



DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

PROF^a. M.Sc. JULIANA H Q BENACCHIO

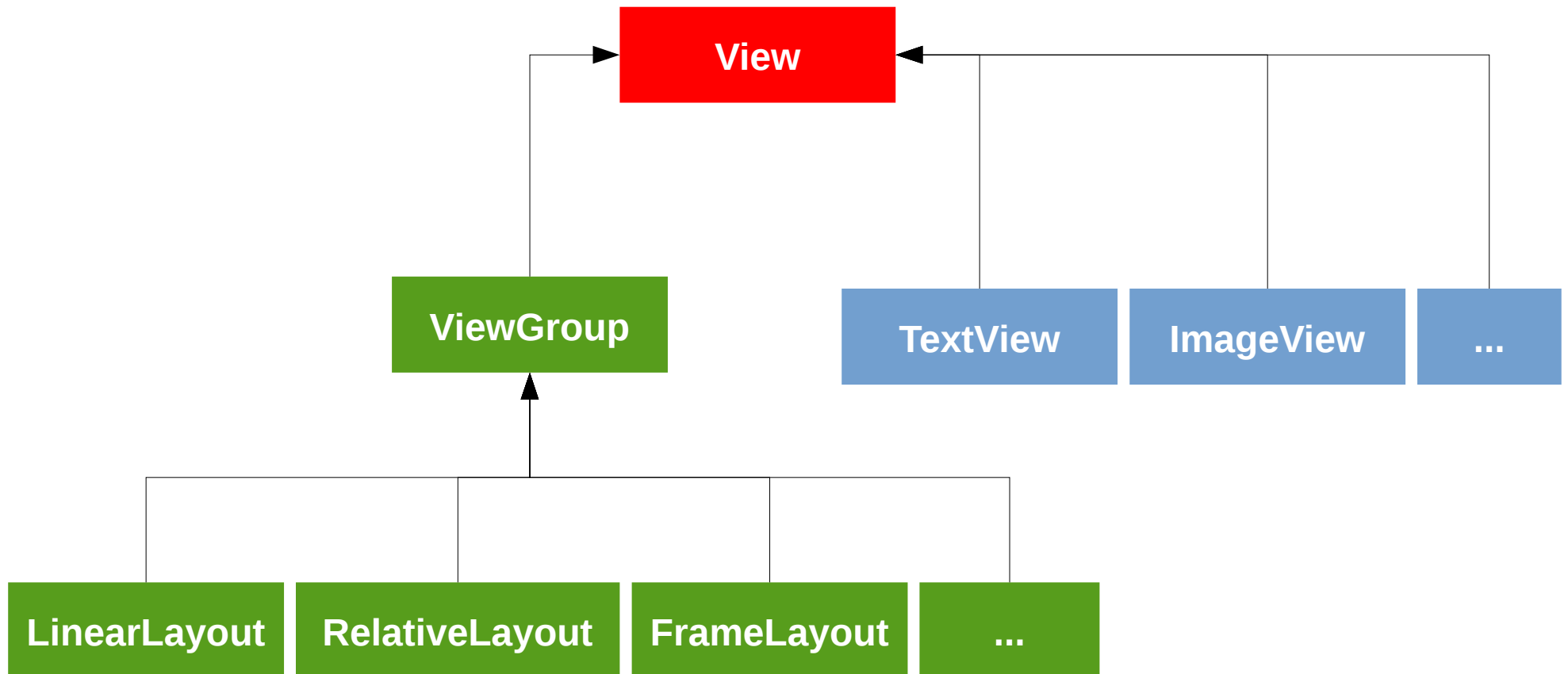


Interfaces Gráficas no Android

- As telas das aplicações são compostas por views
 - Views herdam de **android.view.View**
- As views são organizadas nas telas através de layouts
 - Layouts herdam de **android.view.ViewGroup**



Interfaces Gráficas no Android



Activities e Views



Define as views e trata os eventos.
As views devem estar, de preferência, em arquivos XML

Componentes da tela



Views via Programação



MainActivity.java

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
  
    TextView t = new TextView(this);  
    t.setText("Hello World");  
    setContentView(t);  
}
```

As views são criadas no código



Views via Declaração



`/res/layout/activity_main.xml`

```
...  
<TextView  
    android:text="Hello World"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
>  
...
```

As views são declaradas no XML

`MainActivity.java`

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
}
```



Lendo Views do XML

`/res/layout/activity_main.xml`

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

Criação de ID
para a view

`MainActivity.java`

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    TextView t = (TextView)findViewById(R.id.texto);  
    t.setText("Hello World Android");  
}
```



Views do Android

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



- View: **TextView**
- Exibição de textos na tela

```
<TextView  
    android:text="@string/texto"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
/>
```



Button

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

```
<Button
    android:text="@string/button_text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
/>
```

```
<Button
    android:text="@string/button_text"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:drawableLeft="@drawable/ic_action_mail"
/>
```

Button



- View: **ImageButton**
- Botão composto por ícone

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



Button: Eventos de Clique

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

```
<Button  
    android:text="@string/button_text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:onClick="enviar"  
>
```

Define o método a ser chamado no clique

```
public void enviar(View view) {  
    // Processa a ação  
}
```

Método definido na activity

Button: Eventos de Clique

- É possível também utilizar um *listener*

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

O botão deve ter um ID

```
public class MainActivity extends Activity implements View.OnClickListener {  
  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        Button button = (Button) findViewById(R.id.button);  
        button.setOnClickListener(this);  
    }  
    public void onClick(View v) {  
        // Processa a ação  
    }  
}
```

A activity adiciona um listener ao botão

onClick() é chamado quando houver o clique

Text Field

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

```
<EditText  
    android:id="@+id/text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:hint="@string/text_hint"  
/>
```



Text Field: Input Type

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



Text Field: Input Type

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

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

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

Combina mais de uma característica

Checkbox

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

```
<Checkbox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/futebol" />
<Checkbox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/golfe" />
<Checkbox
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/tenis" />
```



Checkbox: Eventos de Clique

- A *activity* pode ser notificada quando um checkbox é clicado

```
<CheckBox
    android:id="@+id/checkbox_futebol"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/futebol"
    android:onClick="onClickCheckbox" />
<CheckBox
    android:id="@+id/checkbox_golfe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/golfe"
    android:onClick="onClickCheckbox" />
```

Checkbox: Eventos de Clique



```
public void onClickCheckbox(View view) {  
    CheckBox checkbox = (CheckBox) view;  
    boolean checked = checkbox.isChecked();  
  
    if (view.getId() == R.id.checkbox_futebol) {  
        // Processa o clique  
    } else if (view.getId() == R.id.checkbox_golfe) {  
        // Processa o clique  
    }  
}
```



Checkbox: Eventos de Clique

- Um *listener* também pode ser usado

```
public class MainActivity extends Activity implements View.OnClickListener {  
  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        CheckBox checkbox = (CheckBox) findViewById(R.id.checkbox_futebol);  
        checkbox.setOnClickListener(this);  
    }  
  
    public void onClick(View view) {  
        // Processa o evento de clique  
    }  
}
```



Checkbox: Mudança de Estado



- 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



Radio Button



```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/futebol" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/golfe" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/tenis" />

</RadioGroup>
```



Radio Button: Eventos de Clique

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

```
<RadioButton
    android:id="@+id/radiobutton_futebol"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/futebol"
    android:onClick="onClickRadioButton" />
<RadioButton
    android:id="@+id/radiobutton_golfe"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/golfe"
    android:onClick="onClickRadioButton" />
```



Radio Button: Eventos de Clique



```
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: Eventos de Clique

- Um *listener* também pode ser usado

```
public class MainActivity extends Activity implements View.OnClickListener {  
  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        RadioButton rb = (RadioButton) findViewById(R.id.radiobutton_futebol);  
        rb.setOnClickListener(this);  
    }  
  
    public void onClick(View view) {  
        // Processa o evento de clique  
    }  
}
```



Radio Button: Mudança de Estado



- 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



Toggle Button



- 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="@string/desligado"
    android:textOn="@string/ligado" />
<Switch
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOff="@string/desligado"
    android:textOn="@string/ligado" />
```



Toggle Button: Eventos de Clique



- 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="@string/desligado"
    android:textOn="@string/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
    }
}
```



Toggle Button: Eventos de Clique

- Um *listener* também pode ser usado

```
public class MainActivity extends Activity implements View.OnClickListener {  
  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ToggleButton button = (ToggleButton) findViewById(R.id.toggle);  
        button.setOnClickListener(this);  
    }  
  
    public void onClick(View view) {  
        // Processa o evento de clique  
    }  
}
```



Progress Bar



- View: **ProgressBar**
- Mostra um indicador de progresso

```
<ProgressBar  
    android:id="@+id/progress"  
    style="@android:style/Widget.ProgressBar.Large"  
    android:indeterminate="true"  
/>
```



Progress Bar



```
<ProgressBar
  android:id="@+id/progress"
  style="@android:style/Widget.ProgressBar.Horizontal"
  android:indeterminate="true"
/>
```

```
<ProgressBar
  android:id="@+id/progress"
  style="@android:style/Widget.ProgressBar.Horizontal"
  android:indeterminate="false"
  android:progress="30"
  android:max="100"
/>
```



Progress Bar

- O indicador de progresso pode ser configurado também via programação

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    ProgressBar progress = (ProgressBar) findViewById(R.id.progress);  
    progress.setProgress(30);  
}
```



Activity e a Progress Bar

- Um indicador de progresso pode ser exibido também na *activity*
- Não é necessário usar a view **ProgressBar**

```
requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);  
setContentView(R.layout.activity_main);  
setProgressBarIndeterminateVisibility(true);
```

```
requestWindowFeature(Window.FEATURE_PROGRESS);  
setContentView(R.layout.activity_main);  
setProgressBarVisibility(true);  
setProgress(4000);
```



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



Layouts do Android

- O Android possui diversos layouts importantes utilizados para organizar interfaces gráficas
- Localizados no pacote **android.widget**
- Exemplos
 - **LinearLayout**
 - **RelativeLayout**
 - **FrameLayout**
 - **TableLayout**
 - **GridLayout**
 - **ScrollView / HorizontalScrollView**



LinearLayout



- É um dos gerenciadores de layout mais utilizados
- Organiza os componentes na horizontal ou na vertical
- Atributo **android:orientation** define a orientação, sendo horizontal o valor padrão.



LinearLayout



```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/txt_email" />

    <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="@string/btn_processor" />

</LinearLayout>
```

LinearLayout: Weight

- 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" />

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



LinearLayout: Weight

- O componente com o maior peso ocupará o maior espaço na tela
- Sempre que utilizar o atributo **weight**, deve-se definir a largura ou altura da view com **0dp**.

```
<Button  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="2" />  
  
<Button  
    android:layout_width="0dp"  
    android:layout_height="wrap_content"  
    android:layout_weight="1" />
```



LinearLayout: Layout Gravity

- Define a posição da view com relação ao seu layout pai

```
<Button  
    android:layout_gravity="left" />  
  
<Button  
    android:layout_gravity="center" />  
  
<Button  
    android:layout_gravity="right" />
```



RelativeLayout

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



RelativeLayout



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/txt_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" />
```



RelativeLayout



```
<Button
    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: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>
```



FrameLayout

- É um layout capaz de mostrar uma única view
- Caso mais de uma view seja especificada, elas são empilhadas uma sobre a outra
- A última view definida ficará no topo



FrameLayout



```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:src="@drawable/android" />

</FrameLayout>
```



TableLayout

- Cada linha da tabela é formada por um **`android.widget.TableRow`**
- Atributo **`android:stretchColumns`** – faz com que as colunas ocupem o espaço disponível na tela, expandindo-as. Semelhante ao “**`colspan`**” do HTML
- Atributo **`android:shrinkColumns`** – faz com que as colunas especificadas sejam sempre exibidas na tela



TableLayout



```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TableRow>
        <Button android:text="@string/l1c1"/>
        <Button android:text="@string/l1c2"/>
    </TableRow>

    <TableRow>
        <Button android:text="@string/l2c1"/>
        <Button android:text="@string/l2c2"/>
    </TableRow>

    <TableRow>
        <Button android:text="@string/l3c1"/>
        <Button android:text="@string/l3c2"/>
    </TableRow>

</TableLayout>
```

TableLayout e stretchColumns

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:stretchColumns="1">

    <TextView
        android:text="@string/lista_de_produtos"
        android:gravity="center" />

    <View
        android:layout_height="2dp"
        android:background="@android:color/holo_orange_dark"
        android:layout_margin="@dimen/activity_horizontal_margin" />

    <TableRow>
        <TextView
            android:text="@string/prod_a" />
        <TextView
            android:text="@string/reais_100"
            android:layout_gravity="right" />
    </TableRow>
```

TableLayout e stretchColumns



```
<TableRow>
  <TextView
    android:text="@string/prod_b" />
  <TextView
    android:text="@string/reais_200"
    android:layout_gravity="right" />
</TableRow>
<TableRow>
  <TextView
    android:text="@string/prod_c" />
  <TextView
    android:text="@string/reais_300"
    android:layout_gravity="right" />
</TableRow>

<View
  android:layout_height="2dp"
  android:background="@android:color/holo_orange_dark"
  android:layout_margin="@dimen/activity_horizontal_margin" />
</TableLayout>
```

GridLayout

- Organiza as views em linhas e colunas, utilizando os atributos **layout_row** e **layout_column**
- Os atributos **android:layout_columnSpan** e **android:layout_rowSpan** também funcionam de forma similar ao **span** do HTML
- O **GridLayout** só está disponível a partir do Android 4.0 (API level 14)



GridLayout



```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:rowCount="2"
    android:columnCount="2">
```

```
<Button
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="0"
    android:layout_column="0"
    android:text="@string/botao1" />
```

```
<Button
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="0"
    android:layout_column="1"
    android:text="@string/botao2" />
```

GridLayout



```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="1"
    android:layout_column="0"
    android:text="@string/botao3" />
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_row="1"
    android:layout_column="1"
    android:text="@string/botao4" />
```

```
</GridLayout>
```

ScrollView e HorizontalScrollView



- Permite a rolagem da tela caso seja necessário

```
<ScrollView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:id="@+id/lay_screen"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
    </LinearLayout>
</ScrollView>
```

- **HorizontalScrollView** funciona da mesma forma, mas habilita a rolagem horizontal da tela



Margin



- Layouts podem definir margens
 - Esquerda, direita, superior e inferior
- As views inseridas no layout respeitam as margens definidas

```
<LinearLayout  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:layout_marginLeft="40dp"  
    android:layout_marginRight="40dp"  
    android:layout_marginTop="20dp">  
    ...  
</LinearLayout>
```



Padding

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

```
<LinearLayout>
  <Button
    android:layout_width="120dp"
    android:layout_height="wrap_content" />

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



Inclusão de Layouts

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



Inclusão de Layouts



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="@string/txt_header_title"
        android:textSize="20sp"
        android:textColor="#0000FF" />
</LinearLayout>
```



Inclusão de Layouts



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

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

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/lay_header"
        android:layout_centerHorizontal="true"
        android:text="Pressione" />

</RelativeLayout>
```



Usando o *Resource* `dimen`



- Dimensões podem ser especificadas como *resources* para serem reaproveitadas

`dimens.xml`

```
<resources>
  <dimen name="small">50dp</dimen>
  <dimen name="large">130dp</dimen>
</resources>
```

```
<EditText
  android:layout_width="@dimen/small"
  android:layout_height="wrap_content" />
<EditText
  android:layout_width="@dimen/large"
  android:layout_height="wrap_content" />
```





- 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



Layouts de Acordo com a Orientação



- Basta definir o arquivo de layout nas pastas de *resources* correspondentes
- A pasta **/res/layout** pode ser usada quando o layout é o mesmo para ambas as orientações

```
/res/layout-port/activity_layout.xml
```

Layout para a orientação retrato

```
/res/layout-land/activity_layout.xml
```

Layout para a orientação paisagem



Forçando uma 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)
```

