Primeiro vamos acessar a documentação.

Vantagem do Firebase ele tem vários tipos de serviços como

* Banco de dados
* Autenticador
* Etc..

Podemos usar cada serviço desse de forma separada, adicionando no nosso projeto.

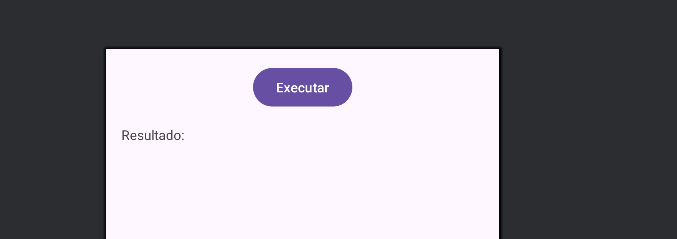
Vamos adicionar o recurso de autenticação

Também vamos adicionar banco de dados e armazenamento

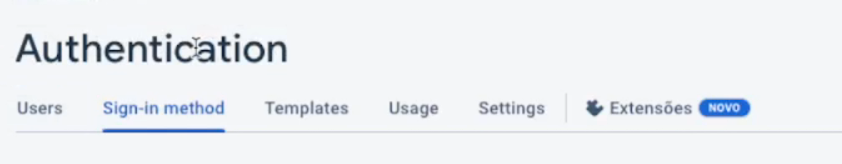
// Firabase Bom  
*implementation*(platform("com.google.firebase:firebase-bom:32.7.2"))  
  
// Analutics  
*implementation*("com.google.firebase:firebase-analytics")  
  
// Autenticação  
*implementation*("com.google.firebase:firebase-auth")  
  
// Cloud Firestore  
*implementation*("com.google.firebase:firebase-firestore")  
  
//Cloud Storage  
*implementation*("com.google.firebase:firebase-storage")

# Agora mao na massa

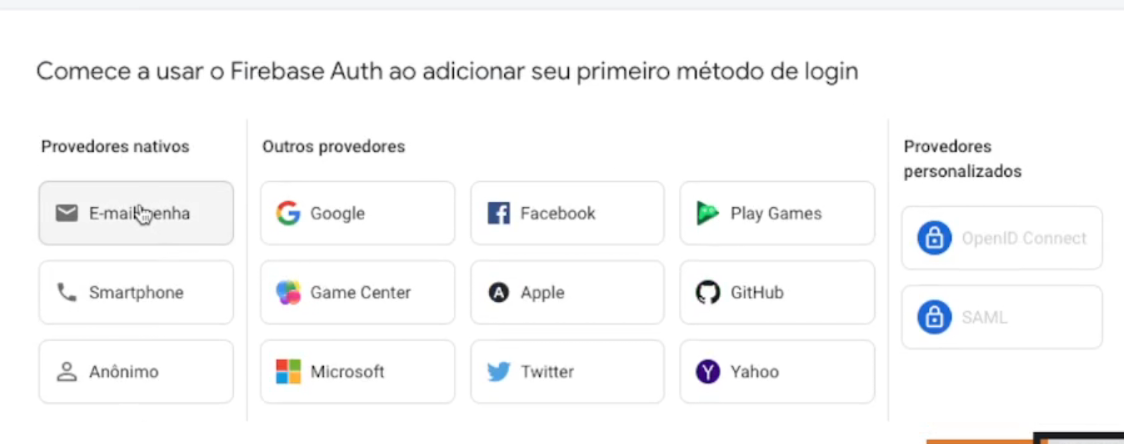
## Criamos uma layout simples



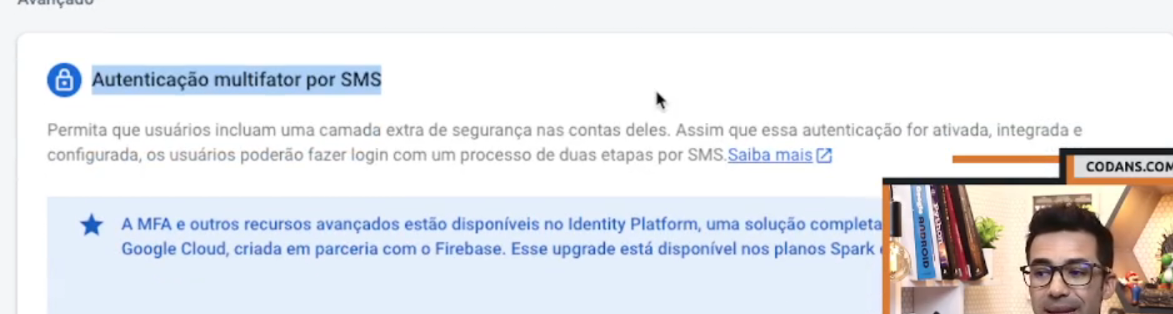
## 2 configurando o autenticador no Firebase



Aqui vamos ter os provedores que podemos usar para autenticar um usuário



Tambem podemos fazer uma autenticação de segunda camada



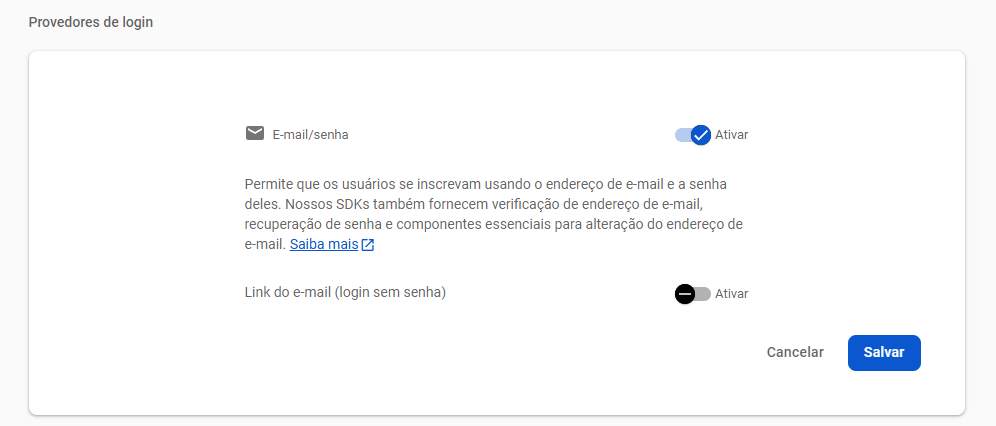
Nesse caso o usuário por exemplo iria adicionar e-mail e senha e depois de confirmar esses dados, ele iria também enviar um SMS para confirmar

\*\* Isso é um recurso pago \*\*

### Ativando um provedor

Nesse nosso exemplo vamos ativar somente e-mail e senha.

1 – clique em e-mail e senha



Agora só ativar e clicar em salvar.

Agora aqui dentro vamos poder ver os usuários, vamos poder adicionar novos métodos de login, ver a ultima vez que o usuário foi logado...

# Cadastrando usuário pelo app

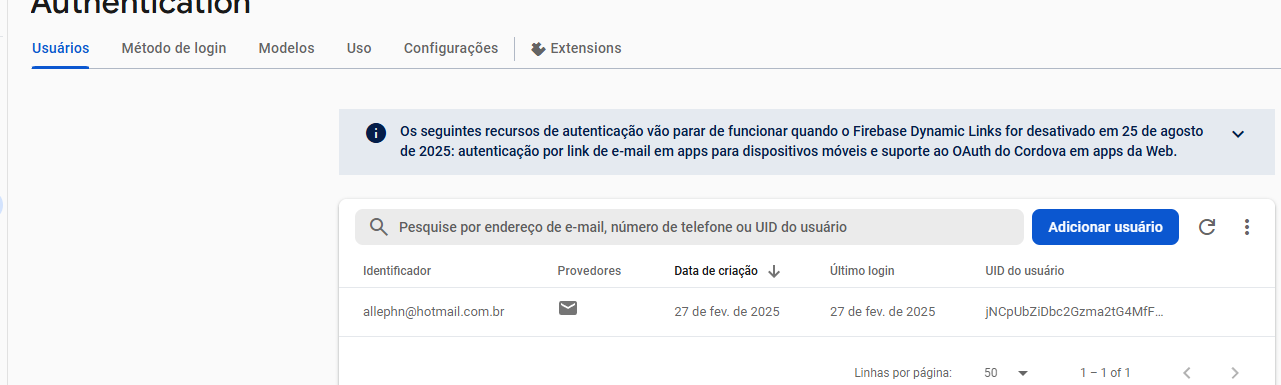
Aqui poderíamos fazer uma caixa de texto para o usuário digitar seus dados, mas vamos deixar como padrão apenas para teste.

Criamos um método para cadastrar o usuário.

binding.btnExecutar.setOnClickListener **{** cadastroDeUsuario()  
 **}** }  
  
 private fun cadastroDeUsuario() {  
 /\* Imagina que estamos capturando os dados que o usuario digitou.  
 \*  
 \* O firebase vai verificar essa senha, ele exige que tenha pelo menos 6 caracteres e seja uma senha forte  
 \* Se a senha for fraca ele nao vai deixar voce cadastrar o usuario.  
 \*  
 \* TELA DE CADASTRO DO USUARIO. \*/  
  
 val email = "allephn@hotmail.com.br"  
 val senha = "teste12345@nogueira"  
  
 /\* USANDO AS CLASSES DO FIREBASE PARA CADASTRO  
 \* Atenção: é recomendado sempre usar a classe especifica para cada recurso que for utilizar  
 \*\*\*\*\* val autenticador = Firebase.auth // Não recomendado \*\*\*\*\*/  
  
 val autenticador = FirebaseAuth.getInstance()

// Vamos pegar as instancias do FirebaseAuth e vamos usar depois no autenticador, imagina que essas instancias seja os métodos para cadastrar, excluir, alterar...  
  
 /\* Dentro do firebaseAuth vamos ter os metodos para criar o usuario  
 Esse metodo serve para criar o usuario com email e senha  
  
 addOnSuccessListener -> Vamos verificar se foi sucesso ou nao ao cadastrar usuario  
 Apos a primeira chave vamos tratar o erro  
 addOnFailureListener -> Em caso de falha vamos cair nesse outro escopo  
  
 exibirMensagem() -> apenas um toast que vai exibir uma mensagem de sucesso ou falha  
 \*/  
  
 autenticador.createUserWithEmailAndPassword(  
 email, senha  
 ).addOnSuccessListener **{** authResult **->** // TESTAR O SUCESSO DO CADASTRO  
 /\* Metodos a baixo para usar se caso o usuario foi cadastrado corretamente \*/  
  
 // Dentro de authResult temos varias informações  
 // Dentro de user vamos ter todos os dados do usuario nesse exemplo capturamos o email apenas  
 val email = authResult.*user*?.*email* // Aqui é para quando o usuario se cadastrar ele vai enviar um email de verificação e só vai ser verificado quando o usuario clicar no link  
 val verificarEmail = authResult.*user*?.sendEmailVerification()  
  
 // Aqui é o identificador do usuario o ID dele  
 val idDoUsuario = authResult.*user*?.*uid* exibirMensagem("Sucesso ao cadastrar usuario $idDoUsuario - $email") // Exibe um toast com os dados  
 binding.textResultado.*text* = "Sucesso: $idDoUsuario - $email"  
  
  
 **}**.addOnFailureListener**{** exception **->** // TESTAR A FALHA DO CADASTRO  
  
 val mensagemErro = exception.message  
  
 exibirMensagem("Falha no cadastro")  
 binding.textResultado.*text* = "FALHA: $mensagemErro"  
 **}** }  
  
 private fun exibirMensagem(texto: String) {  
 Toast.makeText(*applicationContext*, texto, Toast.*LENGTH\_LONG*).show()  
 }  
}

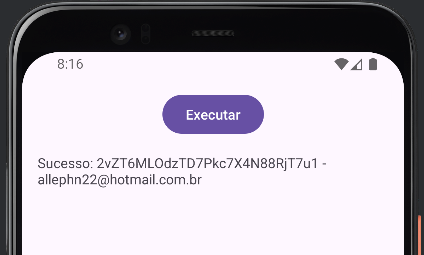
Quando o usuário clicar no botão executar ele vai criar esse cadastro



# Verificar se realmente o usuário foi cadastrado

Pode acontecer erros em cadastrar usuários e para isso vamos fazer testes.

Cadastro realizado com sucesso



# Verificar se o usuário esta logado

Podemos fazer isso dentro do onStart para sempre fazer essa verificação quando o usuário sair e entrar novamente

Imagina que toda vez que essa tela for recriada, vamos novamente chamar o método que esta dentro do onStart e esse método vai ser um método para verificar se o usuário esta logado.

Vamos criar uma nova activity

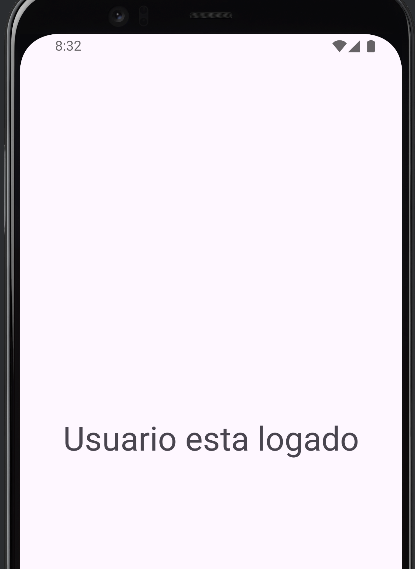
Criamos uma activity principal, aqui nessa tela somente usuário logado vai poder estar aqui.

override fun onStart() {  
 super.onStart()  
 verificarUsuarioLogado()  
 }  
  
 private fun verificarUsuarioLogado() {  
 val usuario = FirebaseAuth.getInstance()  
  
  
 /\* Aqui ele vai pegar o usuario atual \*/  
 usuario.*currentUser* val idUsuario = usuario?.*uid* //

Atenção aqui, quando criamos um usuário o firebase guarda aquele usuário ate ele fazer logaut, nesse caso não fizemos então ele sabe qual foi o ultimo usuário a ser criado/autenticado.

Pegar o id do usuário atual – Isso seria pegar o ID do ultimo usuário que esse dispositivo criou.

if (usuario != null) {  
 // Se der verdadeiro quer dizer que existe usuario logado  
 exibirMensagem("Usuario esta logado com id: $idUsuario")  
 // Se o usuario estiver logado, vamos jogar ele para a proxima tela  
 val intent = Intent(this, PrincipalActivity::class.*java*)  
 startActivity(intent)  
  
 }else{  
 // Não existe usuario logado se der null  
 exibirMensagem("Não tem usuario logado")  
 }  
 }  
  
}



Resumindo sempre que a tela tiver atualização, ele vai chamar o método e vai testar de novo, se retornar o usuário logado ele vai sempre jogar para a nova activity.