

FAZENDO NOSSO LOGIN FUNCIONAR

Professor Crispim Luiz Martins da Silva

CUIDADO

- ▶ Temos nossa tela cadastro e a tela login. Existe o perigo que termos repetido os id de senha e e-mail. Se for o caso devemos mudar. Nas telas login faremos as mudanças.

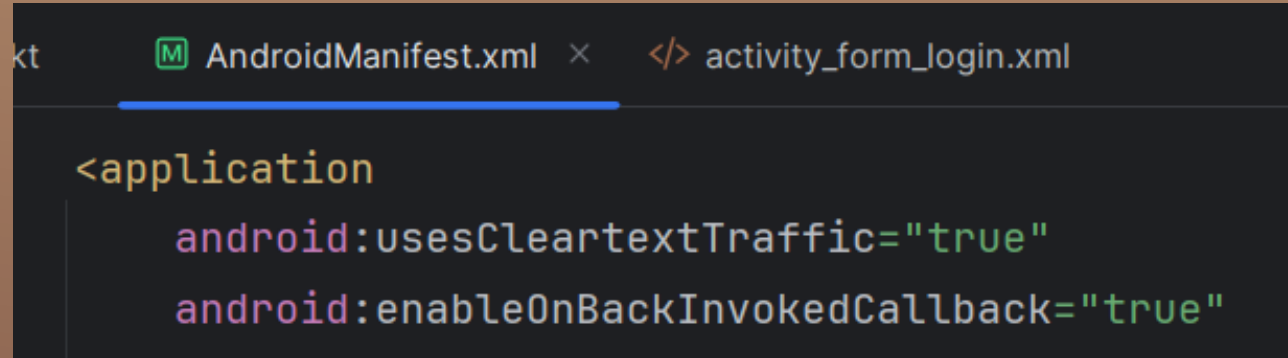
```
<EditText  
    android:id="@+id/edit_email_login"
```

```
<EditText  
    android:id="@+id/edit_senha_login"
```

Lembrem que deve trocar os nomes dos onde são usados

```
app:layout_constraintTop_toBottomOf="@id/edit_senha_login"
```

CONFIGURAÇÕES



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.app">
    <application
        android:usesCleartextTraffic="true"
        android:enableOnBackInvokedCallback="true"
        ...>
    </application>
</manifest>
```


- ▶ **android:usesCleartextTraffic="true":**
- ▶ Essa configuração é usada no arquivo AndroidManifest.xml para permitir tráfego não criptografado (HTTP) em vez de tráfego criptografado (HTTPS) para comunicação com servidores externos.
- ▶ Por padrão, a partir do Android 9 (API nível 28), o tráfego não criptografado é bloqueado por razões de segurança. No entanto, definir android:usesCleartextTraffic="true" permite que o aplicativo use conexões HTTP não criptografadas.
- ▶ Importante: O uso de tráfego não criptografado pode ser inseguro, pois os dados não são protegidos contra interceptação. Portanto, é recomendável usar HTTPS sempre que possível para garantir a segurança dos dados transmitidos entre o aplicativo e o servidor.

- ▶ **android:enableOnBackPressedCallback="true":**
- ▶ Essa configuração está relacionada à navegação por gestos no Android.
- ▶ A partir do Android 13, o sistema de gestos de retorno previsto foi introduzido para melhorar a consistência e a previsibilidade da navegação de retorno.
- ▶ Definir `android:enableOnBackPressedCallback="true"` no arquivo `AndroidManifest.xml` permite que o aplicativo use o callback de retorno ao pressionar o botão de voltar.
- ▶ Isso é útil para lidar com eventos de retorno personalizados quando o usuário pressiona o botão de voltar ou usa gestos de retorno.

```
FormLogin.kt x </> activity_form_login.xml
1 package br.com.faculdade.imepac
2
3 > import ...
10
11 </> class FormLogin : AppCompatActivity() {
12     private lateinit var edit_email: EditText
13     private lateinit var edit_senha: EditText
14     private lateinit var bt_entrada: Button
15     private lateinit var progressbar: ProgressBar
```

Logo encima, já
declaro as variáveis

CRIANDO NOSSA VARIÁVEIS, QUE RECEBERAM A
INFORMAÇÃO DO ELEMENTO DO XML

```
16
17  override fun onCreate(savedInstanceState: Bundle?) {
18     super.onCreate(savedInstanceState)
19     setContentView(R.layout.activity_form_login)
20     //No comando abaixo mando esconder o Toolbar
21     getSupportActionBar()?.hide();
22
23     val linkFormCadastro = findViewById<TextView>(R.id.text_tela_cadastro)
24
25     linkFormCadastro.setOnClickListener{ it: View!
26         val intent = Intent( packageContext: this, FormCadastro::class.java)
27         startActivity(intent)
28     }
29
```


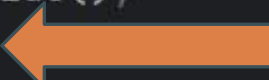
ESSA PARTE JÁ HAVIA, NÃO MEXA!!!
COMEÇAMOS DEPOIS!!!

No final, vamos atribuir os elementos para as variáveis. Está fora da função principal.

```
71     fun IniciarComponentes() {  
72         edit_email = findViewById(R.id.edit_email_login)  
73         edit_senha = findViewById(R.id.edit_senha_login)  
74         bt_entrada = findViewById(R.id.bt_entrada)  
75         progressbar = findViewById(R.id.progressbar)  
76     }  
77 }
```

Vamos chamar essa função dentro de nossa onCreate.

PEGAMOS OS ELEMENTOS E COLOCAMOS NAS VARIÁVEIS.
VEJA QUE É LOGO NO FINAL!!!!


```
18
19  override fun onCreate(savedInstanceState: Bundle?) {
20     super.onCreate(savedInstanceState)
21     setContentView(R.layout.activity_form_login)
22     //No comando abaixo mando esconder o Toolbar
23     getSupportActionBar()?.hide();
24     IniciarComponentes(); 
25     val linkFormCadastro = findViewById<TextView>(R.id.text_tela_cadastro)
26
```

VEJA QUE CHAMAMOS NOSSA FUNÇÃO DENTRO DA ONCREATE.
SE ACASO NÃO DER CERTO E PORQUE FUNÇÃO ESTÁ NO LOCAL ERRADO!!!
A FUNÇÃO ESTÁ FORA DO ONCREATE, LEMBRA!!!!


```

31 |
32 |     bt_entrada.setOnClickListener(){ it: View!
33 |         val email = edit_email.text.toString()
34 |         val senha = edit_senha.text.toString()
35 |         if (email.isEmpty() || senha.isEmpty()) {
36 |             val mensagemErro = "Campos não preenchidos, tente novamente"
37 |             val snackbar = Snackbar.make(it, mensagemErro, Snackbar.LENGTH_LONG)
38 |             snackbar.show()
39 |         }
40 |     }
41 |
42 | }
43 | fun IniciarComponentes() {

```

NO FINAL DO NOSSO ONCREATE, ANTES DE FECHAR, CRIAMOS NOSSA FUNÇÃO QUE CUIDA DO BUTTON BT_ENTRADA, NESSA PRIMEIRA PARTE SIMULAMOS O ERRO DE VAZIO – FAÇA O TESTE E VEJA SE DEU A MENSAGEM DE ERRO

```
32
33     bt_entrada.setOnClickListener(){ it: View!
34         val email = edit_email.text.toString()
35         val senha = edit_senha.text.toString()
36         if (email.isEmpty() || senha.isEmpty()) {
37             val mensagemErro = "Campos não preenchidos, tente novamente"
38             val snackbar = Snackbar.make(it, mensagemErro, Snackbar.LENGTH_LONG)
39             snackbar.show()
40         }else{
41             AutenticarUsuario();
42         }
43     }
44
45 }
```

VEJA QUE A VERIFICAÇÃO SERÁ FEITA
PELA FUNÇÃO AUTENTICARUSUARIO()

```

45 }
46 fun AumentarUsuario(){
47     val email = edit_email.text.toString()
48     val senha = edit_senha.text.toString()
49
50     FirebaseAuth.getInstance().signInWithEmailAndPassword(email, senha).addOnCompleteListener() { task ->
51         if (task.isSuccessful) {
52             progressBar.visibility = View.GONE
53
54             val user = FirebaseAuth.getInstance().currentUser
55             // TODO: Navegar para a próxima tela ou atualizar a interface do usuário
56         } else {
57             task.exception?.message?.let { mensagemErro ->
58                 Snackbar.make(
59                     findViewById(android.R.id.content),
60                     text: "Erro ao autenticar usuário: $mensagemErro",
61                     Snackbar.LENGTH_LONG
62                 ).show()
63             }
64         }
65     }
66 }

```

- 1 – VARIÁVEIS QUE TRAZ OS VALORES
- 2 – FIREBASEAUTH É QUEM FAZ A AUTENTICAÇÃO
- 3 – NELE VEMOS SE CASO SUCESSO OU NÃO

New Android Activity

Empty Views Activity

Creates a new empty activity

Activity Name

TelaPrincipal

☒ Generate a Layout File

Layout Name

activity_tela_principal

☐ Launcher Activity

Package name

br.com.faculdade.imepac

Source Language

Kotlin

mLogin.kt

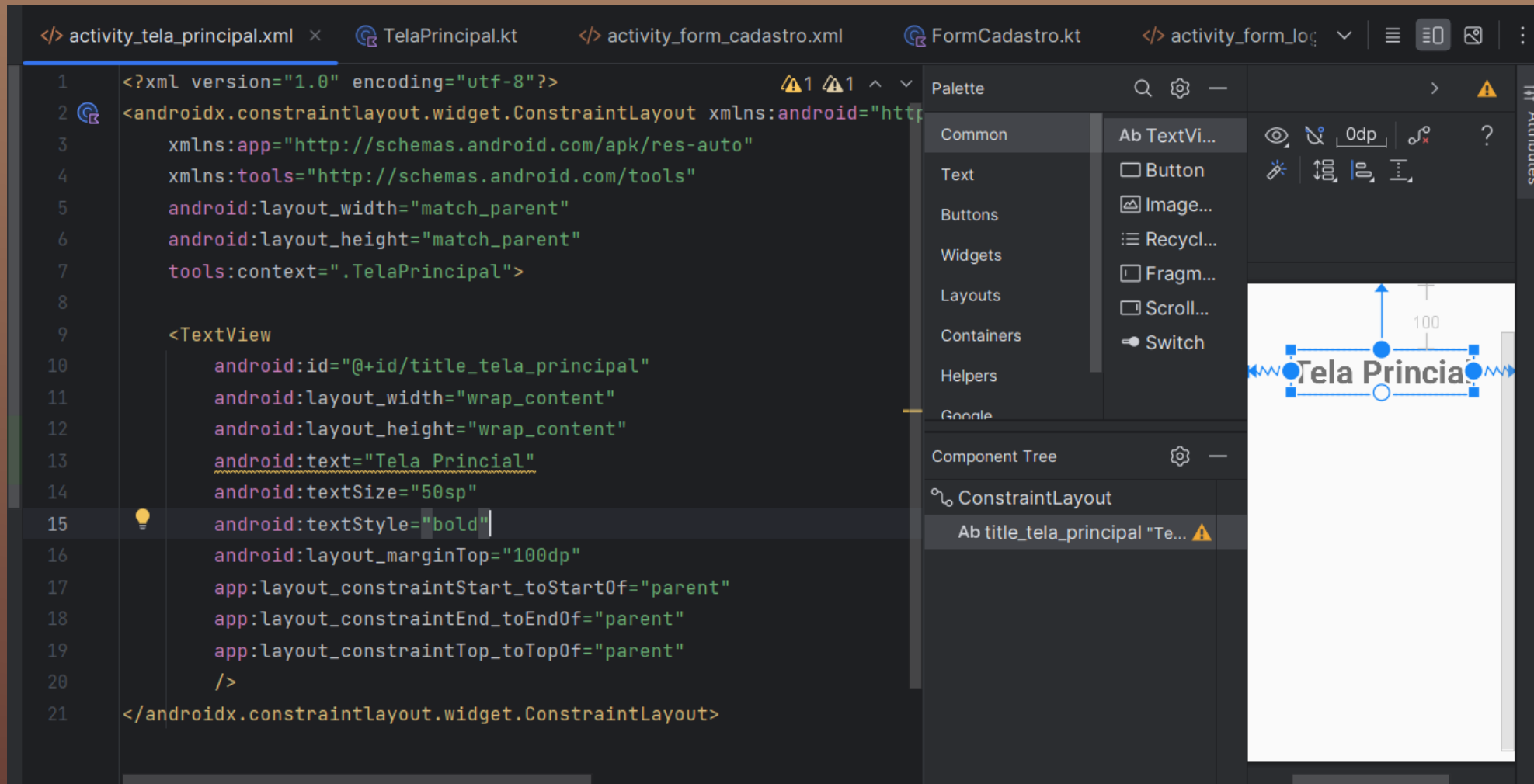
AndroidManifest.xml

activity_tela_principal

```
<activity
    android:name=".TelaPrincipal"
    android:exported="false"
    android:theme="@style/AppTheme" />
```

Coloque
nosso tema

CRIAR UMA TELA PRINCIPAL



APENAS UMA MENSAGEM - TÍTULO

```
FormLogin.kt x </> activity_tela_principal.xml </> activity_form_cadastro.xml FormCadastro.kt </> activity_form_login. v [ ] [ ]
46 fun AutenticarUsuario(){
47     val email = edit_email.text.toString()
48     val senha = edit_senha.text.toString()
49
50     FirebaseAuth.getInstance().signInWithEmailAndPassword(email, senha).addOnCompleteListener() { task ->
51         if (task.isSuccessful) {
52             progressbar.visibility = View.GONE
53
54             val user = FirebaseAuth.getInstance().currentUser
55             val intent = Intent(packageContext: this@FormLogin, TelaPrincipal::class.java)
56             startActivity(intent)
57             finish() // Finaliza a atividade atual para que o usuário não possa voltar para ela
58         } else {
59             task.exception?.message?.let { mensagemErro ->
60                 Snackbar.make(
61                     findViewById(android.R.id.content),
62                     text: "Erro ao autenticar usuário: $mensagemErro",
63                     Snackbar.LENGTH_LONG
64                 ).show()
65             }
66         }
67     }
68 }
69 fun IniciarComponentes() {
```

COMPLETO AGORA – ENVIO PARA A
TELAPRINCIPAL

Projeto IMEPAC ▾

Authentication

Usuários Método de login Modelos **Uso** Configurações Extensions

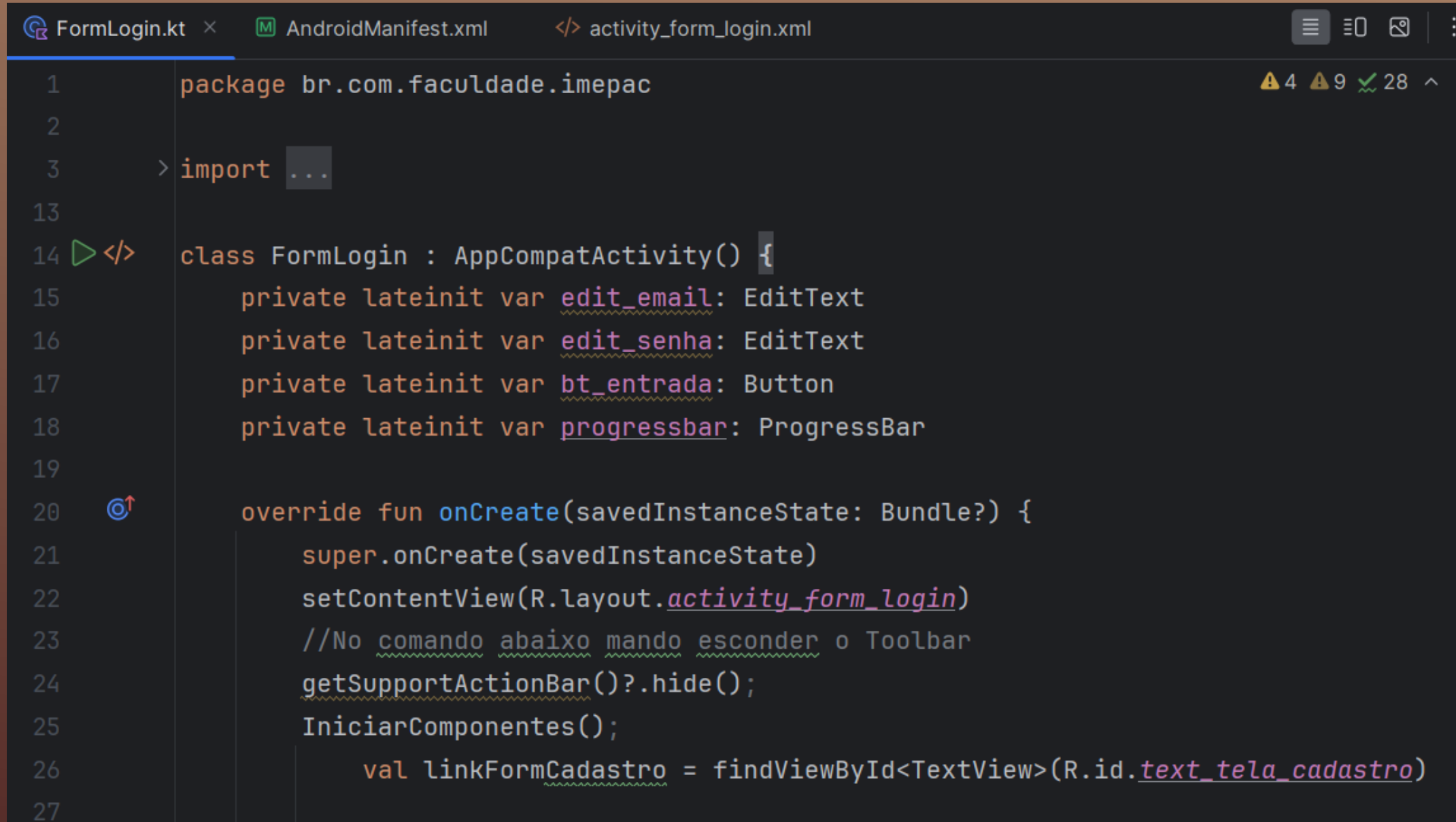
🔍 Pesquise por endereço de e-mail, número de telefone ou UID do usuário [Adicionar usuário](#) ↻ ⋮

Identificador	Provedores	Data de criação ↓	Último login	UID do usuário
joao@live.com	✉	4 de mar. de 2024	4 de mar. de 2024	ex4dGy74cSbeZH50hpZu5XYx...
maria@gmail.com	✉	3 de mar. de 2024	3 de mar. de 2024	tqIYityCU7dGPARlgJzkIgWW9...
crispim.silva@atos.net	✉	26 de fev. de 2024	26 de fev. de 2024	vi1gHJYwsXagQyhOUZBB2TD...
crispimluiz@live.com	✉	15 de fev. de 2024	15 de fev. de 2024	PrRJlqpaOFOWg6OUcwYec0D...
crispim.silva@imepac.e...	✉	12 de fev. de 2024	12 de fev. de 2024	T2pcaMgDQrY3piF1AC4A9Yx...

Linhas por página: 50 ▾ 1 – 5 of 5 < >

VAMOS TESTAR- PEGUE UM USUÁRIO,
LEMBRE A SENHA E TESTE

TELA INTEIRA



```
FormLogin.kt × AndroidManifest.xml </> activity_form_login.xml
1 package br.com.faculdade.imepac
2
3 > import ...
4
13
14 ▶ </> class FormLogin : AppCompatActivity() {
15     private lateinit var edit_email: EditText
16     private lateinit var edit_senha: EditText
17     private lateinit var bt_entrada: Button
18     private lateinit var progressbar: ProgressBar
19
20     override fun onCreate(savedInstanceState: Bundle?) {
21         super.onCreate(savedInstanceState)
22         setContentView(R.layout.activity_form_login)
23         //No comando abaixo mando esconder o Toolbar
24         getSupportActionBar()?.hide();
25         IniciarComponentes();
26         val linkFormCadastro = findViewById<TextView>(R.id.text_tela_cadastro)
27     }
```

```
27
28         linkFormCadastro.setOnClickListener{ it: View!
29             val intent = Intent(packageContext: this, FormCadastro::class.java)
30             startActivity(intent)
31         }
```

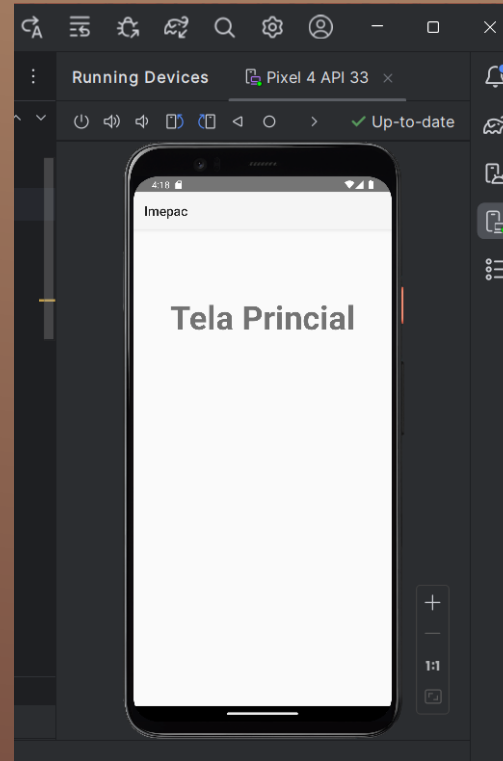
```
32
33         bt_entrada.setOnClickListener(){ it: View!
34             val email = edit_email.text.toString()
35             val senha = edit_senha.text.toString()
36             if (email.isEmpty() || senha.isEmpty()) {
37                 val mensagemErro = "Campos não preenchidos, tente novamente"
38                 val snackbar = Snackbar.make(it, mensagemErro, Snackbar.LENGTH_LONG)
39                 snackbar.show()
40             }else{
41                 AuntenticarUsuario();
42             }
43         }
44     }
45 }
```

```
46 fun AuntenticarUsuario() {
47     val email = edit_email.text.toString()
48     val senha = edit_senha.text.toString()
49
50     FirebaseAuth.getInstance().signInWithEmailAndPassword(email, senha)
51         .addOnCompleteListener { task ->
52             if (task.isSuccessful) {
53                 // Autenticação bem-sucedida
54                 progressbar.visibility = View.GONE
55                 val user = FirebaseAuth.getInstance().currentUser
56                 val intent = Intent(packageContext: this@FormLogin, TelaPrincipal::class.java)
57                 startActivity(intent)
58                 finish()
59             } else {
60                 // Autenticação falhou
61                 val mensagemErro = task.exception?.message
62                 Snackbar.make(
63                     findViewById(android.R.id.content),
64                     text: "Erro ao autenticar usuário: $mensagemErro",
```

```

63         findViewById(android.R.id.content),
64         text: "Erro ao autenticar usuário: $mensagemErro",
65         Snackbar.LENGTH_LONG
66     ).show()
67     }
68 }
69 }
70
71 fun IniciarComponentes() {
72     edit_email = findViewById(R.id.edit_email_login)
73     edit_senha = findViewById(R.id.edit_senha_login)
74     bt_entrada = findViewById(R.id.bt_entrada)
75     progressbar = findViewById(R.id.progressbar)
76     }
77 }

```



TELA PRINCIPAL