

AULA 07

IMD0509 - DESENVOLVIMENTO PARA **DISPOSITIVOS MÓVEIS**

Prof. Emerson Alencar

emerson@imd.ufrn.br



VIEWS - PROGRaMAÇÃO

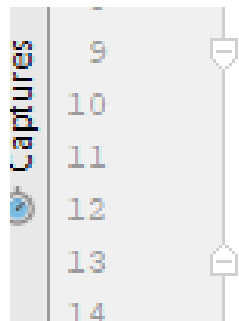
_ X

```
7  <> public class MainActivity extends AppCompatActivity {  
8  
9      @Override  
10     protected void onCreate(Bundle savedInstanceState) {  
11         super.onCreate(savedInstanceState);  
12  
13         TextView textView = new TextView(context: this);  
14         textView.setText("Aula Mobile");  
15         setContentView(textView);  
16  
17     }  
18 }
```

View criada no código

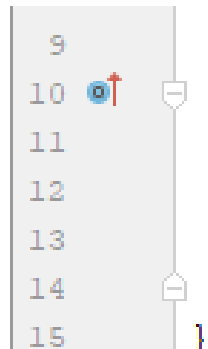
VIEWS - DECLARAÇÃO

_ X



```
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Aula Mobile"
/>
```

Declaração no XML



```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);
}
```

Referenciando no
código

ACESSANDO VIEWS DO XML

_ X

```
8  
9  
10  
11  
12  
13  
14
```

```
<TextView  
    android:id="@+id/texto"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
/>
```

Declaração no XML

```
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19
```

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    setContentView(R.layout.activity_main);  
  
    TextView textView = findViewById(R.id.texto);  
    textView.setText("Aula Mobile!");  
}
```

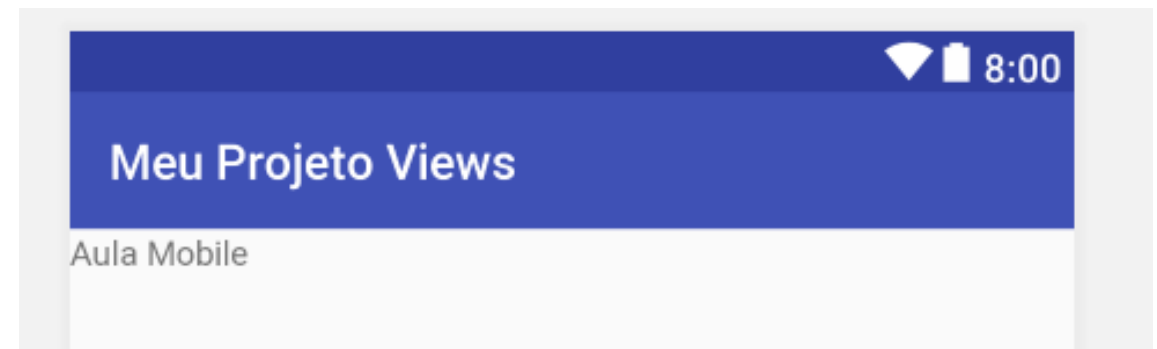
Lendo o XML

- O Android possui diversas views importantes utilizadas para compor interfaces gráficas
- Localizadas no pacote **android.widget**
- Algumas exemplos de tipos de views
 - Text
 - Button
 - Text field
 - Checkbox
 - Radio button
 - Toggle button
 - Progress bar

- View: TextView
- Exibição de Textos na Tela



```
<TextView
    android:id="@+id/texto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Aula Mobile"
/>
```



BUTTON



- View: Button
- Botão composto por texto e/ou ícone

8

9

10

11

12

13

14

15

16

17

18

19

20

<Button

android:text="Enviar"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

/>

<Button

android:text="Enviar"

android:layout_width="wrap_content"


android:layout_height="wrap_content"

android:drawableLeft="@drawable/ic_email"

/>

Meu Projeto Views

ENVIAR

 ENVIAR

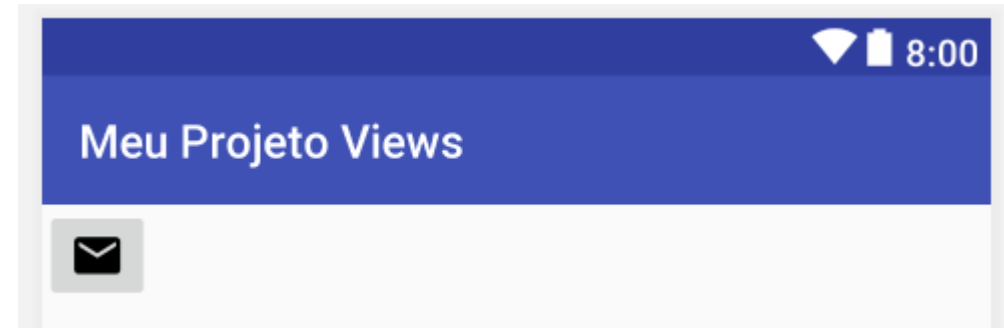
BUTTON



- View: ImageButton
- Botão composto ícone



```
<ImageButton  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:src="@drawable/ic_email"  
/>
```



BUTTON: EVENTO DE CLIQUE

_ X

- Quando um botão é clicado, ele dispara um evento
- Este evento é normalmente tratada pela activity à qual o botão pertence

```
9      <Button
10         android:text="Enviar"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:onClick="enviar"
14     />
```

Define o método que é chamado no clique

```
19
20     public void enviar(View view) {
21         //Ação do botão aqui!
22     }
```

Método definido na Activity

BUTTON: EVENTO DE CLIQUE

_ X

- É possível também utilizar um listener

```
9 <Button
10     android:id="@+id/button"
11     android:text="Enviar"
12     android:layout_width="wrap_content"
13     android:layout_height="wrap_content"
14 />
```

```
8
9 public class MainActivity extends AppCompatActivity implements View.OnClickListener {
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14
15         setContentView(R.layout.activity_main);
16
17         Button button = findViewById(R.id.button);
18         button.setOnClickListener(this);
19     }
20
21     @Override
22     public void onClick(View v) {
23         //Ação do botão
24     }
25 }
```

Chamado quando
houver o clique

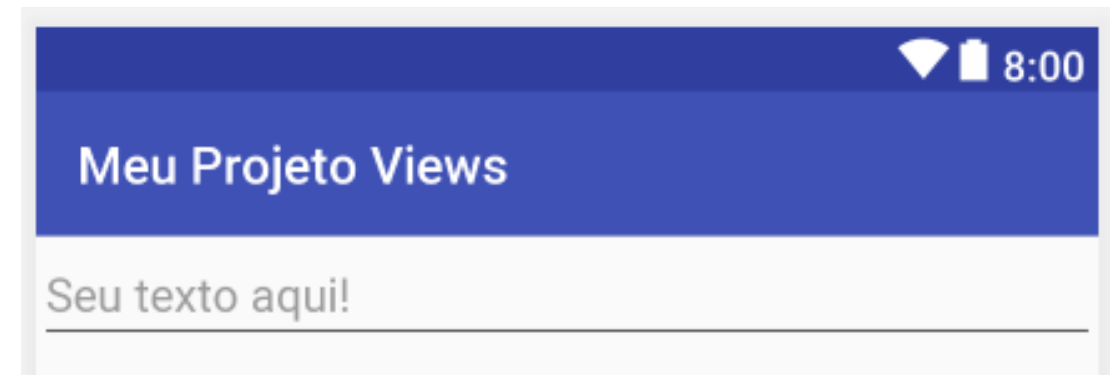
TEXT FIELD

_ X

- View: EditText
- Permite a digitação de textos

```
8
9
10
11
12
13
14
15
```

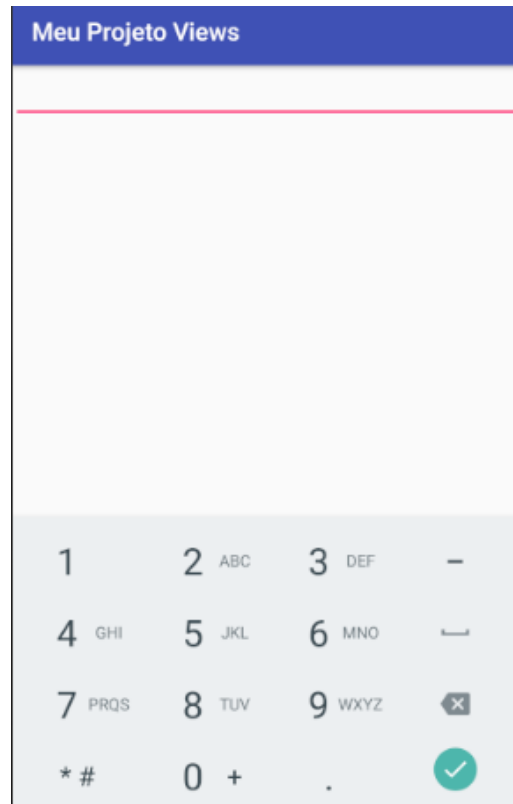
```
<EditText
    android:id="@+id/text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Seu texto aqui!"
/>
```



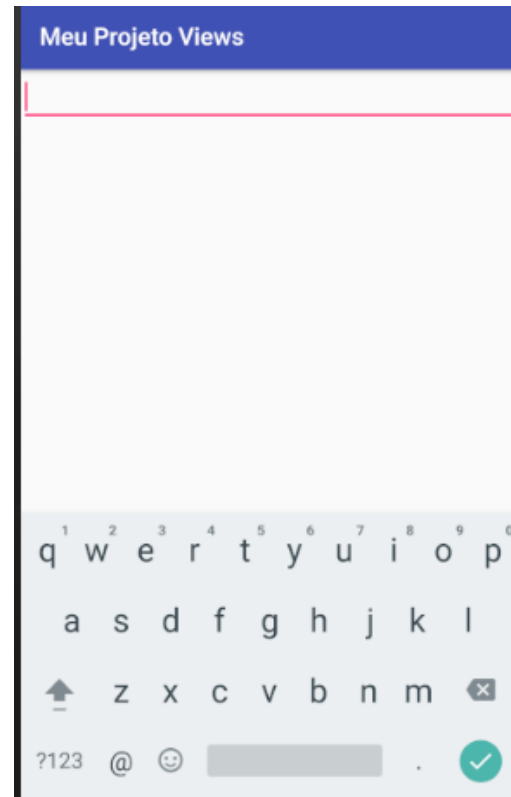
TEXT FIELD: INPUT TYPE

_ X

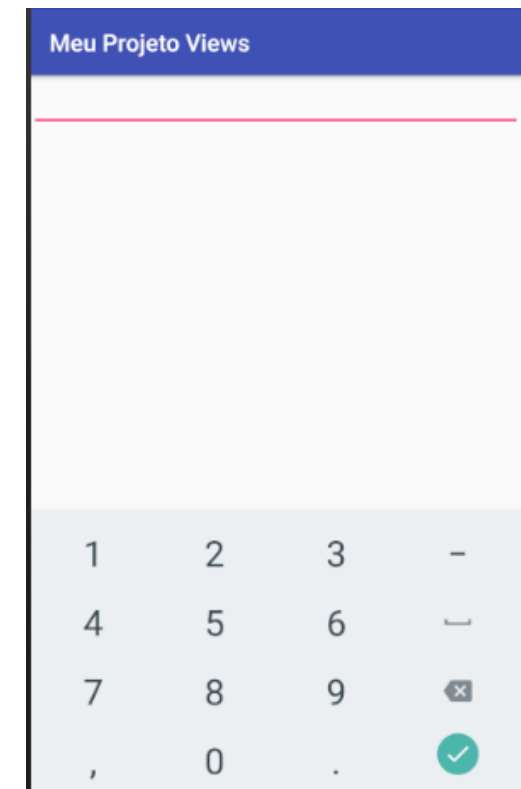
- O atributo `inputType` pode ser usado para identificar o tipo de dado que está sendo digitado
 - Texto qualquer, e-mail, número telefônico, etc



`android:inputType="phone"`



`android:inputType="textEmailAddress"`



`android:inputType="number"`

TEXT FIELD: INPUT TYPE

_ X

- O atributo `inputType` também pode ser utilizado para definir mais características

inputType	Significado
<code>textCapCharacters</code>	Todos os caracteres em maiúsculo
<code>textCapSentences</code>	Toda frase começa com caractere maiúsculo
<code>textMultiLine</code>	Texto com múltiplas linhas
<code>textPassword</code>	Não exibe o caractere real

```
<EditText
    android:id="@+id/text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:inputType="textCapCharacters|textMultiLine"
/>
```

CHECKBOX

_ X

- View: CheckBox
- Permite a seleção de um item ou de itens em um conjunto

```
9  <CheckBox
10      android:layout_width="wrap_content"
11      android:layout_height="wrap_content"
12      android:text="Futebol"
13  />
14
15  <CheckBox
16      android:layout_width="wrap_content"
17      android:layout_height="wrap_content"
18      android:text="Golfe"
19  />
20
21  <CheckBox
22      android:layout_width="wrap_content"
23      android:layout_height="wrap_content"
24      android:text="Tênis"
25  />
26
```

Meu Projeto Views

- ☐ Futebol
- ☐ Golfe
- ☐ Tênis

CHECKBOX: EVENTO CLIQUE

_ X

- A activity pode ser notificada quando um checkbox é clicado

```
9 <CheckBox
10     android:id="@+id/checkbox_futebol"
11     android:layout_width="wrap_content"
12     android:layout_height="wrap_content"
13     android:text="Futebol"
14     android:onClick="onClickCheckBox"
15 />
```

```
17 <CheckBox
18     android:id="@+id/checkbox_golfe"
19     android:layout_width="wrap_content"
20     android:layout_height="wrap_content"
21     android:text="Golfe"
22     android:onClick="onClickCheckBox"
23 />
```

```
21 public void onClickCheckBox(View view) {
22
23     CheckBox checkbox = (CheckBox) view;
24     boolean checked = checkbox.isChecked();
25
26     if (view.getId() == R.id.checkbox_futebol) {
27         // Processa o clique
28     } else if (view.getId() == R.id.checkbox_golfe) {
29         // Processa o clique
30     }
31 }
```

Na Activity

CHECKBOX: EVENTO CLIQUE



- Um listener também pode ser usado

```
10  public class MainActivity extends AppCompatActivity implements View.OnClickListener {  
11  
12      @Override  
13      protected void onCreate(Bundle savedInstanceState) {  
14          super.onCreate(savedInstanceState);  
15  
16          setContentView(R.layout.activity_main);  
17  
18          CheckBox checkbox = findViewById(R.id.checkbox_futebol);  
19          checkbox.setOnClickListener(this);  
20      }  
21  
22  
23      public void onClick(View view) {  
24          // Evento de clique  
25      }  
26  }
```


CHECKBOX: MUDANÇA DE ESTADO _ X

- Além de ler o estado checkbox, é possível alterá-lo também via programação
 - **setChecked(boolean)**
 - Marca ou desmarca o checkbox
 - **toggle()**
 - Troca o estado

RADIO BUTTON



- Views: RadioGroup e RadioButton
- Permite a escolha de um item dentro de um conjunto de itens

```
9      <RadioGroup
10          android:layout_width="match_parent"
11          android:layout_height="wrap_content"
12          android:orientation="vertical">
13          <RadioButton
14              android:layout_width="wrap_content"
15              android:layout_height="wrap_content"
16              android:text="Futebol" />
17          <RadioButton
18              android:layout_width="wrap_content"
19              android:layout_height="wrap_content"
20              android:text="Golfe" />
21          <RadioButton
22              android:layout_width="wrap_content"
23              android:layout_height="wrap_content"
24              android:text="Tênis" />
25      </RadioGroup>
26
```

Meu Projeto Views

- ☐ Futebol
- ☐ Golfe
- ☐ Tênis

RADIO BUTTON: EVENTO CLIQUE

_ X

- A activity pode ser notificada quando um radio button é clicado

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <RadioButton
        android:id="@+id/radiobutton_futebol"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Futebol"
        android:onClick="onClickRadioButton"/>
    <RadioButton
        android:id="@+id/radiobutton_golfe"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Golfe"
        android:onClick="onClickRadioButton"
    />
```

```
public void onClickRadioButton(View view) {
    RadioButton radioButton = (RadioButton) view;
    boolean checked = radioButton.isChecked();

    if (view.getId() == R.id.radiobutton_futebol) {
        // Processa o clique
    } else if (view.getId() == R.id.radiobutton_golfe) {
        // Processa o clique
    }
}
```

RADIO BUTTON: EVENTO CLIQUE



- Um listener também pode ser usado

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);

        RadioButton rb = findViewById(R.id.radiobutton_futebol);
        rb.setOnClickListener(this);
    }

    public void onClick(View view) {
        // Processa o evento de clique
    }
}
```

RADIO BUTTON: ESTADO

_ X

- Além de ler o estado radio button, é possível alterá-lo também via programação
 - **setChecked(boolean)**
 - Marca ou desmarca o radio button
 - **toggle()**
 - Troca o estado

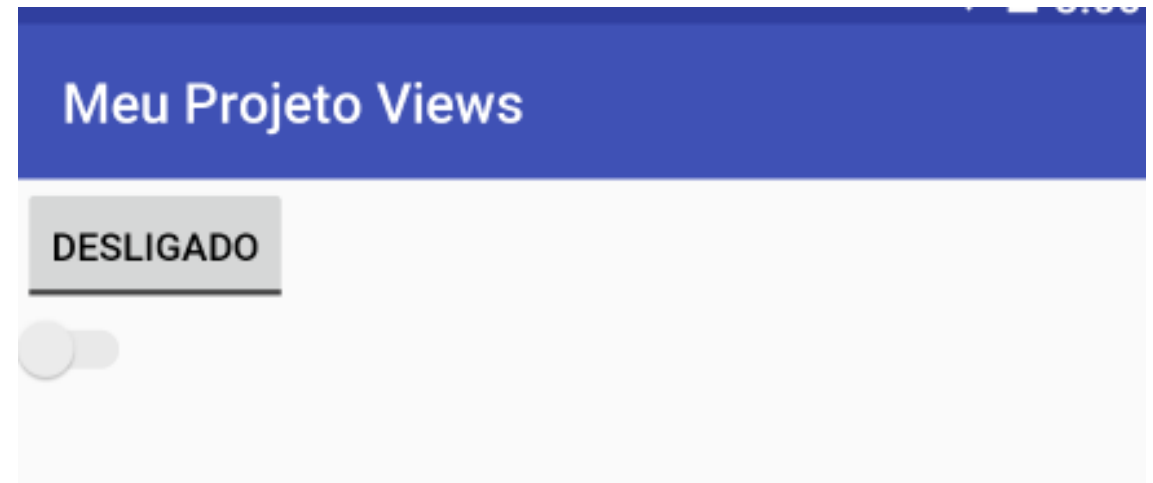
TOGGLEBUTTON E SWITCH



- Views: ToggleButton e Switch
- Permite alternar entre dois estados (ligado ou desligado)

```
<ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOff="Desligado"
    android:textOn="Ligado" />

<Switch
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOff="Desligado"
    android:textOn="Ligado" />
```



TOGGLEBUTTON: EVENTOS

_ X

- A activity pode ser notificada quando um toggle button é clicado

```
<ToggleButton
    android:id="@+id/toggle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOff="Desligado"
    android:textOn="Ligado"
    android:onClick="onClickToggle"
/>
```

```
public void onClickToggle(View view) {

    ToggleButton button = (ToggleButton) view;
    boolean checked = button.isChecked();

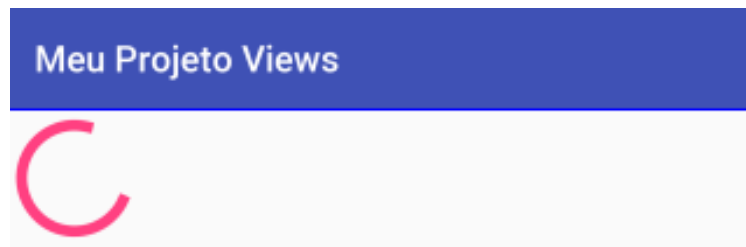
    if (view.getId() == R.id.toggle) {
        // Processa o clique
    }
}
```

PROGRESS BAR

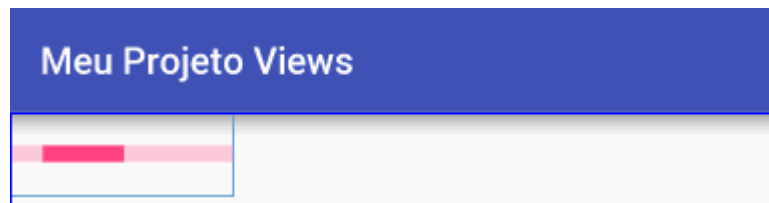


- View: ProgressBar

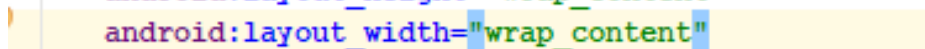
```
<ProgressBar
    android:id="@+id/progress"
    style="?android:attr/progressBarStyleLarge"
    android:indeterminate="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
```



```
<ProgressBar
    android:id="@+id/progress"
    style="?android:attr/progressBarStyleHorizontal"
    android:indeterminate="true"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
/>
```



```
<ProgressBar
    android:id="@+id/progress"
    style="?android:attr/progressBarStyleHorizontal"
    android:indeterminate="false"
    android:progress="30"
    android:max="100"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
/>
```



PROGRESS BAR



- O indicador de progresso pode ser configurado também via programação

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    ProgressBar progress = findViewById(R.id.progress);
    progress.setProgress(30);
}
```

TAMANHO DAS VIEWS



- Toda view tem uma largura e uma altura
 - **layout_width:** define a largura
 - **layout_height:** define a altura
- Estes atributos podem ter os seguintes valores
 - **match_parent:** o tamanho é expandido até ficar igual ao tamanho do layout pai
 - **wrap_content:** o tamanho é o mínimo necessário para comportar o componente
 - **número:** especifica o tamanho em termos numéricos

TAMANHO DAS VIEWS

_ X

- Quando o tamanho é um número, é possível usar as seguintes unidades

Tipo	Abr.	Descrição
Pixels	<i>px</i>	Pixels físicos na tela
Points	<i>pt</i>	Um ponto é 1/72 polegadas
Millimeters	<i>mm</i>	Milímetros
Inches	<i>in</i>	Polegadas
Density-Independent-Pixels	<i>dip</i> ou <i>dp</i>	Usa como base um espaço de 160 pixels e faz o mapeamento
Scale-Independent-Pixels	<i>sp</i>	Usado para definir tamanho de fontes

LINEAR LAYOUT



- Organiza os componentes na horizontal ou na vertical

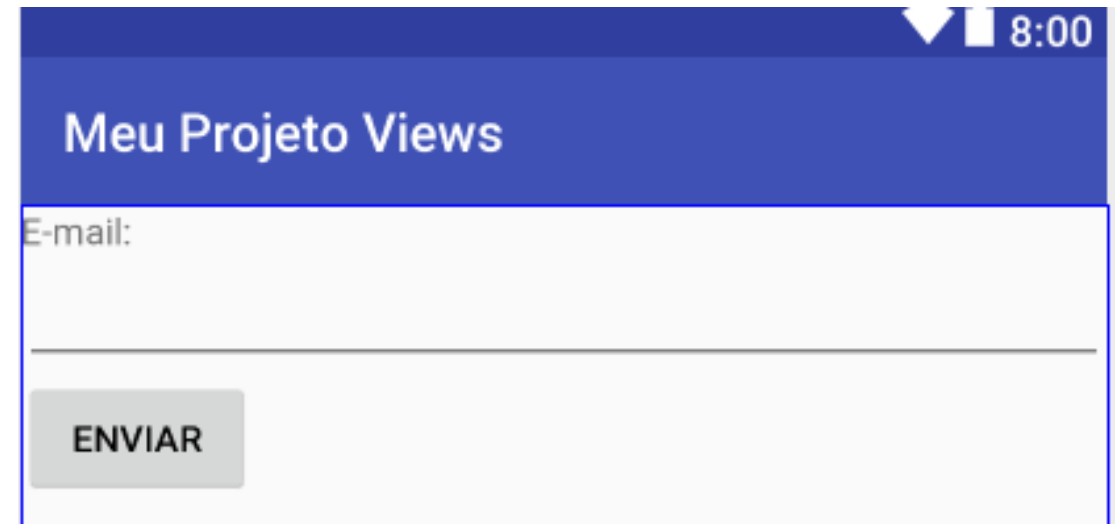
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="E-mail:" />

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="textEmailAddress" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Enviar" />

</LinearLayout>
```

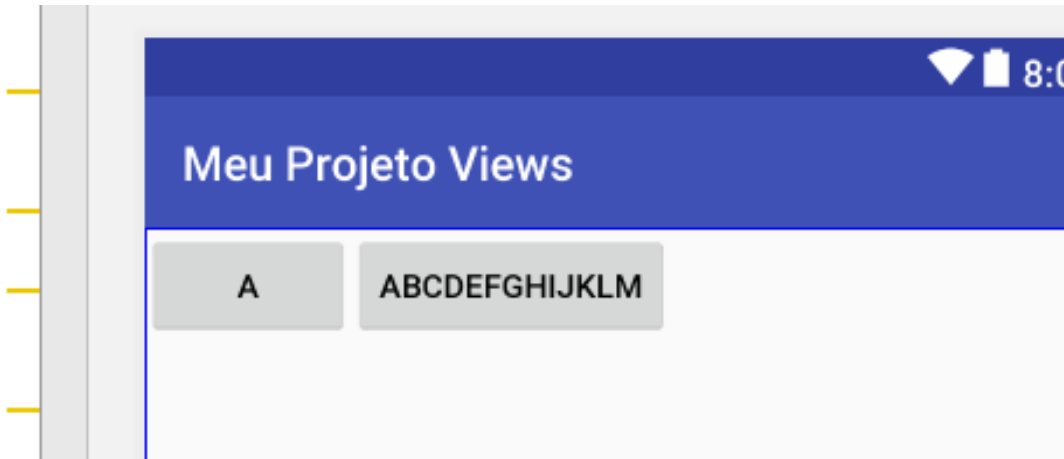


LINEARLAYOUT: WEIGHT

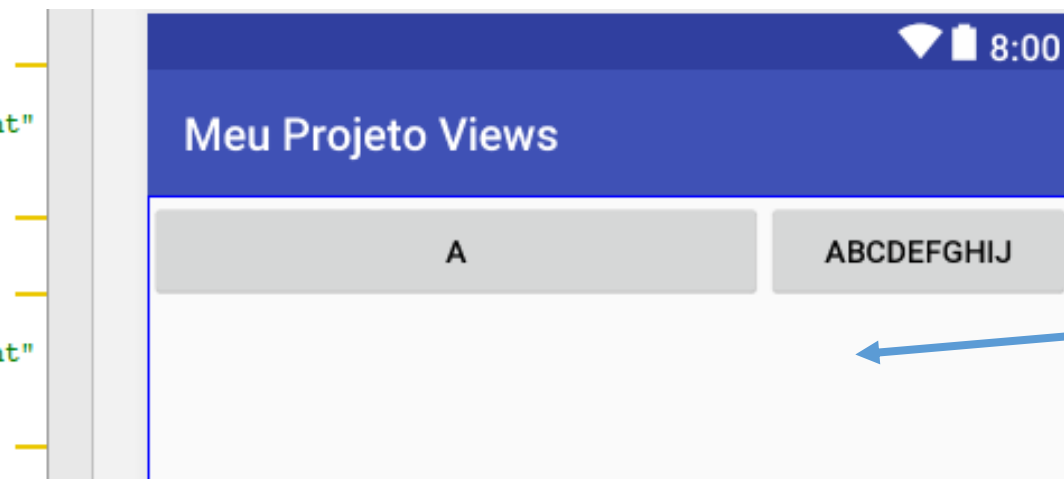
_ X

- Define um peso para a view: Informação usada para definir o tamanho da view com relação a outras views

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="A"
/>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="ABCDEFGHJKLM"
/>
```



```
7  <Button
8      android:layout_width="0dp"
9      android:layout_height="wrap_content"
10     android:layout_weight="2"
11     android:text="A"
12     />
13 <Button
14     android:layout_width="0dp"
15     android:layout_height="wrap_content"
16     android:layout_weight="1"
17     android:text="ABCDEFGHIJ"
18     />
```



Peso

RELATIVE LAYOUT

_ X

- Permite posicionar as views relativamente a outras views
- É um layout bastante poderoso, pois permite criar layouts complexos

RELATIVE LAYOUT



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

    <TextView
        android:id="@+id/txt_nome"
        android:text="Nome: "
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/edt_nome" />

    <EditText
        android:id="@+id/edt_nome"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_toRightOf="@+id/txt_nome" />

    <Button
        android:text="OK"
        android:id="@+id/btn_continuar"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@+id/edt_nome" />

    <Button
        android:text="Cancelar"
        android:id="@+id/btn_cancelar"
        android:layout_width="100dp"
        android:layout_height="wrap_content"
        android:layout_below="@+id/edt_nome"
        android:layout_toLeftOf="@+id/btn_continuar" />

</RelativeLayout>
```

Meu Projeto Views

Nome:

CANCELAR OK

CONSTRAINTLAYOUT



- Similar ao RelativeLayout, só que mais flexível
- Criação de layouts complexos sem aninhar views
- Totalmente integrado com o editor gráfico do Android Studio
- Não é nativo do Android
 - Disponível através da API de compatibilidade

MARGIN



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/a
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginLeft="40dp"
    android:layout_marginRight="40dp"
    android:layout_marginTop="20dp"
>
```

Meu Projeto Views

A

ABCDEFGHIJ

Meu Projeto Views

A

ABCDEFGHIJ

Aplicando Margin ao layout

PADDING



- O padding é um espaço sem uso ao redor da parte interna de uma view
 - Esquerda, direita, acima e abaixo

```
<Button
    android:text="Botão 1"
    android:layout_width="120dp"
    android:layout_height="wrap_content" />
<Button
    android:text="Botão 2"
    android:layout_width="120dp"
    android:layout_height="wrap_content"
    android:paddingTop="20dp"
    android:paddingBottom="20dp"
/>
```

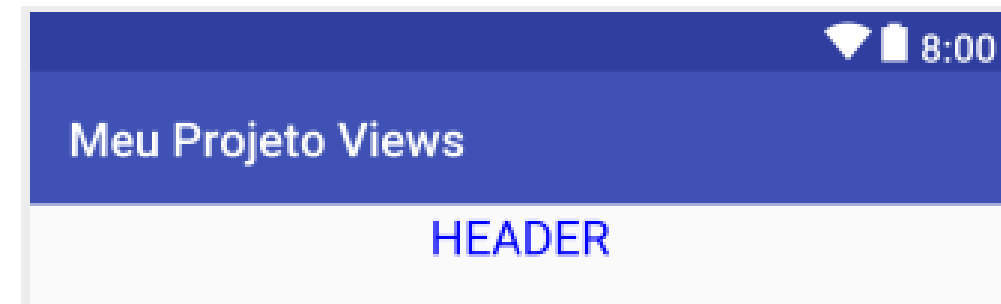
INCLUSÃO DE LAYOUT

_ X

- Um arquivo de layout pode incluir outro arquivo de layout
 - Utilizado quando determinado layout deve ser utilizado em várias telas

header.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="20sp"
        android:gravity="center_horizontal"
        android:text="HEADER"
        android:textSize="20sp"
        android:textColor="#0000FF" />
</LinearLayout>
```



INCLUSÃO DE LAYOUT

_ X

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

    <include layout="@layout/header" android:id="@+id/lay_header"/>

    <Button
        android:text="Botão 1"
        android:layout_width="120dp"
        android:layout_height="wrap_content" />

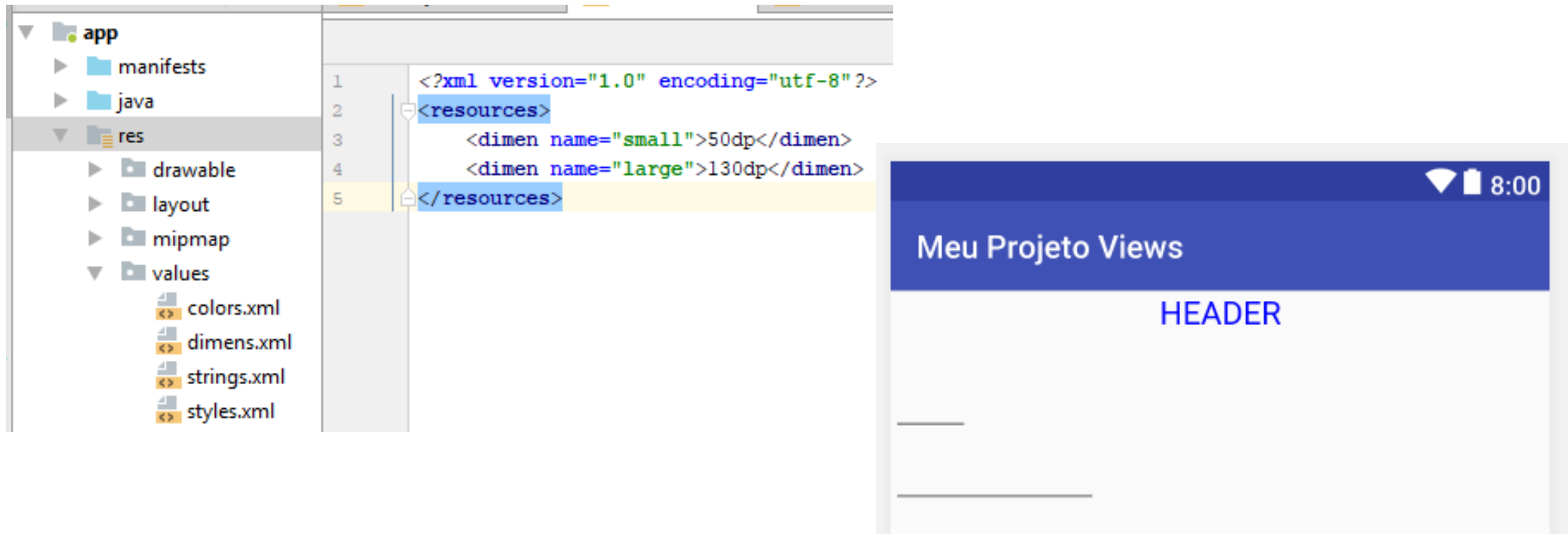
    <Button
        android:text="Botão 2"
        android:layout_width="120dp"
        android:layout_height="wrap_content"
        android:paddingTop="20dp"
        android:paddingBottom="20dp"
    />
</LinearLayout>
```



USANDO RESOURCE *dimen*



- Dimensões podem ser especificadas como resources para serem reaproveitadas



DEFINIR LAYOUT - ORIENTAÇÃO



- Um dispositivo Android pode ter dois tipos de orientação
 - – Retrato (portrait)
 - – Paisagem (landscape)
- O Android faz a adequação do layout dependendo da orientação
- É possível também definir layouts diferentes de acordo com a orientação

DEFINIR LAYOUT - ORIENTAÇÃO

_ X

- Basta definir o arquivo de layout nas pastas de resources correspondentes



FORÇANDO ORIENTAÇÃO

_ X

- Por padrão, uma activity pode funcionar em ambas as orientações
- É possível forçar uma orientação

– Via AndroidManifest.xml

```
<activity android:screenOrientation="portrait">
```

```
<activity android:screenOrientation="landscape">
```

– Via programação

```
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
```

```
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
```

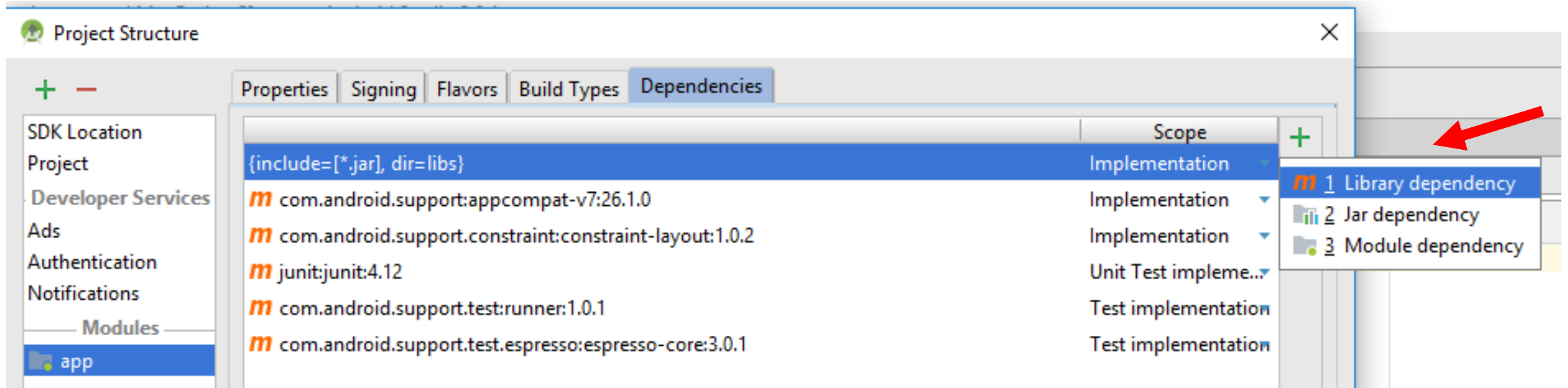

- Verificar se o repositório onde o Android Studio baixa a API de compatibilidade está disponível:
 - Tools -> Android -> SDK Manager
 - Aba: SDK Tools - > Support Repository
- Grade

```
20
21 dependencies {
22     implementation fileTree(dir: 'libs', include: ['*.jar'])
23     implementation 'com.android.support:appcompat-v7:26.1.0'
24     implementation 'com.android.support.constraint:constraint-layout:1.0.2'
25     testImplementation 'junit:junit:4.12'
26     androidTestImplementation 'com.android.support.test:runner:1.0.1'
27     androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'
28 }
29
```



INSERINDO API DE COMPATIBILIDADE _ X

- File -> Project Structure -> App -> Dependencies



EXEMPLO - COMPATIBILIDADE

_ X

```
2
3 import android.support.v7.app.ActionBar;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13
14        ActionBar bar = getSupportActionBar();
15
16    }
17 }
18
```

RUNTIME PERMISSIONS



- Permissão em tempo de execução
- Exemplo: Câmera

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.emerson.meuprojeto2">

    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-feature android:name="android.hardware.camera2"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
```

RUNTIME PERMISSIONS

_ X

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    if (checkSelfPermission(Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {  
        requestPermissions(new String[] {Manifest.permission.CAMERA}, requestCode: 5);  
    }  
}
```

RUNTIME PERMISSIONS

_ X

```
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {  
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);  
  
    if(requestCode == 5){  
        if(permissions.length > 0 && permissions[0].equals(Manifest.permission.CAMERA) &&  
            grantResults[0] == PackageManager.PERMISSION_GRANTED){  
  
            CameraManager cameraManager = (CameraManager) getSystemService(CAMERA_SERVICE);  
            try {  
                String[] ids = cameraManager.getCameraIdList();  
                cameraManager.openCamera(ids[0], new CameraDevice.StateCallback() {  
                    @Override  
                    public void onOpened(@NonNull CameraDevice camera) {  
                        Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
                        startActivity(intent);  
                        Toast.makeText(context, MainActivity.this, text: "Camera Aberta", Toast.LENGTH_LONG).show()  
  
                        camera.close();  
                    }  
  
                    @Override  
                    public void onDisconnected(@NonNull CameraDevice camera) { camera.close(); }  
                }  
            }  
        }  
    }  
}
```

Acessando o Android Studio



The background is a solid blue color. It is decorated with a complex, abstract geometric pattern. This pattern consists of numerous white dots of varying sizes, some of which are connected by thin white lines to form a network of triangles and other polygons. Scattered throughout the design are several yellow elements: small circles, a diamond shape, and plus signs. The overall aesthetic is modern and mathematical.

Perguntas?