Criação de uma interface gráfica - Parte 02

Desenvolvimento Mobile e IoT - Android

Mobile Development and IoT - Android

Prof. Douglas Cabral <douglas.cabral@fiap.com.br>
https://www.linkedin.com/in/douglascabral/

Agenda

- Checkbox
- RadioButton / RadioGroup
- Styles
- Layouts
    - LinearLayout
    - RelativeLayout
    - AbsoluteLayout
    - FrameLayout

Prof. Douglas Cabral <douglas.cabral@fiap.com.br> https://www.linkedin.com/in/douglascabral/

CheckBox

Prof. Douglas Cabral douglas.cabral@fiap.com.br https://www.linkedin.com/in/douglascabral/

Criação de uma interface gráfica - CheckBox

Componentes do tipo **Checkbox** são renderizados em forma de campos quadrados onde permite-se **escolher entre duas opções: marcados ou não marcados**. Caso haja vários componentes do tipo **Checkbox** na tela, poderá ocorrer a múltipla seleção destes componentes, sem anular a escolha anterior (**diferente do RadioButton**).


```
<CheckBox
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Marque-me!"
    android:id="@+id/chkMarqueMe"
    android:checked="true"/>
```

Obs: Para deixar o CheckBox marcado por padrão, basta definir o atributo **android:checked=”true”** na declaração do componente.

Criação de uma interface gráfica - CheckBox

Para verificar no Java **se o CheckBox está selecionado**, basta usar o método **isChecked()** do objeto do tipo **CheckBox**, conforme o exemplo abaixo:

```
CheckBox chkMarqueMe = findViewById(R.id.chkMarqueMe);
if ( chkMarqueMe.isChecked( ) ) {
    Toast.makeText(this, "Marque-me selecionado...", Toast.LENGTH_SHORT).show( );
}
```

RadioButton & RadioGroup

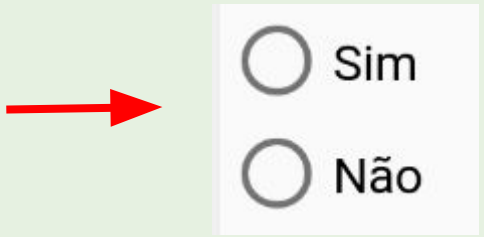
Prof. Douglas Cabral douglas.cabral@fiap.com.br https://www.linkedin.com/in/douglascabral/

Criação de uma interface gráfica - RadioButton/RadioGroup

Componentes do tipo **RadioButton**, são componentes renderizados em formato de círculos e que permitem **escolher apenas um RadioButton por vez** que esteja dentro do mesmo **RadioGroup**. **O RadioGroup** por sua vez, é o componente que agrupa diversos **RadioButtons**.

```
<RadioGroup
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:id="@+id/rdgOpcoes">

    <RadioButton
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Sim"
        android:id="@+id/rdbSim"/>
```



Obs: Para deixar o RadioButton marcado por padrão, basta definir o atributo **android:checked=”true”** na declaração do componente.

Criação de uma interface gráfica - RadioButton/RadioGroup

Para verificar no Java **se o RadioButton está selecionado**, basta usar o método **isChecked()** do objeto do tipo **RadioButton**, conforme o exemplo abaixo:

```
RadioButton rdbSim = findViewById(R.id.rdbSim);
if ( rdbSim.isChecked( ) ) {
    Toast.makeText(this, "Sim selecionado...", Toast.LENGTH_SHORT).show( );
}
```

Criação de uma interface gráfica - RadioButton/RadioGroup

Uma outra forma, é comparar através do método **getCheckedRadioButtonId()** do **RadioGroup**, qual o **ID do RadioButton selecionado**, conforme a imagem abaixo.

```
RadioGroup rdgOpcoes = findViewById(R.id.rdgOpcoes);
int id = rdgOpcoes.getCheckedRadioButtonId();
switch (id) {
    case R.id.rdbSim:
        Toast.makeText(this, "Sim selecionado...", Toast.LENGTH_SHORT).show();
        break;
    case R.id.rdbNao:
        Toast.makeText(this, "Não selecionado...", Toast.LENGTH_SHORT).show();
        break;
}
```

Styles

Prof. Douglas Cabral douglas.cabral@fiap.com.br https://www.linkedin.com/in/douglascabral/

Criação de uma interface gráfica - Styles

Muitas vezes temos muitas propriedades idênticas e compartilhadas entre muitos componentes, fazendo com que nosso código fique muito grande e dificulte a manutenção. Podemos simplificar isso separando a formatação em **Styles**, dentro do arquivo **res/values/styles.xml**.

```
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>
</resources>
```

***Por padrão, o nosso APP já utiliza o Style chamado AppTheme.**

Criação de uma interface gráfica - Styles

Na imagem ao lado, temos as propriedades **android:layout_width=”match_parent”** e **android:layout_height=”wrap_content”** de forma idêntica em todos os **RadiosButtons**.

Vamos simplificá-las criando um novo **Style** no arquivo **res/values/styles.xml** chamado **FIAPRadioButton**, conforme o próximo slide.

```
<RadioButton
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Sim"
    android:id="@+id/rdbSim"/>

<RadioButton
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Não"
    android:id="@+id/rdbNao"/>
```

Criação de uma interface gráfica - Styles

```xml
<resources>
    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <style name="FIAPRadioButton" parent="AppTheme">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
    </style>
</resources>
```

Criação de uma interface gráfica - Styles

No componente, basta retirarmos as propriedades que já estão dentro do **Style** criado, e inserir o atributo **style=”@style/FIAPRadioButton”**.

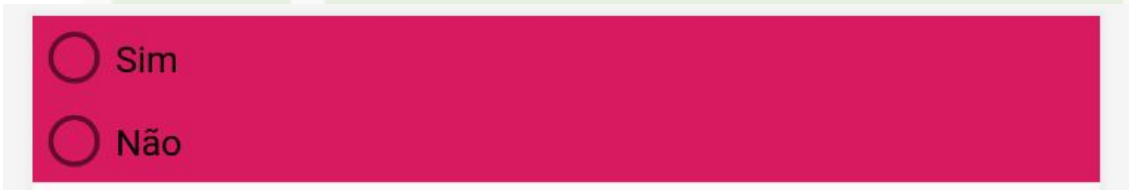
```
<RadioButton
    style="@style/FIAPRadioButton"
    android:text="Sim"
    android:id="@+id/rdbSim"/>

<RadioButton
    style="@style/FIAPRadioButton"
    android:text="Não"
    android:id="@+id/rdbNao"/>
```

Criação de uma interface gráfica - Styles

A vantagem de centralizar as propriedades comuns é que, se um dia for necessário por exemplo, alterar o background de todos estes RadioButtons de nosso APP, basta alterar no estilo compartilhado entre todos:

```
<style name="FIAPRadioButton" parent="AppTheme">
    <item name="android:background">@color/colorAccent</item>
    <item name="android:layout_width">match_parent</item>
    <item name="android:layout_height">wrap_content</item>
</style>
```



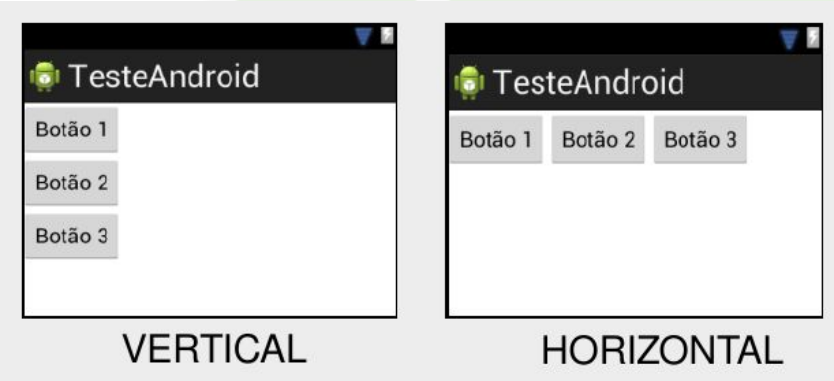
Layouts

Prof. Douglas Cabral douglas.cabral@fiap.com.br https://www.linkedin.com/in/douglascabral/

Criação de uma interface gráfica - LinearLayout

O LinearLayout é conhecido como um dos ViewGroups mais fáceis de utilizar, pois sua função resume-se em alinhar automaticamente os componentes de forma linear seja na vertical ou na horizontal.

A propriedade **android:orientation** define a configuração da forma em que serão organizados os elementos dentro do LinearLayout. Suas opções são: **horizontal** ou **vertical.**



Criação de uma interface gráfica - LinearLayout - Atributos

Para alinhamento dos componentes dentro do LinearLayout, podemos usar o atributo **android:layout_gravity** com os possíveis valores: **center, center_horizontal, center_vertical, start, bottom, end, left, right,** entre outros.

Para colocar os componentes ocupando a mesma proporção do tamanho disponível, basta usar o atributo **android:weight** com pesos iguais para os elementos. Ex: **android:weight=”1”**.

Prof. Douglas Cabral <douglas.cabral@fiap.com.br> https://www.linkedin.com/in/douglascabral/

Criação de uma interface gráfica - RelativeLayout

O **RelativeLayout** é um Layout que permitirá **posicionar os elementos de forma relativa a outros elementos**, ou seja, na esquerda, direita, acima, abaixo de outros componentes existentes.

OBS: o elemento que será referência para o componente a ser posicionado deverá conter sempre um **ID**.

Prof. Douglas Cabral <douglas.cabral@fiap.com.br> https://www.linkedin.com/in/douglascabral/

Criação de uma interface gráfica - RelativeLayout - Atributos

As posições possíveis são:

android:layout_below=”@id/idDaReferencia” - Abaixo do componente referenciado;

android:layout_above=”@id/idDaReferencia” - Acima do componente referenciado;

android:layout_alignParentTop=”true” - Alinhado com o topo do elemento pai;

android:layout_alignParentBottom=”true” - Alinhado com a parte de baixo do elemento pai;

android:layout_alignParentLeft=”true” - Alinhado com a esquerda do elemento pai;

android_layout_alignParentRight=”true” - Alinhado com a direita do elemento pai;

android_toRightOf=”@id/idDaReferencia” - À direita do componente referenciado;

android_toLeftOf=”@id/idDaReferencia” - À esquerda do componente referenciado;

android_alignTop=”@id/idDaReferencia” - Alinha no topo do componente referenciado;

android_alignBottom=”@id/idDaReferencia” - Alinha na parte de baixo do componente referenciado;

android_alignLeft=”@id/idDaReferencia” - Alinha com a esquerda do componente referenciado;

android_alignRight=”@id/idDaReferencia” - Alinha com a direita do componente referenciado;

Criação de uma interface gráfica - AbsoluteLayout

O **AbsoluteLayout** é um **layout depreciado**, ou seja, **não é recomendado sua utilização** e em breve poderá ser removido completamente da plataforma Android, pois com o seu uso, a interface pode ficar errada ou quebrada em diferentes dispositivos com diferentes resoluções de tela.

As posições dos componentes dentro de um AbsoluteLayout são informadas através dos atributos: **android:layout_x** e **android:layout_y** e **seus valores inseridos em pixels**.

Criação de uma interface gráfica - FrameLayout

A **FrameLayout é a layout mais simples** em relação às outras layouts aprendidas até o momento. Sua função é **empilhar uma View sobre a outra**, na ordem em que é inserida na layout, como se fosse uma pilha.

Casos de uso:

- Quando é necessário inserir um vídeo/imagem de fundo de um formulário.

Prof. Douglas Cabral <douglas.cabral@fiap.com.br> https://www.linkedin.com/in/douglascabral/

Criação de uma interface gráfica - Outros ViewGroups

Além do LinearLayout, RelativeLayout, AbsoluteLayout e FrameLayout aprendidos, há outros ViewGroups disponíveis na plataforma Android para uso em casos específicos, como por exemplo:

- TableLayout
- GridLayout
- DrawerLayout
- CoordinatorLayout
- ConstraintLayout
- **entre outros...**

Prof. Douglas Cabral <douglas.cabral@fiap.com.br> https://www.linkedin.com/in/douglascabral/

Dúvidas?

Prof. Douglas Cabral douglas.cabral@fiap.com.br https://www.linkedin.com/in/douglascabral/