Gerenciadores de Layout no Android

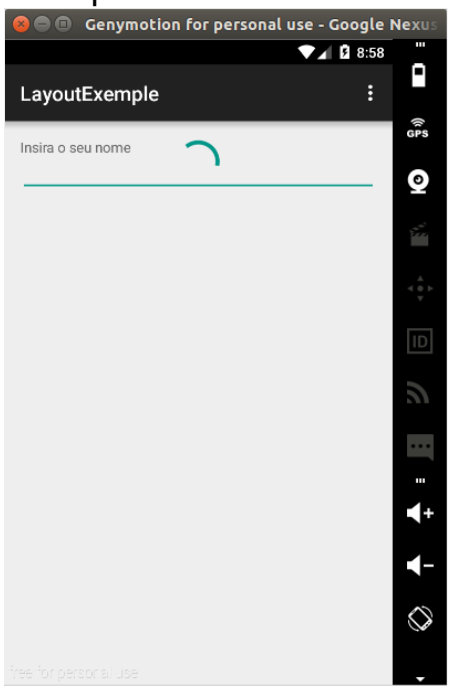
Os gerenciadores de layout no Android são utilizados para organizar a disposição dos componentes na tela, de acordo com o gerenciador de layout escolhido utilizaremos atributos diferentes para posicionar os componentes. A classe-mãe de todos os gerenciadores de layout é a ‘android.view.ViewGroup’, nesse artigo iremos verificar as suas subclasses e o comportamento de cada uma.

FrameLayout

É o mais simples de todos os gerenciadores de layout e é utilizado para empilhar uma view sobre a outra. É possível adicionar vários componentes dentro do FrameLayout. Os últimos componentes ficarão sobre os anteriores, seguindo o conceito de pilha, em que o ultimo elemento fica no topo. Um exemplo de utilização é quando queremos inserir uma ProgressBar acima de outro componente para indicar que estamos carregando alguma coisa. O exemplo a seguir demonstra um FrameLayout utilizando uma ProgressBar sobre um elemento de texto.

```
<FrameLayout 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:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".MainActivity">

    <TextView
        android:text="Insira o seu nome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:layout_marginTop="10dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="top|center"/>

</FrameLayout>
```

E a apresentação dos componentes fica na forma a seguir. O ProgressBar é apresentando de uma forma diferente no Android Studio, por isso decidi exibir o exemplo sendo executado em um smatphone.



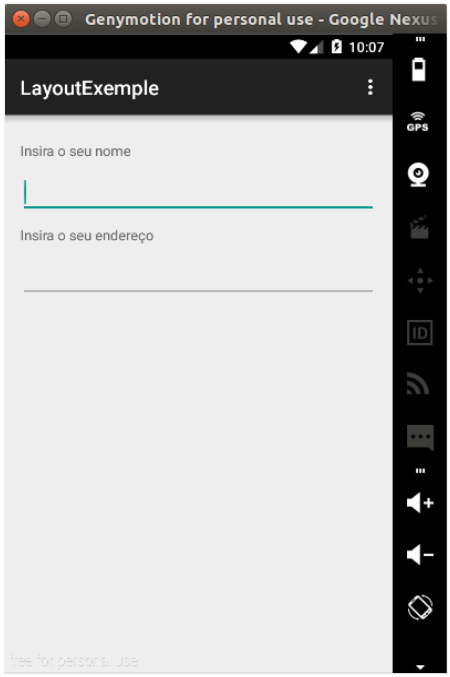
Gerenciadores de Layout no Android – ProgressBar

Linear Layout

O LinearLayout é uma *ViewGroup* que alinha todos os seus componentes em um único sentido configurado através do atributo *android:orientation.* Quando utilizamos a orientação vertical teremos apenas um elemento por linha independente do tamanho do elemento e ao utilizarmos a orientação horizontal teremos apenas uma linha para todos os elementos.

```
<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:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity"
    android:orientation="vertical">


    <TextView
        android:layout_marginTop="10dp"
        android:text="Insira o seu nome"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:layout_marginTop="10dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />


    <TextView
        android:layout_marginTop="10dp"
        android:text="Insira o seu endereço"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
    <EditText
        android:layout_marginTop="10dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Como podemos observar na figura ao utilizar a orientação vertical os elementos são posicionamento um abaixo do outro.

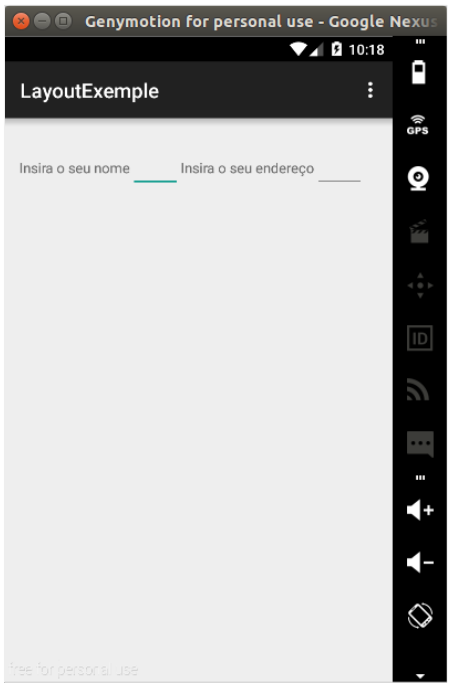


Gerenciadores de Layout no Android – LinearLayout – Vertical

Ao modificarmos a orientação para ‘horizontal’ todos os elementos são posicionados na mesma linha.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
tools:context=".MainActivity"
    android:layout_width="match_parent"
android:layout_height="match_parent"
    android:padding="15dp" android:orientation="horizontal">

    <TextView
        android:layout_marginTop="10dp"
        android:text="Insira o seu nome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <EditText
        android:layout_marginTop="10dp"
        android:layout_width="50dp"
        android:layout_height="wrap_content" />

    <TextView
        android:layout_marginTop="10dp"
        android:text="Insira o seu endereço"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <EditText
        android:layout_marginTop="10dp"
        android:layout_width="50dp"
        android:layout_height="wrap_content" />

</LinearLayout>
```

Como podemos observar na figura os elementos são posicionamento ao lado do outro.

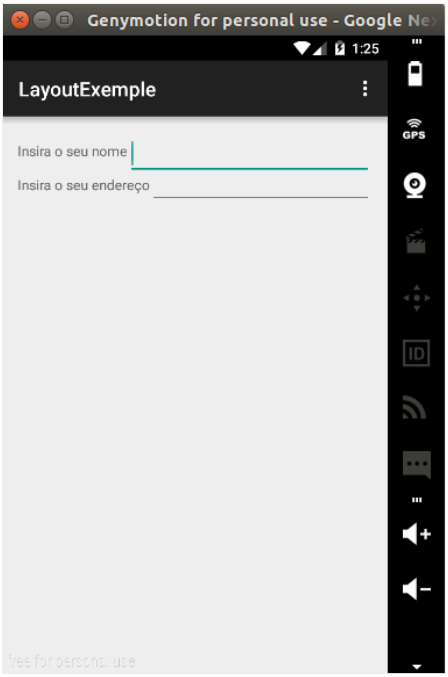


Relative Layout

RelativeLayout é uma *ViewGroup* que posiciona seus os componentes utilizando posições relativas. Essa posição pode ser especificada de acordo com a posição de outro componente (utilizando os atributos below, left, etc) ou de acordo com a posição do próprio RelativeLayout (utilizando os atributos bottom, left ou center). Para realizar o posicionamento relativo devemos identificar os componentes através do seu id. No código a seguir podemos verificar que os atributos *layout_below*, *layout_toRightOf* e *layout_alignParentTop* são utilizados para fazer esse posicionamento relativo.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
tools:context=".MainActivity"
    android:layout_width="match_parent"
android:layout_height="match_parent"
    android:padding="15dp" android:orientation="horizontal">

    <TextView
        android:id="@+id/nomeTextView"
        android:layout_marginTop="10dp"
        android:layout_alignParentTop="true"
        android:text="Insira o seu nome"


        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />


    <EditText
        android:id="@+id/nomeEditText"
        android:layout_toRightOf="@+id/nomeTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />


    <TextView
        android:id="@+id/enderecoTextView"
        android:layout_below="@id/nomeTextView"
        android:text="Insira o seu endereço"
        android:paddingTop="15dp"


        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <EditText
        android:id="@+id/enderecoEditText"
        android:layout_below="@id/nomeTextView"
        android:layout_toRightOf="@id/enderecoTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

</RelativeLayout>
```

Observe que o componente *enderecoEditView* ficou abaixo do *nomeTextView* e a direta do *enderecoTextView.*



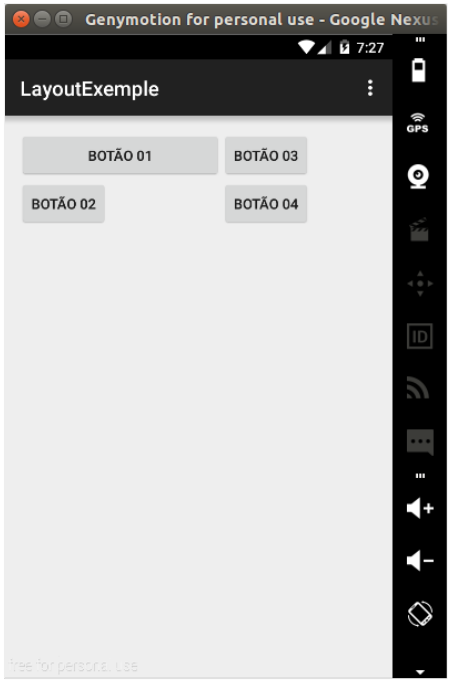
Gerenciadores de Layout no Android – RelativeLayout

Grid View

A classe android.widget.GridLayout organiza as views em linhas e colunas utilizando os atributos layout_row e layout_column. O GridLayout só esta disponível a partir do Android 4.0, mas podemos utilizar a biblioteca de compatibilidade android.support.v7.widget.GridLayout para adicionar o suporte a versões superiores ao Android 2.1.

```
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
tools:context=".MainActivity"
    android:layout_width="match_parent"
android:layout_height="match_parent"
    android:columnCount="2" android:rowCount="2"
    android:padding="15dp">

    <Button
        android:layout_width="200dp" android:layout_height="wrap_content"
        android:layout_column="0" android:layout_row="0"
        android:text="Botão 01"/>


    <Button
        android:layout_width="wrap_content"
android:layout_height="wrap_content"
        android:layout_column="0" android:layout_row="1"
        android:text="Botão 02"/>


    <Button
        android:layout_width="wrap_content"
android:layout_height="wrap_content"
        android:layout_column="1" android:layout_row="0"
        android:text="Botão 03"/>

    <Button
        android:layout_width="wrap_content"
android:layout_height="wrap_content"
        android:layout_column="1" android:layout_row="1"
        android:text="Botão 04"/>

</GridLayout>
```

No exemplo a seguir o primeiro botão recebeu uma largura de 200dp apenas para mostrar que o funcionamento é igual ao de uma tabela normal onde o maior elemento define a largura/altura da tabela.



Gerenciadores de Layout no Android – GridLayout

Utilizando um *ListAdapter* podemos preencher automaticamente essa *GridView*. Além disso podemos utilizar diferentes layouts aninhados para que possamos utilizar cada recurso de acordo com a necessidade. Informações mais completas sobre os gerenciadores de layout no Android estão disponíveis no guia da Google que além de estarem atualizados são bem didáticos.