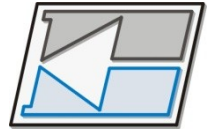


Introdução à Programação Android

Prof. Samir Bonho

Aula 2

Florianópolis, XX de Fevereiro de 2014



Sumário

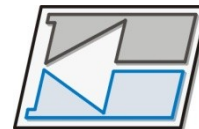
✓ Gerenciadores de *Layout*

- ✓ View / ViewGroup
- ✓ FrameLayout
- ✓ LinearLayout
- ✓ TableLayout
- ✓ RelativeLayout
- ✓ AbsoluteLayout

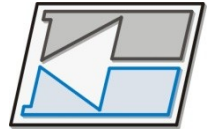
✓ Criação de layouts por API

✓ Fontes:

- ✓ Departamento de Computação | ICEB | Universidade Federal de Ouro Preto
- ✓ Android Developers. Disponível em <http://developer.android.com/>



View/View Group



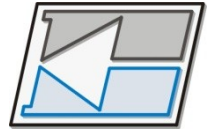
View/View Group

- Introdução

- ✓ O *layout* basicamente diz respeito a forma como os elementos são organizados na tela.

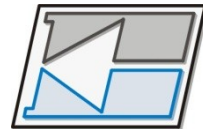
- ✓ Alguns *layouts* podem organizar elementos na horizontal ou mesmo vertical.

- ✓ *Layouts* funcionam basicamente como slots organizadores para a exibição de elementos.



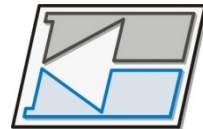
View/View Group: Introdução

- ✓ Classe *android.view.View* é a classe-mãe de todos os componentes visuais do Android.
- ✓ Cada subclasse de *View* precisa implementar o método *onDraw(Canvas)*, responsável por desenhar o componente na tela.
- ✓ Separação entre *widgets* e gerenciadores de *layout*:
 - *Widget* : Elemento simples que herda da classe *View*: Button, ImageView, etc.
 - *Gerenciadores de layout*: Constituem subclasses de *android.view.ViewGroup* e são popularmente chamadas de *layouts*.



View/View Group: Introdução

- ✓ **AbsoluteLayout**: Posiciona elementos na tela através de coordenadas x e y.
- ✓ **FrameLayout**: Utilizado por componentes que precisam ocupar a tela inteira.
- ✓ **LinearLayout**: Organiza os elementos na horizontal ou vertical.
- ✓ **TableLayout**: Organiza os elementos em formato de tabela.
- ✓ **RelativeLayout**: Posiciona elementos na tela utilizado como referência outros componentes.



View/View Group: Introdução

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="fill_parent" android:layout_height="fill_parent"

android:orientation="vertical">

<TextView

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:text="Escolha Sim ou Não" />

<Button android:id="@+id/btSim"

android:layout_width="wrap_content"

android:layout_height="wrap_content"

android:text=" Sim " />

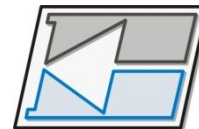
<Button android:id="@+id/btNao"

android:layout_width="wrap_content"

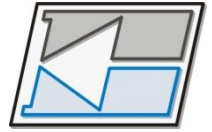
android:layout_height="wrap_content"

android:text=" Não " />

</LinearLayout>



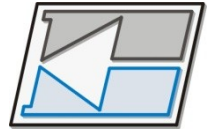
LinearLayout



View/View Group: Introdução

✓ Gerenciador de *layout* <LinearLayout>:
Organiza seus elementos filhos em uma lista vertical ou horizontal.

- `android:orientation="vertical"` : Organiza os elementos em uma lista vertical; e
- `android:orientation="horizontal"` : Organiza os elementos em uma lista horizontal.

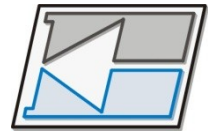


View/View Group: Introdução

✓ *Definição da altura e largura que o layout deve ocupar é feita através de dois outros atributos:*

- **android:layout_width** : Define a **largura** que o *layout* deve ocupar na tela;
- **android:layout_height** : Define a **altura** que o *layout* deve ocupar na tela.

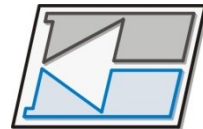
✓ No momento da criação do layout devem ser definidos os elementos de tamanho o orientação.



View/View Group: Introdução

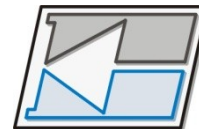
✓ Atributos de largura e altura podem receber os seguintes valores:

Valor	Descrição
número	Número inteiro especificando o tamanho do componente. Ex.: 50px.
fill_parent	Elemento irá ocupar todo o espaço definido pelo seu pai (layout). Deve ser utilizado sempre que algum <i>layout</i> ou componente precisar ocupar a tela inteira ou todo o espaço de <i>layout</i> definido pelo <i>layout</i> -pai.
wrap_content	Componente ocupará apenas o espaço necessário na tela. Ex.: Para <i>TextView</i> e <i>ImageView</i> o tamanho será suficiente para exibir o texto ou imagem requeridos.

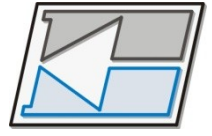


View/View Group: Introdução

- ✓ Os parâmetros android: **layout_width** e **android:layout_height** são definidos pela classe ViewGroup.LayoutParams.
- ✓ Se a tela for construída diretamente via API será necessário conhecer a classe acima.
- ✓ Cada classe de layout define uma subclasse de parâmetros (LayoutParams).
- ✓ Se uma nova classe de layout é criada, então uma nova subclasse de parâmetros também deve ser definida.

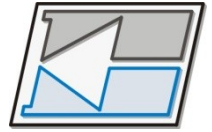


FrameLayout




FrameLayout

- ✓ Definido pela classe `android.widget.FrameLayout`.
- ✓ Mais simples de todos os gerenciadores de *layout*.
- ✓ Utilizado quando a tela possui somente um componente que pode preencher a tela inteira.
- ✓ Componente sempre é posicionado à partir do canto superior esquerdo.
- ✓ Dependendo do tamanho pode ocupar a tela inteira.



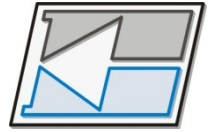
```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#8B8B83">
```




```
<ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

    android:src="@drawable/smile" />
```

```
</FrameLayout>
```



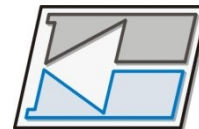
```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="#8B8B83">
```



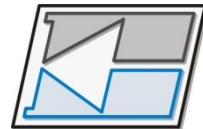
```
<ImageView
    android:layout_width=" fill_parent "
    android:layout_height=" fill_parent "

    android:src="@drawable/smile" />
```

```
</FrameLayout>
```

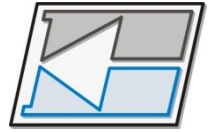



LinearLayout



LinearLayout

- ✓ Definido pela classe *android.widget.LinearLayout*.
- ✓ Gerenciador de *layout* **mais** utilizado.
- ✓ Também contém os atributos “**android:layout_width**” e “**android:layout_height**”.
- ✓ Possível controlar atributos que gerenciam o modo de exibição dos componentes: vertical ou horizontal - **android:orientation**.



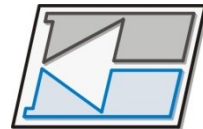
EX.: Exibindo elementos alinhados horizontalmente.

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res-
android"

    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

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

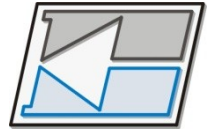
        <ImageView
        android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```



LinearLayout: Exemplo

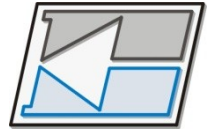
EX.: Exibindo elementos alinhados verticalmente

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res-
android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical" >
    <ImageView
android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/smile" />
    <ImageView
android:layout_width="wrap_content"
```



LinearLayout

- Controle de alinhamento
 - ✓ Componentes dentro de um *LinearLayout* podem ser alinhados de diversas formas na tela.
 - ✓ Atributo *android:layout_gravity* deve ser utilizado para alinhamento dos componentes.
 - ✓ Possíveis valores:
 - *top, bottom, left, right, center_vertical, fill_vertical, center_horizontal, fill_horizontal, center e fill.*



LinearLayout

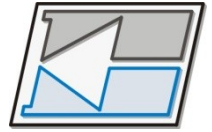
- Controle de alinhamento: Exemplo

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

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

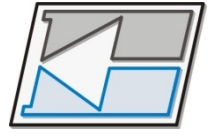
    <ImageView android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/smile2"
        android:layout_gravity="right"/>

</LinearLayout>
```



LinearLayout

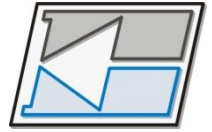
- Controle de peso e relevância
 - ✓ Elementos podem receber um peso cada um.
 - ✓ Os que receberem maior peso, ocuparão um espaço maior na tela.
 - ✓ Peso dado através do atributo “*android:layout_weight*”.
 - ✓ Inteiros como possíveis valores.



LinearLayout

- Controle de peso e relevância: **Exemplo**

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:orientation="vertical">
    <TextView android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Nome: "/>
    <EditText android:layout_width="fill_parent" android:layout_height="wrap_content"
        android:textColor="#ff0000"/>
    <TextView android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="E-mail: "/>
    <EditText android:layout_width="fill_parent" android:layout_height="wrap_content"/>
    <TextView android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Observações: "/>
    <EditText android:layout_width="fill_parent" android:layout_height="fill_parent"
        android:layout_weight="1" />
    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Enviar"/>
</LinearLayout>
```

LinearLayout

- Controle de peso e relevância: **Exemplo 2**

```
<LinearLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="fill_parent"
```

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

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

```
    <EditText android:layout_width="fill_parent"
```

```
        android:layout_height="0px"
```

```
        android:layout_weight="1" android:text="Texto (weight = 1)"/>
```

```
    <EditText android:layout_width="fill_parent"
```

```
        android:layout_height="0px"
```

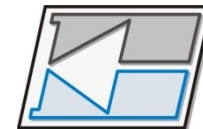
```
        android:layout_weight="2" android:text="Texto (weight = 2)"/>
```

```
    <EditText android:layout_width="fill_parent"
```

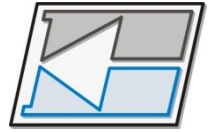
```
        android:layout_height="0px"
```

```
        android:layout_weight="3" android:text="Texto (weight = 3)"/>
```

```
</LinearLayout>
```

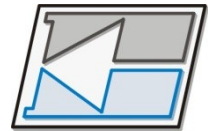


TableLayout



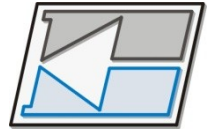
TableLayout

- Controle de peso e relevância
 - ✓ Definido pela classe *android.widget.TableLayout*.
 - ✓ Útil para a construção de formulários.
 - ✓ Cada linha da tabela é por um *android.widget.TableRow*.
 - ✓ Possui atributos como *android:stretchColumns* e *android:shrinkColumns*: Recebem os índices das colunas que devem ser alteradas.

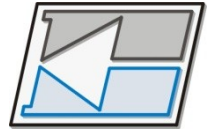


TableLayout

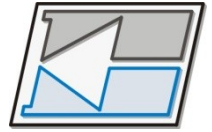
- Ajuste das colunas
 - ✓ Índice das colunas inicia sempre em 0 (zero).
 - ✓ Ex.: Para alterar a segunda coluna, o parâmetro 1 deve ser passado para os atributos **android:stretchColumns** / **android:shrinkColumns**.
 - **android:stretchColumns**: Faz com que as colunas especificadas ocupem o espaço disponível na tela, expandindo-as.
 - **android:shrinkColumns**: Faz com que as colunas especificadas sejam sempre exibidas na tela. Caso o texto seja maior que o espaço, duas ou mais linhas serão exibidas.



```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:shrinkColumns="2">
    <TableRow>
        <TextView android:text="Coluna 1" />
        <TextView android:text="Coluna 2" />
    </TableRow>
    <TableRow>
        <TextView android:text="Coluna 1"/>
        <TextView android:text="Coluna 2"/>
    <TextView android:text="Texto gigantesco mas que irá ser cortado
    automaticamente pelo layout"/>
    </TableRow>
    <TableRow>
        <TextView android:text="Coluna 1"/>
        <TextView android:text="Coluna 2"/>
        <TextView android:text="Coluna 3"/>
    </TableRow>
</TableLayout>
```

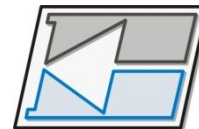


```
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:stretchColumns="1">
<TextView android:text="Lista de Produtos" />
    <View android:layout_height="20px"
android:background="#FF909090"/>
    <TableRow>
<TextView android:text="Produto A" />
<TextView android:text="R$ 100,00"
android:layout_gravity="right" />
    </TableRow>
    <TableRow>
        <TextView android:text="Produto B" />
        <TextView android:text="R$ 200,00" />
    </TableRow>
</TableLayout>
```

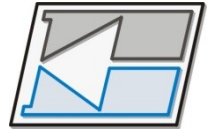


TableLayout: Exemplo 2 (cont.)

```
<TextView android:text="Produto C" />
<TextView android:text="R$ 300,00"
android:layout_gravity="right" />
</TableRow>
<View android:layout_height="20px"
android:background="#FF909090"/>
<LinearLayout android:layout_width="wrap_content"
    android:layout_height="fill_parent"
    android:gravity="bottom">
<Button android:layout_width="wrap_content"
android:layout_height="80px"
    android:text="Cancelar"/>
<Button android:layout_width="wrap_content"
android:layout_height="80px"
    android:text="Comprar"/>
</LinearLayout>
```

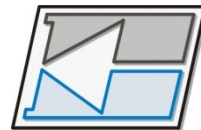


RelativeLayout



RelativeLayout

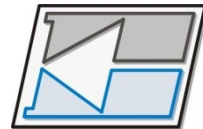
- Introdução
 - ✓ Definido pela classe *android.widget.RelativeLayout*.
 - ✓ Pode posicionar elementos abaixo, acima ou ao lado de um já existente.
 - ✓ Como o próprio nome diz, a posição de um elemento é sempre relativo a outro.
 - ✓ Elemento referenciado precisa aparecer antes no layout.



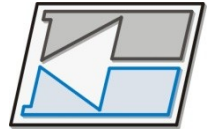
RelativeLayout

✓ Os atributos mostrados abaixo podem ser utilizados para posicionamento de um elemento:

Atributo	Descrição
android:layout_below	Posiciona abaixo do componente indicado.
android:layout_above	Posiciona acima do componente indicado.
android:layout_toRightOf	Posiciona à direita do componente indicado.
android:layout_toLeftOf	Posiciona à esquerda do componente indicado.
android:layout_alignParentTop	Alinha no topo do componente indicado.
android:layout_alignParentBottom	Alinha abaixo do componente indicado.
android:layout_marginTop	Utilizado para definir espaço na margem superior do componente.
android:layout_marginRight	Utilizado para definir um espaço à direita do componente.
android:layout_marginLeft	Utilizado para definir um espaço à esquerda do componente.



```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingBottom="@dimen/activity_vertical_margin"
        android:paddingLeft="@dimen/activity_horizontal_margin"
        android:paddingRight="@dimen/activity_horizontal_margin"
        android:paddingTop="@dimen/activity_vertical_margin"
        tools:context=".MainActivity"
        android:background="#8B8B83"
        android:padding="20px">
<TextView android:id="@+id/labelUsuario"
    android:layout_width="120px"
    android:layout_height="wrap_content"
    android:text="Usuário: "/>
```



RelativeLayout: Exemplo (cont. 1)

<EditText

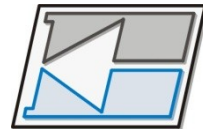
```
    android:id="@+id/campoUsuario"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:background="@android:drawable/editbox_background"  
    android:layout_toRightOf="@id/labelUsuario" />
```

<TextView android:id="@+id/labelSenha"

```
    android:layout_width="120px"  
    android:layout_height="wrap_content"  
    android:layout_below="@id/campoUsuario"  
    android:gravity="left"  
    android:text="Senha: " />
```

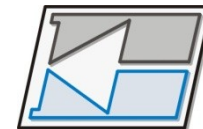
<EditText android:id="@+id/campoSenha"

```
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:background="@android:drawable/editbox_background"  
    android:layout_toRightOf="@id/labelSenha"  
    android:layout_alignTop="@id/labelSenha"  
    android:password="true" />
```

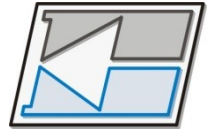


RelativeLayout: Exemplo (cont. 2)

```
<Button android:id="@+id/btLogin"
        android:layout_width="wrap_content"
        android:layout_height="80px"
        android:layout_below="@id/campoSenha"
        android:layout_marginTop="20px"
        android:layout_alignParentRight="true"
        android:text="Login" />
</RelativeLayout>
```



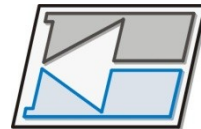
AbsoluteLayout



AbsoluteLayout

- Introdução

- ✓ Definido pela classe *android.widget.AbsoluteLayout*.
- ✓ Permite controlar a posição exata dos elementos na tela através do uso das coordenadas x e y.
 - ✓ Uso dos atributos *android:layout_x* e *android:layout_y*.
- ✓ **Importante**: O Layout definido pode ser diferente em outros dispositivos (uma vez que os valores são absolutos).



<AbsoluteLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout_width="fill_parent"

android:layout_height="wrap_content"

android:background="#005793">

<TextView android:layout_x="5px" android:layout_y="10px"

android:layout_width="wrap_content"

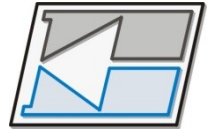
android:layout_height="wrap_content"

android:text="Usuário: "/>

<EditText android:layout_x="60px" android:layout_y="10px"

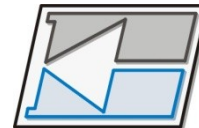
android:layout_width="175px" android:layout_height="wrap_content"

android:background="@android:drawable/editbox_background"/>

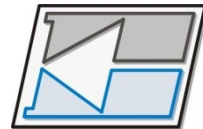


AbsoluteLayout: Exemplo (cont.)

```
<TextView android:layout_x="5px" android:layout_y="55px"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Senha: "/>
<EditText android:layout_x="60px" android:layout_y="55px"
android:layout_width="175px" android:layout_height="wrap_content"
android:background="@android:drawable/editbox_background"
android:password="true"/>
<Button android:layout_x="180px" android:layout_y="100px"
android:layout_width="wrap_content" android:layout_height="60px"
android:text="Login"/>
</AbsoluteLayout>
```

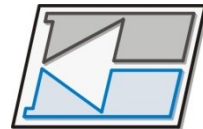


Criação de *layouts* através de API



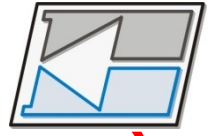
- Introdução

- ✓ Inicialmente, arquivos de *layout* podem ser definidos através do padrão XML.
- ✓ Entretanto, é possível se criar layouts também através da própria API Android.
- ✓ Também por esse método se faz necessário a definição de alguns atributos relacionados aos layouts.
- ✓ Os métodos `setContentView()` e `addView()` continuam a ser utilizados.



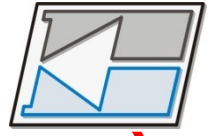
Criação de layouts via API: Exemplo

```
public class MainActivity extends Activity {  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        //Cria layout  
        LinearLayout layout = new LinearLayout(this);  
        layout.setOrientation(LinearLayout.VERTICAL);  
        layout.setLayoutParams(new  
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.FILL_PARENT));  
        layout.setPadding(10, 10, 10, 10);  
        //Cria TextView e EditText  
        TextView nome = new TextView(this);  
        nome.setText("Nome:");  
        nome.setLayoutParams(new  
LayoutParams(LayoutParams.WRAP_CONTENT,LayoutParams.WRAP_CONTENT));  
        layout.addView(nome);  
    }  
}
```



Criação de layouts via API: Exemplo (cont.)

```
EditText tnome = new EditText(this);  
tnome.setLayoutParams(new  
LayoutParams(LayoutParams.FILL_PARENT, LayoutParams.WRAP_CONTENT));  
layout.addView(tnome);  
//Focus  
tnome.requestFocus();  
//Campo senha  
TextView senha = new TextView(this);  
senha.setText("Senha:");  
senha.setLayoutParams(new  
LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));  
layout.addView(senha);
```



Criação de layouts via API: Exemplo (cont.)

```
EditText tsenha = new EditText(this);
```

```
tsenha.setLayoutParams(new  
LayoutParams(LayoutParams.FILL_PARENT,LayoutParams.WRAP_CONTENT));  
layout.addView(tsenha);
```

```
//Botao alinhado a direita
```

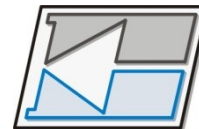
```
Button ok = new Button(this);
```

```
ok.setLayoutParams(new  
LayoutParams(LayoutParams.WRAP_CONTENT,LayoutParams.WRAP_CONTENT));  
ok.setGravity(Gravity.RIGHT);
```

```
ok.setText("OK");
```

```
layout.addView(ok);
```

```
setContentView(layout);
```



Exercícios