Desenvolvimento Para Dispositivos móveis

Prof. Me. Clênio Silva
E-mail: clenio.silva@uniube.br

ScrollView

- O **ScrollView** não permite a inserção de mais de um elemento dentro dele;
- Assim não podemos definir vários **TextView** dentro do **ScrollView**;
- O que podemos fazer é adicionar o **ScrollView** como elemento pai e adicionar um **LinearLayout** dentro dele.
- Assim poderemos incluir vários elementos dentro do **LinearLayout** e poder ver o funcionamento do *scroll.*

ScrollView

```xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

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

        <TextView
            android:layout_width="128dp"
            android:layout_height="128dp"
            android:text="test scroll" />

        <TextView
            android:layout_width="128dp"
            android:layout_height="128dp"
            android:text="test scroll" />


        <TextView
            android:layout_width="128dp"
            android:layout_height="128dp"
            android:text="test scroll" />

        <TextView
            android:layout_width="128dp"
            android:layout_height="128dp"
            android:text="test scroll" />

        <TextView
            android:layout_width="128dp"
            android:layout_height="128dp"
            android:text="test scroll" />
```

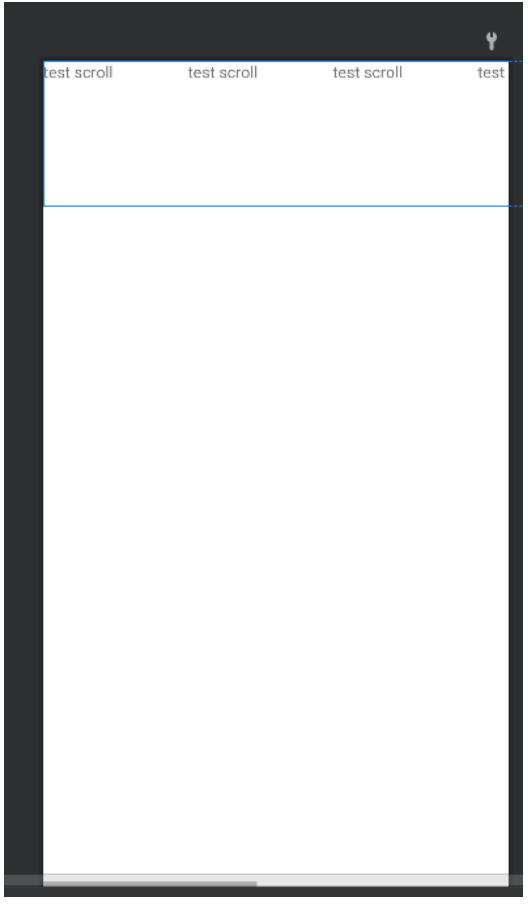
ScrollView


6:43
LinearLayout
test scroll
test scroll
test scroll
test scroll
test scroll

HorizontalScrollView

- Podemos perceber que quando declaramos o elemento pai como **ScrollView** e declaramos a propriedade do **LinearLayout** android:orientation=”vertical”, aparece uma barra de rolagem na vertical para navegar na tela caso tenha mais elementos.
- Agora, vamos mudar o elemento pai para **HorizontalScrollView** e a propriedade android:orientation de vertical para horizontal

HorizontalScrollView



Elemento View

- É usado para fazer uma separação na tela;
- Ele é capaz de dar um estilo no layout;
- Ele não recebe um texto, ou seja ele é um elemento vazio;

Lembrando que a letra v do elemento
View é maiúscula. Caso ocorra de escrever view
com v minusculo, vai dar erro na execução.
Assim a definição dos elementos é *case sensitive*

Elemento View

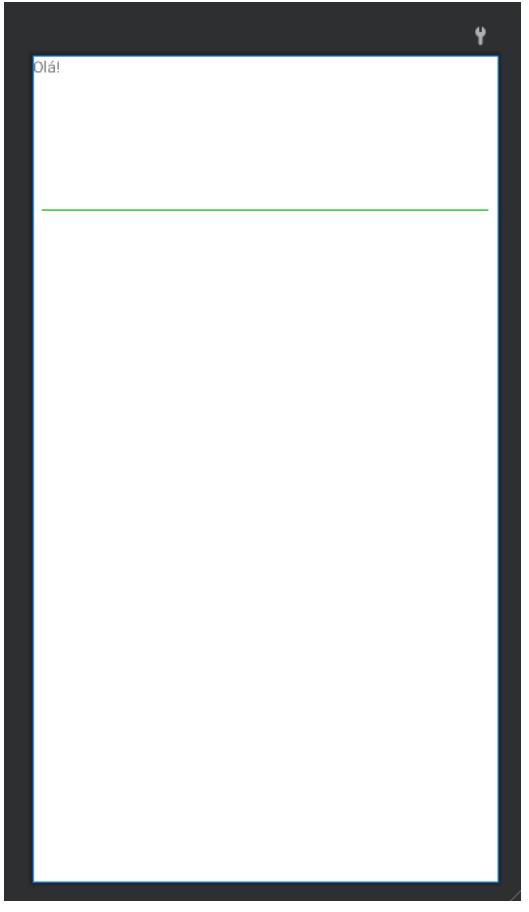
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="128dp"
        android:layout_height="128dp"
        android:text="Olá!" />

    <View
        android:layout_width="match_parent"
        android:layout_height="2dp"
        android:layout_margin="8dp"
        android:background="#0DB514"/>

</LinearLayout>
```

Elemento View



RelativeLayout

- Esse tipo de layout não tem orientação, ou seja, não se deve setar a propriedade **android:orientation=””**;
- *Relative* traduzido para português é relativo. Esse layout apresenta a ideia que os elementos são relativos a eles mesmo e aos elementos pai, ou seja é assim que os elementos sabem como vão se posicionar na tela.

No LinearLayout ou você adiciona os elementos um do lado do outro ou embaixo Um do outro. Não dá para distribuir os elementos de forma relativa ao elemento pai. Já no RelativeLayout é possível.

RelativeLayout

- Como o RelativeLayout permite adicionar elementos e organizar a sua posição em relação ao elemento pai, podemos setar novas propriedades dentro dos elementos, como:
  - Define a posição do elemento na parte inferior da tela:
    - `android:layout_alignParentBottom="true`
  - Define a posição do elemento no centro da tela:
    - `android:layout_centerHorizontal="true"`

RelativeLayout

- Vamos inserir um novo elemento em uma **viewGroup** do tipo **RelativeLayout**;
- O elemento que vamos inserir é o **Button**;
- No elemento **Button**, vamos definir uma propriedade android:id, que é usada para referenciarmos o botão, e vamos setar também a propriedade android:text, que vai conter o texto apresentado no botão.

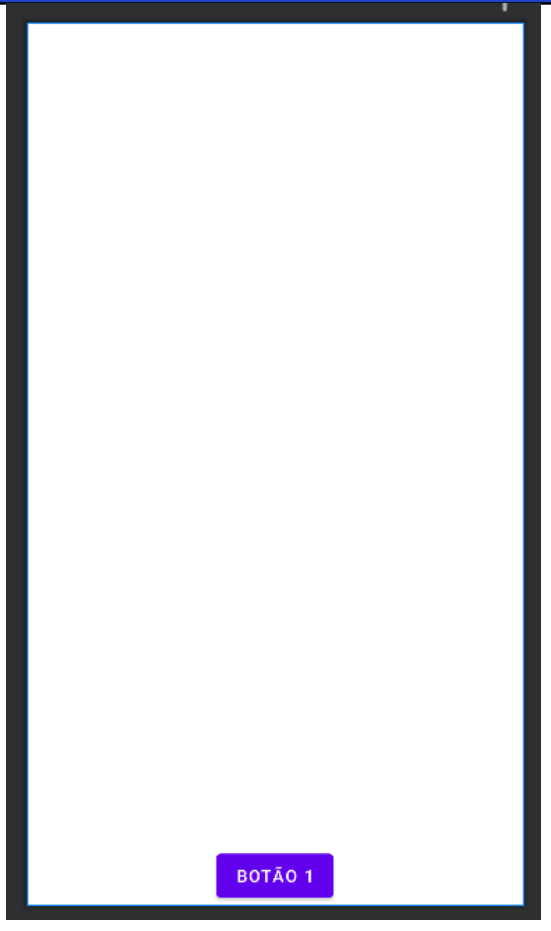
RelativeLayout

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

    <Button
        android:id="@+id/bnt1"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Botão 1" />

</RelativeLayout>
```

RelativeLayout



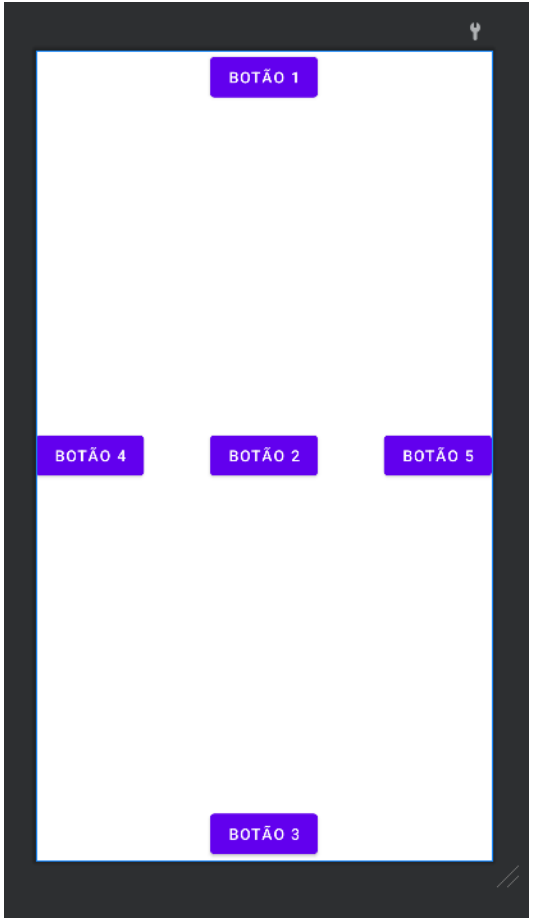
RelativeLayout

- Vamos usar mas propriedades para alinhar vários botões na tela:
  - Centralizar elemento na horizontal:
    - **android:layout_centerHorizontal="true"**
  - Centralizar elemento na vertical:
    - **android:layout_centerVertical="true"**
  - Definir posição do elemento na parte inferior em relação ao elemento pai:
    - **android:layout_alignParentBottom="true"**
  - Definir o elemento na esquerda em relação ao elemento pai:
    - **android:layout_alignParentStart="true"**
  - Definir o elemento a direita em relação ao elemento pai:
    - **android:layout_alignParentEnd="true"**

RelativeLayout

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <Button
        android:id="@+id/bnt1"
        android:layout_centerHorizontal="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Botão 1" />
    <Button
        android:id="@+id/bnt2"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Botão 2" />
    <Button
        android:id="@+id/bnt3"
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Botão 3" />
    <Button
        android:id="@+id/bnt4"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_alignParentStart="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Botão 4" />
    <Button
        android:id="@+id/bnt5"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true"
        android:layout_alignParentEnd="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Botão 5" />
</RelativeLayout>
```

RelativeLayout



RelativeLayout

- O **RelativeLayout** possibilita uma flexibilidade maior que o **LinearLayout** na questão de alinhamento dos elementos;
- A diferença principal do **RelativeLayout** para o **LinearLayout** é que no **LinearLayout** o alinhamento é vertical ou horizontal, já no **RelativeLayout** temos mais flexibilidade para colocar os elementos no centro, a direita, esquerda e embaixo.

Referências

LECHETA, Ricardo. R. Google Android: aprenda a criar aplicações para dispositivos móveis com Android SDK. 5ª ed. São Paulo, SP: Novatec, 2016.