

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

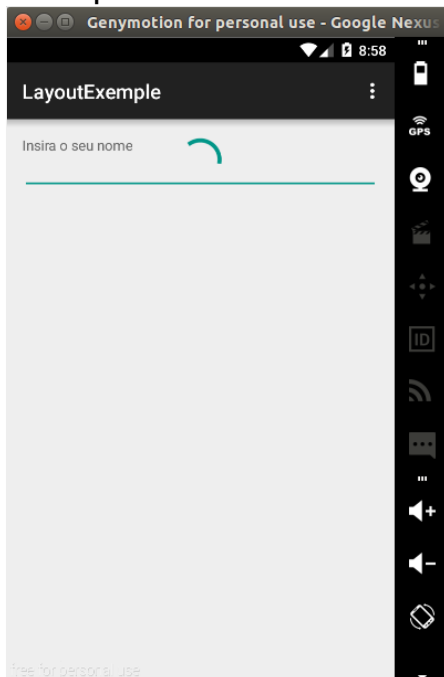
É o mais simples de todos os gerenciadores de layout e é utilizado para empilhar uma view sobre a outra. É possível adicionar vários componentes dentro do `FrameLayout`. Os últimos componentes ficarão sobre os anteriores, seguindo o conceito de pilha, em que o ultimo elemento fica no topo. Um exemplo 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.

```

1  <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:tools="http://schemas.android.com/tools"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:paddingLeft="@dimen/activity_horizontal_margin"
6      android:paddingRight="@dimen/activity_horizontal_margin"
7      android:paddingTop="@dimen/activity_vertical_margin"
8      android:paddingBottom="@dimen/activity_vertical_margin"
9      tools:context=".MainActivity">
10
11      <TextView
12          android:text="Insira o seu nome"
13          android:layout_width="match_parent"
14          android:layout_height="wrap_content" />
15      <EditText
16          android:layout_marginTop="10dp"
17          android:layout_width="match_parent"
18          android:layout_height="wrap_content" />
19
20      <ProgressBar
21          android:layout_width="wrap_content"
22          android:layout_height="wrap_content"
23          android:layout_gravity="top|center"/>
24
25  </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.



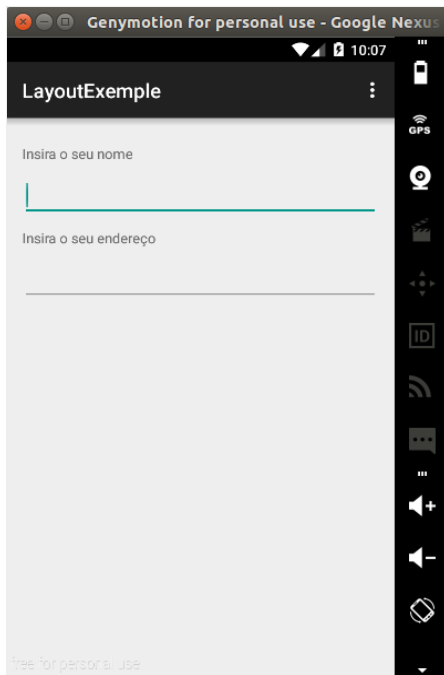
Gerenciadores de Layout no Android – ProgressBar

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.

```
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:paddingLeft="@dimen/activity_horizontal_margin"
6   android:paddingRight="@dimen/activity_horizontal_margin"
7   android:paddingTop="@dimen/activity_vertical_margin"
8   android:paddingBottom="@dimen/activity_vertical_margin"
9   tools:context=".MainActivity"
10  android:orientation="vertical">
11
12  <TextView
13    android:layout_marginTop="10dp"
14    android:text="Insira o seu nome"
15    android:layout_width="match_parent"
16    android:layout_height="wrap_content" />
17  <EditText
18    android:layout_marginTop="10dp"
19    android:layout_width="match_parent"
20    android:layout_height="wrap_content" />
21
22  <TextView
23    android:layout_marginTop="10dp"
24    android:text="Insira o seu endereço"
25    android:layout_width="match_parent"
26    android:layout_height="wrap_content" />
27  <EditText
28    android:layout_marginTop="10dp"
29    android:layout_width="match_parent"
30    android:layout_height="wrap_content" />
31 </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">

```

```

1  <TextView
2      android:layout_marginTop="10dp"
3      android:text="Insira o seu nome"
4      android:layout_width="wrap_content"
5      android:layout_height="wrap_content" />
6
7
8
9
10

```

```

11 <EditText
12     android:layout_marginTop="10dp"
13     android:layout_width="50dp"
14     android:layout_height="wrap_content" />
15
16
17
18
19
20

```

```

21 <TextView
22     android:layout_marginTop="10dp"
23     android:text="Insira o seu endereço"
24     android:layout_width="wrap_content"
25     android:layout_height="wrap_content" />
26
27
28

```

```

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

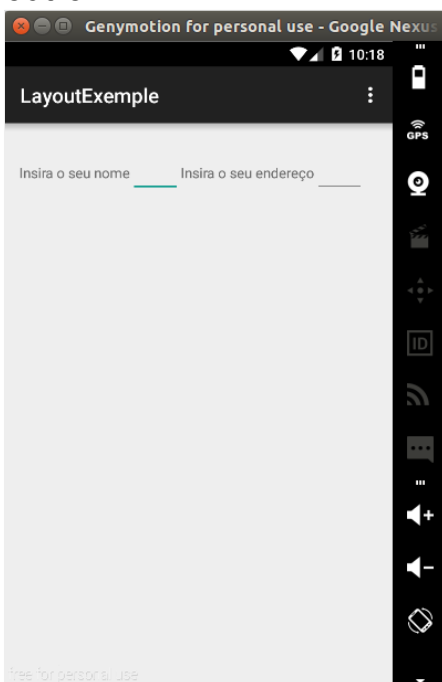
```

```

</LinearLayout>

```

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



Relative Layout

RelativeLayout é uma *ViewGroup* que posiciona seus 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" />
```

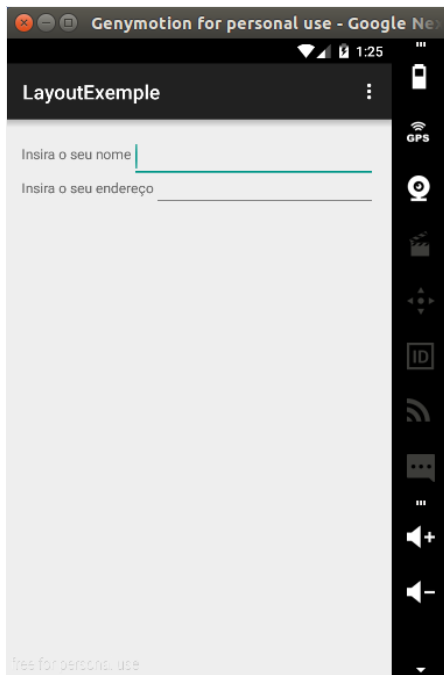
```
    <TextView
        android:id="@+id/enderecoTextView"
        android:layout_below="@id/nomeTextView"
        android:text="Insira o seu endereço"
        android:paddingTop="15dp"

        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 está 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.


```

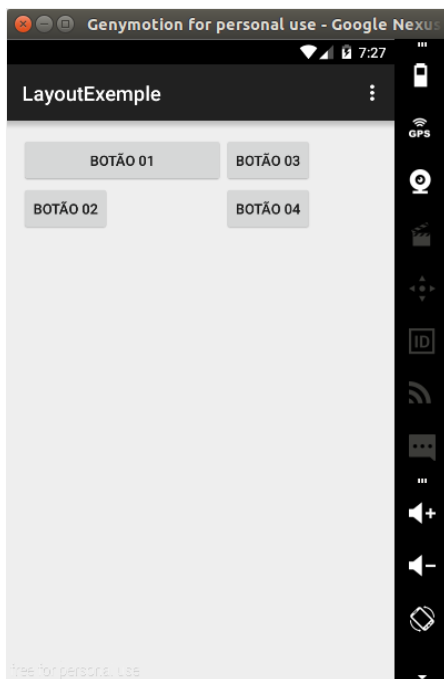
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnCount="2" android:rowCount="2"
    android:padding="15dp">

    <Button
1      android:layout_width="200dp" android:layout_height="wrap_content"
2      android:layout_column="0" android:layout_row="0"
3      android:text="Botão 01"/>
4
5
6
7
8
9
10     <Button
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13     android:layout_column="0" android:layout_row="1"
14         android:text="Botão 02"/>
15
16
17
18
19
20
21     <Button
22         android:layout_width="wrap_content"
23         android:layout_height="wrap_content"
24     android:layout_column="1" android:layout_row="0"
25         android:text="Botão 03"/>
26
27
28
29
30     <Button
31         android:layout_width="wrap_content"
32         android:layout_height="wrap_content"
33     android:layout_column="1" android:layout_row="1"
34         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.