



# Views

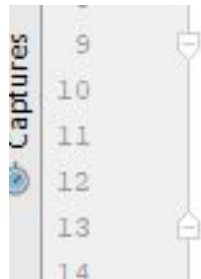
Professor Emerson Alencar  
[emerson@imd.ufrn.br](mailto:emerson@imd.ufrn.br)

# VIEWS - PROGRAMAÇÃO

```
7 class MainActivity : AppCompatActivity() {  
8  
9     override fun onCreate(savedInstanceState: Bundle?) {  
10         super.onCreate(savedInstanceState)  
11  
12  
13         val textView = TextView(context: this)  
14         textView.text = "Aula Mobile"  
15  
16         setContentView(textView)  
17     }  
18 }  
19 }
```

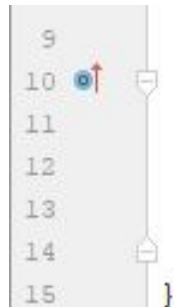
View criada no código

# VIEWS - DECLARAÇÃO



```
<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 VIEW DO XML

```
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

```
7 class MainActivity : AppCompatActivity() {  
8  
9     override fun onCreate(savedInstanceState: Bundle?) {  
10         super.onCreate(savedInstanceState)  
11  
12         setContentView(R.layout.activity_main)  
13  
14         val textView = findViewById<TextView>(R.id.texto)  
15  
16         textView.text = "Aula de Mobile!"  
17  
18     }  
19 }  
20
```

Lendo o XML

# VIEWS DO ANDROID

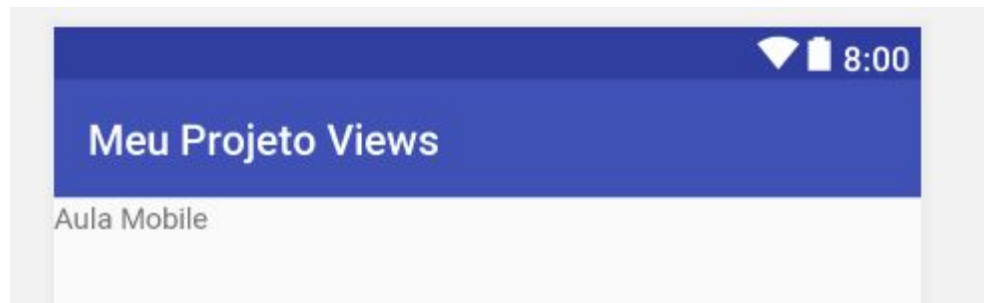
- ▶ O Android possui diversas views importantes utilizadas para compor interfaces gráficas
- ▶ Localizadas no pacote **android.widget**
- ▶ Algumas exemplos de tipos de views
  - Text
  - Button
  - Text field
  - Checkbox
  - Radio button
  - Toggle button
  - Progress bar

# TEXT

- ▷ 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

The image shows an Android Studio interface. On the left, a vertical toolbar contains icons for a hierarchy view, a search icon, and an email icon. The main area displays XML code for two buttons. The first button (lines 9-13) has the text "Enviar". The second button (lines 15-19) also has the text "Enviar" and includes an email icon on the left. On the right, a preview of the app's UI is shown. It features a blue header bar with the text "Meu Projeto Views", a status bar at the top with a Wi-Fi icon, a battery icon, and the time "8:00". Below the header, there are two buttons: a plain grey button with the text "ENVIAR" and a grey button with an email icon and the text "ENVIAR". Blue arrows point from the XML code to the corresponding buttons in the preview.

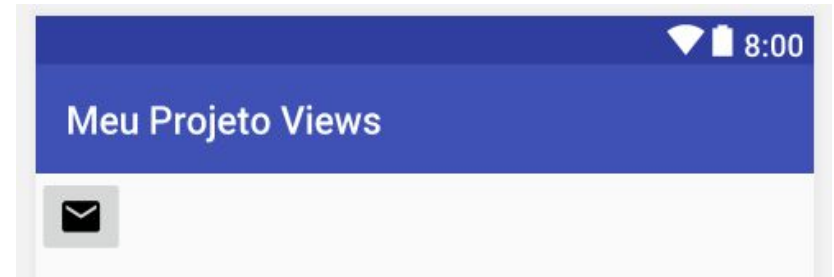
```
8  
9  
10 <Button  
11     android:text="Enviar"  
12     android:layout_width="wrap_content"  
13     android:layout_height="wrap_content"  
14 />  
15  
16 <Button  
17     android:text="Enviar"  
18     android:layout_width="wrap_content"  
19     android:layout_height="wrap_content"  
20     android:drawableLeft="@drawable/ic_email"  
    />
```

# 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

- ▶ 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

```
36
37
38  fun enviar(v: View){
39      // Códigos aqui
40  }
```

Método definido  
na Activity

# BUTTON: EVENTO DE CLIQUE

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

```
<Button
    android:id="@+id/button"
    android:text="Enviar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
```

- ▷ É possível também utilizar um listener

```
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
```

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)

        button.setOnClickListener { it: View!
            Toast.makeText(context: this, text: "Botão clicado!", Toast.LENGTH_LONG).show()
        }
    }
}
```

Chamado  
quando houver  
o clique

# BUTTON: EVENTO DE CLIQUE

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

```
<Button
    android:id="@+id/button"
    android:text="Enviar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
/>
```

- ▶ É possível também utilizar um listener

```
11 class MainActivity : AppCompatActivity(), OnClickListener {
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        button.setOnClickListener(this)
    }

    override fun onClick(v: View?) {
        if(v?.id == R.id.button){
            Toast.makeText(context: this, text: "Botão Clicado!", Toast.LENGTH_LONG).show()
        }
    }
}
```

Chamado  
quando houver  
o clique

# EDITTEXT

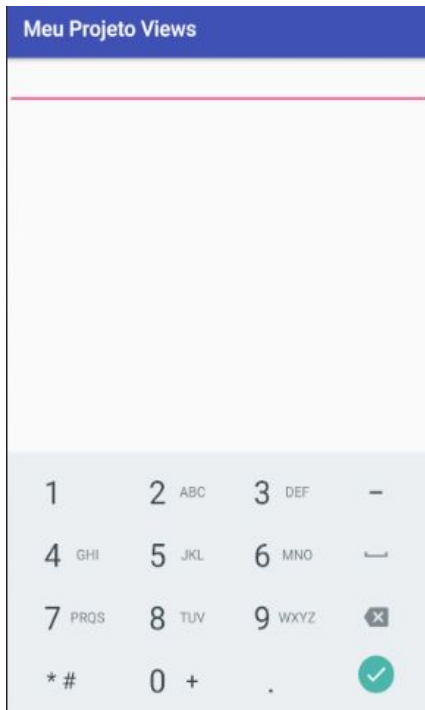
- ▶ View: EditText
- ▶ Permite a digitação de textos

```
8
9
10
11
12
13
14
<EditText
    android:id="@+id/text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="Seu texto aqui!"
/>
```

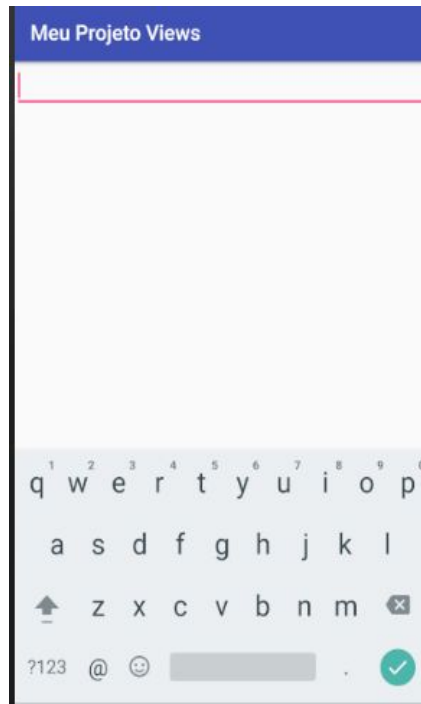


# EDITTEXT: INPUT TYPE

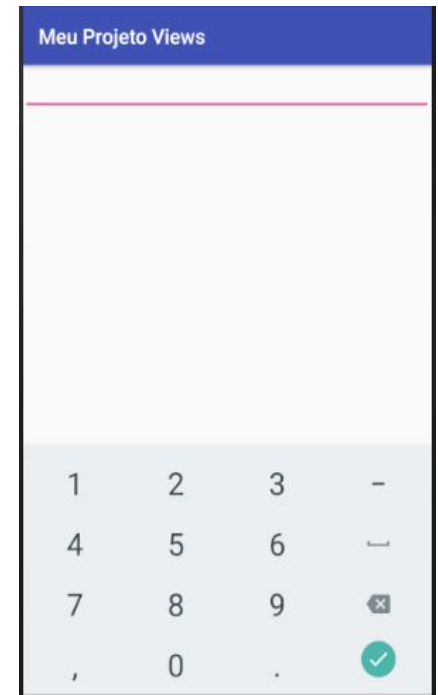
- ▶ O atributo `inputType` pode ser usado para identificar o tipo de dado que está sendo digitado
  - – Texto qualquer, e-mail, número telefônico, etc



`android:inputType="phone"`



`android:inputType="textEmailAddress"`



`android:inputType="number"`

# EDITTEXT: INPUT TYPE

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

inputType	Significado
<b>textCapCharacters</b>	Todos os caracteres em maiúsculo
<b>textCapSentences</b>	Toda frase começa com caractere maiúsculo
<b>textMultiLine</b>	Texto com múltiplas linhas
<b>textPassword</b>	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

- ▶ 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

- ▶ 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 />
```

```
16
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 />
```

```
49
50
51 fun onClickCheckbox(view: View) {
52
53     val checkBox = view as CheckBox
54     var checked = checkBox.isChecked
55
56     if(view.id == R.id.checkbox_futebol){
57         //Processa o clique
58     }else if(view.id == R.id.checkbox_golfe){
59         //Processa o clique
60     }
61 }
```



# CHECKBOX: EVENTO CLIQUE

- ▷ É possível também utilizar um listener

```
13
14 class MainActivity : AppCompatActivity(), View.OnClickListener {
15
16
17
18 override fun onCreate(savedInstanceState: Bundle?) {
19     super.onCreate(savedInstanceState)
20     setContentView(R.layout.activity_main)
21
22     checkbox_futebol.setOnClickListener { this }
23 }
24
25 override fun onClick(v: View?) {
26     // Evento do clique
27 }
```

Chamado  
quando houver  
o clique

# CHECKBOX: MUDANÇA DE ESTADO

- ▶ Além de ler o estado checkbox, é possível alterá-lo também via programação

- `checkbox.isChecked = boolean`

Marca ou desmarca o checkbox

- `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>
```

## Meu Projeto Views

- ☐ Futebol
- ☐ Golfe
- ☐ Tênis

# RADIO BUTTON: EVENTO CLIQUE

- ▶ 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"
    />
```

```
30
31
32
33
34
35
36
37
38
39
40
41
42
```

```
fun onClickRadioButton(view: View) {
    val radiobutton = view as RadioButton
    var checked = radiobutton.isChecked

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

# RADIO BUTTON: EVENTO CLIQUE

- ▶ Um listener também pode ser usado

```
13
14 class MainActivity : AppCompatActivity(), View.OnClickListener {
15
16
17
18 override fun onCreate(savedInstanceState: Bundle?) {
19     super.onCreate(savedInstanceState)
20     setContentView(R.layout.activity_main)
21
22     radiobutton_futebol.setOnClickListener { this }
23 }
24
25 override fun onClick(v: View?) {
26     // Evento do clique
27 }
```

Chamado  
quando houver  
o clique

# RADIO BUTTON: MUDANÇA DE ESTADO

- ▶ Além de ler o estado radio button, é possível alterá-lo também via programação

- `radiobutton.isChecked = boolean`

Marca ou desmarca o radio button

- `radiobutton.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 E SWITCH

- ▶ 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"
/>
```

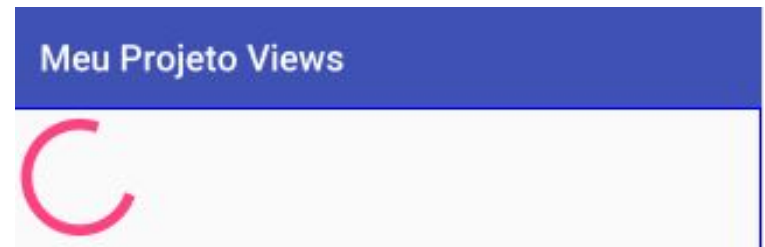
```
30
31
32
33
34
35
36
37
38
39
40
fun onClickToggle(view: View) {
    val togglebutton = view as ToggleButton
    var checked = togglebutton.isChecked
    if(view.id == R.id.toggle){
        //Processa o clique
    }
}
```



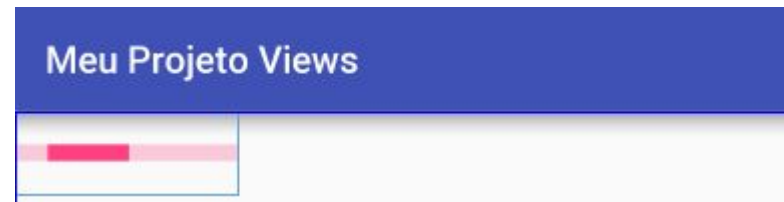
# PROGRESS BAR

## ► 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

```
17
18  override fun onCreate(savedInstanceState: Bundle?) {
19      super.onCreate(savedInstanceState)
20      setContentView(R.layout.activity_main)
21
22      progress.progress = 30
23
24  }
```

# 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

- ▶ 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>
```

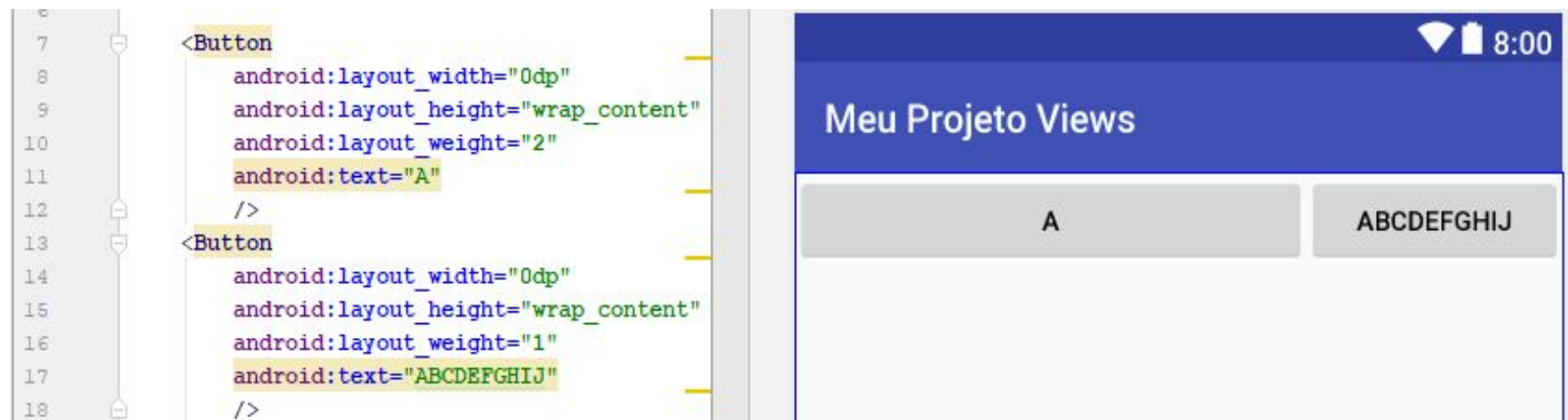
## Meu Projeto Views

E-mail:

ENVIAR

# LINEAR LAYOUT: WEIGHT

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



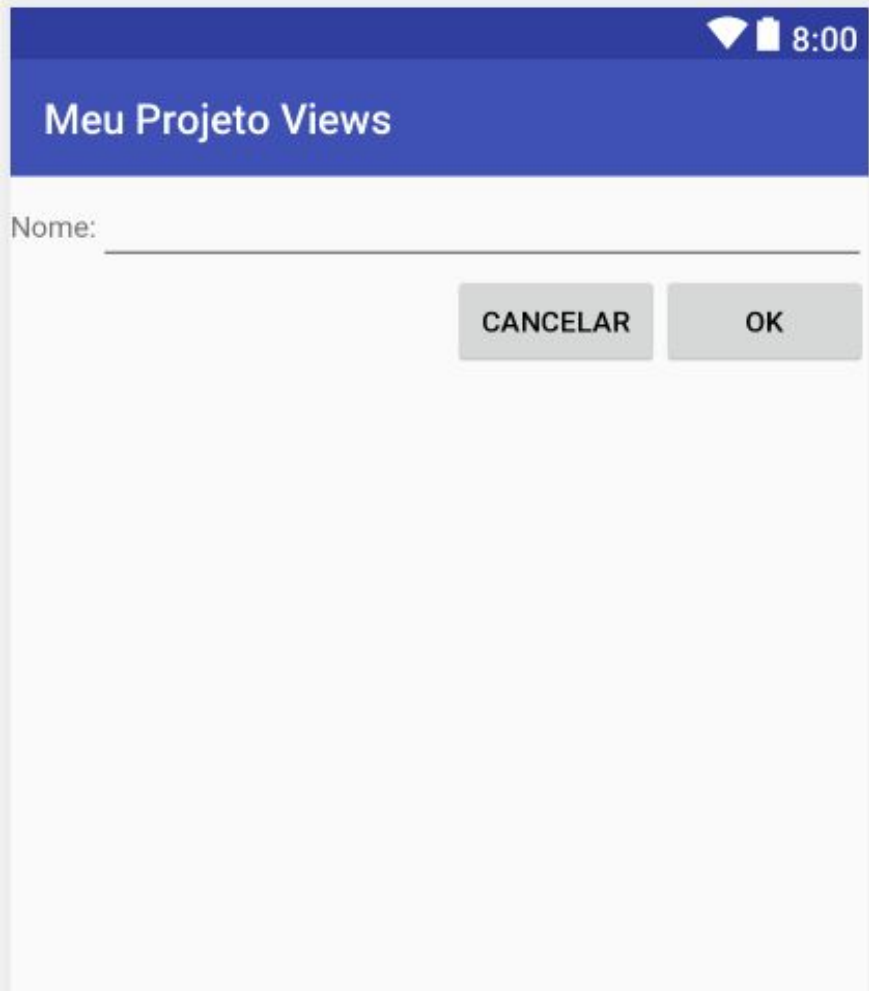
# RELATIVE LAYOUT

- ▷ 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

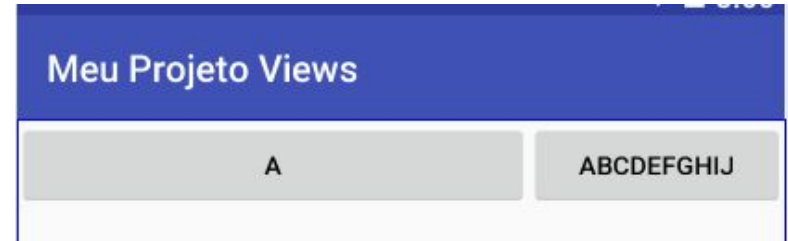


# CONSTRAINT LAYOUT

- ▷ 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"
>
```

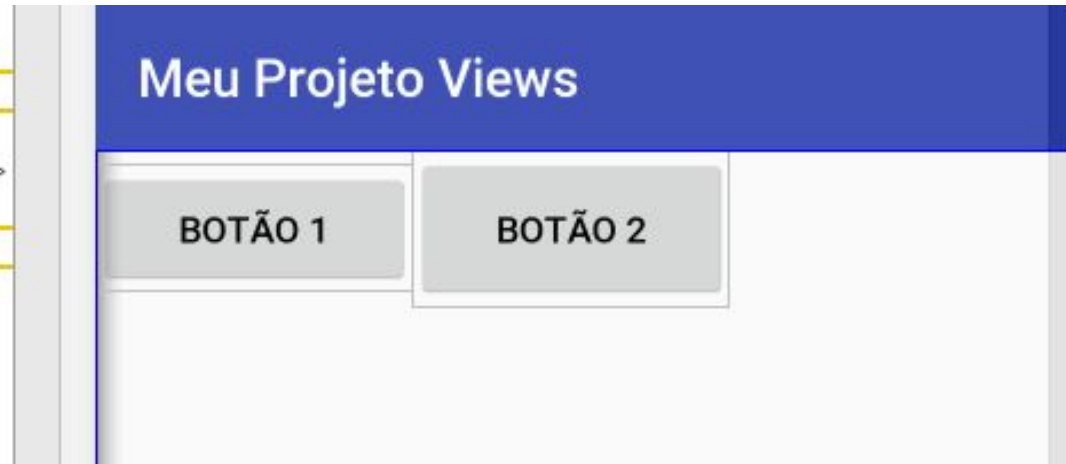


Aplicando Margin ao layout

# 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

- ▶ 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

```
<?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">

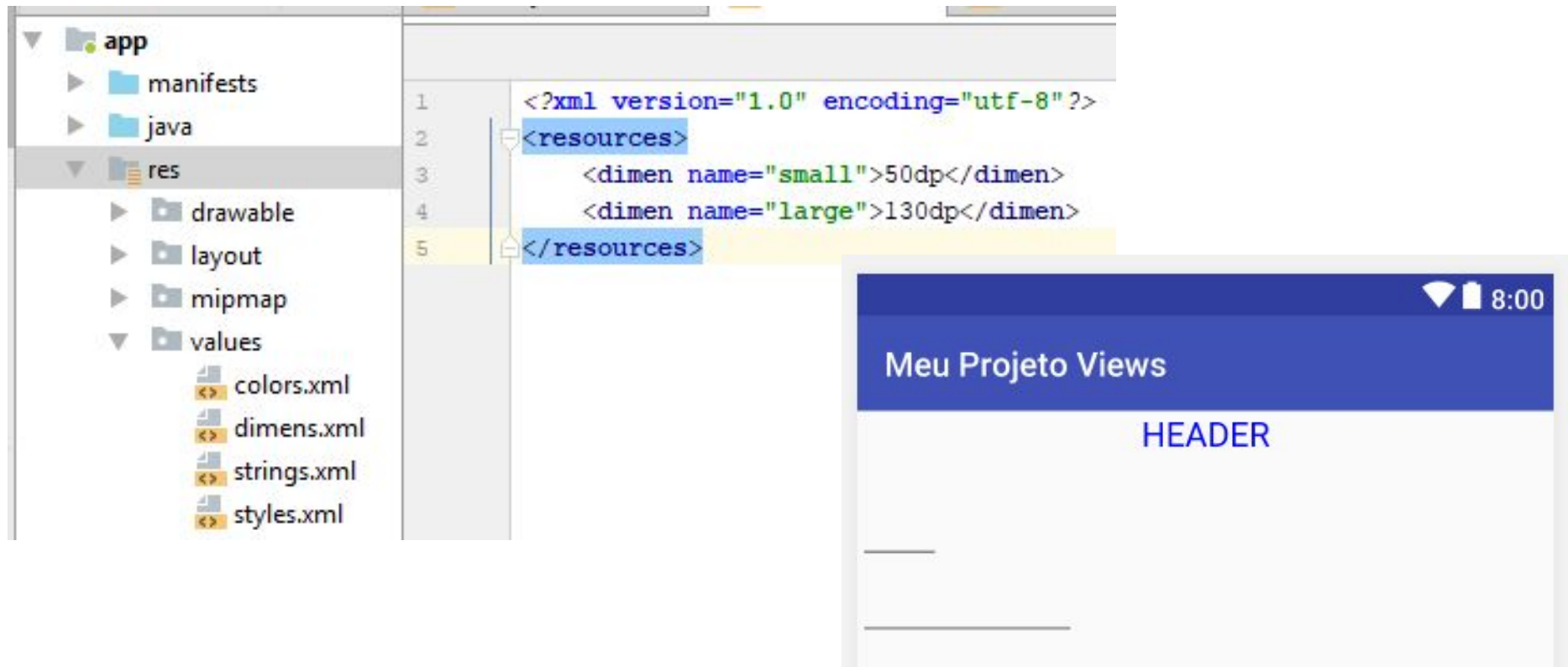
    <include layout="@layout/header" android:id="@+id/lay_header"/>

    <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

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

/res/**layout-port**/activity\_layout.xml



Retrato

/res/**layout-land**/activity\_layout.xml



Paisagem



# FORÇANDO ORIENTAÇÃO

- ▷ 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)  
;
```

# API DE COMPATIBILIDADE

► Verificar se o repositório onde o Android Studio baixa a API de compatibilidade está disponível:

- Tools -> Android -> SDK Manager
- Aba: SDK Tools - > Support Repository

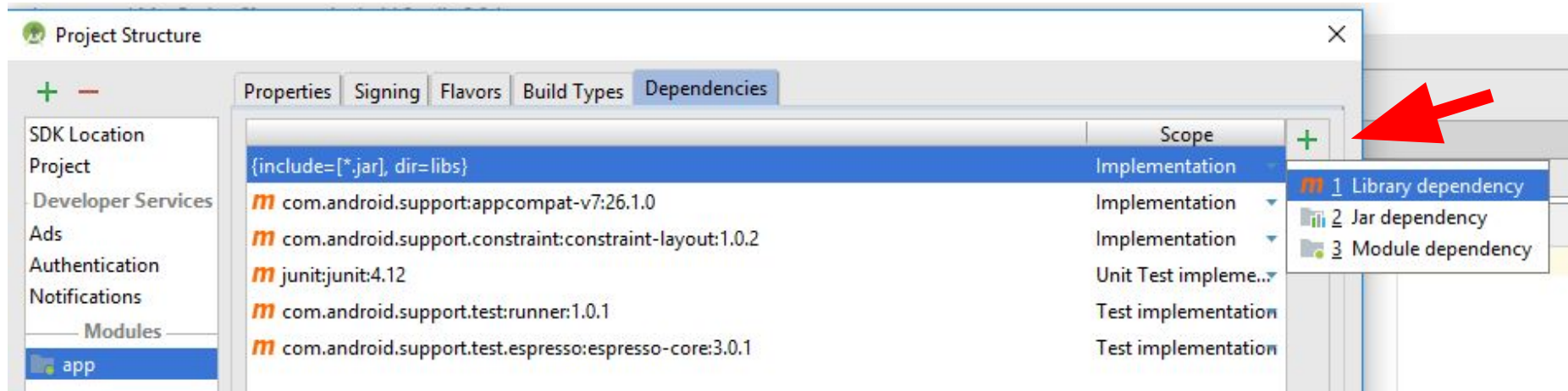
► Grade

```
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

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



# EXEMPLO - COMPATIBILIDADE

```
12
13  import kotlinx.android.synthetic.main.activity_main.*
14  import android.support.v7.app.AppCompatActivity
15
16  class MainActivity : AppCompatActivity() {
17
18      override fun onCreate(savedInstanceState: Bundle?) {
19          super.onCreate(savedInstanceState)
20          setContentView(R.layout.activity_main)
21
22          var bar = supportActionBar
23
24      }
25
26
```

The image shows a code editor snippet with two red arrows. One arrow points from the right towards the class declaration 'class MainActivity : AppCompatActivity() {' on line 16. The other arrow points from the right towards the 'supportActionBar' property access in the line 'var bar = supportActionBar' on line 22.

Obrigado!  
**Dúvidas?**

Professor Emerson Alencar  
[emerson@imd.ufrn.br](mailto:emerson@imd.ufrn.br)