

Linguagem Técnica de Programação Mobile

AULA 9 - Usando Banco de Dados no Android – SQLite – Parte 1

Prof. João Paulo Pimentel
joao.pimentel@projecao.br



SQLite do Android



- No Android existe um **pacote** chamado “**android.sqlite**” para o tratamento de **banco de dados**.
 - Uma das classes do pacote “**android.sqlite**” é a **SQLiteDataBase**, que possui os seguintes métodos, conforme é mostrado nos próximos slides.
-

SQLite do Android

- 1) O método **query**
 - O método **query** realiza uma **consulta SQL** no banco de dados (equivale ao comando **SELECT**).
 - Essa função retorna uma instância (objeto) do tipo **Cursor**. Vejamos a sintaxe do método em seguida:
 - **Cursor** `query(String <tabela>, String[] <colunas>, String <condicao_de_busca>, String[] <selecionArgs>, String <groupby>, String <having>, String <orderby>)`
-

SQLite do Android

- Vamos conhecer os parâmetros que iremos utilizar dessa função:
 - **<tabela>** : Neste parâmetro você informa o **nome da tabela**.
 - **<coluna>** : Neste parâmetro **a(s) coluna(s)** que o banco possui.
 - **<condição de busca>**: Aqui você informa a condição de busca de dados. Este parâmetro funciona como se fosse a **clausula where** do SQL, onde é informada a condição de busca.
 - O restante dos parâmetros iremos trabalhar em seu valor **null**.
-

SQLite do Android

- 2) O método **insert**
 - O método **insert** realiza uma inserção de dados na tabela (equivale ao comando **INSERT** do **SQL**).
Vejam os a sintaxe deste método:
 - ```
long insert(String <tabeLa>,
 String <nullColumnHack>,
 ContentValues <values>)
```
-

# SQLite do Android

- Essa função possui **três** parâmetros:
  - **<tabela>** : Neste parâmetro você informa o nome da tabela.
  - **<nullColumnHack>** : O SQL não permite a inserção de linhas em branco, logo, se o valor de uma coluna for vazio, ele será iniciado com o valor **null**.
  - **<values>** : Este parâmetro possui os valores a serem adicionados na tabela.
-

# SQLite do Android

- 3) O método **update**
  - O método **update** realiza uma atualização de dados na tabela (equivale ao comando **UPDATE** do **SQL**).  
Vejam agora a sintaxe deste método :
  - ```
int update (String <tabela>,  
            ContentValues <values>,  
            String <condição>,  
            String <argumentos_condicao>)
```
-

SQLite do Android

- Essa função possui **quatro** parâmetros:
 - **<tabela>** : Neste parâmetro você informa o nome da tabela.
 - **<values>** : Este parâmetro possui os valores a serem adicionados na tabela.
 - **<condição>**: Aqui você informa a condição para a realização da atualização dos dados.
 - **<argumentos_condicao>**: Aqui você informa os argumentos relativos à condição informada.
-

SQLite do Android

- 4) O método **delete**
 - O método **delete** realiza a remoção de dados na tabela (equivale ao comando **DELETE** do **SQL**).
Veamos a sintaxe deste método:
 - *int* **delete** (*String* **<tabela>**,
 String **<condição>**,
 String **<argumentos_condicao>**)
-



SQLite do Android



- Essa função possui **três** parâmetros:
 - **<tabela>** : Neste parâmetro você informa o nome da tabela.
 - **<condição>**: Aqui você informa a condição para a realização da remoção dos dados da tabela.
 - **<argumentos_condicao>**: Aqui você informa os argumentos relativos à condição informada.
-

SQLite do Android

- 5) O método **execSQL**
 - O método **execSQL** executa uma consulta SQL (como **CREATE TABLE**, **INSERT INTO**, **UPDATE**, **DELETE** e etc.). Não é possível usar a clausula **SELECT** nesta função. Para esse tipo de situação, use o método **query**.
 - Vejamos a sintaxe deste método:
 - *void* **execSQL**(*String sql*)
-

SQLite do Android

- Vamos fazer algumas comparações da função “**execSQL**”, que permite sintaxes de comando SQL com as demais funções, como “**update**”, “**insert**” e “**delete**” para a realização de uma consulta SQL.
-

SQLite do Android

- Exemplos de SQL usando as funções do Android.
- 1) Fazendo uma consulta a tabela (usando **SELECT**)

- Comando SQL :

```
select codusuario, nome, idade  
from cadastro
```

- Usando a função query :

```
query("cadastro", (new String[]  
{"codusuario", "nome", "idade"}), null, null, null,  
null, null);
```

SQLite do Android

- Exemplos de SQL usando as funções do Android.
- 1.1) Agora com a cláusula “where”:

- Comando SQL :

```
select nome  
from cadastro  
where idade > 24
```

- Usando a função query :

```
query("cadastro", (new String[] {"nome"}), "idade > 24",  
    null, null, null, null);
```

SQLite do Android

- Exemplos de SQL usando as funções do Android.
- 2) Inserindo dados (usando **INSERT**)

- Comando SQL :

```
insert into cadastro(nome,idade)  
values('Luciano',23)
```

- Usando a função execSQL

```
execSQL("insert into cadastro(nome,idade) values('Luciano',23); ");
```

Usando a função insert :

```
ContentValues valor = new ContentValues();
```

```
valor.put("nome", "Luciano" );  
valor.put("idade", "23");
```

```
insert("cadastro",null,valor);
```

SQLite do Android

- Exemplos de SQL usando as funções do Android.
- 3) Atualizando dados (usando **UPDATE**)

- Comando SQL :

```
update cadastro set idade = 27  
where codusuario=1
```

- Usando a função execSQL

```
execSQL("update cadastro set idade = 27 where (codusuario=1); ");
```

Usando a função update :

```
ContentValues valor = new ContentValues();  
  
valor.put("idade", "27");  
  
update("cadastro", valor, "codusuario=1", null);
```

SQLite do Android

- Exemplos de SQL usando as funções do Android.
- 4) Removendo dados (usando **DELETE**)

- Comando SQL :

```
delete from cadastro where codusuario=1
```

- Usando a função execSQL

```
execSQL("delete from cadastro where (codusuario=1); ");
```

Usando a função delete :

```
delete("cadastro", "codusuario=1",null);
```

SQLite do Android

- Exemplos de SQL usando as funções do Android.
- 5) Criando uma tabela (usando **CREATE TABLE**)

- Comando SQL :

```
create table cadastro(codusuario integer primary key autoincrement, nome text not null, idade integer not null)
```

- Usando a função execSQL

```
execSQL("create table cadastro(codusuario integer primary key autoincrement, nome text not null, idade integer not null); ");
```

SQLite do Android

- Agora vamos conhecer as funções responsáveis por criar e abrir banco de dados no Android.

```
SQLiteDatabase openOrCreateDatabase(String nome_do_banco,int mode,  
CursorFactory cf);
```

- Essa função abre ou cria um novo banco de dados. Você deve especificar o nome do banco e o modo de abertura (somente leitura ; somente escrita e etc.) e um terceiro parâmetro, que normalmente é **null**. Veja um exemplo abaixo:

```
SQLiteDatabase db;  
  
db = openOrCreateDatabase("dbbanco", Context.MODE_PRIVATE, null);
```

SQLite do Android

- O comando **openOrCreateDatabase** abre ou cria o banco de dados chamado “**dbbanco**”.
 - Quando realizamos uma consulta do tipo **SELECT** usando a função **query**, ela retorna um objeto do tipo **Cursor**, onde nela estão armazenados os registros solicitados pela consulta.
 - Vamos ver nos próximos slides os métodos da classe **Cursor**:
-

Métodos da classe Cursor

Método	Descrição
<code>boolean moveToFirst()</code>	Move o cursor para o primeiro registro da tabela.
<code>boolean moveToPrevious()</code>	Move o cursor para o registro anterior da tabela.
<code>boolean moveToNext()</code>	Move o cursor para o próximo registro da tabela.
<code>boolean moveToLast()</code>	Move o cursor para o último registro da tabela.
<code>int getCount()</code>	Retorna o número de registros da tabela.
<code>int getColumnIndex(String columnName)</code>	Retorna o índice da coluna na tabela, através do seu nome, que é passado como parâmetro.
<code>String getColumnName(int columnIndex)</code>	Retorna o nome da coluna na tabela, através do seu índice, que é passado como parâmetro.

Métodos da classe Cursor

Método	Descrição
<code>int getInt(int columnIndex)</code>	Retorna o valor do campo, tendo como seu parâmetro o seu índice, convertido em int. Lembre-se : o índice do primeiro campo é 0, o índice do segundo campo é 1 e assim por diante.
<code>float getFloat(int columnIndex)</code>	Retorna o valor do campo, tendo como seu parâmetro o seu índice, convertido em float. Lembre-se : o índice do primeiro campo é 0, o índice do segundo campo é 1 e assim por diante.
<code>double getDouble(int columnIndex)</code>	Retorna o valor do campo, tendo como seu parâmetro o seu índice, convertido em double. Lembre-se : o índice do primeiro campo é 0, o índice do segundo campo é 1 e assim por diante.
<code>short getShort(int columnIndex)</code>	Retorna o valor do campo, tendo como seu parâmetro o seu índice, convertido em short. Lembre-se : o índice do primeiro campo é 0, o índice do segundo campo é 1 e assim por diante.

SQLite do Android

- Agora um exemplo para que você tenha um melhor entendimento do mecanismo de banco de dados do Android.
- Imagine uma tabela chamada “**cadastro**” com os seguintes dados abaixo:

nome	idade
Amanda	32
Bianca	30
Bruna	23
Carla	20

SQLite do Android

- Agora, observe a linha de código abaixo:

```
SQLiteDatabase db;
```

```
db = openOrCreateDatabase("dbbanco",  
Context.MODE_PRIVATE, null);
```

```
Cursor c = db.query("cadastro", (new String[] {"nome","idade"}), "idade <  
32", null, null, null, null);
```

- Observe que a linha acima cria um objeto do tipo **Cursor** que vai receber o resultado da consulta da função **query**, que retorna uma instância do mesmo tipo.
 - Logo, a instância retornada pela função **query** na verdade, retorna uma tabela resultante da consulta.
-

SQLite do Android

- Veja o resultado abaixo da execução da query :

nome	idade
Bianca	30
Bruna	23
Carla	20

SQLite do Android

- Continuando a codificação. Veja a linha abaixo:

```
c.moveToFirst();
```

- A linha acima coloca o ponteiro no primeiro registro da tabela. Já a linha:

```
String nome = c.getString(0);
```

- Retorna o valor do campo “**nome**” do primeiro registro, no caso, “**Bianca**”. Veja a próxima linha:

```
int idade = c.getInt(1);
```

- Retorna o valor do campo “**idade**” do primeiro registro, no formato **int**. Neste caso, o valor retornado é **30**.
-

SQLite do Android

- A linha:

```
c.moveToNext();
```

- A linha acima avança para o próximo registro. E aí na sequência a linha:

```
nome = c.getString(0);
```

- Retorna o valor do campo “nome” do segundo registro, no caso, “**Bruna**”. Veja agora a próxima linha:

```
int idade = c.getInt(1);
```

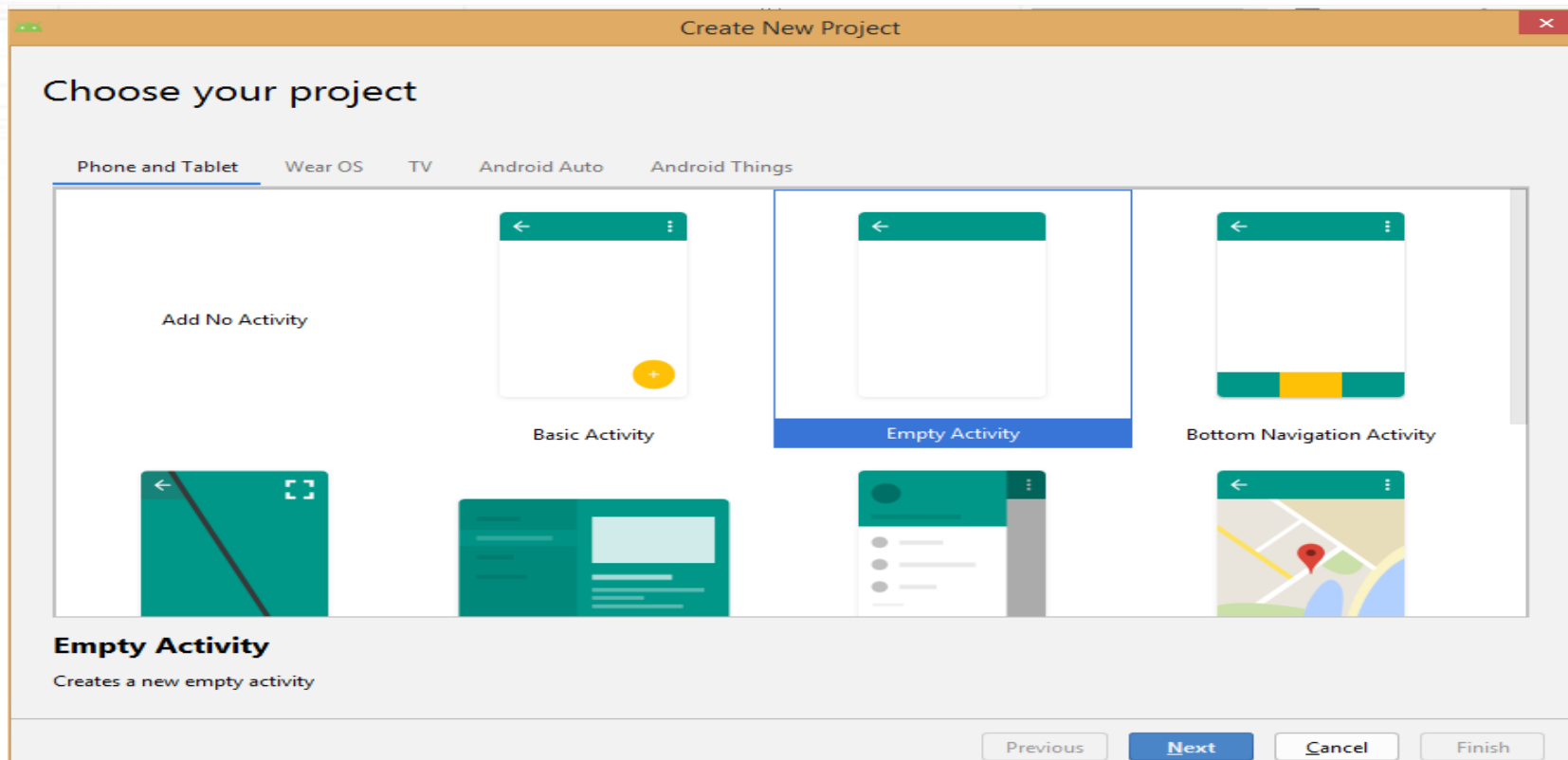
- Retorna o valor do campo “**idade**” do segundo registro, no formato **int**. Neste caso, o valor retornado é **23**.
-

Concluindo... SQLite do Android

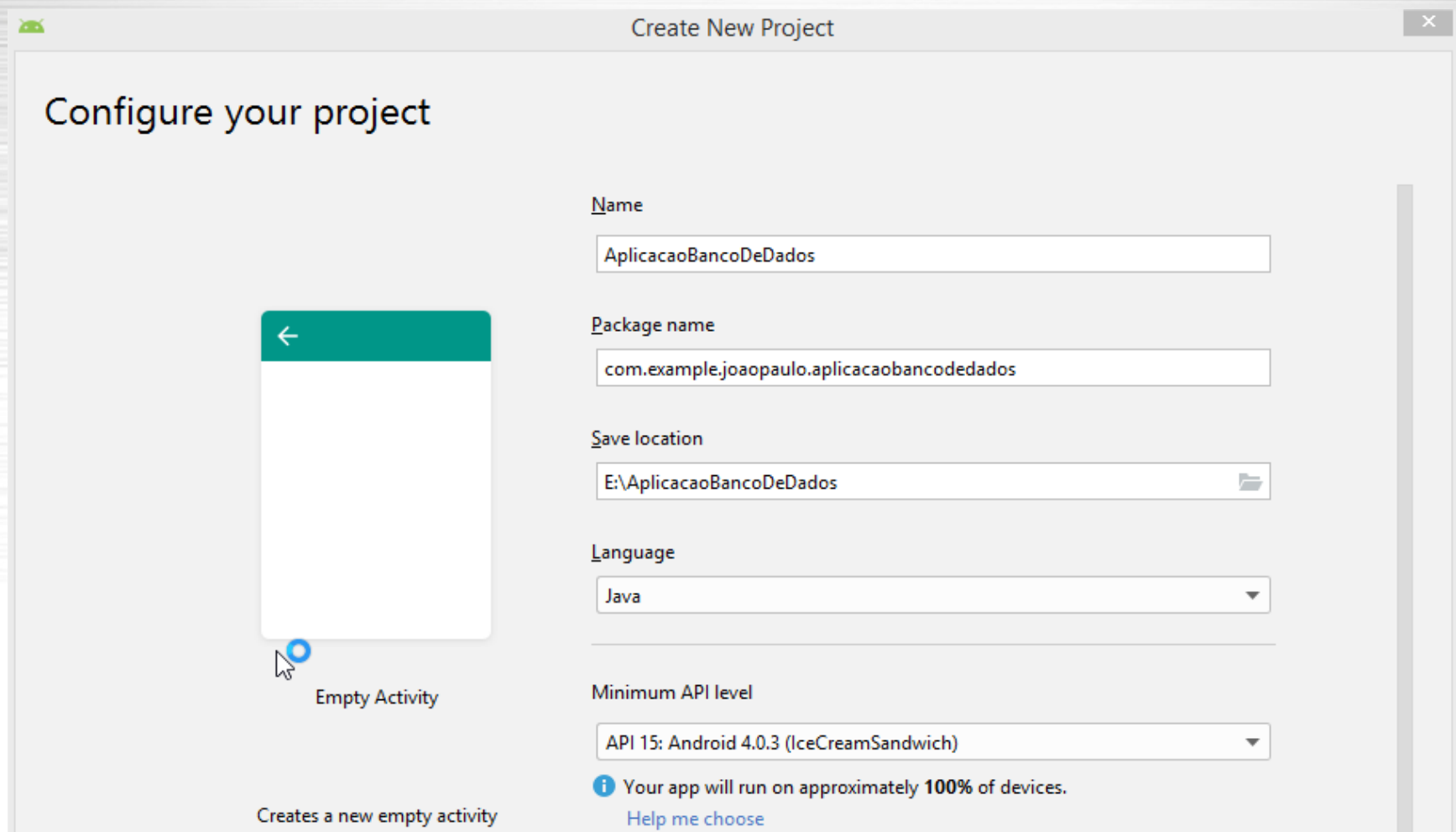
- Bom, com certeza você entendeu como funciona o mecanismo de manipulação de banco de dados no Android.
 - Agora vamos por esse aprendizado na prática, começando desenvolvendo algumas aplicações Android básicas.
 - Mãos a obra?????
 - Primeiramente iremos construir uma aplicação que irá criar o Banco de Dados (**banco_dados**), bem como nossa tabela de usuários (**usuarios**) com os campos (**numreg, nome e email**).
-

Aplicação Android com Banco de Dados

- A nossa primeira aplicação que iremos construir irá criar o nosso **banco**, como também a nossa **tabela** que iremos utilizar para inserir e guardar as nossas informações.
- Para isso, com o Android Studio aberto, vamos criar um **novo projeto** de acordo com as informações abaixo:



Aplicação Android com Banco de Dados



Create New Project

Configure your project

Name
AplicacaoBancoDeDados

Package name
com.example.joaopaulo.aplicacaobancodedados

Save location
E:\AplicacaoBancoDeDados

Language
Java

Minimum API level
API 15: Android 4.0.3 (IceCreamSandwich)

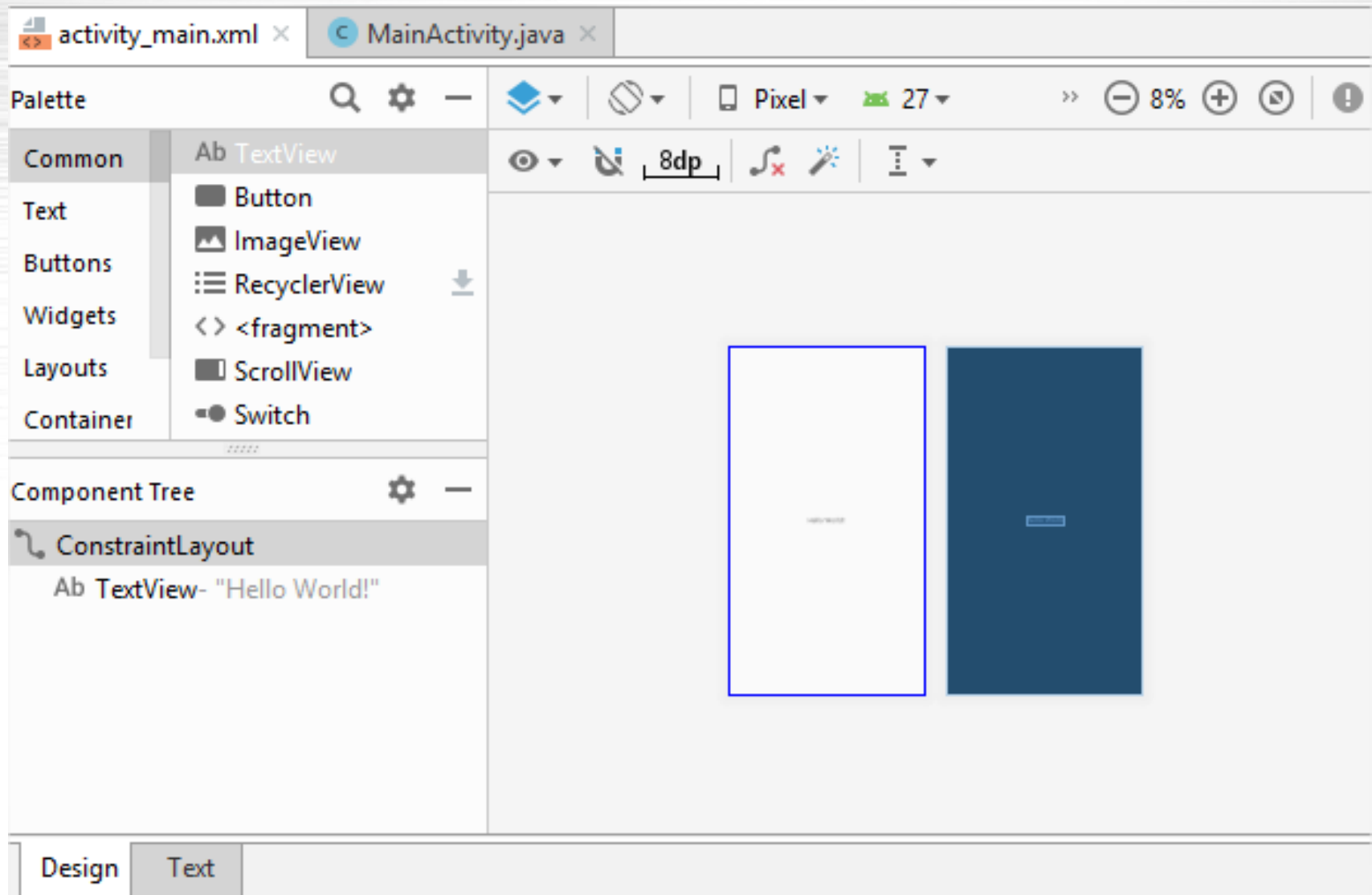
Empty Activity
Creates a new empty activity

Information
Your app will run on approximately **100%** of devices.
[Help me choose](#)

Name: **AplicacaoBancoDeDados**

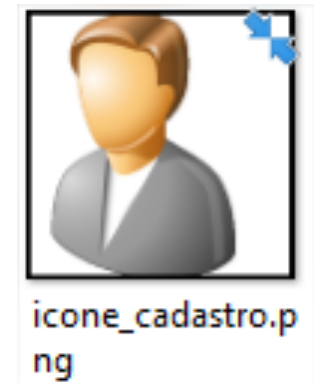
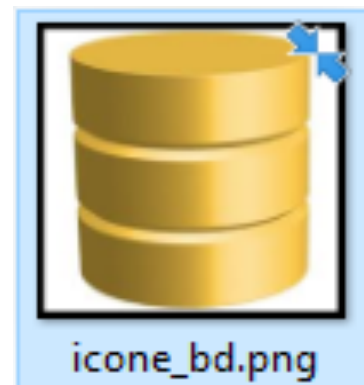
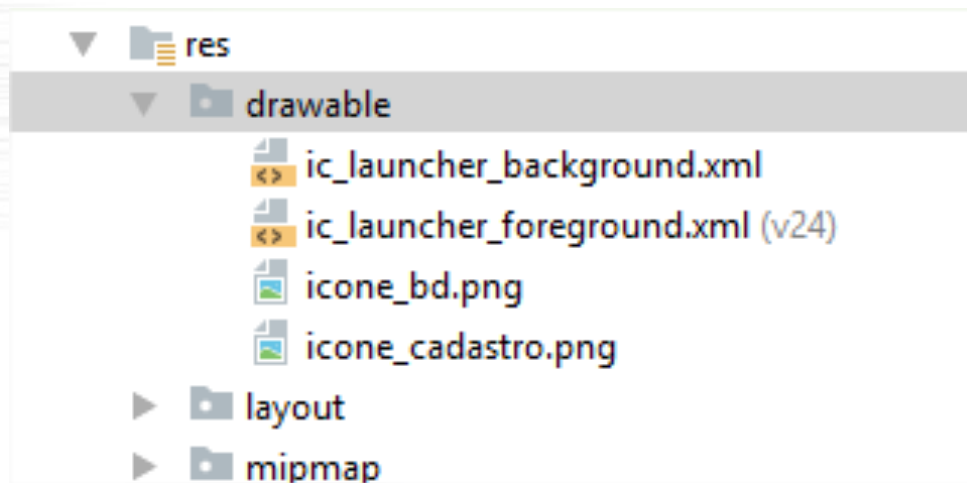
Aplicação Android com Banco de Dados

- Vejam como ficou:



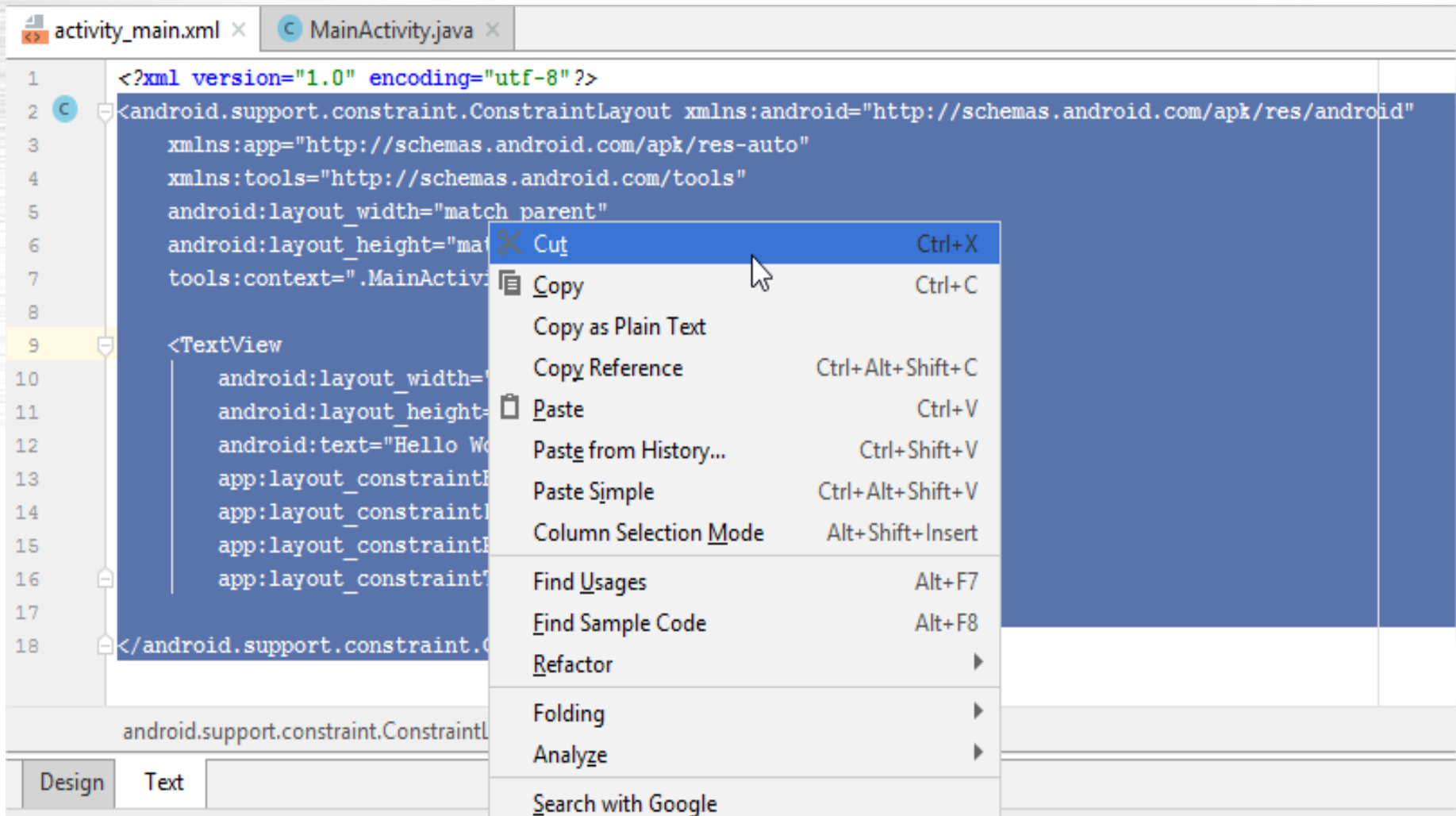
Aplicação Android com Banco de Dados

- Depois de criar o nosso projeto no Android Studio vamos colocar dentro do diretório “**drawable**” (diretório de imagens, presente dentro da pasta “res” do projeto) as imagens “icone_bd.png” e icone_cadastro.png que estão disponibilizadas no Blog.



Aplicação Android com Banco de Dados

- Vamos recortar/deletar todo o código XML da aplicação conforme abaixo:



Aplicação Android com Banco de Dados

- Vamos escrever o seguinte código XML para a tela da nossa aplicação:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3ba0e2" >
        <ImageView
            android:id="@+id/imageView1"
            android:layout_width="96dp"
            android:layout_height="96dp"
            android:src="@drawable/icone_bd" />
    </LinearLayout>
</LinearLayout>
```

Aplicação Android com Banco de Dados

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
/>
```

```
<LinearLayout
    android:id="@+id/LayoutCorFundo"
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:background="#E5BE40"
    android:gravity="center" >
```

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Trabalhando com Banco de
    Dados no Android"
    android:textAppearance=
    "?android:attr/textAppearanceLarge"
    android:textColor="#ffffff" />
```

Aplicação Android com Banco de Dados

```
</LinearLayout>
```

```
</LinearLayout>
```

```
<LinearLayout
```

```
    android:id="@+id/LayoutImagemFundo"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="fill_parent"
```

```
    android:orientation="vertical" >
```

```
    <TextView
```

```
        android:id="@+id/textView4"
```

```
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
```

```
        android:text="Escolha uma das opções abaixo:"
```

```
        android:textAppearance="?android:attr/textAppearanceLarge"
```

```
        android:textSize="20sp" />
```

```
    <LinearLayout
```

```
        android:layout_width="match_parent"
```

```
        android:layout_height="fill_parent"
```

```
        android:gravity="center"
```

```
        android:orientation="vertical" >
```

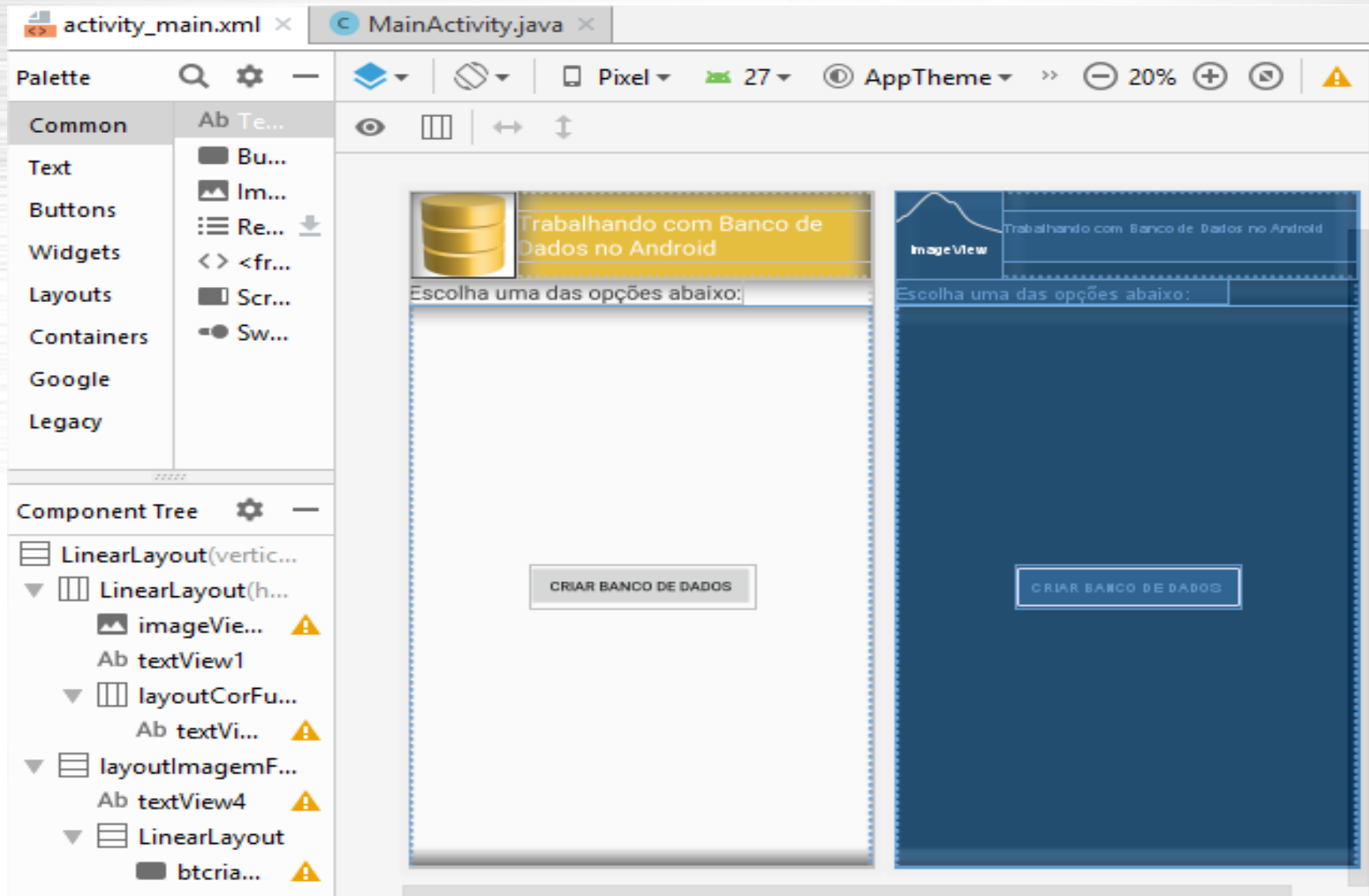
Aplicação Android com Banco de Dados

- Última parte do nosso código XML:

```
<Button
    android:id="@+id/btcriarbanco"
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:text="Criar Banco de Dados" />
</LinearLayout>
</LinearLayout>
</LinearLayout>
```

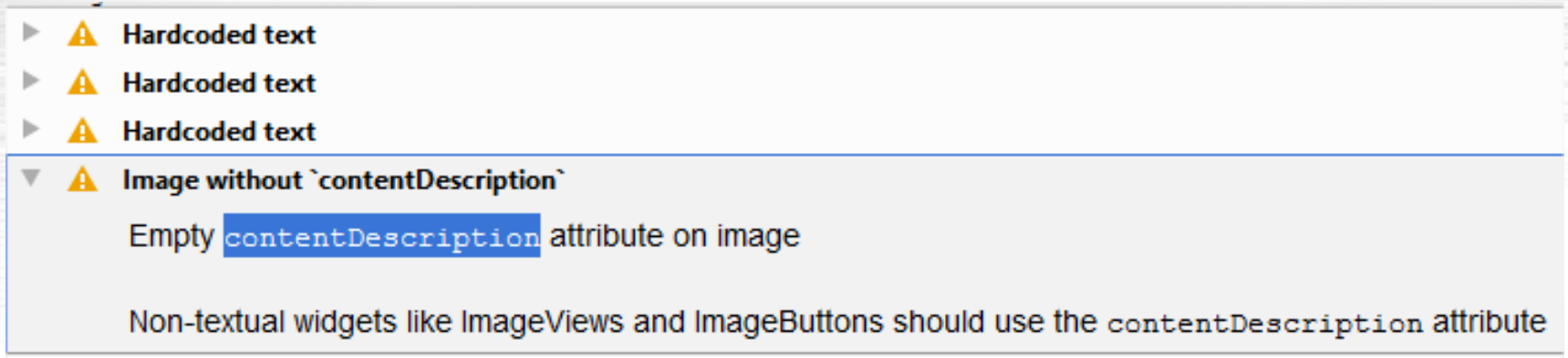
Aplicação Android com Banco de Dados

- Feito isso teremos o seguinte resultado abaixo:

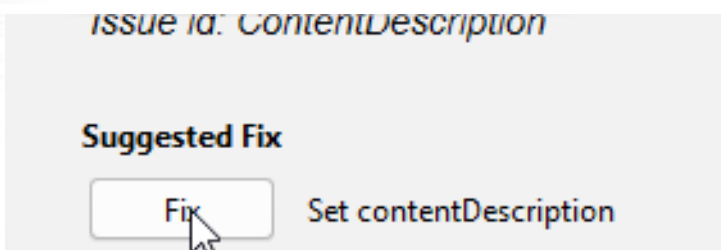


Aplicação Android com Banco de Dados

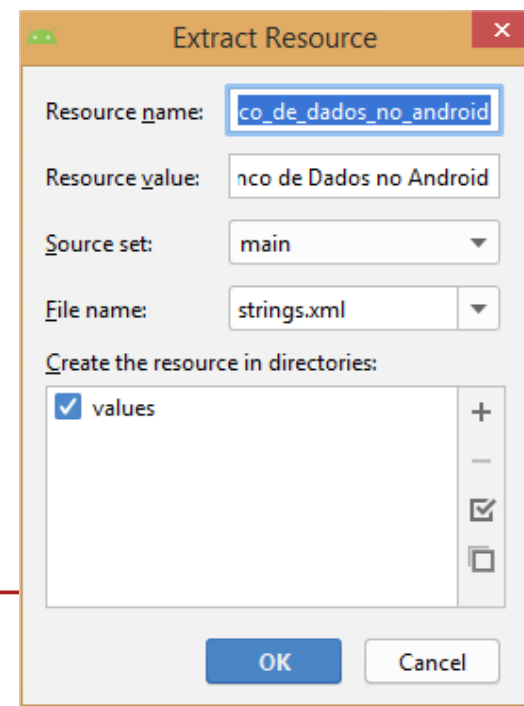
- Como resolver as alertas?



A screenshot of the Android Studio interface showing a list of lint warnings. The first three warnings are 'Hardcoded text'. The fourth warning, 'Image without `contentDescription`', is expanded. It shows the message: 'Empty `contentDescription` attribute on image' and a note: 'Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute'.



A screenshot of the 'Suggested Fix' panel for the 'Image without contentDescription' warning. It shows the text 'Issue id: ContentDescription' and a 'Suggested Fix' section with a 'Fix' button and the text 'Set contentDescription'.

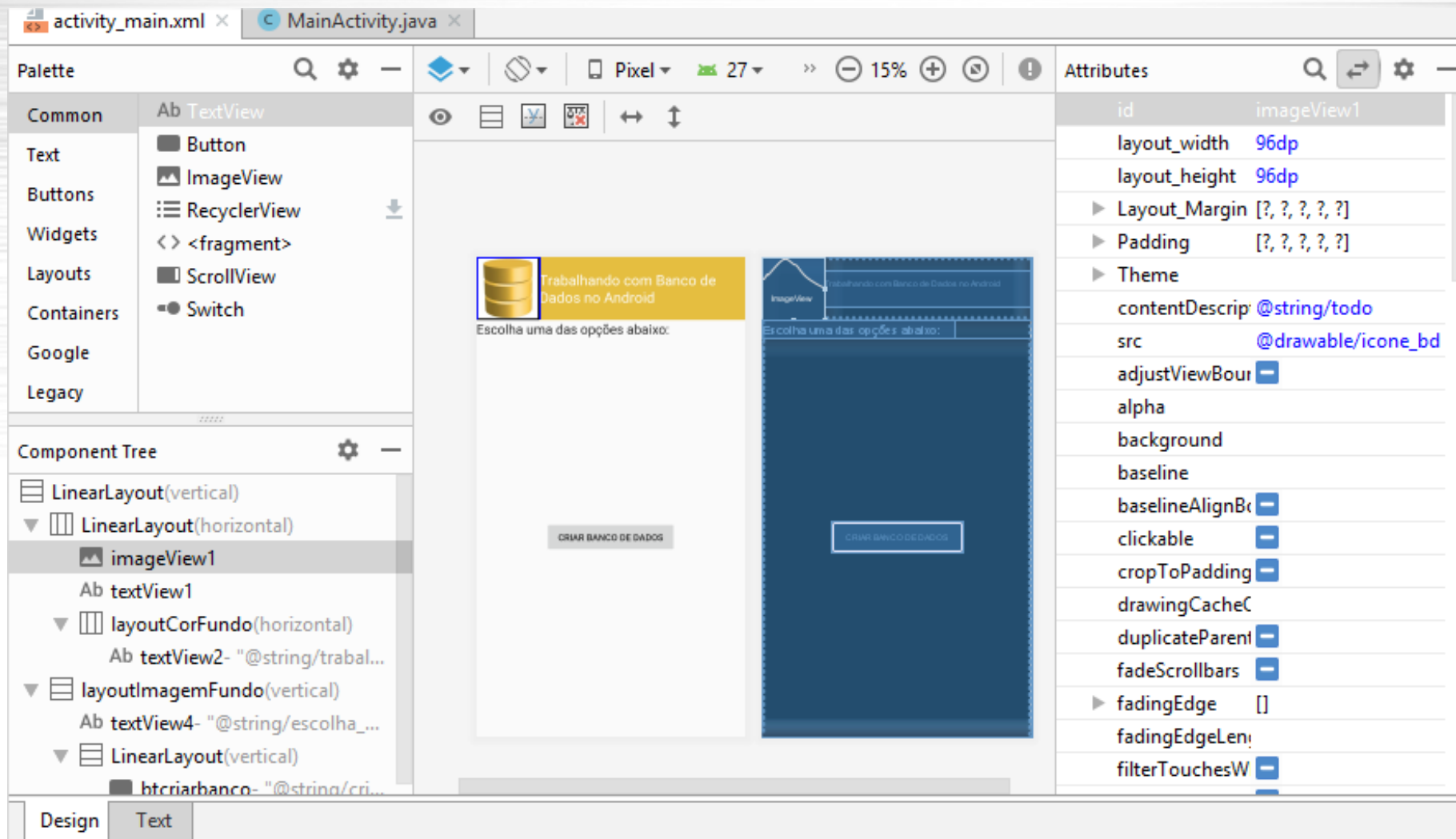


A screenshot of the 'Extract Resource' dialog box. It contains the following fields and options:

- Resource name: `co_de_dados_no_android`
- Resource value: `co de Dados no Android`
- Source set: `main`
- File name: `strings.xml`
- Create the resource in directories: ☒ values
- Buttons: OK, Cancel

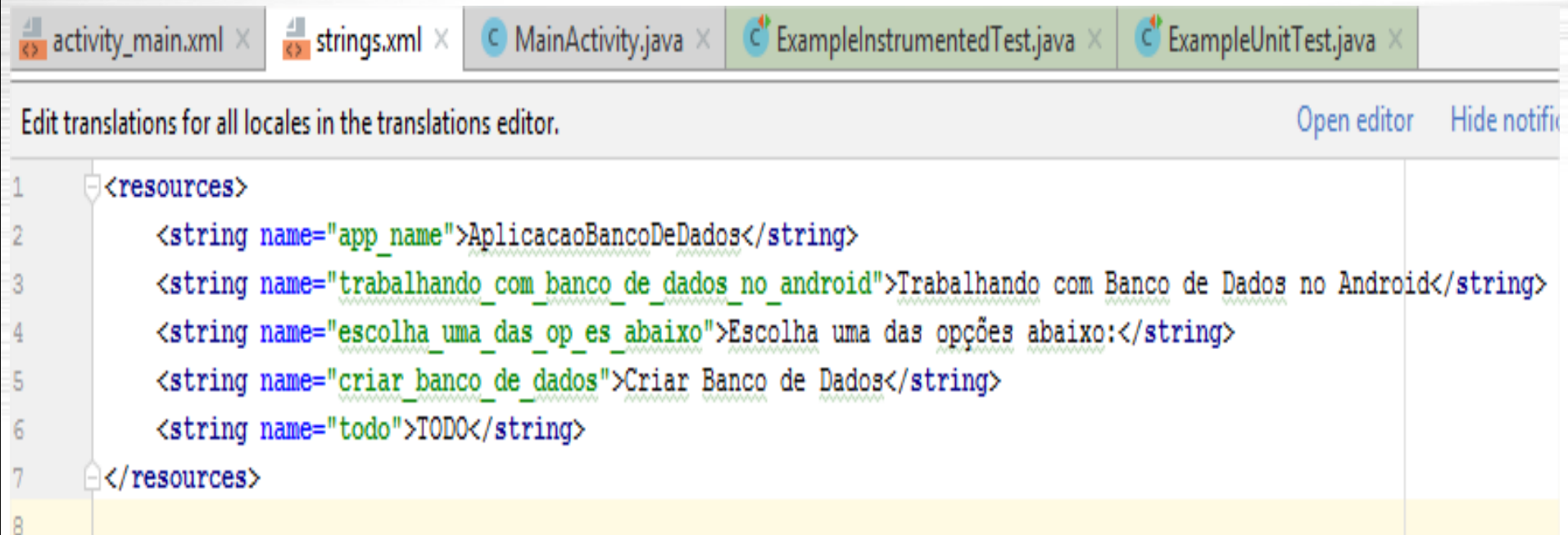
Aplicação Android com Banco de Dados

- Feito isso teremos o seguinte resultado abaixo:



Aplicação Android com Banco de Dados

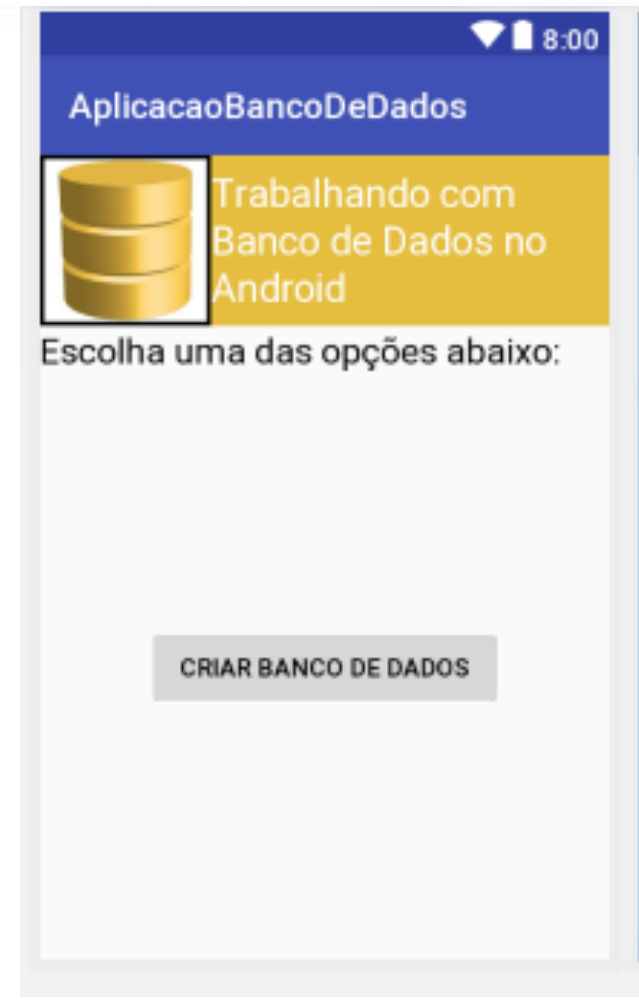
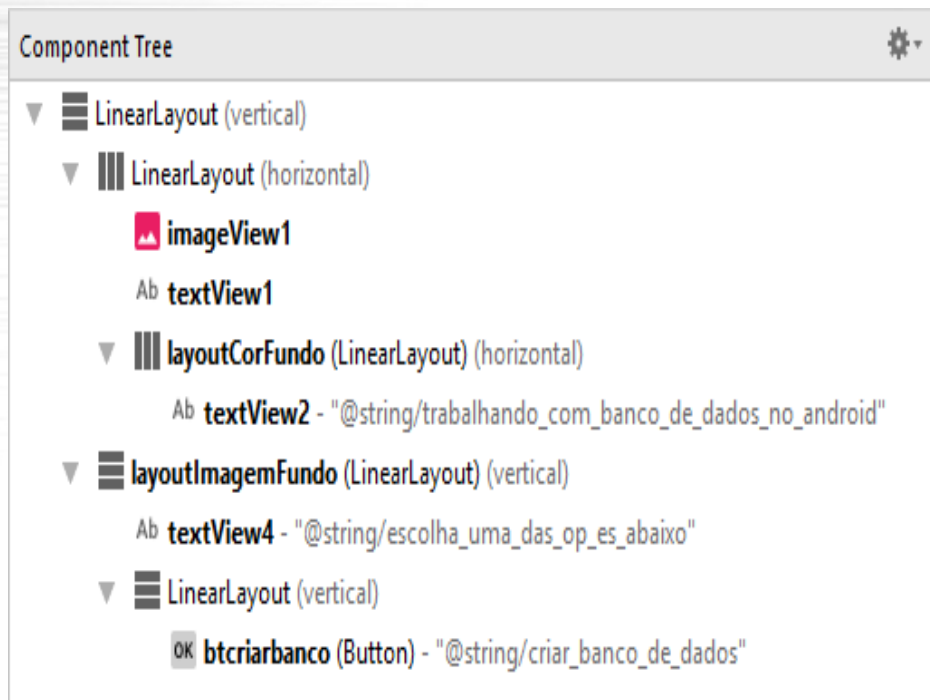
- string.xml ficará assim:



```
1 <resources>
2     <string name="app_name">AplicacaoBancoDeDados</string>
3     <string name="trabalhando_com_banco_de_dados_no_android">Trabalhando com Banco de Dados no Android</string>
4     <string name="escolha_uma_das_opcoes_abaxo">Escolha uma das opções abaixo:</string>
5     <string name="criar_banco_de_dados">Criar Banco de Dados</string>
6     <string name="todo">TODO</string>
7 </resources>
8
```

Aplicação Android com Banco de Dados

- Feito isso teremos o seguinte resultado abaixo:



Aplicação Android com Banco de Dados

- Agora vamos abrir o arquivo “**MainActivity.java**” para digitarmos o seguinte código seguinte:

```
1      package com.example.joaopaulo.aplicacaobancodedados;
2
3      import android.app.Activity;
4      import android.os.Bundle;
5      import android.app.AlertDialog;
6      import android.content.Context;
7      import android.database.sqlite.SQLiteDatabase;
8      import android.view.View;
9      import android.widget.*;
10
```

Aplicação Android com Banco de Dados

```
11 public class MainActivity extends Activity {  
12  
13     Button btcriarbanco;  
14     SQLiteDatabase db;  
15  
16     @Override  
17     protected void onCreate(Bundle savedInstanceState) {  
18         super.onCreate(savedInstanceState);  
19         setContentView(R.layout.activity_main);  
20         btcriarbanco = findViewById(R.id.btcriarbanco);  
21         btcriarbanco.setOnClickListener(new View.OnClickListener() {  
22  
23
```

Aplicação Android com Banco de Dados

```
23
24
25 ①↑
26
27     @Override
28     public void onClick(View view) {
29         try {
30             db = openOrCreateDatabase( name: "banco_dados",
31                                     Context.MODE_PRIVATE, factory: null);
32             db.execSQL("create table if not exists " +
33                 " usuarios(numreg integer primary key " +
34                 " autoincrement, nome text not null, telefone text " +
35                 " not null, " + " email text not null) ");
36             AlertDialog.Builder dialogo = new
37                 AlertDialog.Builder( context: MainActivity.this);
38             dialogo.setTitle("Aviso")
39                 .setMessage("Banco de dados criado com sucesso!")
40                 .setNeutralButton("OK", null)
41                 .show();
42         } catch (Exception e) {
43         }
44     }
45
46     });
47
48 }
49
50 }
```

Aplicação Android com Banco de Dados

- Comentando sobre algumas linhas de código do programa. A instrução :

```
db = openOrCreateDatabase("banco_dados",  
Context.MODE_PRIVATE, null);
```

- Cria (ou abre, caso já exista) o nosso banco de dados, cujo nome se chama “banco_dados” através do método **openOrCreateDataBase** do objeto *db* .
-

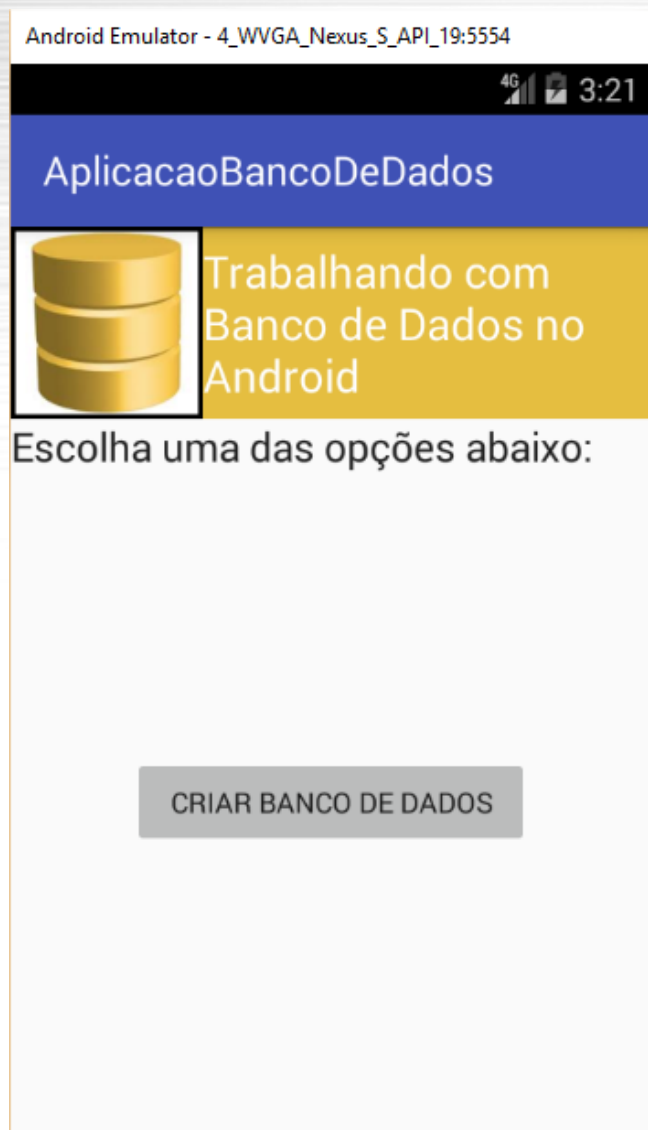
Aplicação Android com Banco de Dados

- A próxima instrução :

```
db.execSQL("create table if not exists " +  
" usuarios(numreg integer primary key " +  
" autoincrement, nome text not null, telefone text " +  
+ "not null," + "email text not null)");
```

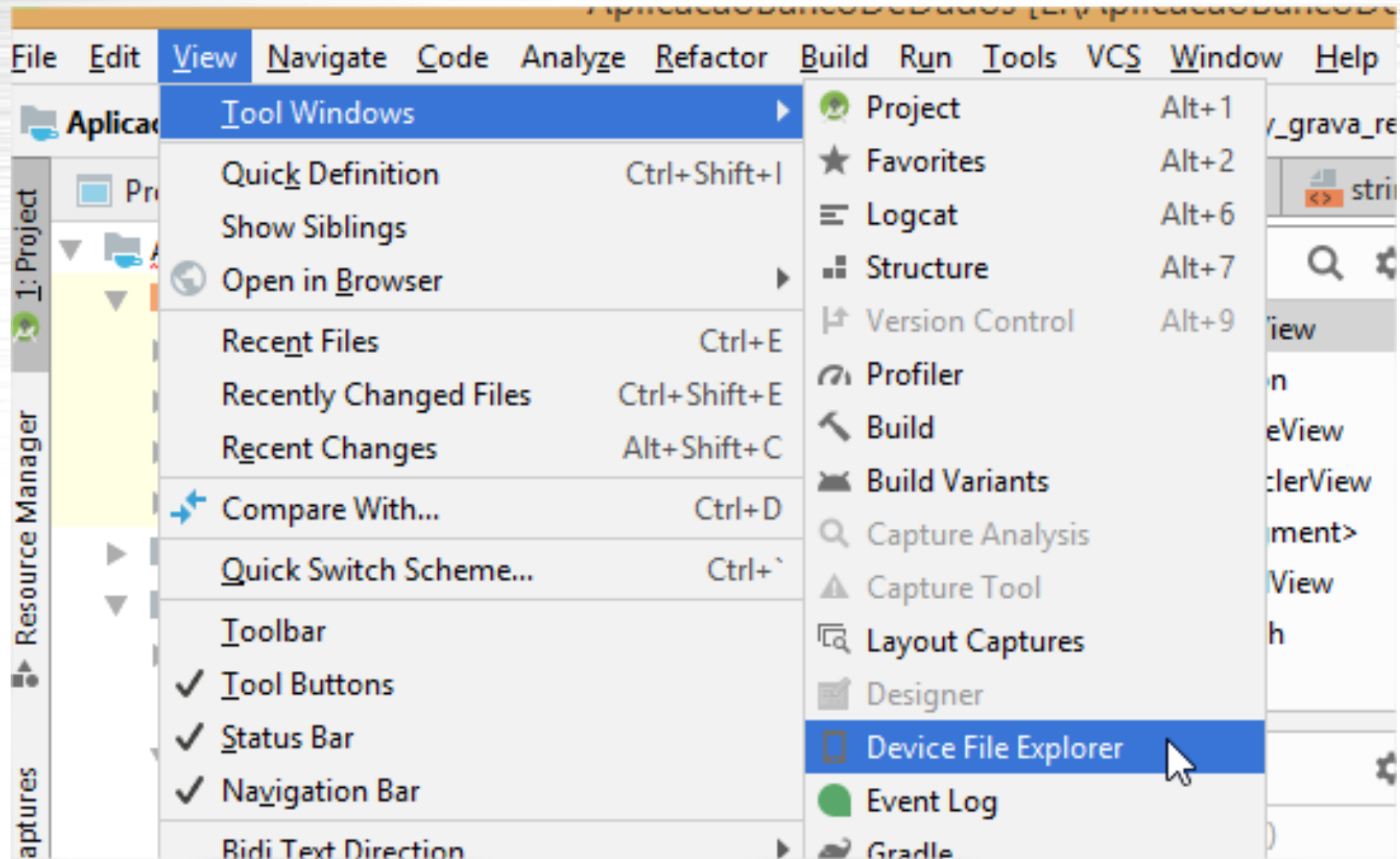
- Cria uma tabela (caso ela não exista) dentro do banco de dados chamada “usuarios”, com os seus devidos campos, através do método **execSQL**. No parâmetro do método passamos uma “string” que contém uma instrução SQL do tipo **create table**, que irá criar a tabela que iremos trabalhar em nossa aplicação.
-

Executando a aplicação



Verificando se o Banco foi criado

- View → Tool Windows → Device File Explorer



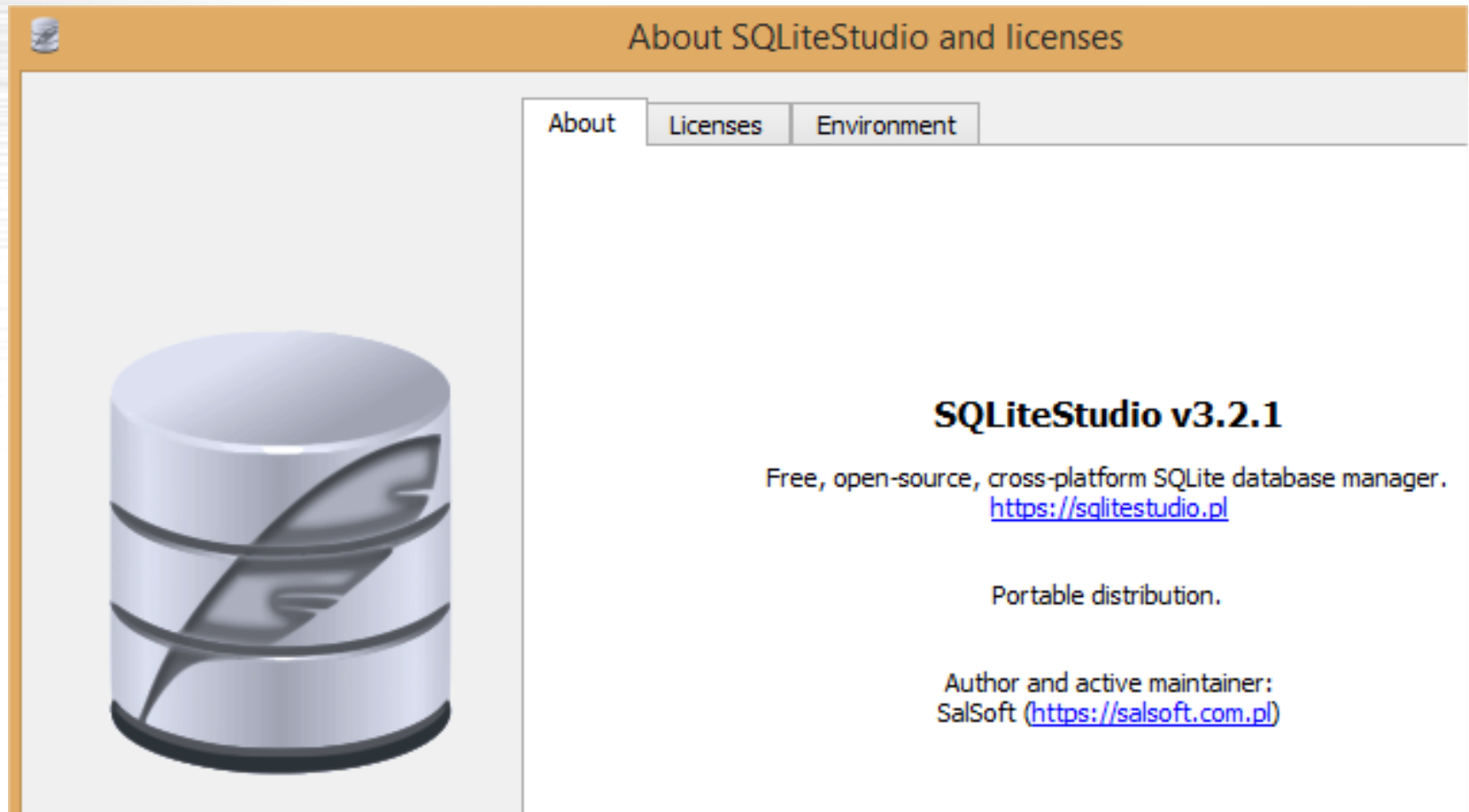
Verificando se o Banco foi criado

- O diretório onde o nosso banco de dados foi criado dentro do Android é o:
- “data/data/usuario.app.aplicacaobancodedados/databases/”.

>	jp.co.omronsoft.openwnn		2017-10-26	15:12	drwxr-x--x	
▼	usuario.app.aplicacaobancodedados		2017-10-26	15:21	drwxr-x--x	
>	cache		2017-10-26	15:15	drwxrwx--x	
▼	databases		2017-10-26	15:21	drwxrwx--x	
	banco_dados	20480	2017-10-26	15:21	-rw-rw----	
	banco_dados-journal	8720	2017-10-26	15:21	-rw-----	
	lib		2017-10-26	15:15	lrwxrwxrwx	-> /data/a...
>	dontpanic		2017-10-26	15:11	drwxr-x---	

SQLiteStudio

- O **SQLiteStudio** é uma ferramenta para a criar e manipular de bases de dados dos nossos APP's.

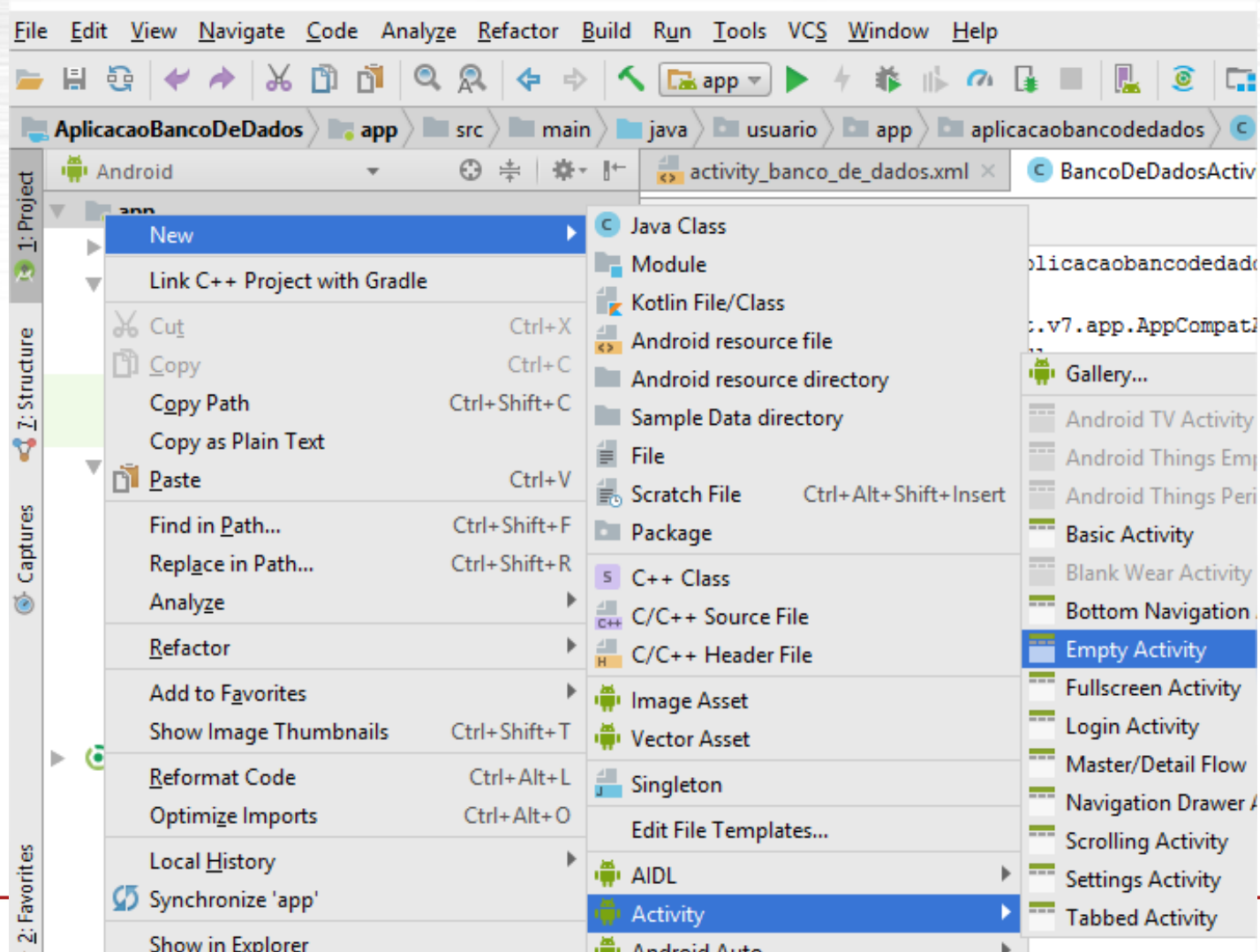


Registrando informações no Banco

- Vamos construir agora, dentro do mesmo projeto, uma Activity que irá registrar as informações dentro da nossa tabela dentro do Banco de Dados (a tabela “usuarios”). Essa será a primeira versão, onde iremos trabalhar com o método **execSQL** para a inserção das informações na tabela (através de comando SQL).
 - Vamos criar agora um nova Activity, seguindo os passos a seguir:
 - Clique com o botão direito do mouse sobre o diretório “app” do nosso projeto e selecione “New / Activity / Empty Activity”. Veja na figura seguinte :
-


Registrando informações no Banco

- Criando uma nova Activity...



Registrando informações no Banco

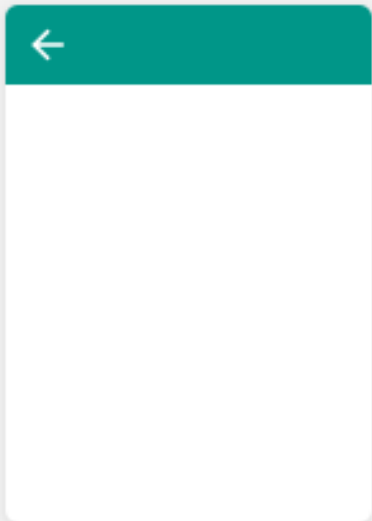
- Configurando a nova Activity...



Configure Activity

Android Studio

Creates a new empty activity



Activity Name:

☒ Generate Layout File

Layout Name:

☐ Launcher Activity

☐ Backwards Compatibility (AppCompat)

Package name: ▼

Source Language: ▼

Target Source Set: ▼

Registrando informações no Banco

- Arquivo “activity_grava_registros.xml”

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="#3ba0e2" >

        <ImageView
            android:id="@+id/imageView1"
            android:layout_width="72dp"
            android:layout_height="72dp"
            android:src="@drawable/icone_cadastro" />
```

Registrando informações no Banco

- Arquivo “activity_grava_registros.xml”

```
<TextView
    android:id="@+id/textView1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textAppearance="?android:attr/textAppearanceLarge"
/>
```

```
<LinearLayout
    android:id="@+id/layoutCorFundo"
    android:layout_width="fill_parent"
    android:layout_height="match_parent"
    android:background="#f4797d"
    android:gravity="center" >
```

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

Registrando informações no Banco

- Arquivo “activity_grava_registros.xml”

```
        android:text="Cadastro de Usuário"  
        android:textAppearance=  
            "?android:attr/textAppearanceLarge"  
        android:textColor="#ffffff" />
```

```
    </LinearLayout>
```

```
</LinearLayout>
```

```
<LinearLayout
```

```
    android:id="@+id/layoutImagemFundo"  
    android:layout_width="match_parent"  
    android:layout_height="fill_parent"
```

```
    android:orientation="vertical" >
```

Registrando informações no Banco

- Arquivo “activity_grava_registros.xml”

<TextView

```
    android:id="@+id/textView4"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Esta aplicação irá registrar as  
    informações do usuário no Banco de Dados."  
    android:textAppearance="?android:attr/textAppearanceLarge"  
    android:textSize="18sp" />
```

<TextView

```
    android:id="@+id/textView3"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:paddingTop="10dp"  
    android:text="Nome:"  
    android:textColor="#000000"  
    android:textSize="18sp" />
```

Registrando informações no Banco

- Arquivo “activity_grava_registros.xml”

```
<EditText
    android:id="@+id/ednome"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10" >

    <requestFocus />
</EditText>

<TextView
    android:id="@+id/TextView01"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="10dp"
    android:text="Telefone:"
    android:textColor="#000000"
    android:textSize="18sp" />
```

Registrando informações no Banco

- Arquivo “activity_grava_registros.xml”

```
<EditText
    android:id="@+id/edtelefone"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10" />

<TextView
    android:id="@+id/TextView02"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="10dp"
    android:text="E-Mail:"
    android:textColor="#000000"
    android:textSize="18sp" />

<EditText
    android:id="@+id/edemail"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ems="10" />
```

Registrando informações no Banco

- Arquivo “activity_grava_registros.xml”

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="fill_parent"
    android:gravity="center" >

    <Button
        android:id="@+id/btcadastrar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Cadastrar" />

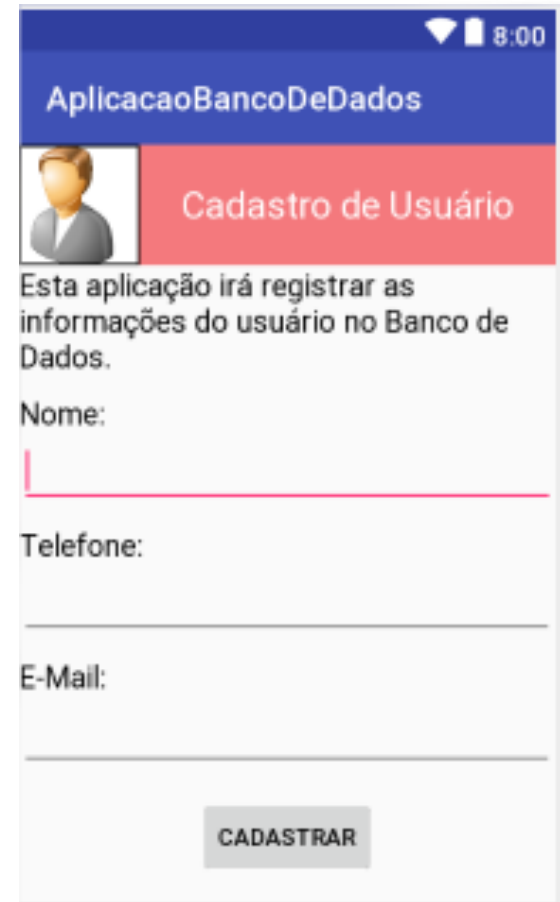
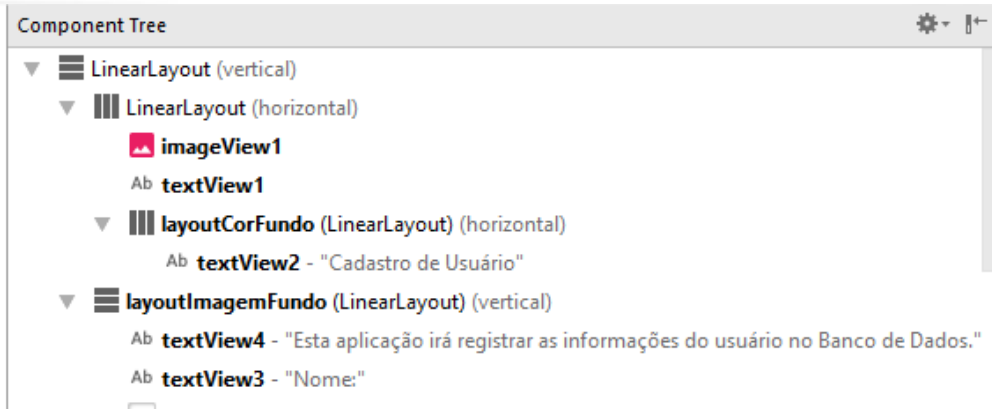
</LinearLayout>

</LinearLayout>

</LinearLayout>
```

Registrando informações no Banco

- Feito o que foi solicitado, salve o arquivo XML.
- Vejamos na figura seguinte o resultado:



Registrando informações no Banco

- Agora dentro do arquivo “GravaRegistrosActivity.java” vamos digitar o seguinte código abaixo:

```
1 package com.example.joaopaulo.aplicacaobancodedados;
2
3 import android.app.Activity;
4 import android.os.Bundle;
5 import android.app.AlertDialog;
6 import android.content.Context;
7 import android.database.sqlite.SQLiteDatabase;
8 import android.view.View;
9 import android.widget.*;
10
```

Registrando informações no Banco

- Agora dentro do arquivo “GravaRegistrosActivity.java” vamos digitar o seguinte código abaixo:

```
public class GravaRegistrosActivity extends Activity {

    Button btcadastrar;
    EditText ednome, edtelefone, edemail;
    SQLiteDatabase db;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_grava_registros);
        btcadastrar = (Button) findViewById(R.id.btcadastrar);
        ednome = (EditText) findViewById(R.id.ednome);
        edtelefone = (EditText) findViewById(R.id.edtelefone);
        edemail = (EditText) findViewById(R.id.edemail);
        try {
            db = openOrCreateDatabase("banco_dados",
                                    Context.MODE_PRIVATE, null);
        }
        catch(Exception e)
        {
            MostraMensagem("Erro : " + e.toString());
        }
    }
}
```


Registrando informações no Banco

- Agora dentro do arquivo “GravaRegistrosActivity.java” vamos digitar o seguinte código abaixo:

```
btcadastrar.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View arg0) {  
        String nome = ednome.getText().toString();  
        String telefone = edtelefone.getText().  
            toString();  
        String email = edemail.getText().toString();  
        try {  
            db.execSQL("insert into usuarios(nome," +  
                "telefone, email) values('" + nome + "','" +  
                + telefone + "','" + email + "')");  
            MostraMensagem("Dados cadastrados com sucesso");  
        }  
        catch(Exception e)  
        {  
            MostraMensagem("Erro : " + e.toString());  
        }  
    }  
});  
}
```

Registrando informações no Banco

- Agora dentro do arquivo “GravaRegistrosActivity.java” vamos digitar o seguinte código abaixo:

```
public void MostraMensagem(String str)
{
    AlertDialog.Builder dialogo = new
        AlertDialog.Builder(GravaRegistrosActivity.this);
    dialogo.setTitle("Aviso");
    dialogo.setMessage(str);
    dialogo.setNeutralButton("OK", null);
    dialogo.show();
}
```

Registrando informações no Banco

- Vamos a explicação de algumas linhas de código. A instrução:

```
String nome = ednome.getText().toString();  
String telefone = edtelefone.getText().toString();  
String email = edemail.getText().toString();
```

- Obtém as informações digitadas nos campos “nome”, “telefone” e “email” (representado respectivamente pelos componentes/objetos *ednome*, *edtelefone* e *edmail*) e atribui os seus valores para as variáveis de mesmo nome : *nome*, *telefone* e *email*.
-

Registrando informações no Banco

- Na instrução seguinte :

```
db.execSQL("insert into usuarios(nome,telefone, email) values('" +  
nome + "', '" + telefone + "', '" + email + "')");
```

- Realiza a inserção dos dados na tabela “usuários” por meio da instrução “insert” da linguagem SQL, através do método **execSQL**.
 - Vamos abrir agora o arquivo “**activity_banco_de_dados.xml**” (no “modo Design”) para inserirmos o seguinte componente abaixo, abaixo do botão rotulado “**Criar Banco de Dados**”):
-

No activity_banco_de_dados.xml

- Button → Acrescentar o Botão de **Cadastrar Dados**

Button

Propriedade	Valor
id	btcadastrardados
text	Cadastrar Dados
layout:width	200dp

```
<Button  
    android:id="@+id/btcadastrardados"  
    android:layout_width="200dp"  
    android:layout_height="wrap_content"  
    android:text="@string/cadastrar_dados" />
```

- E no strings.xml :

```
<string name="cadastrar_dados">Cadastrar Dados</string>
```

Registrando informações no Banco

- Vejam o resultado:



Registrando informações no Banco

- Agora vamos abrir novamente o arquivo “MainActivity.java” para acrescentar as seguintes instruções do botão **btcadastrardados**:

```
1      package com.example.joaopaulo.aplicacaobancodedados;
2
3      import android.app.Activity;
4      import android.os.Bundle;
5      import android.app.AlertDialog;
6      import android.content.Context;
7      import android.database.sqlite.SQLiteDatabase;
8      import android.view.View;
9      import android.widget.*;
10     import android.content.Intent;
11
```

Registrando informações no Banco

- Agora vamos abrir novamente o arquivo “MainActivity.java” para acrescentar as seguintes instruções do botão **btcadastrardados**:

```
public class MainActivity extends Activity {  
  
    Button btcriabanco;  
    Button btcadastrardados;  
    SQLiteDatabase db;
```

Registrando informações no Banco

- Arquivo “MainActivity.java” //continuação

```
18
19     @Override
20     protected void onCreate(Bundle savedInstanceState) {
21         super.onCreate(savedInstanceState);
22         setContentView(R.layout.activity_main);
23
24         btcriarbanco = findViewById(R.id.btcriarbanco);
25         btcadastrardados = findViewById(R.id.btcadastrardados);
26
27         btcadastrardados.setOnClickListener(new View.OnClickListener() {
28             @Override
29             public void onClick(View args0) {
30                 Intent gravaRegistroActivity = new Intent (MainActivity.this,
31                     GravaRegistrosActivity.class);
32                 MainActivity.this.startActivity(gravaRegistroActivity);
33             }
34         });
```

Registrando informações no Banco

- Arquivo "MainActivity.java" //continuação

```
35
36      btcriarbanco.setOnClickListener(new View.OnClickListener() {
37          @Override
38          public void onClick(View view) {
39              try {
40                  db = openOrCreateDatabase("banco_dados",
41                      Context.MODE_PRIVATE, null);
42                  db.execSQL("create table if not exists " +
43                      " usuarios(numreg integer primary key " +
44                      " autoincrement, nome text not null, telefone text " +
45                      " not null, " + " email text not null) ");
46                  AlertDialog.Builder dialogo = new
47                      AlertDialog.Builder(MainActivity.this);
48                  dialogo.setTitle("Aviso")
49                      .setMessage("Banco de dados criado com sucesso!")
50                      .setNeutralButton("OK", null)
51                      .show();
52              } catch (Exception e) {
53              }
54          }
55      });
56  }
57  }
58
```

Registrando informações no Banco

- Execute o app e faça um cadastro...




Trabalhando com Banco de Dados no Android

Escolha uma das opções abaixo:

Criar Banco de Dados

Cadastrar Dados



Cadastro de Usuário

Esta aplicação irá registrar as informações do usuário no Banco de Dados.

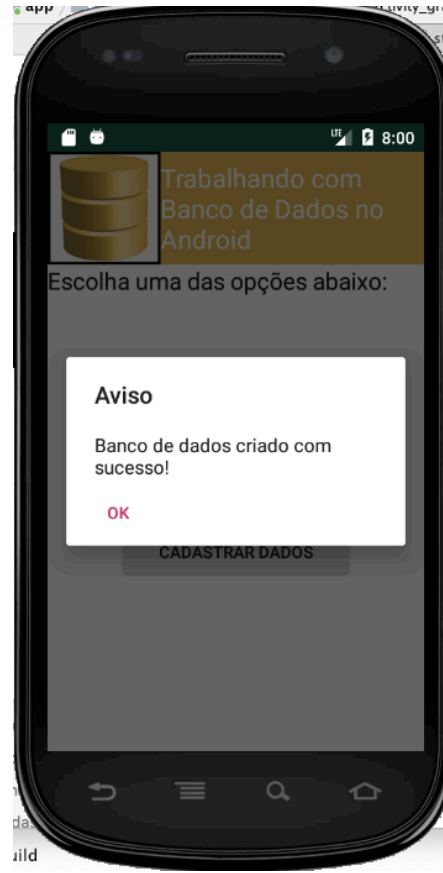
Nome:

Telefone:

E-Mail:

Cadastrar

APP desenvolvido até aqui





Por hoje é só !!!

Até a próxima aula...
