CANAL: STACK MOBILE.**

**DATA DE ACESSO: 25/07/22**

**1.1.VÍDEO: Curso de Firebase – Introdução | Recursos| Preços| Criando uma conta do zero #1**

**Tempo: 15min47s, link: https://youtu.be/fSfiCIwpx6E?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**

**Usaremos a linguagem Kotlin, em Android Studio e o Firebase.**

Vamos criar uma conta na plataforma. No Chrome, digite Firebase e acesse o site: <https://firebase.google.com> O Firebase é um serviço de BackEnd que pertence a Google; vai nos permitir que trabalhe no backend das aplicações, como trabalhar com banco de dados, com sistema de autenticação, armazenamento de arquivos.

Senão usasse o Firebase, teria que criar um backend do zero para nossa aplicação, seja web ou Mongo, e teríamos que pensar em toda a parte de segurança dos dados do usuário, autenticação e lado Servidor. O Firebase já faz a parte estrutural. Temos que integrar o Firebase dentro da aplicação.

O Firebase , tem os tipos de integrações; Google Ads, AdMob, Google Marketing Platform, Play Store, Data Studio, BigQuery, Stack, Jira e PagerDuty. Exemplos de clientes: DuoLingo, Trivafo, GameLoft e afins.

Os principais serviços são: Realtime Database (banco de dados em tempo real), Remote Config, Firebase ML, Cloud Functions, Authentication (sistema de login e cadastro), Cloud Messaging, Hosting, Cloud Storage (armazenamento de arquivos: imagens, json, arquivos de vídeo), Cloud Firestore (banco de dados baseados em coleção de dados, é o mais usado).

O Firebase pode usar em qualquer projeto Android ou Web, é gratuito para pequenas aplicações, é o Plano Spark. Pra qualquer serviço Firebase geralmente precisa pagar, mesmo assim é mais barato do que os demais.

Para criar uma conta, clica em fazer login, precisa da conta do Google. Depois de logado, clique em “Ir para o Console”. No console, vai aparecer os projetos. Pode criar os projetos para integração.

**1.2.VÍDEO: Curso de Firebase – Configurando Firebase no Android Studio #2**

**Tempo: 13min22s, link: https://youtu.be/hm6TjO1C1Pg?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**

Nessa aula vamos configurar o firebase no Android Studio. Abra o Android Studio, a versão do vídeo para o software (Android) é Android Studio Artic Fox 2020.3.1 Patch 4. Se for uma versão superior, não tem problema. Precisa se preocupar com os códigos, que serão os mesmos. Geralmente o Google muda o IDE. Clica em New Project, Empty Activity, next, Nome do app: AplicativoDeTeste, packagename: com.stackmobile.aplicativodeteste, save location: C:\Users\marco\AndroidStudioProjects\AplicativodeTeste2, language: Kotlin, Minimum SDK: API 21 Android 5.0 (Lollipop)

Com o projeto criado, volta no Firebase, vá no Console. Vá em Adicionar o Projeto. Coloca o mesmo nome de projeto que estiver no Android, como: Aplicativo de Teste. Depois, clica em Continuar, ativar o Google Analitics. Se ativar o Google Analitics vai pedir pra criar conta. Clica em criar projeto, continuar. Cai no painel principal do Firebase, e abaixo da criação vem os nomes dos serviços.

Precisamos configurar o Firebase para o projeto: iOS, Android, Web e Games (Unity). Onde tem escrito: “Comece adicionando o Firebase ao seu aplicativo”, embaixo dele tem alguns ícones, clique no que tiver o Android.

Na tela: Adicionar o Firebase ao seu App Android, temos algumas informações a serem passadas:

**1. Registrar app:**

**Nome do pacote do Android (package name).** Acessa o Android Studio; para achar o nome do pacote, vá em Gradle Scripts, build gradle (Module: Aplicativo\_de\_Teste.app), aplication id é o nome do pacote. Copia e cola.

**Apelido do app:** é o nome do aplicativo. Vai ser: aplicativo de teste.

**Certificado de assinatura: (opcional).** Usado quando vai usar o login do Google ou Facebook, precisa do SHA-1 que é o certificado de assinatura. Não vamos usar agora.

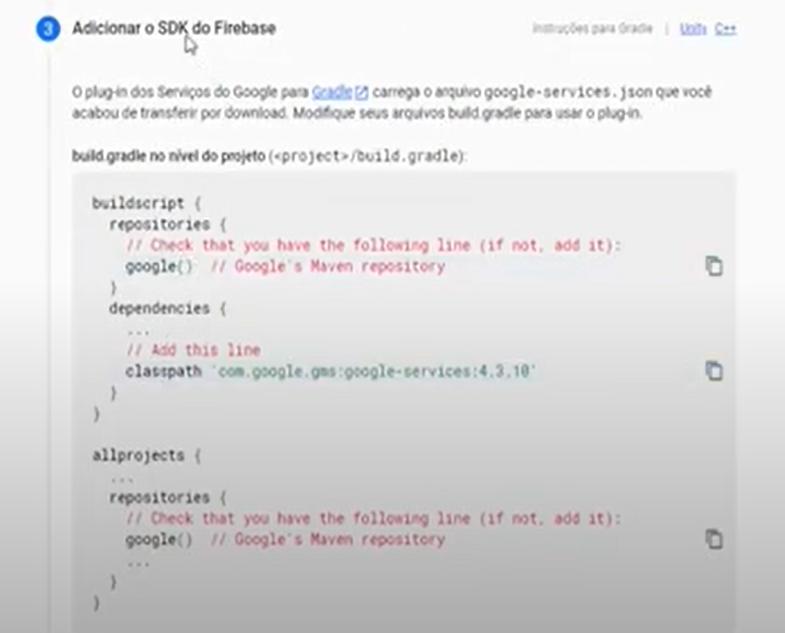
Clica em Registrar app

**2. Fazer o download do arquivo de configuração**

Precisaremos baixar o arquivo de configuração, que é o Google Service Json (GSON). É um arquivo JSON que vem várias configurações do nosso projeto. Clica em “Fazer download de google-services.json”. **Precisa colocar o GSON dentro do nosso projeto no Android Studio. Vai ficar dentro da pasta app, src, google-services.json. Coloque no local indicado, ele cola direto dentro do projeto no Android Studio. Clica em ok.**

Dentro do Firebase, clique em Próximo

**3.Adicione o SDK do Firebase**

**build gradle no nível do projeto (<project>/build.gradle)**

buildscript{

repositories{

// Check that you have the following line (if not add it)

google() // google’s haven repository

}

dependencies{

.....

// Add this line

classpath: ‘com.google.gms:google-services:4.3.10’

}

}

allprojects{

...

repositories{

// Check that you have the following line (if not add it)

google() // google’s haven repository

}

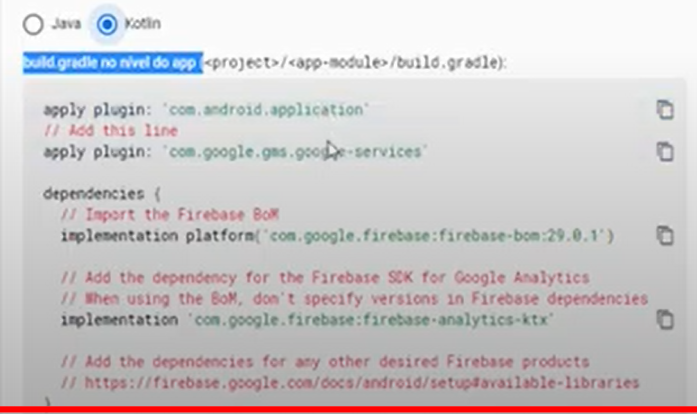
}

**Então, para colocar o SDK do Firebase dentro do Android Studio:**

Vá em Gradle Scripts, build.gradle (Project: Aplicativo\_de\_Teste)

Colar dentro de dependencies, a classpath do google

**A configuração a nível de app:**



**build gradle a nível de app (<project>/<app-module></build.gradle>**

apply plugin: ‘com.android.application’

// Add this line

apply plugin: ‘com.google.gms:google-services’

dependencies{

// Import the firebase sdk

Implementation platform: (‘com.google.firebase:firebase-bom:29.0.1’)

// Add this dependency for the firebase sdk for Google Analytics

// When using the BoM don’t specify versions in Firebase dependencies

Implementation ‘com.google.firebase:firebase-analytics-ktx’

// Add the dependencies for any other desired firebase products

// https://firebase.google.com/docs/android/setup4avaliable-libraries

}

**No Android Studio, vá em:**

Android Studio, Gradle Scripts, build.gradle(Module:Aplicativo\_de\_Teste.app)

Copia o android.application e o google.services, cola em cima de dependencies (no Android Studio).

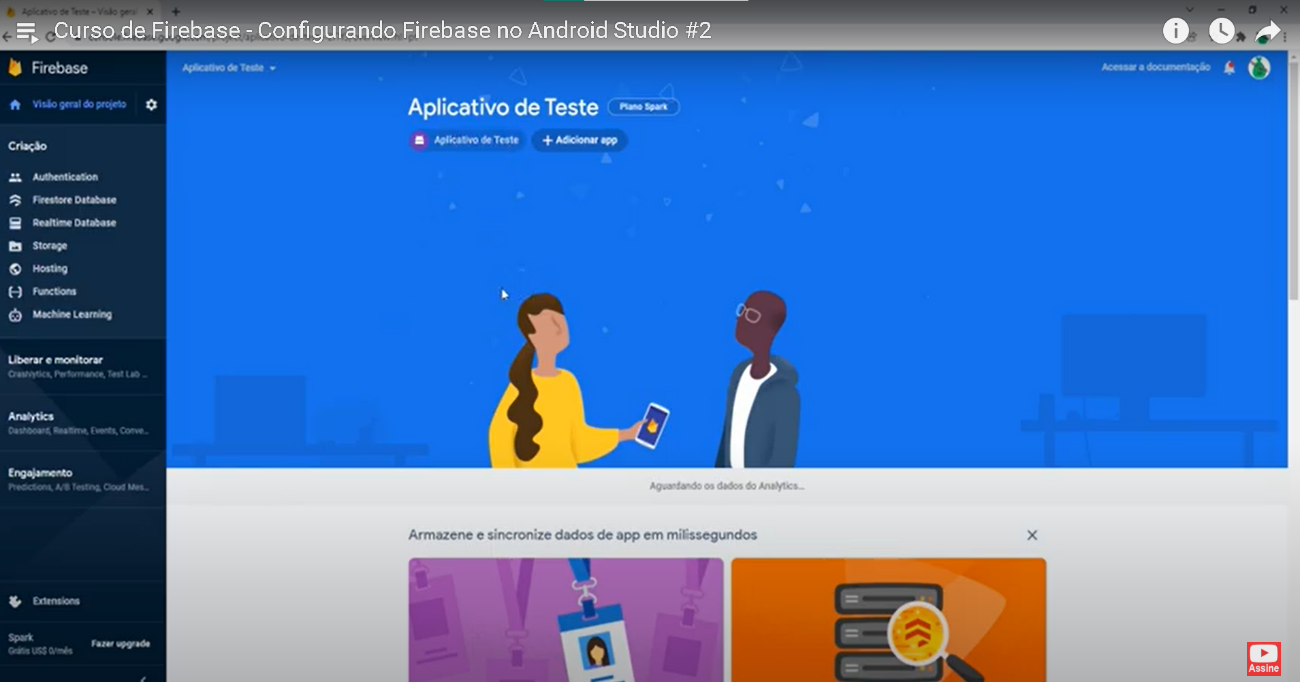
Dentro de dependencies, adicionar o firebase-bom e firebase-analytics. Cola as duas dependências depois da última androidTesteImplementation.

**Agora precisamos sincronizar o projeto, clique em Sync Now no Android Studio.**

No site do Firebase, clique em Próximo

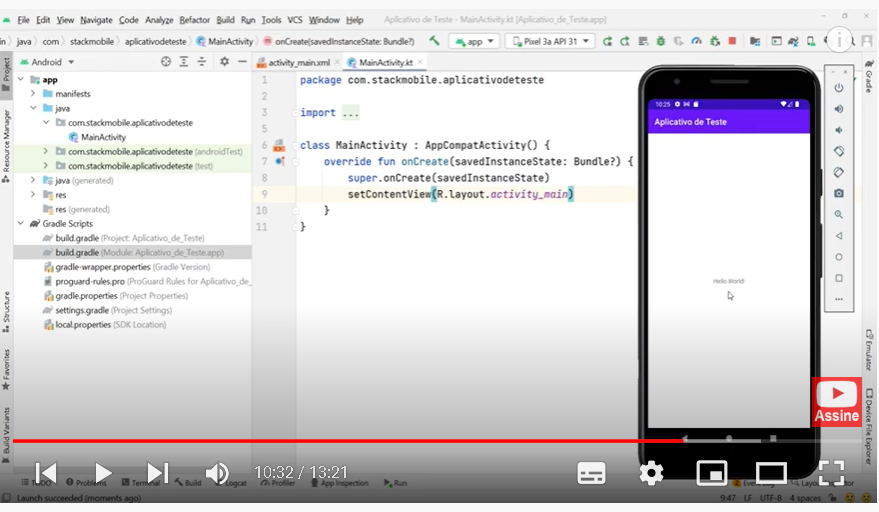
**4. Próximas etapas.**

Alega que está tudo pronto e te deixa de volta no Console. No Console do Firebase, aparece o nome: Aplicativo de Teste conforme figura adiante:

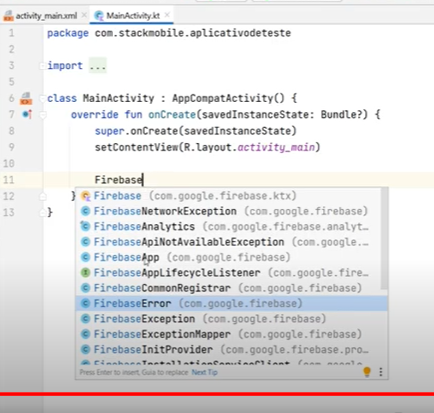


Vai no Android Studio, feche as abas do Build.Gradle e clique para executar o projeto no emulador (ícone de play). Senão tiver emulador, vá em AVD Manager pra criar um. Usa o Pixel 3a API 31. O emulador precisa do Google Play instalado.

O Projeto acabou de ser executado (nome do app na área roxa e hello world no meio da tela):

Executar o emulador e não quebrar significa que funciona. Outro teste é ir no MainActivity e digitar Firebase com classes nativas, significa que ele foi bem implementado. Logo, em ambos os casos foi bem configurado no projeto.

Outro teste é reiniciar o Android Studio, pois as dependências podem não carregar e as classes não carregarem.

****

**1.3.VÍDEO : Configurando Firebase no Android Studio – Nova Forma (2022)**

**Tempo: 15min11s, link:** [**https://www.youtube.com/watch?v=NDlNM8jPO4w&list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9&index=3**](https://www.youtube.com/watch?v=NDlNM8jPO4w&list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9&index=3)

O Android Studio está na versão: BumbleBee 2021.1.1 Patch 2. Tivemos uma grande atualização do Android Studio, e mudou a forma pra configurar o Firebase no Android; pois mudou o build.gradle.

Abre o Android Studio, New Project, Empty Activity, next. Name: teste, package name: com.marcos.teste, save location: C:\Users\marco\AndroidStudioProjects\Teste, language: Java, Minimum SDK: API 21: Android 5.0 (Lollipop)

Aguarda o projeto ser iniciado. Vai configurar o Firebase. Acessa o Painel do Firebase através de um conta do Gmail. Clica em Adicionar Projeto. Nome do projeto: Teste. Clica em Continuar. Deixa o Analytics desativado. Clica em Criar Projeto. Após carregado, clica em continuar. No painel do aplicativo, clica no ícone do Android.

**1.Registrar app**

Nome do pacote: (vai em Android Studio, Gradle Scripts, build.gradle (module: teste.app), e copia o applicationID de defaultConfig.

Apelido: Teste

Clica no botão “Registrar app”

**2. Fazer o download do Google Json.**

Clica no botão “Fazer o download de google-services.json”.

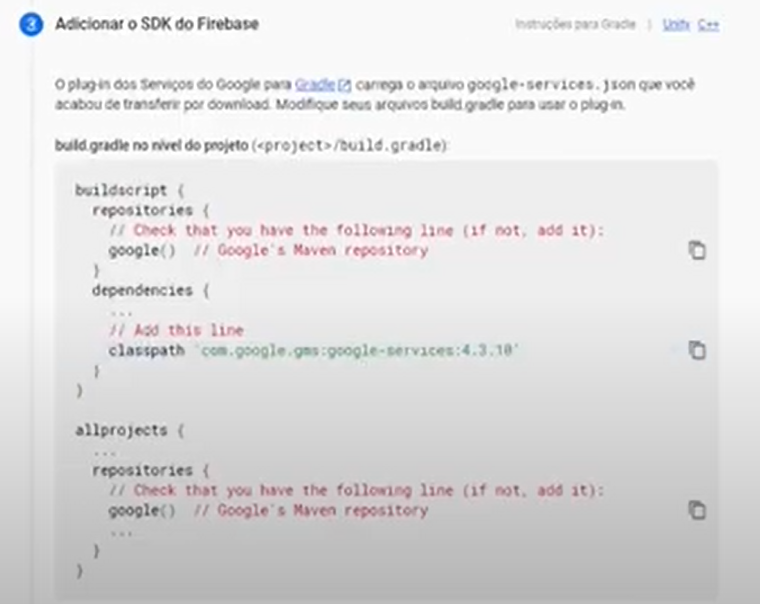
Dentro do Android Studio, clique na Visualização, deixe no Modo Projeto.

Copie o arquivo e vá na pasta app, depois na pasta src e cole dentro da pasta src. Dê o ok.

Dentro do Firebase, clique no botão Próxima.

**3. Adicionar o SDK de Firebase**

**É NESSE ITEM QUE TEM A MUDANÇA. O build gradle a nível de projeto está totalmente repaginado.**



**BUILD.GRADLE (Project)**

**Dentro do Android Studio, o Build Gradle a nível de projeto está modificado.**

**Não tem mais as dependências, apenas os plugins.**

Copia a classpath:

dependencies{

classpath: ‘com.google.gms:google-services:4.3.10’

}

**No Android Studio, acessa o Gradle Scripts, build.gradle a nível de app.**

Em plugins, terá:

plugins{

id ‘com.android.application’

// Cola e faz essas alterações

id ‘com.google.gms.google-services’

}

**A versão do Google Services defino no Build. Gradle a nível de PROJECT:**

plugins{

id ‘com.android.application’ version ‘7.1.2’ apply false

id ‘com.android.library’ version ‘7.1.2.’ apply false

// Cola o google services com alterações, version é um método

id ‘com.google.gms.google-services’ version ‘4.3.10’ apply false

}

**BUILD GRADLE A NÍVEL DE APP (pelo Firebase)**

No build a nível de app NÃO precisa do apply plugin google-services porque já foi definido no build.gralde em app como um id. 

**Copia o firebase bom , coloca dentro das dependências do build.gradle a nível de app (**caminho: Android Studio, Gradle Scripts, build.gradle(Module:Teste.app)

dependencies{

androidTestImplementation ‘androidx.test.espresso:espresso-core:3.4.0’

// Adiciona na linha debaixo integralmente

Implementation platform (‘com.google.firebase-bom:29.1.0 ’)

}

Dá um sync now no Android Studio.

Clique no botão próximo do Firebase

4. Vai parabenizar e voltar pro Console do Firebase.

Vá no Android Studio, deixe o Emulador como Pixel 4 API 31, rode o emulador. Depois escreva Firebase na Main Activity e espere as classes carregarem. Ambos os testes deram ok. Quando escreve Firebase, clica na lâmpada para importar a biblioteca, depois vai aparecer as classes correspondentes.

**Para ver dependências: no firebase, clique em Acessar Documentação, Primeiros Passos para Android, depois o artigo: Adicionar o Firebase ao projeto Android vai ter uma tabela das dependências para acionar os serviços do Firebase.**

Copia a dependência, vai no build.gradle a nível de app, dentro de dependencies, coloque: implementation ‘com.google.firebase:firebase-firestore’. Depois dá um build. Depois vá em Main Activity e escreva Firebase, veja se tem a classe da FireStore.

**1.4.VÍDEO: Curso de Firebase – cadastro de usuários com e-mail e senha #3**

Tempo: 38min52s, link: https://youtu.be/abZdiJ-H1eI?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9

Hoje vamos começar a implementar o primeiro serviço que é o Authentication: fazer login, identificar usuário logado. Para criar um sistema de login, precisa criar o layout de cadastro e login.

Primeiro precisamos cadastrar. **Usar o projeto AplicativoDeTeste2**

**CADASTRANDO USUÁRIOS DO FIREBASE**

No Android Studio, vai na pasta app, depois na pasta java, depois na pasta com.stackmobile.aplicativodeteste. Clica em cima da pasta do packagename, botão direito, new, package. O nome será view. Na pasta view ficará todas as atividades do nosso projeto (todas as atividades). Clica na pasta view, botão direito, new, activity, empty activity. ActivityName: FormCadastro, layout name: activity\_form\_cadastro, package name: com.stackmobile,.aplicativodeteste.view, lnaguage: Kotlin, Finish.

Depois de criada a Atividade, aparecerá aberto o arquivo: FormCadastro.kt

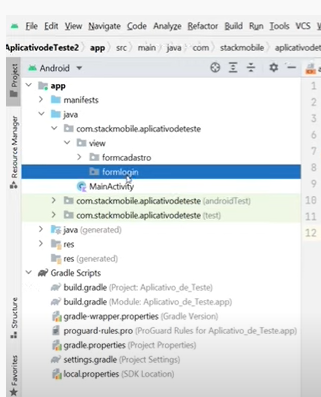
Importar o R.

**Dentro da pasta view:**  clica com o botão direito, new, package, package: formcadastro

**Dentro da pasta view:** clica com o botão direito, new, package, package: formlogin

**Clica em cima do arquivo FormCadastro**  e arraste pra dentro da pasta formcadastro. Pede pra refatorar, ok. Atualiza o Android Studio com Build, Rebuild Project.

Estrutura de pastas:

Vamos excluir a MainActivity. Delete.

Depois vai na pasta app, pasta res, pasta layout, exclui o activity\_main.xml

Vai na pasta app, pasta manifests, AndroidManifest.xml. Dentro de

<activity>

Mude o android:name (segunda activity, onde estava android:name = MainActivity)

android:name=”view.formcadastro.FormCadastro”

(O padrão é lançar a Main Activity). Queremos iniciar pela FormCadastro.

Na primeira activity, exclui.

Na **activity\_form\_cadastro.xml** , deixa em split. Vai ter 2 campos de texto (email e senha) e um botão para cadastrar o usuário. **Todos os textos ficarão armazenados no arquivo: strings.xml (caminho: app,res,values, strings.xml)**

**ARQUIVO: ACTIVITY\_FORM\_CADASTRO.XML**

<? xml version=’1.0’ encoding=”utf-8”?>

**// Colocando o ScrollView se o usuário estiver usando o dispositivo deitado**

<androidx.core.widget.NestedScrollView

android:layout\_width=”match\_parent” // altura máxima do container

android:layout\_height=”match\_parent” // largura máxima do container

android:fillViewport = “true”

xmlns:android=”http://schemas.android.com/apk/res/android”>

<LinearLayout

xmlns:tools=” <http://schemas.android.com/tools>”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

android:orientation=”vertical”

android:background=”@color/light\_gray”

tools:contexto=”.view.formcadastro.FormCadastro”>

<TextView

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:text=”@string/titulo\_form\_cadastro”

android:textSize=”25sp”

androidtextColor=”@color/black”

android:textStyle=”bold”

android:layout\_gravity=”center”

android:layout\_marginTop=”100dp” // afasta externamente da margem

/>

// Componentes de texto: Email

<EditText

android:id=”@+id/edit\_email”

android:layout\_width=”match\_content”

android:layout\_height=”wrap\_content”

android:hint = “@string/hint\_email” // é o placeholder

android:background=”@color/white”

android:padding=”15dp” // espaçamento interno de todos os lados de 15dp

android:maxLines=”1”// para evitar de descer a linha e quebrar a aplicação

android:inputType=”textEmailAddress” //define o tipo de teclado

android\_layout\_marginStart=”20dp” // é a margem inicial da esquerda

android\_layout\_marginEnd=”20dp” // é a margem final da direita

android:layout\_marginTop=”20dp” // afasta da margem de cadastre-se

/>

// Componentes de texto: Senha

<EditText

android:id=”@+id/edit\_senha”

android:layout\_width=”match\_content”

android:layout\_height=”wrap\_content”

android:hint = “@string/hint\_senha” // é o placeholder

android:background=”@color/white”

android:padding=”15dp” // espaçamento interno de todos os lados de 15dp

android:maxLines=”1”// para evitar de descer a linha e quebrar a aplicação

android:inputType=”numberPassword” //não vai poder ver a senha enquanto o usuário digita

android\_layout\_marginStart=”20dp” // é a margem inicial da esquerda

android\_layout\_marginEnd=”20dp” // é a margem final da direita

android:layout\_marginTop=”10dp” // afasta da margem de cadastre-se

/>

<Button

android:id=”@+id/bt\_cadastrar”

android:layout\_width=”match\_content”

android:layout\_height=”wrap\_content”

android:text=”@string/texto\_botao\_cadastrar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textSize=”bold”

android:textAllCaps =“false” // somente a primeira letra é maiúscula

android:padding=”15dp”

android:backgroundTint = “@color/black” // pinta a cor interna do botão

android:layout\_margin=”20dp” // margem geral tem espaçamento 20 pra todo mundo

/>

</LinearLayout>

</androidx.core.widget.NestedScrollView>

**NO ARQUIVO: STRINGS.XML**

<resources>

<string name=”app\_name”> Aplicativo de Teste </string>

<string name=”titulo\_form\_cadastro”> Cadastre-se </string>

<string name=”hint\_email”>Email </string>

<string name=”hint\_senha”>Senha </string>

<string name=”texto\_botao\_cadastrar”>Cadastrar </string>

</resources>

**Pra criar cor de fundo:**

**Caminho: pasta app, pasta res, pasta layout, arquivo: colors.xml**

**ARQUIVO: colors.xml**

<? xml version = “1.0” encoding=”utf-8”?>

<resources>

<color name= “purple\_200”> #FF8886FC</color>

<color name= “purple\_500”> #FF6200EE</color>

<color name= “purple\_700”> #FF3700B3</color>

<color name= “teal\_200”> #FF03DAC5</color>

<color name= “teal\_700”> #FF018786</color>

<color name= “black”> #FF000000</color>

<color name= “white”> #FFFFFFFF</color>

<color name=”light\_gray”>#DADADA</color>

</resources>

**Agora precisa recuperar a referência de cada campo da tela de Cadastro pra conseguir começar o processo de cadastro dele. Tem 2 formas de fazer isso:**

**1. Método nativo: findViewById**

**2. Usar a biblioteca do viewBinding, é a biblioteca mais atual do Android para recuperar os componentes.**

Acessa o build.gradle a nível de app

Embaixo de:

kotlinOption{

jvmTarget=’1.0’

}

viewBinding{

enabled true

}

Dá um sync now

**ARQUIVO: FORMCADASTRO.KT**

class FormCadastro: AppCompatActivity(){

private lateinit var binding: ActivityFormCadastroBinding // Quando coloca o nome da activity, ele pega o nome do cadastro, e o binding que é a biblioteca que faz isso

private val auth = FirebaseAuth.getInstance() // classe FirebaseAuth.recupera a instancia do servidor

override fun onCreate (savedInstanceState: Bundle?){

super.onCreate(savedInstanceState)

binding = ActivityFormCadastroBinding.inflate(layoutInflater) // método get está oculto

setContentView(binding.root) // método que exibe o layout de Linear Layout (setContentView)

// Propriedade it herda de uma view. Renomeia pra view

binding.btCadastrar.setOnClickListener{ view ->

// binding.recuperarCampo.capturarformato.transformaemString

val email = binding.editEmail.text.toString()

val senha = binding.editSenha.text.toString()

if (email.isEmpty() || senha.isEmpty()){

//vou criar uma snack bar

val snackbar = SnackBar.make(view, “Preencha todos os campos!”, SnackBar.LENGTH\_SHORT )

// coloco a cor de fundo na snackbar

snackbar.setBackgroundTint (Color.RED)

snackbar.show()

} else{

// Cadastra o usuário no Firebase. O método addOnCompleteListener é o primeiro da lista

auth.createUserWithEmailAndPassword(email, senha).addOnCompleteListener {cadastro ->

if (cadastro.isSuccessful){

val snackbar = SnackBar.make(view, “Sucesso ao cadastrar o usuário”, SnackBar.LENGTH\_SHORT )

// coloco a cor de fundo na snackbar

snackbar.setBackgroundTint (Color.BLUE)

snackbar.show()

// limpa os campos de email e senha

binding.editEmail.setText(“”)

binding.editSenha.setText(“”)

} // fecha cadastro.isSucessful

// trata as mensagens de erro

}.addOnFailureListener{ exception ->

val mensagemErro = when(exception){

// se o usuário colocar a senha menor que 6 números/caracteres

is FirebaseAuthWeakPasswordException -> “Digite uma senha com no mínimo 6 caracteres!”

// email inválido, sem arroba, sem o .com

is FirebaseAuthInvalidCredentialException -> “Digite um email válido”

// se o usuário cadastrar uma conta que já foi cadastrada

is FirebaseAuthUserCollisionException -> “Esta conta já foi cadastrada!”

// se o usuário for criar uma conta e não tiver conexão com a internet

is FirebaseNetworkException -> “Sem conexão com a internet!”

// Erro genérico

else -> “Erro ao cadastrar usuário!”

}// fim do when

val snackbar = SnackBar.make(view, mensagemErro, SnackBar.LENGTH\_SHORT )

snackbar.setBackgroundTint (Color.RED)

snackbar.show()

}

}

}

}

}

**Agora vamos ter que validar os campos de email e senha, tem que validar o formulário. Se o usuário não preencher email e senha, vai aparecer mensagem de erro. Só cadastra usuário que tiver todos os campos preenchidos.** Quando clicar em cadastrar, vai fazer a verificação. Preciso colocar um evento de clique no botão.

Para isso  **veja o método setOnClickListener**

**Para cadastrar o usuário a partir do formulário de cadastro preenchido, vá no Painel do Firebase. Primeiros passos para o Android, procure por Authentication. Ele cadastra os usuários e faz a autenticação deles. Copia a dependência:** com.google.firebase:firebase-auth

Vá no build.gradle a nível de app. Cola dentro de dependencies, depois da implementação do analytics. Fica:

Implementation ‘com.google.firebase:firebase-auth’

Dá um sync now

**Volta pro Painel Principal do Firebase, veja se está no Aplicativo de Teste. Clique em Authentication. Clica em Primeiros Passos. Em Sign in Method, tem todos os provedores de login: desde email e senha até o GitHub, de forma anônima. VAMOS USAR O EMAIL E SENHA. Clica em Email e Senha, ativar, salvar.**

**A aba Users fica habilitado. VEJA O ELSE DO SETONCLICKLISTENER**

**Depois de fazer o if-else do botão Cadastrar, testa a aplicação. O FIREBASE ACEITA SENHA DE NO MÍNIMO, 6 caracteres. Volta no Firebase e veja o usuário cadastrado. Podemos ou cadastrar pela própria aplicação ou pelo Authentication.**  Exemplo do uso do Authentication é quando precisa criar um perfil de Administrador, dá para fazer testes.

**PROBLEMA: se cadastrar como:**

[**marcos500@gmail.com**](mailto:marcos500@gmail.com) **e a senha:12345 ele não aparece nenhuma mensagem de erro, mas NÃO CADASTROU. Se tentar cadastrar uma senha que não existe não tem erro, se o usuário não tiver internet , tem que aparecer o erro.**

**1.5.VÍDEO: Curso de Firebase – Tratando exceptions no Firebase #4**

**Tempo: 12min43s, link:** [**https://youtu.be/79UVs1kuGwI?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**](https://youtu.be/79UVs1kuGwI?list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9)

Vamos tratar as mensagens de erro, as exceptions.

**Vamos tratar as falhas no addOnFailureListener**

**PRECISA COLOCAR UMA PERMISSÃO DE INTERNET**

**ARQUIVO: ANDROIDMANIFEST.XML**

**Caminho: pasta app, pasta manifest, arquivo:androidManifest.xml**

<? xml version = “1.0” encoding=”utf-8”?>

<manifest xmlns: android=”<http://schemas.android.com/apk/res/android>”

package=”com.stackmobile.aplicativodeteste”>

<uses-permission android:name=”androdi.permission.INTERNET”>

Emula o aplicativo e testa cada situação de erro, que são corretamente capturados.

**1.6.VÍDEO: Curso de Firebase – Autenticando usuários com e-mail e senha #5**

**Tempo: 18min10s, link:** [**https://youtu.be/A6FQOzRwyBI?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**](https://youtu.be/A6FQOzRwyBI?list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9)

Vamos criar o layout da tela de login e a autenticação de usuário usando o Firebase Auth. No projeto, acesse Java, pasta com.stackmobile.aplicativodeteste.view, dentro dela tem as pastas formcadastro e formlogin.

Dentro de formlogin, clique com o botão direito, new, activity, empty activity. Nome: FormLogin, finish. Dá um alt+ enter para importar a classe R. Veja se a FormLogin.kt está dentro da pasta formlogin.

Acesse o layout da formlogin (activity\_form\_login.xml). A form login será muito parecida com o layout da formcadastro. Então, copie de formcadastro.

**ARQUIVO: ACTIVITY\_FORM\_LOGIN.xml**

<? xml version=’1.0’ encoding=”utf-8”?>

**// Colocando o ScrollView se o usuário estiver usando o dispositivo deitado**

<androidx.core.widget.NestedScrollView

android:layout\_width=”match\_parent” // altura máxima do container

android:layout\_height=”match\_parent” // largura máxima do container

android:fillViewport = “true”

xmlns:android=”http://schemas.android.com/apk/res/android”>

<LinearLayout

xmlns:tools=” <http://schemas.android.com/tools>”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

android:orientation=”vertical”

android:background=”@color/light\_gray”

tools:contexto=”.view.formcadastro.FormCadastro”>

<TextView

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:text=”@string/titulo\_form\_login”

android:textSize=”25sp”

androidtextColor=”@color/black”

android:textStyle=”bold”

android:layout\_gravity=”center”

android:layout\_marginTop=”100dp” // afasta externamente da margem

/>

// Componentes de texto: Email

<EditText

android:id=”@+id/edit\_email”

android:layout\_width=”match\_content”

android:layout\_height=”wrap\_content”

android:hint = “@string/hint\_email” // é o placeholder

android:background=”@color/white”

android:padding=”15dp” // espaçamento interno de todos os lados de 15dp

android:maxLines=”1”// para evitar de descer a linha e quebrar a aplicação

android:inputType=”textEmailAddress” //define o tipo de teclado

android\_layout\_marginStart=”20dp” // é a margem inicial da esquerda

android\_layout\_marginEnd=”20dp” // é a margem final da direita

android:layout\_marginTop=”20dp” // afasta da margem de cadastre-se

/>

// Componentes de texto: Senha

<EditText

android:id=”@+id/edit\_senha”

android:layout\_width=”match\_content”

android:layout\_height=”wrap\_content”

android:hint = “@string/hint\_senha” // é o placeholder

android:background=”@color/white”

android:padding=”15dp” // espaçamento interno de todos os lados de 15dp

android:maxLines=”1”// para evitar de descer a linha e quebrar a aplicação

android:inputType=”numberPassword” //não vai poder ver a senha enquanto o usuário digita

android\_layout\_marginStart=”20dp” // é a margem inicial da esquerda

android\_layout\_marginEnd=”20dp” // é a margem final da direita

android:layout\_marginTop=”10dp” // afasta da margem de cadastre-se

/>

<Button

android:id=”@+id/bt\_entrar”

android:layout\_width=”match\_content”

android:layout\_height=”wrap\_content”

android:text=”@string/texto\_botao\_entrar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textSize=”bold”

android:textAllCaps =“false” // somente a primeira letra é maiúscula

android:padding=”15dp”

android:backgroundTint = “@color/black” // pinta a cor interna do botão

android:layout\_margin=”20dp” // margem geral tem espaçamento 20 pra todo mundo

/>

// Para que o usuário acesse a tela de cadastro

<TextView

android:id=”@id/txt\_tela\_cadastro”

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:text=”@string/texto\_tela\_cadastro”

android:textSize=”16sp”

androidtextColor=”@color/black”

android:textStyle=”bold”

android:layout\_gravity=”center”

android:layout\_marginTop=”50dp” // afasta externamente da margem

/>

</LinearLayout>

</androidx.core.widget.NestedScrollView>

**Abra o arquivo de strings.xml**

**ARQUIVO:STRINGS.XML**

<resources>

<string name=”app\_name”> Aplicativo de Teste </string>

<string name=”titulo\_form\_cadastro”> Cadastre-se </string>

<string name=”titulo\_form\_login”>Login do Usuário</string>

<string name=”hint\_email”>Email </string>

<string name=”hint\_senha”>Senha </string>

<string name=”texto\_botao\_cadastrar”>Cadastrar </string>

<string name=”texto\_botao\_entrar”>Entrar </string>

<string name=”texto\_tela\_cadastro”>Não tem uma conta? Cadastre-se agora mesmo! </string>

</resources>

**Abra o FormLogin.kt**

**ARQUIVO: FORMLOGIN.KT**

class FormLogin: AppCompatActivity(){

private lateinit var binding : ActivityFormLoginBinding

private auth = FirebaseAuth.getInstance()

override fun onCreate (savedInstanceState: Bundle?){

super.onCreate(savedInstanceState)

binding = ActivityFormLoginBinding.inflate(layoutInflater)

setContentView(binding.root)

// Para validar os campos de entrada do login

binding.btEntrar.setOnClickListener{ view ->

val email = binding.editEmail.text.toString()

val senha = binding.editSenha.text.toString()

if (email.isEmpty() || senha.isEmpty()){

val snackbar = Snackbar.make(view, “Preencha todos os campos”, Snackbar.LENGTH\_SHORT)

snackbar.setBackgroundTint(Color.RED)

snackbar.show()

}else{

// Faz o login autenticado

auth.signInWithEmailAndPassword(email, senha).addOnCompleteListener{ autenticação ->

if (autenticação.isSuccessful){

// Manda pra tela principal

navegarTelaPrincipal()

}.addOnFailureListener{

// Erro geral

val snackbar = Snackbar.make(view, “Erro ao fazer o login do usuário!”, Snackbar.LENGTH\_SHORT)

snackbar.setBackgroundTint(Color.RED)

snackbar.show()

}

}

}

}

// Para que o usuário acesse a tela cadastro assim que clicar no texto e executa normalmente

binding.txtTelaCadastro.setOnClickListener{

val intent = Intent(this, FormCadastro:: class.java)

startActivity(intent)

}

}

// Cria o método pra usar pra chamar a tela principal

private fun navegarTelaPrincipal(){

val intent = Intent(this, telaPrincipal :: class.java)

startActivity(Intent)

finish() // finaliza a tela de login( que é o contexto) depois que entrar na tela principal

}

}

**Para iniciar na tela de login, abra o Manifest (pasta app, pasta manifests, arquivo: AndroidManifest.xml)**

Primeira Activity:

<activity

android:name=”.view.formcadastro.FormCadastro”

android:exported = “false”

/>

Segunda Activity:

<activity

android:name=”.view.formlogin.FormLogin”

android:exported = “true”

<intente-filter>

<action android:name=”android.intent.action.MAIN”/>

<category android:name=”android.intent.category.LAUNCHER”/>

</intent-filter>

</activity>

</application>

</manifest>

**Tenho que acessar a FormCadastro na tela de FormLogin.xml. Tenho que colocar um texto, como: “Não tem conta? Cadastre-se agora mesmo.”**

**Para criar a tela PRINCIPAL.**

Abra a pasta app, pasta java, pasta com.stackmobile.aplicativodeteste.view, clica na pasta da view, botão direito, new, package, nome: telaprincipal.

Dentro da pasta tela principal, clica com o botão direito, new, Activity, emptyActivity, nome: TelaPrincipal. Espera carregar o arquivo e dá alt+enter para carregar o R.

**Volte para FormLogin.kt**

**Fiz a autenticação do login, mas quando volto pra aplicação tenho que logar novamente. Eu tenho que ver se o usuário está logado.**

**1.7.VIDEO: Curso de Firebase – Verificando usuário logado e deslogando usuário do sistema #6**

**Tempo: 10min50s, link:** [**https://youtu.be/oqKOm6vjtzs?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**](https://youtu.be/oqKOm6vjtzs?list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9)

Precisamos verificar se o usuário está logado e quando deslogar.

**ARQUIVO: FORMLOGIN.KT**

**// Abaixo do método navegarTelaPrincipal, CRIA UM CICLO DE VIDA**

override fun onStart(){

super.onStart()

val usuarioAtual = FirebaseAuth.getInstance().currentUser

if(usuarioAtual != null) {

navegarTelaPrincipal()

}

}

**Se o usuário for nulo, ele tem que ser deslogado; porém não temos como deslogar. Por isso, precisa ser criado um botão de deslogar no layout da tela principal.**

**ABRA O ARQUIVO: activity\_tela\_principal.xml**

<?xml version = “1.0” enconding=”utf-8”>

<LinearLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”<http://schemas.android.com/apk/res-auto>”

android:orientation=”vertical”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

tools:contexto=”.view.telaprincipal.TelaPrincipal”>

<Button

android:id=”@id/bt\_deslogar”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=” texto\_botao\_deslogar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

</LinearLayout>

**ARQUIVO: STRINGS.XML**

<resources>

<string name=”app\_name”> Aplicativo de Teste </string>

<string name=”titulo\_form\_cadastro”> Cadastre-se </string>

<string name=”titulo\_form\_login”>Login do Usuário</string>

<string name=”hint\_email”>Email </string>

<string name=”hint\_senha”>Senha </string>

<string name=”texto\_botao\_cadastrar”>Cadastrar </string>

<string name=”texto\_botao\_entrar”>Entrar </string>

<string name=”texto\_botao\_deslogar”>Deslogar </string>

<string name=”texto\_tela\_cadastro”>Não tem uma conta? Cadastre-se agora mesmo! </string>

</resources>

**VÁ PARA A TELAPRINCIPAL.KT**

**ARQUIVO: TELAPRINCIPAL.KT**

class TelaPrincipal: AppCompatActivity(){

private lateinit var binding : ActivityTelaPrincipalBinding

override fun onCreate (savedInstanceState: Bundle?){

super.onCreate(savedInstanceState)

binding = ActivityFormLoginBinding.inflate(layoutInflater)

setContentView(binding.root)

binding.btDeslogar.setOnClickListener{

// signOut é o método para deslogar do Firebase

FirebaseAuth.getInstance().signOut()

// Pra voltar pra tela de login

val voltarTelaLogin = Intent(this, FormLogin:: class.java)

startActivity(voltarTelaLogin)

finish()

}

}

}

**// Executa o teste pelo botão de deslogar na tela principal.**

**1.8.VÍDEO: Curso de Firebase – Iniciando com o Cloud Firestore – Guia absolutamente completo #7**

**Tempo: 24min58s, link:** [**https://youtu.be/Vy135ddtnzo?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**](https://youtu.be/Vy135ddtnzo?list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9)

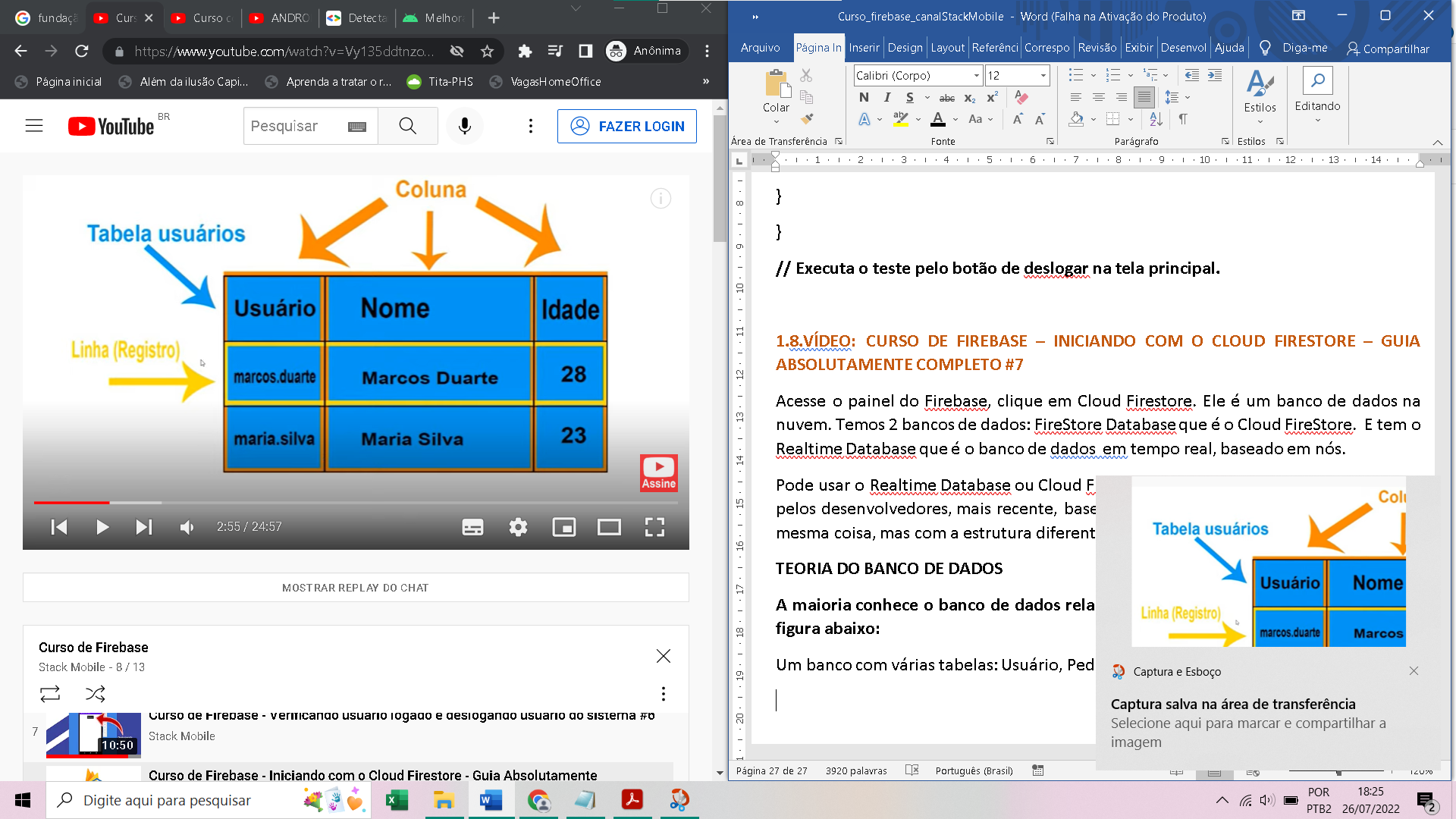
Acesse o painel do Firebase, clique em Cloud Firestore. Ele é um banco de dados na nuvem. Temos 2 bancos de dados: FireStore Database que é o Cloud FireStore. E tem o Realtime Database que é o banco de dados em tempo real, baseado em nós.

Pode usar o Realtime Database ou Cloud Firestore. O FireStore é o banco mais usados pelos desenvolvedores, mais recente, baseado em coleção de dados. Eles vão fazer a mesma coisa, mas com a estrutura diferente.

**TEORIA DO BANCO DE DADOS**

**A maioria conhece o banco de dados relacional, o banco de dados SQL. Conforme a figura abaixo:**

Um banco com várias tabelas: Usuário, Pedidos e afins.

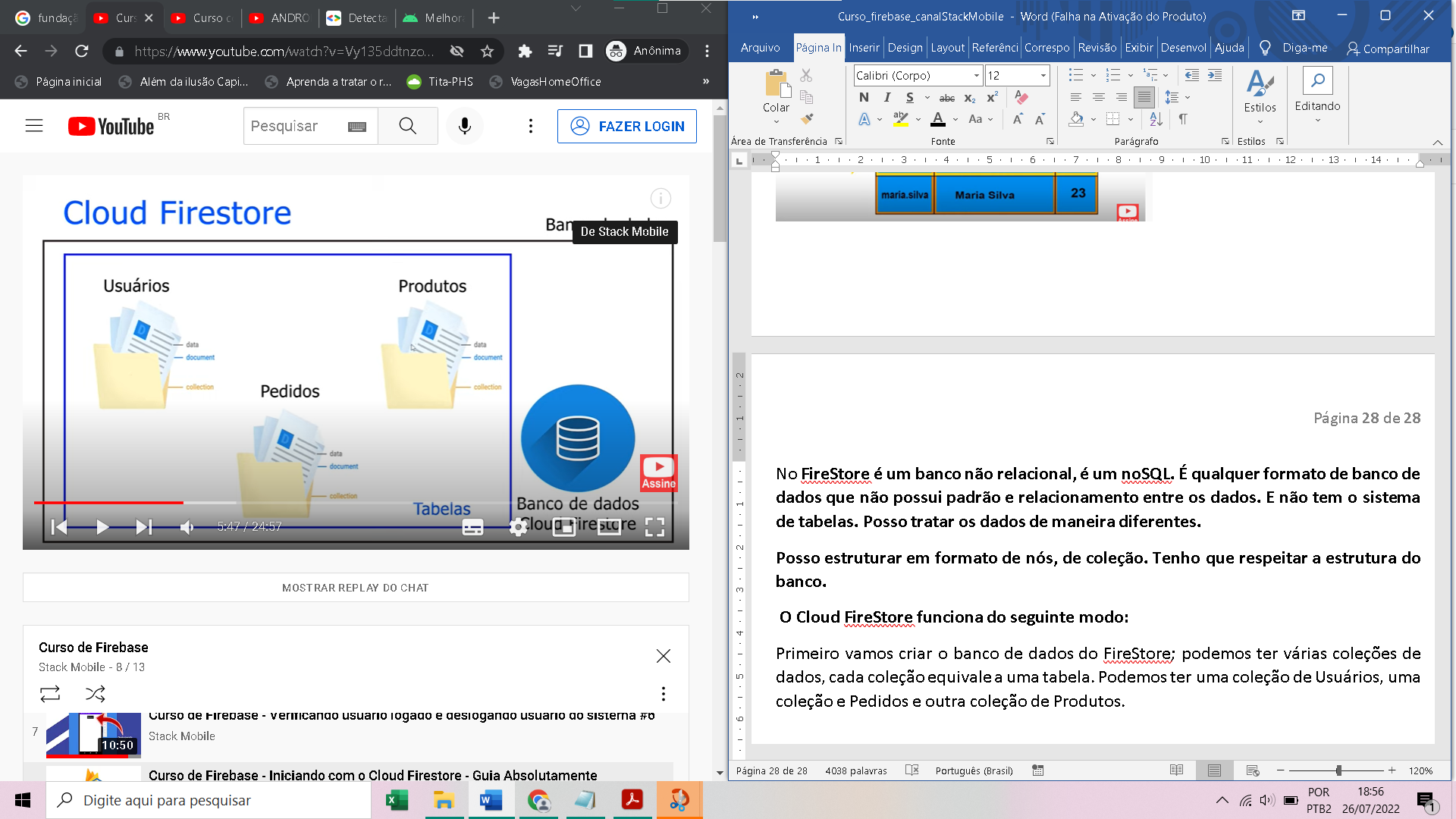
No exemplo, tem a tabela de Usuários, com colunas e linhas. Cada registro é uma linha. Colunas: Usuário, Nome, Idade.

No **FireStore é um banco não relacional, é um noSQL. É qualquer formato de banco de dados que não possui padrão e relacionamento entre os dados. E não tem o sistema de tabelas. Posso tratar os dados de maneira diferentes.**

**Posso estruturar em formato de nós, de coleção. Tenho que respeitar a estrutura do banco.**

**O Cloud FireStore funciona do seguinte modo:**

Primeiro vamos criar o banco de dados do FireStore; podemos ter várias coleções de dados, cada coleção equivale a uma tabela. Podemos ter uma coleção de Usuários, uma coleção e Pedidos e outra coleção de Produtos.

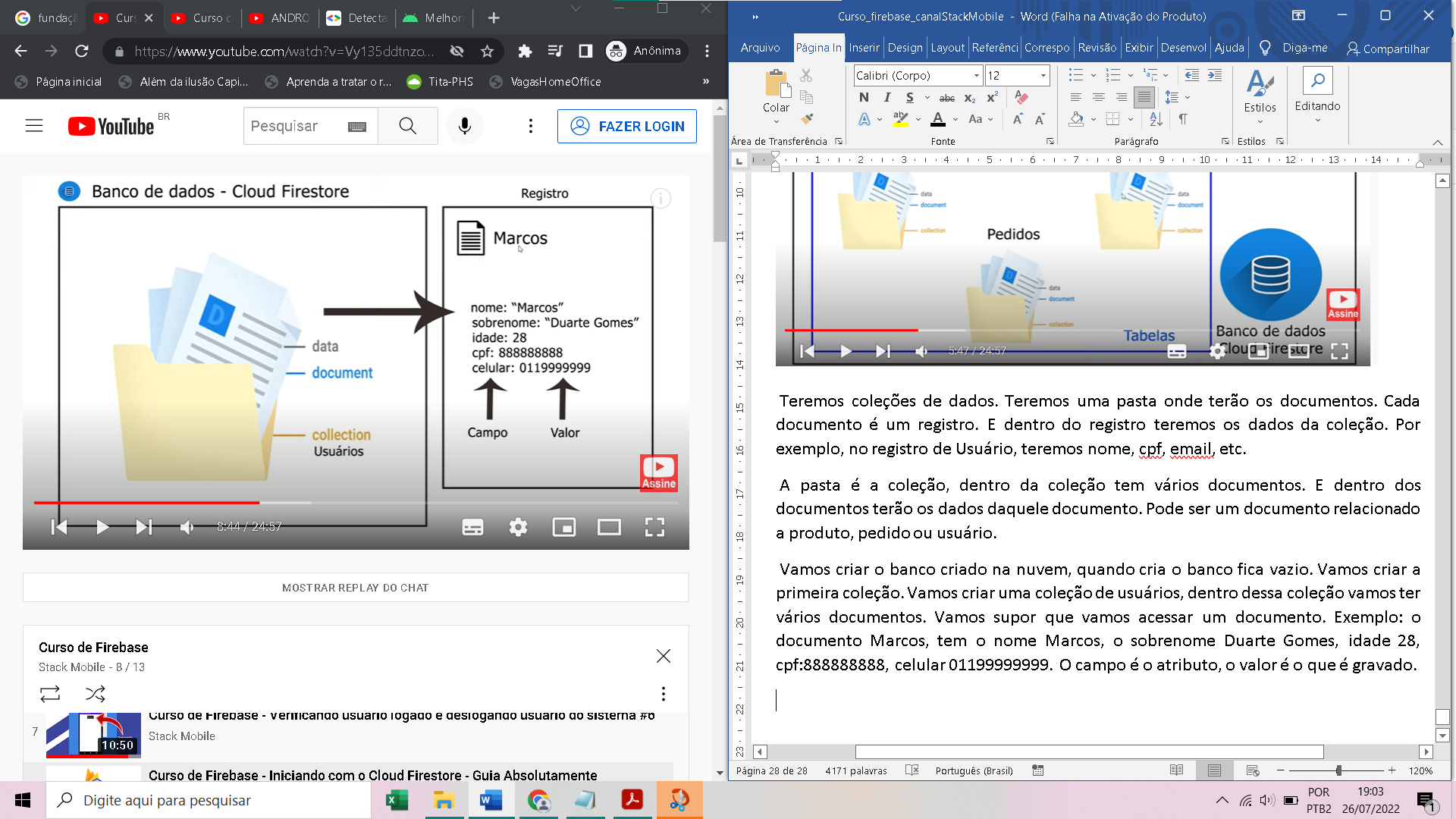


Teremos coleções de dados. Teremos uma pasta onde terão os documentos. Cada documento é um registro. E dentro do registro teremos os dados da coleção. Por exemplo, no registro de Usuário, teremos nome, cpf, email, etc.

A pasta é a coleção, dentro da coleção tem vários documentos. E dentro dos documentos terão os dados daquele documento. Pode ser um documento relacionado a produto, pedido ou usuário.

Vamos criar o banco criado na nuvem, quando cria o banco fica vazio. Vamos criar a primeira coleção. Vamos criar uma coleção de usuários, dentro dessa coleção vamos ter vários documentos. Vamos supor que vamos acessar um documento. Exemplo: o documento Marcos, tem o nome Marcos, o sobrenome Duarte Gomes, idade 28, cpf:888888888, celular 01199999999. O campo é o atributo, o valor é o que é gravado.

O data são os dados. Podemos ter vários dados do usuário, o nome é o campo e o que é preenchido é o valor. Dentro da coleção Usuários teremos vários documentos.



No site do Firebase, entre no projeto, clique em Cloud Firestore, clique no botão “Criar banco de dados”. Pode iniciar no modo de produção (quando publicar o aplicativo, tem que colocar o banco na produção. Pra colocar o banco na produção, precisa definir algumas regras de segurança. Quando o seu aplicativo estiver pronto, você muda as regras de segurança e automaticamente o banco de dados estará no modo de produção). Outra forma de iniciar é no modo de teste.

Vamos usar o “Iniciar em modo de teste”, porque eu quero testar as funções do banco. Existe um tempo limitado pra isso. Mas, podemos mudar as regras de segurança para deixá-lo em modo de produção. O tempo é de 30 dias. Clique em avançar. Deixa o local do Cloud FireStore como padrão. Clique em ativar. O banco de dados foi criado. Vamos em regras. Também podemos ver índices, usos. No uso, tem as conexões ativas, por exemplo. Podemos usar gratuitamente, mas tem que verificar qual o limite para ser gratuito.

Na ala de dados, vamos criar a primeira coleção de banco de dados. Clique no botão “Iniciar coleção”, nome: Usuários. Clique em Próximo, precisa criar os documentos. Pode deixar como código automático, ou o nome do documento como o nome do próprio usuário. Nome do documento: Marcos.

Campos, tipo e valor (respectivamente):

Nome: String e Marcos

sobrenome: String e Duarte Gomes

idade: number e 28

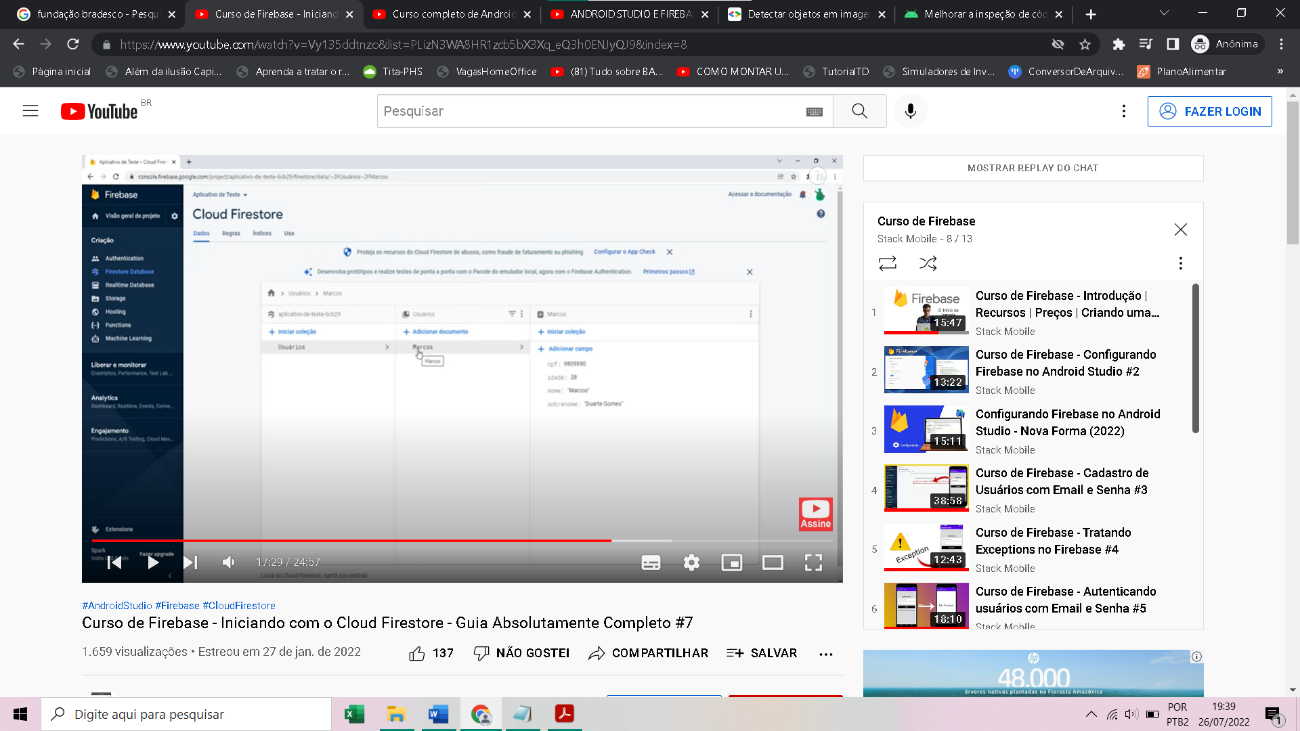
cpf: number e 888888888

Tipos mais usados: strings, boolean, number, timestamp

Para adicionar o campo, clique em Adicionar campo.

Clica em salvar.

Ele criou dentro do bd a coleção de Usuários e conseguimos visualizar os documentos.



Pode colocar um identificador único, um número pra identificar o documento. Do lado do documento consigo visualizar o json. Vou criar um novo documento, e tenho que criar os mesmos campos.

Nome do documento: Maria

Campos, tipo e valor (respectivamente):

Nome: String e Maria

sobrenome: String e da Silva

idade: number e 25

cpf: number e 484848484

Clica em salvar.

Posso criar outras coleções dentro do mesmo banco de dados.

Clico em criar coleção, nome: Produtos. Clica em Próximos. Eu quero gerar um código aleatório. Clico em gerar código automático.

Os campos, tipo e valor (respectivamente):

Nome: string e placa de vídeo

Preco: number e 800,00

Salvar

Pode ter várias coleções para separar as funções. **As coleções não se comunicam entre si.**

**Para EXCLUIR,**  clica nos três pontinhos e pode excluir documento e excluir campos do documento. Cada um é separado. A exclusão dos campos limpa o json; a exclusão do documento tira o registro daquele documento. **Para excluir a coleção,** clico nos três pontos e tenho que colocar exatamente o nome da coleção e depois clica em Excluir.

**1.9.VÍDEO: Curso de Firebase – Como gravar dados no banco de dados #8**

Tempo: 14min20s, link: <https://youtu.be/1ey3NnyYuns?list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9>

Vamos colocar um botão dentro do activity\_tela\_principal.xml para gravar no banco de dados. Para usar o Cloud Firestore, tem que criar o banco no painel principal (da Firebase) e depois implementar as dependências dentro do Android.

Acesse a documentação, primeiros passos para android, ver dependências do Cloud Firebase. Vamos acessar o build.gradle a nível de app.

**ARQUIVO: BUILD.GRADLE (Module: Aplicativo de teste app)**

dependencies{

// depois do implementation da google play services

Implementation ‘com.google.firebase:firebase-firestore’

}

Dá um sync now

**Arquivo: activity\_tela\_principal.xml**

<?xml version = “1.0” enconding=”utf-8”>

<LinearLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”<http://schemas.android.com/apk/res-auto>”

android:orientation=”vertical”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

tools:contexto=”.view.telaprincipal.TelaPrincipal”>

<Button

android:id=”@id/bt\_deslogar”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=” texto\_botao\_deslogar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Botão para gravar no banco de dados

<Button

android:id=”@id/bt\_gravarDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=” gravar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

</LinearLayout>

**Vá para o arquivo: TelaPrincipal.kt**

class TelaPrincipal: AppCompatActivity(){

private lateinit var binding : ActivityTelaPrincipalBinding

private val db = FirebaseFirestore .getInstance()

override fun onCreate (savedInstanceState: Bundle?){

super.onCreate(savedInstanceState)

binding = ActivityFormLoginBinding.inflate(layoutInflater)

setContentView(binding.root)

binding.btDeslogar.setOnClickListener{

// signOut é o método para deslogar do Firebase

FirebaseAuth.getInstance().signOut()

// Pra voltar pra tela de login

val voltarTelaLogin = Intent(this, FormLogin:: class.java)

startActivity(voltarTelaLogin)

finish()

}

binding.btGravarDadosDB.setOnClickListener{

// usa o hashmap para criar os campos com valores dentro do documento

val usuariosMap = hashMapOf(

“nome” to “Maria”,

“sobrenome” to “da Silva”,

“idade” to 25

)

db.collection(“Usuarios”).document(“Maria”)

.set(usuariosMap).addOnCompleteListener{

// d de debug

Log.d(“db”, “sucesso ao salvar os dados do usuário!”)

}.addOnFailureListener{

}

}

}

}

**1.10 VÍDEO: Curso de Firebase – Como fazer a leitura dos dados no banco de dados #9**

**Tempo: 10min47s, link:** [**https://youtu.be/DeX0oiUwNSg?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**](https://youtu.be/DeX0oiUwNSg?list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9)

Vamos aprender a fazer a leitura do banco de dados.

Vai criar mais um botão na activity\_tela\_principal.xml

**Arquivo: activity\_tela\_principal.xml**

<?xml version = “1.0” enconding=”utf-8”>

<LinearLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”<http://schemas.android.com/apk/res-auto>”

android:orientation=”vertical”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

tools:contexto=”.view.telaprincipal.TelaPrincipal”>

<Button

android:id=”@id/bt\_deslogar”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=” texto\_botao\_deslogar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Botão para gravar no banco de dados

<Button

android:id=”@id/bt\_gravarDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=” gravar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Botão para leitura no banco de dados

<Button

android:id=”@id/bt\_lerDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=”ler”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layout\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Para aparecer os dados recuperados do banco

<TextView

android:id=”@id/txt\_resultadoNome”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=””

android:textSize=”18sp”

android:layout\_gravity=”center”

android:layout\_marginTop =”20dp”

/>

<TextView

android:id=”@id/txt\_resultadoSobrenome”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=””

android:textSize=”18sp”

android:layout\_gravity=”center”

android:layout\_marginTop =”20dp”

/>

<TextView

android:id=”@id/txt\_resultadoIdade”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=””

android:textSize=”18sp”

android:layout\_gravity=”center”

android:layout\_marginTop =”20dp”

/>

</LinearLayout>

**Vá para TelaPrincipal.kt**

class TelaPrincipal: AppCompatActivity(){

private lateinit var binding : ActivityTelaPrincipalBinding

private val db = FirebaseFirestore .getInstance()

override fun onCreate (savedInstanceState: Bundle?){

super.onCreate(savedInstanceState)

binding = ActivityFormLoginBinding.inflate(layoutInflater)

setContentView(binding.root)

binding.btDeslogar.setOnClickListener{

// signOut é o método para deslogar do Firebase

FirebaseAuth.getInstance().signOut()

// Pra voltar pra tela de login

val voltarTelaLogin = Intent(this, FormLogin:: class.java)

startActivity(voltarTelaLogin)

finish()

}

binding.btGravarDadosDB.setOnClickListener{

// usa o hashmap para criar os campos com valores dentro do documento

val usuariosMap = hashMapOf(

“nome” to “Maria”,

“sobrenome” to “da Silva”,

“idade” to 25

)

db.collection(“Usuarios”).document(“Maria”)

.set(usuariosMap).addOnCompleteListener{

// d de debug

Log.d(“db”, “sucesso ao salvar os dados do usuário!”)

}.addOnFailureListener{}

}

binding.btLerDadosDB.setOnClickListener{

db.collection(“Usuários”).document(“Maria”)

// escolhe o segundo método de addSnapshotListener, dá enter 2x

// value é o valor do documento, e error é a exception de addOnFailure

.addSnapshotListener{ documento, error ->

if(documento != null){

val idade = documento.getLong(“idade”)

// recupero pela chave, a chave é o campo. Fica 1 binding pra cada campo

binding.txtResultadoNome.text = documento.getString(“nome”)

binding.txtResultadoSobrenome.text = documento.getString(“sobrenome”)

// Idade é number no banco de dados, então usamos uma variável idade

binding.txtResultadoIdade.text = idade.toString()

}

// Posso dar vário textView, um para cada campo do banco de dados no seu layout

}

}

}

}

**1.11.VÍDEO: Curso Firebase – Como atualizar os dados no banco de dados #10**

**Tempo: 8min42s, link:** [**https://youtu.be/5jQnqKlYE30?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**](https://youtu.be/5jQnqKlYE30?list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9)

Na activity\_tela\_principal.xml, teremos que criar um botão para atualizar os dados, e ele vai ficar antes de gravar.

**Arquivo: activity\_tela\_principal.xml**

<?xml version = “1.0” enconding=”utf-8”>

<LinearLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”<http://schemas.android.com/apk/res-auto>”

android:orientation=”vertical”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

tools:contexto=”.view.telaprincipal.TelaPrincipal”>

<Button

android:id=”@id/bt\_deslogar”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=” texto\_botao\_deslogar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Botão para atualizar dados no banco de dados

<Button

android:id=”@id/bt\_atualizarDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=”Atualizar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Botão para gravar no banco de dados

<Button

android:id=”@id/bt\_gravarDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=” gravar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Botão para leitura no banco de dados

<Button

android:id=”@id/bt\_lerDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=”ler”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layout\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Para aparecer os dados recuperados do banco

<TextView

android:id=”@id/txt\_resultadoNome”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=””

android:textSize=”18sp”

android:layout\_gravity=”center”

android:layout\_marginTop =”20dp”

/>

<TextView

android:id=”@id/txt\_resultadoSobrenome”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=””

android:textSize=”18sp”

android:layout\_gravity=”center”

android:layout\_marginTop =”20dp”

/>

<TextView

android:id=”@id/txt\_resultadoIdade”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=””

android:textSize=”18sp”

android:layout\_gravity=”center”

android:layout\_marginTop =”20dp”

/>

</LinearLayout>

**Com o botão Atualizar criado, vamos configurar o evento de clique no TelaPrincipal.kt**

**Vá para TelaPrincipal.kt**

class TelaPrincipal: AppCompatActivity(){

private lateinit var binding : ActivityTelaPrincipalBinding

private val db = FirebaseFirestore .getInstance()

override fun onCreate (savedInstanceState: Bundle?){

super.onCreate(savedInstanceState)

binding = ActivityFormLoginBinding.inflate(layoutInflater)

setContentView(binding.root)

binding.btDeslogar.setOnClickListener{

// signOut é o método para deslogar do Firebase

FirebaseAuth.getInstance().signOut()

// Pra voltar pra tela de login

val voltarTelaLogin = Intent(this, FormLogin:: class.java)

startActivity(voltarTelaLogin)

finish()

}

binding.btGravarDadosDB.setOnClickListener{

// usa o hashmap para criar os campos com valores dentro do documento

val usuariosMap = hashMapOf(

“nome” to “Maria”, // “nome” to “Marcos”,

“sobrenome” to “da Silva”, // “sobrenome” to “Duarte Gomes”

“idade” to 25 // “idade” to 28

)

db.collection(“Usuarios”).document(“Maria”)

.set(usuariosMap).addOnCompleteListener{

// d de debug

Log.d(“db”, “sucesso ao salvar os dados do usuário!”)

}.addOnFailureListener{}

}

binding.btLerDadosDB.setOnClickListener{

db.collection(“Usuários”).document(“Marcos”)

// escolhe o segundo método de addSnapshotListener, dá enter 2x

// value é o valor do documento, e error é a exception de addOnFailure

.addSnapshotListener{ documento, error ->

if(documento != null){

val idade = documento.getLong(“idade”)

// recupero pela chave, a chave é o campo. Fica 1 binding pra cada campo

binding.txtResultadoNome.text = documento.getString(“nome”)

binding.txtResultadoSobrenome.text = documento.getString(“sobrenome”)

// Idade é number no banco de dados, então usamos uma variável idade

binding.txtResultadoIdade.text = idade.toString()

}

// Posso dar vários textView, um para cada campo do banco de dados no seu layout

}

}

binding.btAtualizarDadosDB.setOnClickListener{

// quer atualizar a idade, pega o primeiro método de update

db.collection(“Usuários”).documento(“Marcos”) // Maria, Ferreira, 30; Marcos, da Silva, 21

.update(“sobrenome”, “Duarte Gomes”, “idade”, 28).addOnCompleteListener{

Log.d(“db\_update”, “sucesso ao atualizar os dados do usuário!”)

}

}

}

}

**1.12.VÍDEO: Curso Firebase – Como deletar os dados no banco de dados #11**

**Tempo:06min29s, link:** [**https://youtu.be/jMN1LewT-W0?list=PLizN3WA8HR1zcb5bX3Xq\_eQ3h0ENJyQJ9**](https://youtu.be/jMN1LewT-W0?list=PLizN3WA8HR1zcb5bX3Xq_eQ3h0ENJyQJ9)

Vamos deletar um documento da coleção. Para isso, vamos acrescentar um botão no layout da activity\_tela\_principal.xml e depois capturar o botão no TelaPrincipal.kt

**Arquivo: activity\_tela\_principal.xml**

<?xml version = “1.0” enconding=”utf-8”>

<LinearLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”<http://schemas.android.com/apk/res-auto>”

android:orientation=”vertical”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

tools:contexto=”.view.telaprincipal.TelaPrincipal”>

<Button

android:id=”@id/bt\_deslogar”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=” texto\_botao\_deslogar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Botão para atualizar dados no banco de dados

<Button

android:id=”@id/bt\_atualizarDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=”Atualizar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20dp”

>

// Botão para deletar dados no banco de dados

<Button

android:id=”@id/bt\_deletarDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=”Deletar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20dp”

>

// Botão para gravar no banco de dados

<Button

android:id=”@id/bt\_gravarDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=” gravar”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layouit\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Botão para leitura no banco de dados

<Button

android:id=”@id/bt\_lerDadosDB”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=”ler”

android:textSize=”18sp”

android:textColor=”@color/white”

android:textStyle=”bold”

android:textAllCaps=”false”

android:backgroundTint=”@color/black”

android:layout\_gravity=”center”

android:padding = “15dp”

android:layout\_marginTop =”20sp”

>

// Para aparecer os dados recuperados do banco

<TextView

android:id=”@id/txt\_resultadoNome”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=””

android:textSize=”18sp”

android:layout\_gravity=”center”

android:layout\_marginTop =”20dp”

/>

<TextView

android:id=”@id/txt\_resultadoSobrenome”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=””

android:textSize=”18sp”

android:layout\_gravity=”center”

android:layout\_marginTop =”20dp”

/>

<TextView

android:id=”@id/txt\_resultadoIdade”

android:layout\_width=”wrap\_parent” // pega a altura máxima

android:layout\_height=”wrap\_parent”// pega a largura máxima

android:text=””

android:textSize=”18sp”

android:layout\_gravity=”center”

android:layout\_marginTop =”20dp”

/>

</LinearLayout>

**Com o botão Deletar criado, vamos configurar o evento de clique no TelaPrincipal.kt**

**Vá para TelaPrincipal.kt**

class TelaPrincipal: AppCompatActivity(){

private lateinit var binding : ActivityTelaPrincipalBinding

private val db = FirebaseFirestore .getInstance()

override fun onCreate (savedInstanceState: Bundle?){

super.onCreate(savedInstanceState)

binding = ActivityFormLoginBinding.inflate(layoutInflater)

setContentView(binding.root)

binding.btDeslogar.setOnClickListener{

// signOut é o método para deslogar do Firebase

FirebaseAuth.getInstance().signOut()

// Pra voltar pra tela de login

val voltarTelaLogin = Intent(this, FormLogin:: class.java)

startActivity(voltarTelaLogin)

finish()

}

binding.btGravarDadosDB.setOnClickListener{

// usa o hashmap para criar os campos com valores dentro do documento

val usuariosMap = hashMapOf(

“nome” to “Maria”, // “nome” to “Marcos”,

“sobrenome” to “da Silva”, // “sobrenome” to “Duarte Gomes”

“idade” to 25 // “idade” to 28

)

db.collection(“Usuarios”).document(“Maria”)

.set(usuariosMap).addOnCompleteListener{

// d de debug

Log.d(“db”, “sucesso ao salvar os dados do usuário!”)

}.addOnFailureListener{}

}

binding.btLerDadosDB.setOnClickListener{

db.collection(“Usuários”).document(“Marcos”)

// escolhe o segundo método de addSnapshotListener, dá enter 2x

// value é o valor do documento, e error é a exception de addOnFailure

.addSnapshotListener{ documento, error ->

if(documento != null){

val idade = documento.getLong(“idade”)

// recupero pela chave, a chave é o campo. Fica 1 binding pra cada campo

binding.txtResultadoNome.text = documento.getString(“nome”)

binding.txtResultadoSobrenome.text = documento.getString(“sobrenome”)

// Idade é number no banco de dados, então usamos uma variável idade

binding.txtResultadoIdade.text = idade.toString()

}

// Posso dar vários textView, um para cada campo do banco de dados no seu layout

}

}

binding.btAtualizarDadosDB.setOnClickListener{

// quer atualizar a idade, pega o primeiro método de update

db.collection(“Usuários”).documento(“Marcos”) // Maria, Ferreira, 30; Marcos, da Silva, 21

.update(“sobrenome”, “Duarte Gomes”, “idade”, 28).addOnCompleteListener{

Log.d(“db\_update”, “sucesso ao atualizar os dados do usuário!”)

}

}

binding.btDeletarDados.setOnClickListener{

// quer deletar o documento todo, pega o primeiro método de addOnCompleteListener

db.collection(“Usuários”).documento(“Maria”)

.delete().addOnCompleteListener{

// quando exclui o último documento, ele exclui a coleção

Log.d(“db\_delete”, “sucesso ao excluir os dados do usuário!”)

}

}

}

}

**Tem que ser dados dinâmicos, conforme a aplicação. No tutorial foram usados dados estáticos.**

**2.CURSO DE FIREBASE , CANAL ÁREA TECH BRASIL**

**Fonte:** [**https://www.youtube.com/watch?v=FX7XSQC284U**](https://www.youtube.com/watch?v=FX7XSQC284U) **Acesso em 27/07/22**

**Tempo: 02h21min11s**

**2.1.VÍDEO: CURSO COMPLETO DE ANDROID + FIREBASE**

**Tempo: 02h21min11s, link:** [**https://www.youtube.com/watch?v=FX7XSQC284U**](https://www.youtube.com/watch?v=FX7XSQC284U) **(Canal Área Tech Brasil)**

**2.1.1. Criação do projeto no Android Studio**

Cria um projeto no Android Studio e fazer a tela de login, colocando fotos para atualizar o perfil, comprar o produto e afins. Cria um projeto chamado AndroidFirebaseProject na linguagem Java, pacote br.com.areatechbrasil.androidfirebaseprojetc, api: 17 Android 4.2 (JellyBean)

**Arquivo: MainActivity.java**

public class MainActivity extends AppCompatActivity {

@override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

}

}

**Vá na pasta br.com.areatechbrasil.androidfirebaseprojetc, botão direito, new, package. Nome: Activity. Depois clique na pasta Activity, botão direito, new, Activity, Empty Activity, Nome: Login, finish.**

**Arquivo: Login.java**

**Dá alt+ enter para importações**

public class Login extends AppCompatActivity{

@Override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

}

}

**2.1.2. Criação do design da tela de login**

**Vá em activity\_login.xml**

**Arquivo: activity\_login.xml**

// Colocar em modo split, layout é o Linear Layout

<? xml version=”1.0” enconding =”utf-8”?>

<LinearLayout.widget.ConstraintLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”http:// schemas.android.com/apk/res-auto”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

android:orientation=”vertical”

android:gravity=”center”

android:background=”@drawable/background”

tools:context=”.Activity.Login”>

<TextView

android:layout\_width=”match\_parent”

android:layout\_height=”wrap\_ content”

android:text=”Firebase Project”

android:textColor=”@color/white”

android:textSize=”40sp”

android:layout\_marginBottom=”120dp” // margem pra baixo

android:textAlignment=”center”

/>

// Pro usuário colocar o E-mail

<EditText

android:id=”@+id/edt\_email”

android:layout\_width=”1800dp”

android:layout\_height=”wrap\_ content”

android:hint=”E-mail”

android:inputType=”textEmailAddress”

android:textColorHint=”@color/white”

android:textColor=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

/>

// Para o usuário colocar a Senha

<EditText

android:id=”@+id/edt\_senha”

android:layout\_width=”300dp”

android:layout\_height=”wrap\_content”

android:hint=”Senha”

android:inputType=”textPassword” // é a senha mascarada com bolinhas

android:textColorHint=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

android:layout\_marginTop=”10dp”

android:textColor=”@color/white”

/>

// Para dar a opção de mostrar a senha

<CheckBox

android:id=”@+id/ckb\_mostrar\_senha”

android:layout\_width=”200dp”

android:layout\_height=”wrap\_ content”

android:text=”Mostrar senha”

android:textColor =”@color/white”

android:buttonInt=”@color/white” // deixar o quadrado branco

android:layout\_marginRight=”70dp”

android:layout\_marginTop=”10dp”

/>

// Botão para logar

<Button

android:id=”@+id/btn\_login”

android:layout\_width=”380dp”

android:layout\_height=”wrap\_ content”

android:background=”@color/white”

android:textColor=”@android:color/holo\_red\_light”

android:text=”Login”

android:layout\_marginTop=”20dp”

/>

// Botão para registrar

<Button

android:id=”@+id/btn\_registrar”

android:layout\_width=”380dp”

android:layout\_height=”wrap\_ content”

android:background=”#30FFFFFF”

android:textColor=”@color/white”

android:text=”Registrar”

android:layout\_marginTop=”20dp”

/>

// Barra de carregando. O padrão é um círculo em azul anil carregando

<ProgressBar

android:id=”@+id/loginProgressBar”

android:layout\_width=”match\_parent”

android:layout\_height=”wrap\_ content”

android:indeterminateTint=”@color/white” // cor da barra carregando

android:indeterminate=”true”

style=”@style/ProgressBar\_Horizontal” // Para deixar uma barra horizontal

android:visibility=”invisible” // só vai ficar visível quando o usuário tocar no botão login

android:layout\_marginTop=”20dp”

</LinearLayout>

**Para criar o background, vá na pasta res, pasta drawable, new Drawable Resource File. File name: background**

**O shape é o corpo. Para colocar as cores precisa atualizar o arquivo: colors.xml, dentro da pasta values**

**ARQUIVO: Background.xml**

<? xml version=”1.0” enconding =”utf-8”?>

<shape xlmns:android=”http:// schemas.android.com/apk/res/android”>

<gradient android:endColor=”@color/mid”

android:startColor=”@color/start”

android:angle=”90”/>

</shape>

**ARQUIVO: Colors.xml**

<? xml version = “1.0” encoding=”utf-8”?>

<resources>

<color name= “purple\_200”> #FF8886FC</color>

<color name= “purple\_500”> #FF6200EE</color>

<color name= “purple\_700”> #FF3700B3</color>

<color name= “teal\_200”> #FF03DAC5</color>

<color name= “teal\_700”> #FF018786</color>

<color name= “black”> #FF000000</color>

<color name= “white”> #FFFFFFFF</color>

<color name= “start”> #E94057 </color>

<color name= “mid”> #A020f0 </color>

</resources>

Volta pro arquivo: activity\_login.xml

Layout da tela de Login pronta

Vá na pasta res, pasta themes, arquivo: themes.xml

**Arquivo: Themes.xml**

<resources xmlns:tools=”<http://schemas.android.com/tools>”>

<!-- Base aplication theme -->

<style name= “Theme\_AndroidFirebaseProject” parent=”Theme.AppCompat.Light.NoActionBar”> // parent com outro tema é pro app não aparecer a barra de título

<!--Primary base color -->

<item name=”colorPrimary”>@color/purple\_500</item>

<item name=”colorPrimaryVariant”>@color/purple\_700</item>

<item name=”colorOnPrimary”>@color/white</item>

<!--Secondary base color -->

<item name=”colorSecondary”>@color/teal\_200</item>

<item name=”colorSecondaryVariant”>@color/teal\_700</item>

<item name=”colorOnSecondary”>@color/black</item>

<!--Status bar color -->

<item name=”android:statusBarColor” tools:targetApi=”1”>?attr/colorPrimaryVariant</item>

<!--Customize your theme here -->

<item name=”windowActionBar”> false </item>

<item name=”colorPrimaryDark”> @color/mid </item> // cor padrão do aplicativo

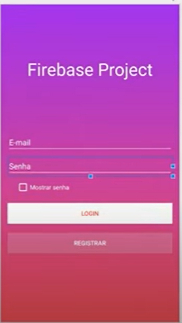
<item name=”windowNoTitle”> true </item>

</style>

</resources>

**Após atualizar o theme, o layout fica com o design do lado esquerdo.**

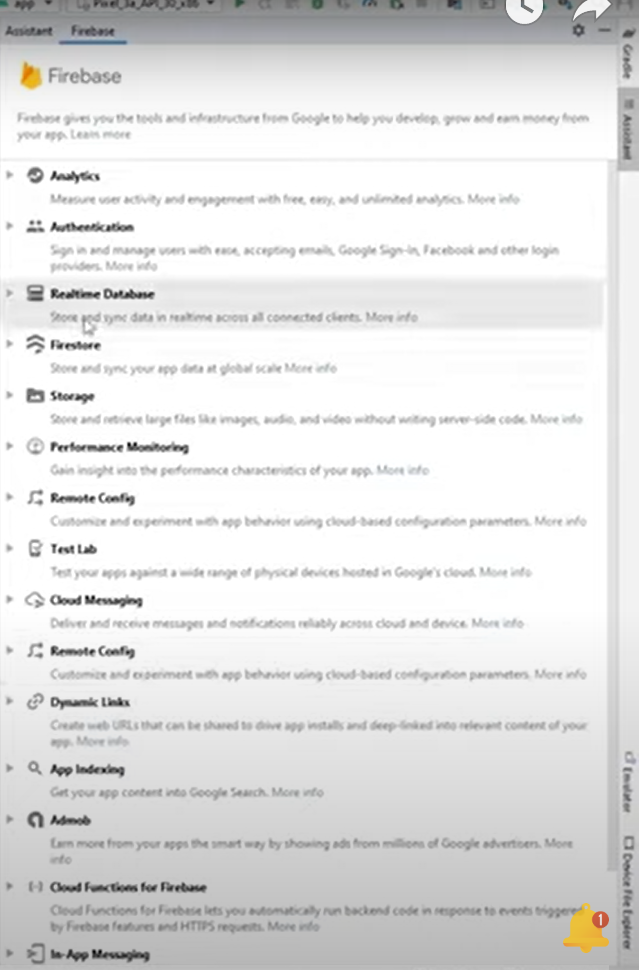
**Iremos manipular as entradas dos dados com o android:id de cada componente.**



**2.1.3.Integração do projeto no Android Studio com o Firebase Authentication**

O login vai estar hospedado na nuvem, e não no aplicativo.

**Para integrar com o Firebase, vá dentro do projeto no Android Studio, guia Tools, Firebase. Abre a aba Firebase, com todos os recursos que a plataforma Firebase nos oferece. Vamos conectar com o serviço Authentication, com email e senha (email and password authentication).**



Para isso:

1. Logue no site do Firebase, veja que a aba Firebase Console já está na url o packageName=”br.com.areatechbrasil.androidfirebaseprojetc” e o nome do projeto para o FIrebase é projectDisplayName=”AndroidFirebaseProjetc”

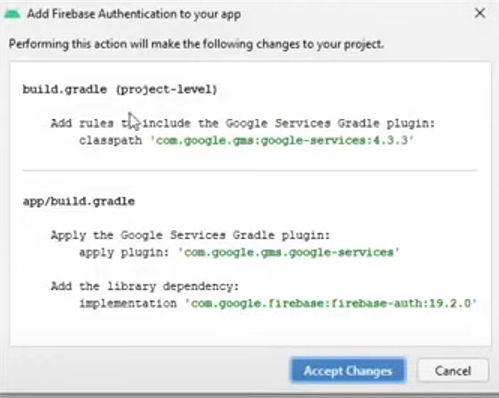
2. Clique em Criar um novo projeto, coloque como nome AndroidFirebaseProjetc. Clique em continuar.

3. Mantenha ativado o Google Analytics. Qualquer problema ou crédito que o app venha a dar, ele pode ser resolvido pelo relatório do Google Analytics. Ele vem ativo como padrão.

4. Configure o Google Analytics, deixe a localização do centro de dados como Brasil (o padrão é o EUA); deixe habilitado a opção de usar as configurações padrão do Google Analytics. Deixe ativo a opção de aceitas os termos de proteção, e aceite os termos do Google Analytics. Clique no botão criar projeto. Ele cria uma instância no datacenter da Google. O projeto foi criado. Vai aparecer a tela com os dizeres:”Seu app Android foi criado no Firebase”. Clique em conectar, ele vai conectar localmente o Android Studio com a instância da nuvem.

5. Vai abrir uma guia com localhost:52707; com os dizeres que o projeto do Android Studio foi conectado com a aplicação do Firebase.

6. Volte no projeto do Android Studio, pela tela do Assistente do Firebase, guia Firebase. Siga a instrução de adicionar as dependências de autenticação para o seu app. Clique no botão “Add Firebase Authentication to your app”. Ele vai abrir a janela com os códigos:



O google service usa a conta do Google para manipular o Android Studio. E, também, o Firebase Auth é o serviço de autenticação que usa como opções de login: Gmail, Facebook, Twitter, Instagram, GitHub, login anônimo, login personalizado e afins. Na janela dos códigos de autenticação, clique em Accept Changes.

Feche o assistente.

Vá na pasta Gradle Scripts, build.gradle (Module:AndroidFirebaseProjetc.app), dê dois cliques para abrir. Confira que nas dependências foi colocada a implementation ‘com.google.firebase:firebase-auth:19.2.0’, assim como na parte de plugin. O mesmo ocorre com o build.gradle a nível de projeto.

7. Vá na página do Firebase, dê um F5, e clique no retângulo com nome AndroidFirebaseProjetc. Vá no menu lateral com nome de Criação, e clique em Authentication. Clique em primeiros passos. Depois clique em Sign in method. Deixe ativo apenas o Email e senha, clicando na caneta, depois na opção de ativar. Após, clique no botão Salvar.

**2.1.4. Serviços que a plataforma Firebase oferece**

Além do **Authentication** que serve para autenticar os usuários e controlar a sessão; há o **Cloud Firestore.**

O **Cloud Firestore** é para usar o banco de dados por documentos; é como se tivesse uma pasta que é a coleção, e dentro da coleção tem documentos. Cada documento é um registro. Cada coleção é uma tabela.

Há o banco de dados **Realtime Database.** Ele é criado no JSON e cada tabela seria um nó.

No **Storage** é para fazer o armazenamento. Ele armazena e recupera arquivos gerados pelo usuário, como imagens, áudio e vídeos sem o código do servidor. No projeto, usaremos uma imagem de perfil.

No **Hosting** é usado quando se deseja criar um site. Ele implementa apps da Web para dispositivos móveis em segundos usando uma rede de fornecimento de contéudo global segura. Exemplo, vou criar um projeto web vou hospedar no hosting.

No **Functions** ele executa um código de backend para dispositivos móveis sem gerenciar servidores. Por exemplo, um Web Service precisaria estar integrando no seu projeto. Pode criar um WebService do paypal; na function cria um checkout e ele estaria mandando a requisição para verificar o cartão de crédito, se a compra foi aprovada ou não e afins.

No **Machine Learning** é um aprendizado de máquina. Ele resolve problemas comuns nos seus aplicativos com o aprendizado de máquina.

Ao clicar no ícone de casa, tem a **visão geral do projeto.** Tem o resumo do projeto e os ícones de cada serviço. Em configurações, há o pacote que foi criado o projeto no Android Studio que abre a configuração de SDK. É nessa configuração que posso remover o dispositivo (remover o projeto atual do Android). Ele lista todos os aplicativos que estão listados nesse projeto. Por exemplo, se tivesse um iOS, outro aplicativo do Android. Um aplicativo com versão servidor e outra versão de cliente, mas o banco de dados das duas versões seria o mesmo. Pode integrar com aplicativo web. (Html, angular, react). Mais para cima, ele mostra a chave, nome do projeto, nome do pacote e etc. Se quiser remover o projeto, existe o botão excluir projeto.

**2.1.5. Mudando o Android Manifest e Main Activity**

Teste o aplicativo no emulador. Use como emulador o Pixel 3a API 30. Ao executar o projeto, ela está com a tela padrão onde há o retângulo roxo e o hello world. Isso ocorre porque o arquivo AndroidManifest.xml não está atualizado. Vá na pasta app, na pasta manifests, arquivo: AndroidManifest.xml

**ARQUIVO: AndroidManifest.xml**

<? xml version=”1.0” enconing=”utf-8”/>

<manifest xmlns:android=”<http://schemas.android.com/apk/res/android> “

package =”br.com.areatechbrasil.androidFirebaseprojetc”>

<application

android:allowbackup=”true”

android:icon=”@mipmap/ic\_launcher”

android:label=”AndroidFirebaseProjetc”

android:roundIcon=”@mipmap/ic\_launcher\_round”

android:suportsRtl=”true”

android:theme=”@style/Theme\_AndroidFirebaseProjetc”>

<activity android:name=”.Activity.Login”></activity>

<activity android:name=”.MainActivity”>

<Intent-filter>

<action android:name=”android.intent.action.MAIN/>

<category android:name=”android.intent.category.LAUNCHER” />

</intente-filter>

</activity>

</application>

</manifest>

A tela de login foi criada na pasta Activity com o arquivo Login.java

**Para resolver, vá na pasta app, pasta java, pasta br.com.areatechbrasil.androidFirebaseprojetc, arquivo: MainActivity.java**

**O Firebase Auth é responsável por ver se tem uma sessão de autenticação aberta e se está fazendo a autenticação ou o logout dessa sessão.**

**ARQUIVO: MainActivity.java**

public class MainActivity extends AppCompatActivity {

private FirebaseAuth mAuth;

@override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

}

// Primeira ação que será executada quando iniciar o aplicativo é o método onStart, ou seja, se ninguém logou vai pra tela de login

protected void onStart(){

super.onStart();

// pega o primeiro método de getInstance, corrige manualmente o tipo de currentuser para FirebaseUser

FirebaseAuth currentuser = FirebaseAuth.getInstance().getCurrentUser();

If(currentuser == null){

// O new Intent passa como parâmetros: a activity atual, a activity que eu quero ir

Intent intent = new Intent(MainActivity.this, Login.class);

startActivity(intent);

// Para finalizar a Main Activity

finish();

}

}

}

}

Rodou no emulador e foi pra imagem da tela de login.

**2.1.6. Colocando ação no botão de Login**

Atualmente, a tela de login está estática e os botões sem ação. Vá no arquivo: Login.java

**ARQUIVO: Login.java**

public class Login extends AppCompatActivity{

// mesmos nomes capturados no activity\_login.xml

private EditText edt\_email;

private EditText edt\_senha;

private Button btn\_login;

private Button btn\_registrar;

private FirebaseAuth mAuth;

private ProgressBar loginProgressBar;

private CheckBox ckb\_mostrar\_senha;

@Override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

// Relacionar as variáveis com os componentes dos ids; findViewById pega do layout

mAuth = firebaseAuth.getInstance();

edt\_email = findViewById(R.id.edt\_email);

edt\_senha = findViewById(R.id.edt\_senha);

btn\_login = findViewById(R.id.btn\_login);

btn\_registrar = findViewById(R.id.btn\_registrar);

loginProgressBar = findViewById(R.id.loginProgressBar);

ckb\_mostrar\_senha = findViewById(R.id.ckb\_mostrar\_senha);

// Vamos capturar a ação do botão login, que ao ser clicado ele deve executar uma ação, um método

btn\_login.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick(view v){

// captura email e senha com os dados de entrada

String loginEmail = edt\_email.getText().toString();

String loginSenha = edt\_senha.getText().toString();

// Valida se o usuário vai clicar no login sem digitar email e senha

if (!TextUtils.isEmpty(loginEmail) || !TextUtils.isEmpty(loginSenha)){

// só vai ser exibido após preencher os dados e clicar no botão login

loginProgressBar.setVisibility(View.VISIBLE);

// vamos consultar o firebase Auth pra ver se o usuário existe

mAuth.signInWithEmailAndPassword(loginEmail, loginSenha)

// se o mAuth achou o registro

.addOnCompleteListener(new OnCompleteListener<AuthResult>){

@Override

public void onComplete(@NonNull Task<AuthResult> task){

if (task.isSuccesful()){

// vamos voltar pra main activity

abrirTelaPrincipal();

}else{

String error = task.getException().getMessage();

Toast.makeText(Login.this, “” +error, Toast.LENGTH\_SHORT).show();

loginProgressBar.setVisibility(view.INVISIBLE);

}

}

});

}

}

});

}

private void abrirTelaPrincipal(){

Intent intent = new Intent (Login.this, MainActivity.class);

startActivity(intent);

// finaliza a tela de login

finish();

}

}

}

Primeiro, vamos abrir o activity\_login.xml para capturar os nomes dos ids e dentro do arquivo: login.java vamos criar uma variável para cada id, que facilita no processamento dos dados.

**2.1.7. Colocando ação no botão de logout e ativando o checkbox**

Vá no site do Firebase, clique em Authentication e confira que a lista está vazia. Por isso, para logar precisa inserir os usuários no banco de dados. Em primeiro plano, vai colocar os usuários manualmente no Authentication.

Email: [teste@teste.com](mailto:teste@teste.com) e senha: 123456 e depois clique em adicionar usuário. O usuário foi criado, porém não foi conectado.

**Se fechar o aplicativo e abrir novamente, ele continua logado. Ele vai direto pra MainActivity, devido ao método onStart da MainActivity. Ele mostra na página do Authentication a hora e dia de última conexão.**

Vamos colocar uma ação no checkbox e fazer um botão de logout.

**Vá na MainActivity.java**

**ARQUIVO: MainActivity.java**

public class MainActivity extends AppCompatActivity {

private FirebaseAuth mAuth;

private Button btn\_logout;

@override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

mAuth = firebaseAuth.getInstance();

btn\_logout = findViewById(R.id.btn\_logout);

btn\_logout.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick(View v){

mAuth.signOut();

Intent intent = new Intent(MainActivity.this,Login.class);

// vai pra tela de login

startActivity(intent);

// finaliza a tela principal

finish();

}

});

}

// Primeira ação que será executada quando iniciar o aplicativo é o método onStart, ou seja, se ninguém logou vai pra tela de login

protected void onStart(){

super.onStart();

// pega o primeiro método de getInstance, corrige manualmente o tipo de currentuser para FirebaseUser

FirebaseAuth currentuser = FirebaseAuth.getInstance().getCurrentUser();

If(currentuser == null){

// O new Intent passa como parâmetros: a activity atual, a activity que eu quero ir

Intent intent = new Intent(MainActivity.this, Login.class);

startActivity(intent);

// Para finalizar a Main Activity

finish();

}

}

}

}

**Vamos no arquivo de layout da main, a activity\_main.xml**

**ARQUIVO: Activity\_main.xml**

// Colocar em modo split, layout é o Linear Layout

<? xml version=”1.0” enconding =”utf-8”?>

<LinearLayout.widget.ConstraintLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”http:// schemas.android.com/apk/res-auto”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

android:gravity=”center”

android:orientation=”vertical”

tools:context=”.Activity.Login”>

<Button

Android:id=”@id/btn\_logout”

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:text=”logout”

/>

</LinearLayout>

**O uso do checkbox é para mostrar a senha quando o usuário criou senhas complexas com letras, números e arroba. Sua ação fica na tela de login.**

**Vá para tela de login.**

**ARQUIVO: Login.java**

public class Login extends AppCompatActivity{

// mesmos nomes capturados no activity\_login.xml

private EditText edt\_email;

private EditText edt\_senha;

private Button btn\_login;

private Button btn\_registrar;

private FirebaseAuth mAuth;

private ProgressBar loginProgressBar;

private CheckBox ckb\_mostrar\_senha;

@Override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

// Relacionar as variáveis com os componentes dos ids; findViewById pega do layout

mAuth = firebaseAuth.getInstance();

edt\_email = findViewById(R.id.edt\_email);

edt\_senha = findViewById(R.id.edt\_senha);

btn\_login = findViewById(R.id.btn\_login);

btn\_registrar = findViewById(R.id.btn\_registrar);

loginProgressBar = findViewById(R.id.loginProgressBar);

ckb\_mostrar\_senha = findViewById(R.id.ckb\_mostrar\_senha);

// Vamos capturar a ação do botão login, que ao ser clicado ele deve executar uma ação, um método

btn\_login.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick(view v){

// captura email e senha com os dados de entrada

String loginEmail = edt\_email.getText().toString();

String loginSenha = edt\_senha.getText().toString();

// Valida se o usuário vai clicar no login sem digitar email e senha

if (!TextUtils.isEmpty(loginEmail) || !TextUtils.isEmpty(loginSenha)){

// só vai ser exibido após preencher os dados e clicar no botão login

loginProgressBar.setVisibility(View.VISIBLE);

// vamos consultar o firebase Auth pra ver se o usuário existe

mAuth.signInWithEmailAndPassword(loginEmail, loginSenha)

// se o mAuth achou o registro

.addOnCompleteListener(new OnCompleteListener<AuthResult>){

@Override

public void onComplete(@NonNull Task<AuthResult> task){

if (task.isSuccesful()){

// vamos voltar pra main activity

abrirTelaPrincipal();

}else{

String error = task.getException().getMessage();

Toast.makeText(Login.this, “” +error, Toast.LENGTH\_SHORT).show();

loginProgressBar.setVisibility(view.INVISIBLE);

}

}

});

}

}

});

// Ação do checkbox

ckb\_mostrar\_senha.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() ){

@Override

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked){

if (isChecked){

// ele vai chamar a classe hidereturns para tirar a máscara

edt\_senha.setTransformationMethod(HideReturnsTransformationMethod.getInstance())

} else{

// mantém a senha com máscara

edt\_senha.setContentTransitionManager(PasswordTransformationMethod.getInstance());

}

}

});

}

private void abrirTelaPrincipal(){

Intent intent = new Intent (Login.this, MainActivity.class);

startActivity(intent);

// finaliza a tela de login

finish();

}

}

}

**2.1.8.Criando a tela de registro e botão registrar ativado**

O aplicativo vai criar o registro. Vá na pasta app, pasta java, pasta br.com.androidManifest.androidFirebaseprojetc, pasta Activity. Na pasta Activity, clica com o botão direito, new Activity, empty Activity. Nome: Register.

**Arquivo: Register.java**

**Dá alt+ enter para importações**

public class Register extends AppCompatActivity{

private FirebaseAuth mAuth;

private EditText edt\_email\_register;

private EditText edt\_senha\_register;

private EditText edt\_confirmar\_senha\_register;

private CheckBox ckb\_mostrar\_senha\_register;

private Button btn\_registrar\_register;

private Button btn\_login\_register;

private ProgressBar loginProgressBar\_register;

@Override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

mAuth = firebaseAuth.getInstance();

edt\_email\_register = findViewById(R.id.edt\_email\_register);

edt\_senha\_register = findViewById(R.id.edt\_senha\_register);

edt\_confirmar\_senha\_register = findViewById(R.id.edt\_confirmar\_senha\_register);

btn\_registrar\_register = findViewById(R.id.btn\_registrar\_register);

btn\_login\_register = findViewById(R.id.btn\_login\_register);

loginProgressBar\_register = findViewById(R.id.loginProgressBar\_register);

ckb\_mostrar\_senha\_register = findViewById(R.id.ckb\_mostrar\_senha\_register);

// Ação do checkbox

ckb\_mostrar\_senha\_register.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() ){

@Override

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked){

if (isChecked){

// ele vai chamar a classe hidereturns para tirar a máscara

edt\_senha\_register.setTransformationMethod(HideReturnsTransformationMethod.getInstance())

edt\_confirmar\_senha\_register.setTransformationMethod(HideReturnsTransformationMethod.getInstance())

} else{

// mantém a senha com máscara

edt\_senha\_register.setContentTransitionManager(PasswordTransformationMethod.getInstance());

edt\_cofirmar\_senha\_register.setContentTransitionManager(PasswordTransformationMethod.getInstance());

}

}

});

// Ação do botão registrar

btn\_registrar\_register.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick (View v){

String registerEmail = edt\_email\_register.getText().toString();

String senha = edt\_senha\_register.getText().toString();

String confirmarSenha = edt\_confirmar\_senha\_register.getText().toString();

if(!(TextUtils.toEmpty(registerEmail) || TextUtils.toEmpty(senha) || TextUtils.toEmpty(confirmarSenha))){

if(senha.equals(confirmarSenha)){

LoginProgressBar\_register.setVisibility(View.VISIBLE);

mAuth.createUserWithEmailAndPassword(registerEmail,senha).addOnCompleteListener(new OnCompleteListener<AuthResult>(){

@Override

public void onComplete(@NonNull Task<AuthResult> task){

if (task.isSuccesful()){

abrirTelaPrincipal();

}else{

String error = task.getException().getMessage();

Toast.makeText(Register.this, “”+error, Toast.LENGTH\_SHORT).show();

}

loginProgressBar\_register.setVisibility(View.INVISIBLE);

}

});

}else{

Toast.makeText(Register.this, “A senha deve ser a mesma em ambos os cmapos!”, Toast.LENGTH\_SHORT).show();

}

}

}

});

btn\_login\_register.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick (View v){

Intent intent = new Intent (Register.this, Login.class);

startActivity(intent);

finish();

}

});

}

private void abrirTelaPrincipal(){

Intent intent = new Intent (Register.this, MainActivity.class);

startActivity(intent);

// finaliza a tela de login

finish();

}

}

**Precisamos desenhar a tela de registro, com o arquivo: activity\_registro.xml. Ele vai ser bem parecido com a tela de login no layout. Para identação, pressione Ctrl+A e depois Ctrl+Alt+L, e o id sempre vem primeiro.**

**ARQUIVO:activity\_register.xml**

<? xml version=”1.0” enconding =”utf-8”?>

<LinearLayout.widget.ConstraintLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”http:// schemas.android.com/apk/res-auto”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

android:orientation=”vertical”

android:gravity=”center”

android:background=”@drawable/background”

tools:context=”.Activity.Register”>

<TextView

android:layout\_width=”match\_parent”

android:layout\_height=”wrap\_ content”

android:text=”Criar conta”

android:textColor=”@color/white”

android:textSize=”40sp”

android:layout\_marginBottom=”120dp” // margem pra baixo

android:textAlignment=”center”

/>

// Pro usuário colocar o E-mail

<EditText

android:id=”@+id/edt\_email\_register”

android:layout\_width=”1800dp”

android:layout\_height=”wrap\_ content”

android:hint=”E-mail”

android:inputType=”textEmailAddress”

android:textColorHint=”@color/white”

android:textColor=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

/>

// Para o usuário colocar a Senha

<EditText

android:id=”@+id/edt\_senha\_register”

android:layout\_width=”300dp”

android:layout\_height=”wrap\_content”

android:hint=”Senha”

android:inputType=”textPassword” // é a senha mascarada com bolinhas

android:textColorHint=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

android:layout\_marginTop=”10dp”

android:textColor=”@color/white”

/>

<EditText

android:id=”@+id/edt\_confirmar\_senha\_register”

android:layout\_width=”300dp”

android:layout\_height=”wrap\_content”

android:hint=”Redigite a Senha”

android:inputType=”textPassword” // é a senha mascarada com bolinhas

android:textColorHint=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

android:layout\_marginTop=”10dp”

android:textColor=”@color/white”

/>

// Para dar a opção de mostrar a senha

<CheckBox

android:id=”@+id/ckb\_mostrar\_senha\_register”

android:layout\_width=”200dp”

android:layout\_height=”wrap\_ content”

android:text=”Mostrar senha”

android:textColor =”@color/white”

android:buttonInt=”@color/white” // deixar o quadrado branco

android:layout\_marginRight=”70dp”

android:layout\_marginTop=”10dp”

/>

// Botão para registrar

<Button

android:id=”@+id/btn\_registrar\_register”

android:layout\_width=”380dp”

android:layout\_height=”wrap\_ content”

android:background=”@color/white”

android:textColor=”@android:color/holo\_red\_light”

android:text=”Registrar”

android:layout\_marginTop=”20dp”

/>

// Botão para logar

<Button

android:id=”@+id/btn\_login\_register”

android:layout\_width=”380dp”

android:layout\_height=”wrap\_ content”

android:background=”#30FFFFFF”

android:textColor=”@color/white”

android:text=”Login”

android:layout\_marginTop=”20dp”

/>

// Barra de carregando. O padrão é um círculo em azul anil carregando

<ProgressBar

android:id=”@+id/loginProgressBar\_register”

android:layout\_width=”match\_parent”

android:layout\_height=”wrap\_ content”

android:indeterminateTint=”@color/white” // cor da barra carregando

android:indeterminate=”true”

style=”@style/ProgressBar\_Horizontal” // Para deixar uma barra horizontal

android:visibility=”invisible” // só vai ficar visível quando o usuário tocar no botão login

android:layout\_marginTop=”20dp”

</LinearLayout>

Faça o teste com o emulador.

**2.1.9 Criação de perfil**

Vai usar o banco de dados Realtime Database para cadastrar dados de perfil. Veja a documentação com os primeiros passos. Ele é um banco de dados noSQL, armazenado na nuvem. Tem saída Json.

Geralmente quando se usa o banco de dados em mySQL, constrói uma API para que a saída de dados seja Json. O banco de dados é compatível com a maioria das plataformas. A integração com o Android é mais simplificada. Armazena um cache local.

Para sistema de compras, as vezes só tem internet em um local. As tratativas tem que ser automáticas, se aumentar o preço não faz a compra. Por exemplo, o programa de mensagens, tem o chat. Dá para escalonar o Banco de Dados, tem vários bancos de dados em diferentes países. O Realtime Database tem persistência de dados.

Voltando para o Android Studio, o tema da aula é criar um usuário no aplicativo; mas esse usuário está sendo atribuído na autenticação. Não tem persistência de dados com o Realtime Database. Vamos supor que eu criei o usuário, mas quero colocar outros dados além do email e senha, como nome, telefone, cpf, foto, endereço e afins. Essas outras informações exigem o uso de um banco de dados, como o Realtime Database.

Dentro do site do Firebase, clica em Realtime Database, clica no botão “Criar banco de dados”. Datacenter no Estados Unidos. Avançar. No modo bloqueado eu não tenho permissão pra ler e gravar. O modo teste é onde se tem 30 dias de teste, após isso ele vai cobrar pelo excesso de dados.

Vamos deixar no modo bloqueado, pois vamos alterar as regras manualmente. Clica em ativar, e espera o banco ser criado. Criou a estrutura nula. Poderia criar manualmente, no formato chave-valor. userId, nome, sobrenome. O userId é conhecido como nó; e o que vem dentro dele são os atributos. Depois clique em Adicionar. Pode importar o json ou exportando. Usa o editor VSCode. Tem como configurar. Do jeito que está, não conseguiria armazenar; por isso que tem que mexer nas regras. O aplicativo tem que estar atribuído neste projeto. Mesmo sem autenticação, ele conseguiria gravar.

Vá na parte de Regras

{

“rules”{

“root”: “auth” is null

“write”: “auth” is null

}

}

Dá Ctrl+S para salvar.

No nosso aplicativo, vamos compor o nome e sobrenome.

Vá no arquivo: **activity\_register.xml**

**ARQUIVO: Activity\_register.xml**

<? xml version=”1.0” enconding =”utf-8”?>

<LinearLayout.widget.ConstraintLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”http:// schemas.android.com/apk/res-auto”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

android:orientation=”vertical”

android:gravity=”center”

android:background=”@drawable/background”

tools:context=”.Activity.Register”>

<TextView

android:layout\_width=”match\_parent”

android:layout\_height=”wrap\_ content”

android:text=”Criar conta”

android:textColor=”@color/white”

android:textSize=”40sp”

android:layout\_marginBottom=”120dp” // margem pra baixo

android:textAlignment=”center”

/>

// Pro usuário colocar o nome

<EditText

android:id=”@+id/edt\_nome\_register”

android:layout\_width=”1800dp”

android:layout\_height=”wrap\_ content”

android:hint=”Nome”

android:layout\_marginTop=”10dp”

android:inputType=”textPersonName”

android:textColorHint=”@color/white”

android:textColor=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

/>

// Pro usuário colocar o sobrenome

<EditText

android:id=”@+id/edt\_sobrenome\_register”

android:layout\_width=”1800dp”

android:layout\_height=”wrap\_ content”

android:hint=”Sobrenome”

android:layout\_marginTop=”10dp”

android:inputType=” textPersonName”

android:textColorHint=”@color/white”

android:textColor=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

/>

// Pro usuário colocar o E-mail

<EditText

android:id=”@+id/edt\_email\_register”

android:layout\_width=”1800dp”

android:layout\_height=”wrap\_ content”

android:hint=”E-mail”

android:inputType=”textEmailAddress”

android:textColorHint=”@color/white”

android:textColor=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

/>

// Para o usuário colocar a Senha

<EditText

android:id=”@+id/edt\_senha\_register”

android:layout\_width=”300dp”

android:layout\_height=”wrap\_content”

android:hint=”Senha”

android:inputType=”textPassword” // é a senha mascarada com bolinhas

android:textColorHint=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

android:layout\_marginTop=”10dp”

android:textColor=”@color/white”

/>

<EditText

android:id=”@+id/edt\_confirmar\_senha\_register”

android:layout\_width=”300dp”

android:layout\_height=”wrap\_content”

android:hint=”Redigite a Senha”

android:inputType=”textPassword” // é a senha mascarada com bolinhas

android:textColorHint=”@color/white”

android:backgroundTint=”@color/white” // é a cor debaixo da escrita de E-mail

android:layout\_marginTop=”10dp”

android:textColor=”@color/white”

/>

// Para dar a opção de mostrar a senha

<CheckBox

android:id=”@+id/ckb\_mostrar\_senha\_register”

android:layout\_width=”200dp”

android:layout\_height=”wrap\_ content”

android:text=”Mostrar senha”

android:textColor =”@color/white”

android:buttonInt=”@color/white” // deixar o quadrado branco

android:layout\_marginRight=”70dp”

android:layout\_marginTop=”10dp”

/>

// Botão para registrar

<Button

android:id=”@+id/btn\_registrar\_register”

android:layout\_width=”380dp”

android:layout\_height=”wrap\_ content”

android:background=”@color/white”

android:textColor=”@android:color/holo\_red\_light”

android:text=”Registrar”

android:layout\_marginTop=”20dp”

/>

// Botão para logar

<Button

android:id=”@+id/btn\_login\_register”

android:layout\_width=”380dp”

android:layout\_height=”wrap\_ content”

android:background=”#30FFFFFF”

android:textColor=”@color/white”

android:text=”Login”

android:layout\_marginTop=”20dp”

/>

// Barra de carregando. O padrão é um círculo em azul anil carregando

<ProgressBar

android:id=”@+id/loginProgressBar\_register”

android:layout\_width=”match\_parent”

android:layout\_height=”wrap\_ content”

android:indeterminateTint=”@color/white” // cor da barra carregando

android:indeterminate=”true”

style=”@style/ProgressBar\_Horizontal” // Para deixar uma barra horizontal

android:visibility=”invisible” // só vai ficar visível quando o usuário tocar no botão login

android:layout\_marginTop=”20dp”

</LinearLayout>

**Agora vamos criar um modelo de usuário. Vá na pasta app, pasta java, pasta br.com.areatechbrasil.firebase.projetc. Clique com o botão direito na pasta br.com.areatechbrasil.firebase.projetc, new, package, nome: Model. Na pasta Model, clique com o botão direito, new, class, UserModel.**

**Com o arquivo: UserModel.java aberto, teremos que:**

**ARQUIVO: UserModel.java**

public class UserModel{

private String id;

private String nome;

private String sobrenome;

private String email;

// Para criar o construtor vazio, pressione alt+insert, constructor.Select nome

public UserModel(){

}

public UserModel (String id, String nome, String sobrenome, String email){

this.id = id;

this.nome = nome;

this.sobrenome = sobrenome;

this.email = email;

}

// Getters

public String getId(){

return id;

}

public String getNome(){

return nome;

}

public String getSobrenome(){

return sobrenome;

}

public String getEmail(){

return email;

}

// Setters

public void setId(String id){

this.id = id;

}

public void setNome(String nome){

this.nome = nome;

}

public void setSobrenome(String sobrenome){

this.sobrenome = sobrenome;

}

public void setEmail(String email){

this.email = email;

}

// Método para salvar no Firebase

public void salvar(){

// inicia a instância do Firebase e coloca a referência do banco de dados, que seria o caminho pra chegar no banco.

DatabaseReference reference = FirebaseDatabase.getInstance().getReference();

// criando o nó, que seria um campo

reference.child(“Usuarios”).child(getId()).setValue(this);

}

}

**Tem que adicionar as dependências do Realtime Database no projeto.Vou em Tools, Firebase, Realtime Database. Save and retrieve data. Clique em Add the Realtime Database to our app.**

**No build.gradle a nível de projeto:** classpath: ‘com.google.gms:google-services-4.3.4‘

**No build.gradle a nível de app:** classpath:’com.google.firebase:firebase-database-19.5.1’

Clica em Accept Changes. Aguarda a sincronização. Ambos aparecem como implementation ao invés de classpath. Ambos dentro do setor de dependencies.

Vamos abrir a classe Register.

**Arquivo: Register.java**

public class Register extends AppCompatActivity{

private FirebaseAuth mAuth;

private EditText edt\_nome\_register;

private EditText edt\_sobrenome\_register;

private EditText edt\_email\_register;

private EditText edt\_senha\_register;

private EditText edt\_confirmar\_senha\_register;

private CheckBox ckb\_mostrar\_senha\_register;

private Button btn\_registrar\_register;

private Button btn\_login\_register;

private ProgressBar loginProgressBar\_register;

@Override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

mAuth = firebaseAuth.getInstance();

edt\_nome\_register = findViewById(R.id.edt\_nome\_register);

edt\_sobrenome\_register = findViewById(R.id.edt\_sobrenome\_register);

edt\_email\_register = findViewById(R.id.edt\_email\_register);

edt\_senha\_register = findViewById(R.id.edt\_senha\_register);

edt\_confirmar\_senha\_register = findViewById(R.id.edt\_confirmar\_senha\_register);

btn\_registrar\_register = findViewById(R.id.btn\_registrar\_register);

btn\_login\_register = findViewById(R.id.btn\_login\_register);

loginProgressBar\_register = findViewById(R.id.loginProgressBar\_register);

ckb\_mostrar\_senha\_register = findViewById(R.id.ckb\_mostrar\_senha\_register);

// Ação do checkbox

ckb\_mostrar\_senha\_register.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() ){

@Override

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked){

if (isChecked){

// ele vai chamar a classe hidereturns para tirar a máscara

edt\_senha\_register.setTransformationMethod(HideReturnsTransformationMethod.getInstance())

edt\_confirmar\_senha\_register.setTransformationMethod(HideReturnsTransformationMethod.getInstance())

} else{

// mantém a senha com máscara

edt\_senha\_register.setContentTransitionManager(PasswordTransformationMethod.getInstance());

edt\_cofirmar\_senha\_register.setContentTransitionManager(PasswordTransformationMethod.getInstance());

}

}

});

// Ação do botão registrar

btn\_registrar\_register.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick (View v){

// Instanciando a classe UserModel

UserModel userModel = new UserModel();

userModel.setEmail = edt\_email\_register.getText().toString();

userModel.setNome = edt\_nome\_register.getText().toString();

userModel.setSobrenome = edt\_sobrenome\_register.getText().toString();

String senha = edt\_senha\_register.getText().toString();

String confirmarSenha = edt\_confirmar\_senha\_register.getText().toString();

if(!TextUtils.isEmpty(userModel.getNome()) || !TextUtils.isEmpty(userModel.getSobrenome())||

(!TextUtils.isEmpty(userModel.getEmail()) || !TextUtils.isEmpty(senha) || !TextUtils.isEmpty(confirmarSenha)){

if(senha.equals(confirmarSenha)){

LoginProgressBar\_register.setVisibility(View.VISIBLE);

mAuth.createUserWithEmailAndPassword(userModel.getEmail(),senha).addOnCompleteListener(new OnCompleteListener<AuthResult>(){

@Override

public void onComplete(@NonNull Task<AuthResult> task){

if (task.isSuccesful()){

// Para capturar o Id e salvar no Realtime Database. Excluir no Realtime Database NÃO exclui no Authentication porque é independente.

userModel.setId(mAuth.getUId());

userModel.salvar();

abrirTelaPrincipal();

}else{

String error = task.getException().getMessage();

Toast.makeText(Register.this, “”+error, Toast.LENGTH\_SHORT).show();

}

loginProgressBar\_register.setVisibility(View.INVISIBLE);

}

});

}else{

Toast.makeText(Register.this, “A senha deve ser a mesma em ambos os cmapos!”, Toast.LENGTH\_SHORT).show();

}

}

}

});

btn\_login\_register.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick (View v){

Intent intent = new Intent (Register.this, Login.class);

startActivity(intent);

finish();

}

});

}

private void abrirTelaPrincipal(){

Intent intent = new Intent (Register.this, MainActivity.class);

startActivity(intent);

// finaliza a tela de login

finish();

}

}

**2.1.10 Tratativa de erros de cadastro**

A classe mAuth tem uma tratativa padrão, porém em inglês. Temos que traduzir pro português. Vamos usar o arquivo: Register.java

**ARQUIVO: Register.java**

public class Register extends AppCompatActivity{

private FirebaseAuth mAuth;

private EditText edt\_nome\_register;

private EditText edt\_sobrenome\_register;

private EditText edt\_email\_register;

private EditText edt\_senha\_register;

private EditText edt\_confirmar\_senha\_register;

private CheckBox ckb\_mostrar\_senha\_register;

private Button btn\_registrar\_register;

private Button btn\_login\_register;

private ProgressBar loginProgressBar\_register;

@Override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

mAuth = firebaseAuth.getInstance();

edt\_nome\_register = findViewById(R.id.edt\_nome\_register);

edt\_sobrenome\_register = findViewById(R.id.edt\_sobrenome\_register);

edt\_email\_register = findViewById(R.id.edt\_email\_register);

edt\_senha\_register = findViewById(R.id.edt\_senha\_register);

edt\_confirmar\_senha\_register = findViewById(R.id.edt\_confirmar\_senha\_register);

btn\_registrar\_register = findViewById(R.id.btn\_registrar\_register);

btn\_login\_register = findViewById(R.id.btn\_login\_register);

loginProgressBar\_register = findViewById(R.id.loginProgressBar\_register);

ckb\_mostrar\_senha\_register = findViewById(R.id.ckb\_mostrar\_senha\_register);

// Ação do checkbox

ckb\_mostrar\_senha\_register.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() ){

@Override

public void onCheckedChanged(CompoundButton buttonView, boolean isChecked){

if (isChecked){

// ele vai chamar a classe hidereturns para tirar a máscara

edt\_senha\_register.setTransformationMethod(HideReturnsTransformationMethod.getInstance())

edt\_confirmar\_senha\_register.setTransformationMethod(HideReturnsTransformationMethod.getInstance())

} else{

// mantém a senha com máscara

edt\_senha\_register.setContentTransitionManager(PasswordTransformationMethod.getInstance());

edt\_cofirmar\_senha\_register.setContentTransitionManager(PasswordTransformationMethod.getInstance());

}

}

});

// Ação do botão registrar

btn\_registrar\_register.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick (View v){

// Instanciando a classe UserModel

UserModel userModel = new UserModel();

userModel.setEmail = edt\_email\_register.getText().toString();

userModel.setNome = edt\_nome\_register.getText().toString();

userModel.setSobrenome = edt\_sobrenome\_register.getText().toString();

String senha = edt\_senha\_register.getText().toString();

String confirmarSenha = edt\_confirmar\_senha\_register.getText().toString();

if(!TextUtils.isEmpty(userModel.getNome()) && !TextUtils.isEmpty(userModel.getSobrenome()) &&

(!TextUtils.isEmpty(userModel.getEmail()) && !TextUtils.isEmpty(senha) && !TextUtils.isEmpty(confirmarSenha)){

if(senha.equals(confirmarSenha)){

LoginProgressBar\_register.setVisibility(View.VISIBLE);

mAuth.createUserWithEmailAndPassword(userModel.getEmail(),senha).addOnCompleteListener(new OnCompleteListener<AuthResult>(){

@Override

public void onComplete(@NonNull Task<AuthResult> task){

if (task.isSuccesful()){

// Para capturar o Id e salvar no Realtime Database. Excluir no Realtime Database NÃO exclui no Authentication porque é independente.

userModel.setId(mAuth.getUId());

userModel.salvar();

abrirTelaPrincipal();

}else{

String error;

try{

throw task.getException();

}catch(FirebaseAuthWeakPasswordException e ){

error = “A senha deve conter no mínimo 6 caracteres”;

}catch(FirebaseAuthInvalidCredentialException e){

error = “E-mail inválido”;

}catch(FirebaseAuthUserCollisionException e){

error = “E-mail já existe”;

}catch(Exception e){

error = “Erro ao efetuar o cadastro”;

e.printStrackTrace();

}

Toast.makeText(Register.this, error, Toast.LENGTH\_SHORT).show();

}

loginProgressBar\_register.setVisibility(View.INVISIBLE);

}

});

}else{

Toast.makeText(Register.this, “A senha deve ser a mesma em ambos os cmapos!”, Toast.LENGTH\_SHORT).show();

} else {

Toast.makeText(Register.this, “Preencha todos os campos”, Toast.LENGTH\_SHORT).show();

}

}

}

});

btn\_login\_register.setOnClickListener(new View.OnClickListener(){

@Override

public void onClick (View v){

Intent intent = new Intent (Register.this, Login.class);

startActivity(intent);

finish();

}

});

}

private void abrirTelaPrincipal(){

Intent intent = new Intent (Register.this, MainActivity.class);

startActivity(intent);

// finaliza a tela de login

finish();

}

}

**3. CURSO DE FIREBASE (CANAL EXTRA MATCH 007)**

**3.1. PLAYLIST FIREBASE**

<https://www.youtube.com/watch?v=PE_riDivk6Y&list=PLfs7O23seKWmhdu3tJlrshIXpSczlgmQy>

**3.1.1. VIDEO: Aula 56 – CRUD: Android e Firebase – Insert**

**Tempo: 23min19s, link:** [**https://www.youtube.com/watch?v=PE\_riDivk6Y&**](https://www.youtube.com/watch?v=PE_riDivk6Y&) **Acesso em 28/07/2022**

Acesse o site do Firebase, faça o login com a conta do Google, adicione um projeto. Nome: Firebase, localização:EUA. Clique em Database. Vai abrir o Realtime Database. Vá na guia de Regras. Deixe como:

{

“rules”: {

“read”: true

“write”: true

}

}

Vá para o Android Studio, file, new, Project. Nome: MyFIrebase, API 21 (Lollipop), Java, empty activity. Package: wgc.myfirebase. Volte no FIrebase, clique no projeto, clique em “Adicionar o Firebase na aplicação Android”, coloca o nome do pacote.

Tem que buscar o SHA-1. Clique em Gradle Projects, refresh, clique em MyFirebase, MyFirebase (root), pasta tasks, pasta android, signingReport. Após carregar, clica no botão de gráfico. Procure por SHA-1, que no caso é SHA1: 3F:00:16:D1:32:5C:B2:F8:8C:C8:1E:65:09:5A:7D:E0:A9:EE:25:A3; copie o código e cole na criação do projeto. Registrar app. Baixe o package-json do projeto clicando no download.

Dentro do android studio, deixe a visão no modo projeto. Após,vá em MyApplication, pasta app, pasta src, e cole exatamente abaixo de build.gradle. Após, coloque as dependências do Firebase. Adicione a nível de projeto como compile dentro da área dependencies{} e a nível de app o plugin dentro de dependencies{} como compile. Também coloque a implementação do firebase-database versionado.(com.google.firebase:firebase-database:9.6.1). Espere ser gerado no firebase essa configuração; e depois volte no Android e dê um sync now.

Vá na pasta app, pasta AndroidManifest. Acesse o arquivo: AndroidManifest.xml.

**ARQUIVO: AndroidManifest.xml**

<? xml version=”1.0” enconing=”utf-8”/>

<manifest xmlns:android=”<http://schemas.android.com/apk/res/android> “

package =”wgc.myfirebase”>

<uses-permission android:name = “android.permission.INTERNET”/>

<application

android:allowbackup=”true”

android:icon=”@mipmap/ic\_launcher”

android:label=”MyFirebase”

android:roundIcon=”@mipmap/ic\_launcher\_round”

android:suportsRtl=”true”

android:theme=”@style/.MainActivity”>

<Intent-filter>

<action android:name=”android.intent.action.MAIN/>

<category android:name=”android.intent.category.LAUNCHER” />

</intent-filter>

</activity>

</application>

</manifest>

Vai criar a classe do objeto que irá usar. Vá na pasta app, pasta java, pasta wgc.myfirebase. Clique com o botão direito na pasta wgc.myfirebase, new, package, nome: modelo. Clique com o botão direito na pasta modelo, new, class, nome: Pessoa.

**ARQUIVO: Pessoa.java**

public class Pessoa{

private String uid;

private String nome;

private String email;

public Pessoa(){

}

public Pessoa (String uid, String nome, String email){

this.uid = uid;

this.nome = nome;

this.email = email;

}

public String uid() {

return uid;

}

public String nome() {

return nome;

}

public String email(){

return email;

}

public void setUid (String uid){

this.uid = uid;

}

public void setNome (String nome){

this.nome = nome;

}

public void setEmail (String email){

this.email = email;

}

@Override

public String toString(){

return nome;

}

}

Agora ele vai criar um menu. Vá na pasta res, botão direito, new, Image Action. Icon type: Action Bar and Tab Items.

Primeiro botão com nome: ic\_novo, clip art para + circular com fundo branco, theme: holo\_dark. Clique em Next. Finish.

Segundo botão com nome: ic\_atualiza, clip art para disquete com fundo branco, theme: holo\_dark. Clique em Next. Finish.

Terceiro botão com nome: ic\_deleta, clip art para - circular com fundo branco, theme: holo\_dark. Clique em Next. Finish.

Para criar a aplicação do menu com botões, clica com o botão direito na pasta res, new, Android Resource File. File name: menu\_main , resource type: Menu, directory name: menu.

**Arquivo: menu\_main.xml**

<?xml version=”1.0” enconding=”utf-8”>

<menu xmlns: android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”<http://schemas.android.com/apk/res-auto>”>

// Cria novo usuário

<item android:id=”@+id/menu\_novo”

android:title=”Novo”

android:icon=”@drawable/ic\_novo”

app:showAsAction=”always”/>

// Atualiza o usuário

<item android:id=”@+id/menu\_atualiza”

android:title=”Atualiza”

android:icon=”@drawable/ic\_atualiza”

app:showAsAction=”always”/>

//Exclui o usuário

<item android:id=”@+id/menu\_deleta”

android:title=”Deleta”

android:icon=”@drawable/ic\_deleta”

app:showAsAction=”always”/>

</menu>

Vamos para a activity\_main.xml

**ARQUIVO: Activity\_main.xml**

<? xml version=”1.0” enconding =”utf-8”?>

<LinearLayout.widget.ConstraintLayout xmlns:android=”<http://schemas.android.com/apk/res/android>”

xmlns:app=”http:// schemas.android.com/apk/res-auto”

xmlns:tools=”http://schemas.android.com/tools”

android:layout\_width=”match\_parent”

android:layout\_height=”match\_parent”

android:orientation=”vertical”

tools:context=”.wgc.myfirebase.MainActivity”>

// Campo de entrada de nome

<EditText

android:id=”@+id/edtNome”

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:layout\_weight = “1”

android:ems=”10”

android:inputType=”textPersonName”

android:hint=”Name”/>

// Campo de entrada de email

<EditText

android:id=”@+id/edtEmail”

android:layout\_width=”wrap\_content”

android:layout\_height=”wrap\_content”

android:layout\_weight = “1”

android:ems=”10”

android:inputType=”textEmailAddress”

android:hint=”Email”/>

// Lista com os resultados obtidos

<ListView

android:id=”@+id/listV\_dados”

android:layout\_width=”wrap\_content”

android:layout\_height=”match\_content”

/>

</LinearLayout>

**Vá para a MainActivity.java**

**ARQUIVO: MainActivity.java**

public class MainActivity extends AppCompatActivity{

EditText edtNome, edtEmail;

ListView listV\_dados;

FirebaseDatabase firebaseDatabase;

DatabaseReference databaseReference;

@Override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

edtEmail = (EditText) findViewById(R.id.edtEmail);

edtNome = (EditText) findViewById(R.id.edtNome);

listV\_dados = (ListView) findViewById(R.id.listV\_dados);

inicializarFirebase();

}

private void inicializarFIrebase(){

FirebaseApp.initializeApp(MainActivity.this);

firebaseDatabase = FirebaseDatabase.getInstance();

databaseReference = firebaseDatabase.getReference();

}

public boolean onCreateOptionsMenu (Menu menu){

// parâmetros: contexto, menu do método

getMenuInflater().inflate(R.menu.menu\_main, menu);

return super.onCreateOptionMenu(menu);

}

public boolean onOptionsItemSelected(MenuItem item){

int id = item.getItemId();

if (id == R.id.menu\_novo){

Pessoa p = new Pessoa();

p.setUid(UUID.randomUUID().toString());

p.setNome(edtNome.getText().toString());

p.setEmail(edtEmail.getText().toString());

// A coleção pessoa tem como nó o id que encontra o valor p

databaseReference.child(“Pessoa”).child(p.getUid()).setValue(p);

limparCampos();

}

return true;

}

private void limparCampos(){

edtEmail.setText(“”);

edtNome.setText(“”);

}

}

Faz o teste com o emulador.

**3.1.2.VÍDEO: Aula 57 – CRUD: Android e Firebase – Select**

Tempo: 08min35s, link: <https://www.youtube.com/watch?v=D1_85HekEYc>

**ARQUIVO: MainActivity.java**

public class MainActivity extends AppCompatActivity{

EditText edtNome, edtEmail;

ListView listV\_dados;

FirebaseDatabase firebaseDatabase;

DatabaseReference databaseReference;

private List<Pessoa> listPessoa = new ArrayList<Pessoa>();

private ArrayAdapter<Pessoa> arrayAdapterPessoa;

@Override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

edtEmail = (EditText) findViewById(R.id.edtEmail);

edtNome = (EditText) findViewById(R.id.edtNome);

listV\_dados = (ListView) findViewById(R.id.listV\_dados);

inicializarFirebase();

eventoDatabase();

}

private void eventoDatabase(){

databaseReference.child(“Pessoa”).addValueEventListener(new ValueEventListener(){

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

listPessoa.clear();

for (DataSnapshot objSnapshot: dataSnapshot.getChildren()) {

Pessoa p = objSnapshot.getValue(Pessoa.class);

listPessoa.add(p);

}

arrayAdapterPessoa = new ArrayAdapter<Pessoa>(MainActivity.this,android.R.layout.simple\_list\_item\_1, listPessoa);

listV\_dados.setAdapter(arrayAdapterPessoa);

}

@Override

public void onCancelled(DatabaseError databaseError){

}

});

}

private void inicializarFIrebase(){

FirebaseApp.initializeApp(MainActivity.this);

firebaseDatabase = FirebaseDatabase.getInstance();

firebaseDatabase.setPersistenceEnabled(true); // salva e altera arquivos na nuvem e no app

databaseReference = firebaseDatabase.getReference();

}

public boolean onCreateOptionsMenu (Menu menu){

// parâmetros: contexto, menu do método

getMenuInflater().inflate(R.menu.menu\_main, menu);

return super.onCreateOptionMenu(menu);

}

public boolean onOptionsItemSelected(MenuItem item){

int id = item.getItemId();

if (id == R.id.menu\_novo){

Pessoa p = new Pessoa();

p.setUid(UUID.randomUUID().toString());

p.setNome(edtNome.getText().toString());

p.setEmail(edtEmail.getText().toString());

// A coleção pessoa tem como nó o id que encontra o valor p

databaseReference.child(“Pessoa”).child(p.getUid()).setValue(p);

limparCampos();

}

return true;

}

private void limparCampos(){

edtEmail.setText(“”);

edtNome.setText(“”);

}

}

**3.1.3.VÍDEO: Aula 58 – CRUD: Android e Firebase – Update e Delete**

Tempo: 10min50s, link: <https://youtu.be/NnZKRMszgSM>

**ARQUIVO: MainActivity.java**

public class MainActivity extends AppCompatActivity{

EditText edtNome, edtEmail;

ListView listV\_dados;

FirebaseDatabase firebaseDatabase;

DatabaseReference databaseReference;

private List<Pessoa> listPessoa = new ArrayList<Pessoa>();

private ArrayAdapter<Pessoa> arrayAdapterPessoa;

Pessoa pessoaSelecionada;

@Override

protected void onCreate (Bundle savedInstanceState){

super.onCreate(savedInstanceState);

setContentView.layout.activity.main);

edtEmail = (EditText) findViewById(R.id.edtEmail);

edtNome = (EditText) findViewById(R.id.edtNome);

listV\_dados = (ListView) findViewById(R.id.listV\_dados);

inicializarFirebase();

eventoDatabase();

listV\_dados.setOnItemClickListener(new AdapterView.OnItemClickListener(){

@Override

public void onItemSelected(AdapterView<?> parent, View view, int position, long id){

pessoaSelecionada = (Pessoa) parent.getItemAtPosition(position);

edtNome.setText(pessoaSelecionada.getNome());

edtEmail.setText(pessoaSelecionada.getEmail());

}

});

}

private void eventoDatabase(){

databaseReference.child(“Pessoa”).addValueEventListener(new ValueEventListener(){

@Override

public void onDataChange(DataSnapshot dataSnapshot) {

listPessoa.clear();

for (DataSnapshot objSnapshot: dataSnapshot.getChildren()) {

Pessoa p = objSnapshot.getValue(Pessoa.class);

listPessoa.add(p);

}

arrayAdapterPessoa = new ArrayAdapter<Pessoa>(MainActivity.this,android.R.layout.simple\_list\_item\_1, listPessoa);

listV\_dados.setAdapter(arrayAdapterPessoa);

}

@Override

public void onCancelled(DatabaseError databaseError){

}

});

}

private void inicializarFIrebase(){

FirebaseApp.initializeApp(MainActivity.this);

firebaseDatabase = FirebaseDatabase.getInstance();

firebaseDatabase.setPersistenceEnabled(true); // salva e altera arquivos na nuvem e no app

databaseReference = firebaseDatabase.getReference();

}

@Override

public boolean onCreateOptionsMenu (Menu menu){

// parâmetros: contexto, menu do método

getMenuInflater().inflate(R.menu.menu\_main, menu);

return super.onCreateOptionMenu(menu);

}

@Override

public boolean onOptionsItemSelected(MenuItem item){

int id = item.getItemId();

// Se quiser inserir novo usuário

if (id == R.id.menu\_novo){

Pessoa p = new Pessoa();

p.setUid(UUID.randomUUID().toString());

p.setNome(edtNome.getText().toString());

p.setEmail(edtEmail.getText().toString());

// A coleção pessoa tem como nó o id que encontra o valor p

databaseReference.child(“Pessoa”).child(p.getUid()).setValue(p);

limparCampos();

}else if (id == R.id.menu\_atualiza){ // Se quiser atualizar o usuário

Pessoa p = new Pessoa();

p.setUid(pessoaSelecionada.getUid());

p.setNome(edtNome.getText().toString().trim()); // trim tira o espaço em branco

p.setEmail(edtEmail.getText().toString().trim());

databaseReference.child(“Pessoa”).child(p.getUid()).setValue(p);

limparCampos();

}else if (id == R.id.menu\_deleta){ // Se quiser deletar o usuário

Pessoa p = new Pessoa();

p.setUid(pessoaSelecionada.getUid());

databaseReference.child(“Pessoa”).child(p.getUid()).removeValue();

limparCampos();

}

return true;

}

private void limparCampos(){

edtEmail.setText(“”);

edtNome.setText(“”);

}

}

**4. CURSO DE FIREBASE (CANAL JONE FRANCO)**

**4.1.1. VÍDEO: Firebase Firestore – Regras de Segurança – Parte 1 – Criando regras de segurança**

Tempo: 12min44s, link: <https://youtu.be/v3SXmnwAJd0,> acessado em 29/07/22

Data de criação do vídeo: 03/05/2021

07/05/2021 foi a parte 05.

Abre o site do Firebase, cria um projeto chamado FirebaseCursoSF, abra o Cloud Firestore, guia Regras. As regras foram inspiradas no Cloud Storage.

**Cloud Firestore**

rules\_version = ‘2’;

// regra de serviço

service cloud.firestore{

// caminho – obrigatório, referencia a raiz do banco de dados. Acessa a primeira coluna.

match/databases/(database)/documents {

// Regra padrão

match/ (document\*\*\*){ // coringa, representa qualquer tipo de conexão, e qualquer subpasta que esteja dentro das coleções

// permite ler, inserir e remover sem segurança, pode não usar o app. Tem que ser false.

// Para ler, remover, inserir e atualizar somente quem está logado no app

// Para gravar a informação no banco, tem a comunicação do app com o banco de dados do firebase. O app vai mandar uma mensagem pro firebase pedindo acesso as informações do banco de dados. O firebase responde pedindo as informações do usuário. Uma das informações que se pede é a identificação do usuário que está logado naquele momento.

allow read: if request.auth.uid != null;

allow write: if request.auth.uid != null;

}

}

}

**4.1.2. VÍDEO: Firebase Firestore Regras de Segurança – Parte 2 – Regras para cada coleção**

**Tempo: 13min12s, acessado em: 01/08/2022, link:** [**https://www.youtube.com/watch?v=\_5hCEeHUejQ**](https://www.youtube.com/watch?v=_5hCEeHUejQ) **, data de criação do vídeo: 04/05/2021**

Você pode querer criar regras separadas para cada coleção. Para o banco de dados, temos as coleções de categorias e gerentes. Vai aplicar uma regra para cada coleção.

Logo:

switch /databases/(database)/documents{

match/Categorias/(document){

// só faz algo se tiver logado

allow read: if request.auth.uid ! = null;

allow write: if request.auth.uid !=null;

match/usuários/(document\*\*\*){

allow read: if true;

allow write: if true;

/\*

match/usuários/(document\*\*\*){

allow read: if false;

allow write: if false;

}\*/

}

}

// qualquer pessoa estará sem acesso à informação, independente de estar logada ou não.

match/Gerentes/(document\*\*\*){

allow read: if false;

allow write: if false;

}

}

Para cada coleção criada deve criar uma regra, e isso também deve ser feito para cada subpasta. Dentro das coleções, vá na pasta onde tem a coleção Usuários. Agora vai querer colocar uma regra para coleção interna de Usuários. Tem 2 documentos, com algumas informações. Quer outra regra, outro controle.

Se tentar fazer um conjunto de chaves após o match de gerentes, vai ter um problema. Porque todos os documentos e subpastas de dentro de categorias sejam aplicadas uma regra de segurança para ler apenas sobre autenticação. Como usuários está dentro da pasta Categorias, dará conflito de interesse/regra. O ideal é apagar a parte final de document\*\*\*. Ou seja, deixar apenas document; e o erro vai desaparecer. A sub coleção usuários deve estar no mesmo conjunto de chaves de Categorias.

Em Categorias há 50 documentos. O document é como se fosse o número do documento. Cada documento é uma subpasta. Pode substituir document por nomedapasta.

Na situação de usuários, vamos pegar o documento da subpasta. Dentro da posição usuários, vamos aplicar a regra para cada documento. Dentro da pasta de usuários, quer aplicar uma regra diferente para uma subcoleção. Logo, vamos construindo o par de chaves por encadeamento. Depois de finalizado, clica em publicar.

Faça o teste com o emulador do Android Studio. Quando tentou acessar a parte de alunos no aplicativo, deu erro de falta de permissão; pois no conjunto de regras, os usuários não podem ler nem escrever os dados. Mudando para true ele aceita, assim como deixar o coringa como document\*\*\*.

**4.1.3. VÍDEO: Firebase Firestore Regras de Segurança – Parte 3 – Regras para cada usuário**

**Tempo: 08min39s, acessado em: 01/08/2022, link:** [**https://www.youtube.com/watch?v=xRz4pMh-iXM**](https://www.youtube.com/watch?v=xRz4pMh-iXM) **, data de criação do vídeo: 05/05/2021**

Vamos restringir ainda mais o acesso ao banco de dados. Quando acessa a coleção categorias, encontra os documentos e a subcoleção de usuários que carregam os itens. Aparece como: Categorias/6/usuários. Se olhar a parte de autenticação, o documento armazena pelo uid de identificação do authenticator. Quer que o [leo@gmail.com](mailto:leo@gmail.com) tenha acesso ao banco, somente ele. Cada usuário terá acesso apenas a sua pasta. Para criar essa regra:

switch /databases/(database)/documents{

match/Categorias/(document){

// só faz algo se tiver logado

allow read: if request.auth.uid ! = null;

allow write: if request.auth.uid !=null;

// forma específica

match/usuarios/( UbkmBK09og6XF7JIBWj5NzFiwy02)/(document\*\*\*){

allow read: if UbkmBK09og6XF7JIBWj5NzFiwy02 != UbkmBK09og6XF7JIBWj5NzFiwy02;

allow write: if UbkmBK09og6XF7JIBWj5NzFiwy02 != UbkmBK09og6XF7JIBWj5NzFiwy02;

}

// forma generalizada

match/usuários/(uidUsuario)/(document\*\*\*){

allow read : if request.auth.uid != uidUsuario;

allow write : if request.auth.uid != uidUsuario;

}

}

// qualquer pessoa estará sem acesso à informação, independente de estar logada ou não.

match/Gerentes/(document\*\*\*){

allow read: if false;

allow write: if false;

}

}

A variável nomepasta pode significar, tanto o documento identificado pelo uid do authenticator. O uid vai ser uma identificação de quem está logado no aplicativo. O nome da pasta é o que ele está tentando acessar. Exemplo: nome é [leo@gmail.com](mailto:leo@gmail.com), e o leo.uid será o uid = UbkmBK09og6XF7JIBWj5NzFiwy02

Quando troca o uid e nomepasta pelo identificador, teria que fazer uma regra para cada usuário; em grandes bancos ficaria inviável. Em um formato geral, seria:

match/usuários/(uidUsuario)/(document\*\*\*){

allow read : if request.auth.uid != uidUsuario;

allow write : if request.auth.uid != uidUsuario;

}

Faça o teste com o emulador do Android Studio.

**APLICAR CONFIGURAÇÃO DE PARÁGRAFO DENTRO DO MICROSOFT OFFICE: ITENS 4.1.4 E 4.1.5**

**4.1.4. VÍDEO: Firebase Firestore Regras de Segurança – Parte 4 – Regras para cada usuário Write Parte Final**

**Tempo: 03min50s, acessado em: 01/08/2022, link: https://www.youtube.com/watch?v=8OBvhsvqj\_s , data de criação do vídeo: 06/05/2021**

A partir da configuração da última aula, em Firebase Cloud Firestore:

match / databades/{database}/documents{

match/Categories/{nomepasta} {

allow read: if request.auth.uid != null;

allow write: if request.auth.uid != null;

match/usuarios{uidUsuario]/ {document\*\*\*}{

allow read: if request.auth.uid != uidUsuario;

// Temos 3 funções em separado: atualização, remoção e inserção de dados

allow update: if request.auth.uid != uidUsuario;

allow delete: if false; // nunca deleta a informção;

allow create: if request.auth.uid != uidUsuario;

// forma genérica de fazer o update, delete e create:

allow write: if request.auth.uid != uidUsuario;

}

}

}

Teste no Android Studio. Verifique a documentação, em primeiros passos para configurar regras de segurança em ambientes inseguros:

service cloud.firestore {

match /databases/{database}/documents {

// For attribute-based access control, Check a boolean `admin` attribute

allow write: if get(/databases/$(database)/documents/users/$(request.auth.uid)).data.admin == true;

allow read: true;

// Alterntatively, for role-based access, assign specific roles to users

match /some\_collection/{document} {

allow read: if get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role == "Reader"

allow write: if get(/databases/$(database)/documents/users/$(request.auth.uid)).data.role == "Writer"

}

}

}

**4.1.5. VÍDEO: Firebase Firestore Regras de Segurança – Parte 5 – Usando Get para acessar Firestore**

**Tempo: 08min16s, acessado em: 08/08/2022, link: https://www.youtube.com/watch?v=bWEdEg-dszQ , data de criação do vídeo: 07/05/2021**

Dentro do Firestore, vá em Categorias, iniciar Coleção, nome: Teste. Código: admin, campo: nome, string, valor: Jone. Vou ter que criar uma regra pra coleção.

Consulte no Banco de dados para ver se ele existe ou não. Por exemplo, usando o get:

A partir da configuração da última aula, em Firebase Cloud Firestore:

match / databases/{database}/documents{

match/Categories/{nomepasta} {

allow read: if request.auth.uid != null;

allow write: if request.auth.uid != null;

match/usuarios{uidUsuario]/ {document=\*\*}{

allow read: if get(databases/$(database)/documents/teste/admin).data.nome == ‘Jone’ ; //se o usuário acessar o conteúdo da coleção Usuários (coleção de itens), só entra se estiver na segunda condição (de acessar o Jone)

allow write: if request.auth.uid != uidUsuario;

}

}

match/teste/{document =\*\*]{

allow read: if true

allow write: if true;

}

match/Gerentes/{document =\*\*]{

allow read: if request.auth.uid != uidUsuario;

allow write: if request.auth.uid != uidUsuario;

}

}

Ele quer fazer o acesso do banco pra buscar o campo Jone dentro da coleção Teste. Quando o usuário tentar acessar os itens de usuários (como alunos), somente vão conseguir ler somente se o valor de dentro do banco no campo for igual a Jone.

**Se dentro do teste, colocar o tipousuário como admin, teremos:**

match/usuarios{uidUsuario]/ {document=\*\*}{

allow read: if get(databases/$(database)/documents/teste/admin).data.tipousuario == ‘admin’ ; //se o usuário acessar o conteúdo da coleção Usuários (coleção de itens), só entra se estiver na segunda condição (de acessar o Jone)

allow write: if request.auth.uid != uidUsuario;

}

}

Se o usuário for Jone, ele não terá acesso; ele tem que ser o tipo de usuário como admin. Ou seja, somente o admin pode ler o banco de dados: teste. Testa o emulador.

(colocar como nível 1 no sumário, item 5)