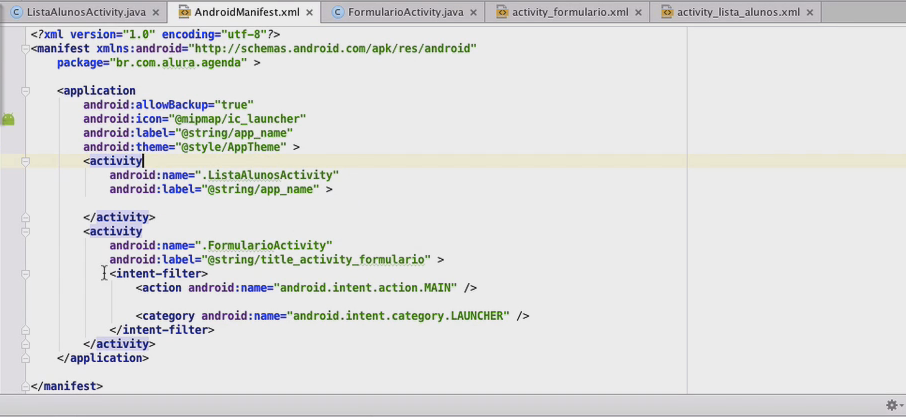
## **Floating button**

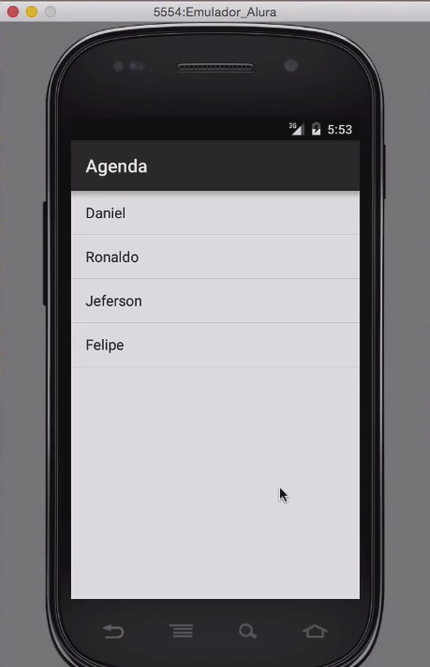
Nossa aplicação já tem campos , botão de salvar e uma mensagem automática. Vamos retornar a lista de alunos? Vamos na aba AndroidManifest.xml onde encontraremos a seguinte tela:



Para que a lista de alunos volte a aparecer na tela do emulador é preciso retornar a intent para a activity da ListaAlunosActivity. Selecionamos com o mouse a intent e deslocamos ela para a activity da lista de alunos. Damos um "Comand+X" ou "Ctrl+X" e um "Comand+V" ou "Ctrl+V" na activity da lista de alunos. Ficará assim:

<application   
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"   
 android:label="@string/app\_name"   
 android:theme="@style/AppTheme" >   
 <activity   
 android:name=".ListaAlunosActivity"   
 android:label="@string/app\_name" >   
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name=“android.intent.category.LAUNCHER" />   
 </intent-filter>  
 </activity>   
 <activity  
 android:name=".FormularioActivity"  
 android:label="@string/title\_activity\_formulario" >   
 </activity>  
 </application>

Pronto! Se rodarmos a aplicação o que vamos encontrar é a lista de alunos mais uma vez.



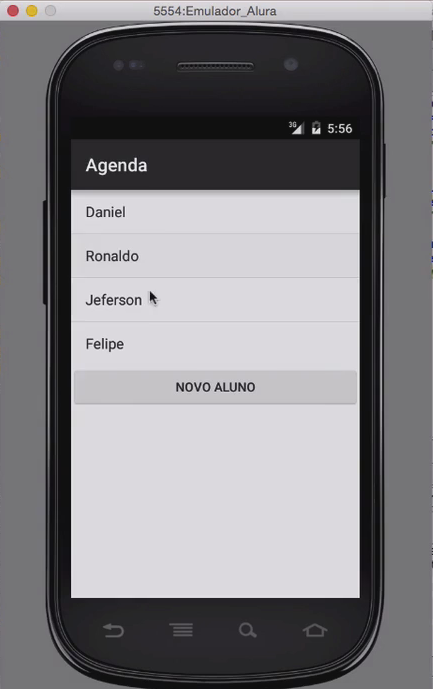
É preciso criar uma forma de sair da lista para chegar no formulário, dessa forma, inserimos um botão "novo aluno". Para modificar a tela da lista alunos vamos na activity\_lista\_alunos.xml . Encontramos o seguinte:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"   
 android:layout\_height="match\_parent"   
 android:orientation="vertical">  
  
 <Listview   
 android:layout\_width="match\_parent"   
 android:layout\_height=“match\_parent"   
 android:id="@+id/lista\_alunos“/>   
  
</LinearLayout>

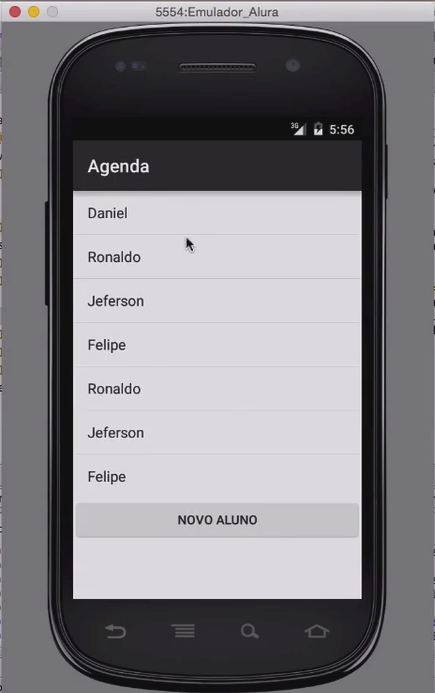
Embaixo do LinearLayout vamos acrescentar uma Viewque represente um botão, a Button. Nesse Button definiremos os parâmetros de altura e largura. A layout\_width como largura, acompanhada de um match\_parent e layout\_heightde altura, acompanhada de um wrap\_content. Por fim, definimos também o texto do botão através da text cujo conteúdo será "Novo aluno". Ficaremos com o seguinte:

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"   
 android:layout\_height="match\_parent"   
 android:orientation="vertical">  
  
 <Listview   
 android:layout\_width="match\_parent"   
 android:layout\_height=“match\_parent"   
 android:id="@+id/lista\_alunos“/>   
  
 <Button   
 android:layout\_width="match\_parent"   
 android:layout\_height="wrap\_content"   
 android:text="Novo aluno"/>  
  
</LinearLayout>

Para que o botão apareça na tela temos que pensar sobre as dimensões do ListViewque tem altura e largura definida no parâmetro match\_parent e por isso ocupa a tela inteira e empurra nosso botão para fora. Isso ocorre também porque utilizamos o LinearLayoutque indica orientação vertical. Para resolver essa situação trocamos o match\_parent referente a altura da ListView por um wrap\_content, assim a lista passa a ocupar apenas o necessário. Vamos salvar e rodar a aplicação! Teremos o seguinte:



Se quisermos acrescentar novos alunos vamos na aba 'ListaAlunosActivity.java' no String, onde acrescentamos novos nomes. Rodando o emulador mais uma vez veremos o que temos:



Podemos adicionar ainda mais nomes e inclusive a lista vai rolar, mas continuaremos com o problema do botão não aparecer. O ListView possui internamente um Scroll, por isso a lista rola automaticamente.

Como fazemos para que o botão fique fixo e apareça sempre na tela? Desejamos que ele apareça no fim da lista, mas que não saia nunca da tela.

Perceba que já fizemos isso antes! Nós colocamos um componente em cima do outro de forma involuntária. Lembre-se do inicio, quando nossa activity\_lista\_alunos.xmlcontinha um RelativeLayout. A RelativeLayout organiza os componentes dispondo-os de maneira relativa, isto é, distribuídos da maneira a ficarem uns a cima dos outros.

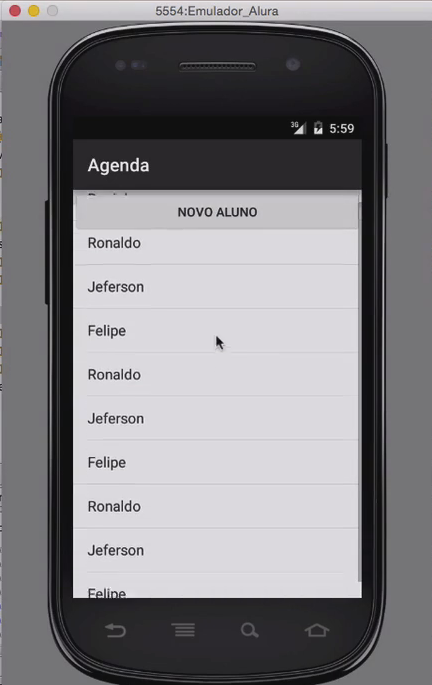
Poderíamos acrescentar o atributo bellow no Button. Adicionaríamos o android:layout\_bellow seguido do "@id/Lista\_alunos" que indica o local onde o botão deve ficar. Essas alterações seriam feitas na activity\_lista\_alunos.xml. E teríamos:

<Button   
 android:layout\_width="match\_parent"   
 android:layout\_height="wrap\_content"   
 android:text="Novo aluno"   
 android:layout\_below="@id/lista\_alunos"/>

Mas, ao fazer isso, teremos o mesmo comportamento do LinearLayout e não é isso que queremos, então, vamos apagar o android:layout\_below="@id/lista\_alunos" e o LinearLayout. No lugar deste último vamos adicionar o comportamento RelativeLayout. Teremos o seguinte:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"   
 android:layout\_width="match\_parent"   
 android:layout height="match\_parent"   
 android:orientation=“vertical">   
  
 <ListView   
 android:layout\_width="match\_parent"   
 android:layout\_height=”match\_parent"   
 android:id="@+id/lista\_alunos"/>   
  
 <Button android:layout\_width="match\_parent"   
 android:layout\_height="wrap\_content"   
 android:text="Novo aluno” />   
  
</RelativeLayout>

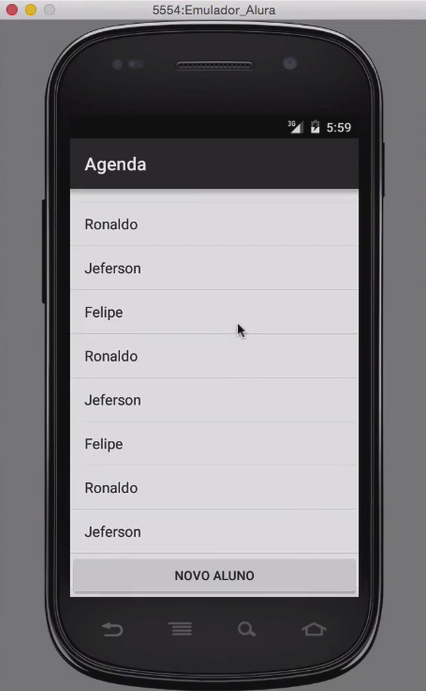
Vamos rodar o emulador para ver como fica. Ficaremos com:



O botão aparece no topo da lista e não embaixo, como queremos. Para deslocar o botão utilizaremos o atributo RelativeLayout que é alignParentBottom, o que significa alinhar na parte de baixo do "pai". Acrescentaremos, ainda, o true, para dizer que é verdadeiro. Ficaremos com:

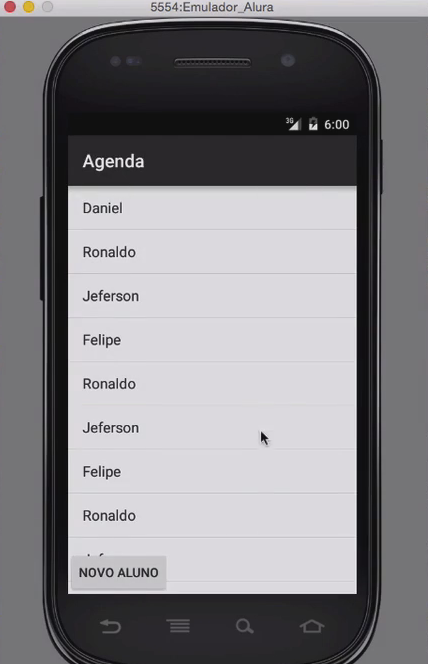
<Button android:layout\_width="match\_parent"   
 android:layout\_height="wrap\_content"   
 android:text="Novo aluno"   
 android:layout\_alignParentBottom="true"/>

Vamos rodar o emulador mais uma vez para verificar como ficou:



Agora ele está alinhado como queremos! Mas, vamos diminuir o tamanho desse botão. Para isso basta alterar no Button a largura do botão, adicionando wrap\_content e teremos android:layout\_width="wrap\_content".

Vamos rodar o emulador para ver como ficou:



Note que o botão ficou no lado esquerdo e ainda está em cima dos nomes dos alunos.

Vamos jogar ele para o lado direito para que não atrapalhe a leitura. Para isso vamos acrescentar outro alignParent, mas dessa vez diremos que ele se alinha com o "pai" à direita, teremos um layout\_alignParentRight e adicionamos o true.

<Button android:layout\_width="wrap\_parent"   
 android:layout\_height="wrap\_content"   
 android:text="Novo aluno"   
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentRight="true"/>

Vamos rodar o emulador para ver como ficou:



A *google* recomenda que o padrão dos botões que aparecem em seus aplicativos destaquem a ação mais importante que ele realiza. Por exemplo, o aplicativo da google do *gmail* indica com um botão redondo e vermelho com um sinal de "mais" é a ação de adicionar. Vamos alterar nosso botão para que ele fique nesse padrão dando um destaque nele.

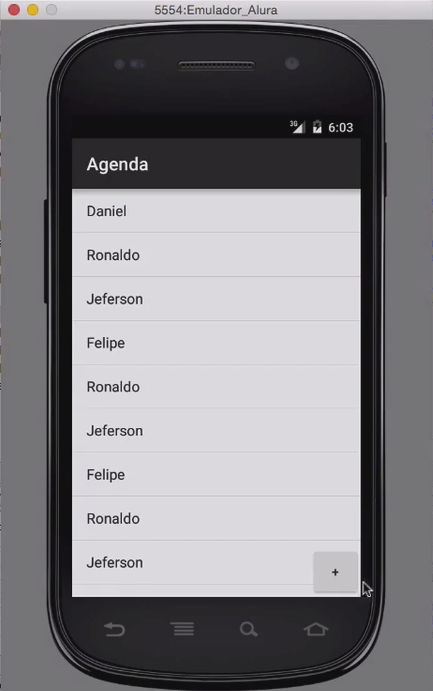
Vamos voltar para a activity\_lista\_alunos.xml e alterar o texto do botão, antes tínhamos "novos alunos", agora vamos apagar isso e digitar um sinalzinho de "+" para ficar naquele padrão bacana da *google*. Ficaremos com android:text="+".

Vamos alterar também o tamanho do nosso botão, vamos redefinir um tamanho padrão para a largura e altura dos componentes. Podemos escrever o tamanho "na mão", isto é, digitando as medidas que desejamos, aqui utilizaremos 56dp.

Ficaremos com <android:layout\_width="56dp" e <android:layout\_height="56dp". O "56dp" indica que o botão terá esse medida em uma tela de celular de quatro polegadas ou em um *tablet* de 10 polegadas. Essa medida converte para a realidade e esse "56dp" indica que essa medida equivale a aproximadamente 2cm em qualquer dispositivo.

<Button android:layout\_width="56dp"   
 android:layout\_height="56dp"   
 android:text="+"   
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentRight="true"/>

Vamos rodar e ver como ficou:



O botão está próximo ao que queremos, mas ainda está quadrado e grudado ao canto da tela.

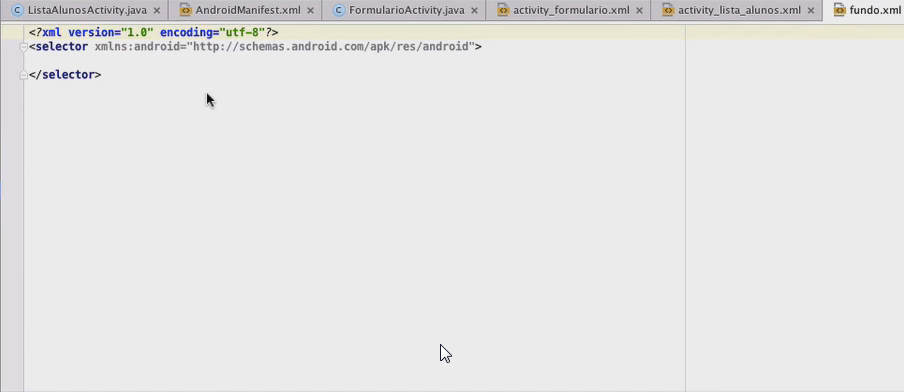
Para alterar isso voltamos na activity\_lista\_alunos.xml vamos acrescentar uma margem, com marginBottom. Vai ser pedido uma medida também e aqui vamos escolher "16 dp" e ficaremos com android:layout\_marginBottom="16dp" e acrescentaremos também que essa margem ficará a direita e terá o mesmo tamanho, escrevendo android:layout\_marginRight="16dp". Teremos:

<Button android:layout\_width="56dp"  
 android:layout\_height="56dp"   
 android:text="+"   
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentright="true"  
 android:layout\_marginBottom="16dp"  
 android:layout\_marginRight="16dp" />

Entretanto, isso ainda não vai resolver o problema da forma do botão. Para isso teremos que criar um novo xml que representará a forma do botão.

Para fazer isso, vamos na pasta "res" e clicamos com o botão direito do mouse em "drawable" e "New > Drawable resource file". O *Drawable* é um arquivo "desenhado", ou seja, algo no qual se pode desenhar. Seguindo esse caminho vai abrir uma janela e pedirá o "File Name", vamos denominar de "Fundo", pois queremos criar o fundo do botão e damos um "Ok".

O seguinte '.xml' vai ser criado:



Nesse '.xml' vamos preencher com a *tag* shape e nele vamos definir a forma que vamos utilizar, para isso utilizaremos o atributo shape. Na linha de baixo, após digitarmos shape ele vai pedir qual a forma que queremos dar, no caso, oval, selecionamos o oval e ficaremos com android:shape="oval". Dentro disso também conseguimos definir a cor de preenchimento da nossa forma, nesse caso queremos uma bola vermelha totalmente preenchida.

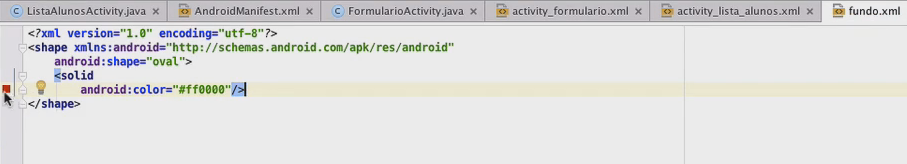
Na linha de baixo adicionamos a *tag* solid para indicar que todo o círculo oval será preenchido e vamos inserir também a cor que desejarmos. Na próxima linha digitamos o atributo color e como queremos que ela seja vermelha, adicionaremos o padrão web, #ff0000.

Teremos o seguinte:

<?xml version="1.0" encoding="utf-8"?>   
<shape xmlns:android="http://schemas.android.com/apk/res/android"   
 android:shape="oval">   
 <solid   
 android:color="#ff0000'/>  
  
</shape>

***Dica***

Se você não conhece o padrão de cores da *web* basta escrever qualquer coisa e fechar do lado esquerdo irá aparecer um quadradinho vermelho e clicando nele vai abrir a paleta de cores, como as imagens na sequência mostram:



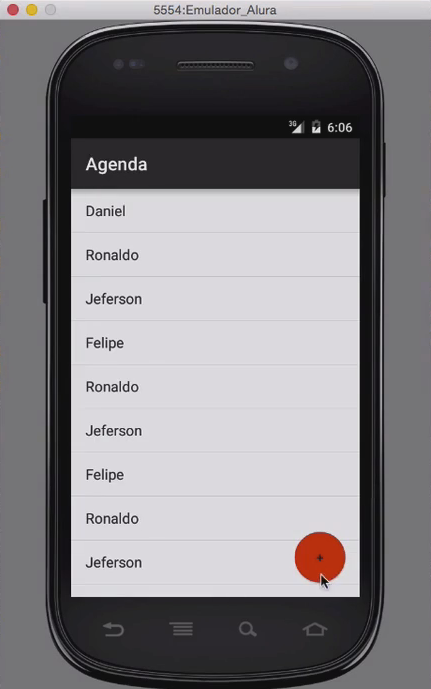


Aqui você escolhe qual a cor que você deseja, clica nela e dando um "Choose" já preencherá o código automaticamente. No caso, usaremos o #ff000" do padrão da web.

Definimos que será oval e vermelho, então, agora voltamos na activity\_lista\_alunos.xml e vamos dizer que o fundo do botão é a imagem que acabamos de definir na fundo.xml. Para tanto usamos background e definimos que é um *Drawable*, pois está dentro da pasta "drawable" e pra isso digitamos com @seguido de drawable e /. Vamos também indicar o nome dele, no caso 'fundo' e escreveremos isso depois da barra. Teremos o seguinte android:background="@drawable/fundo" e o código ficará assim:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"   
 android:layout\_height="match\_parent"  
 android:orientation="vertical">  
  
 <ListView   
 android:layout\_width="match\_parent"   
 android:layout\_height=“match\_parent"   
 android:id="@+id/lista\_alunos“/>   
  
 <Button android:layout\_width="56dp"   
 android:layout\_height="56dp"   
 android:text="+"   
 android:textColor="#ffffff"   
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentright="true"  
 android:layout\_marginBottom="16dp"  
 android:layout\_marginRight="16dp"   
 android:background="@drawable/fundo" />  
  
  
</RelativeLayout>

Vamos rodar o emulador e ver como ficou:

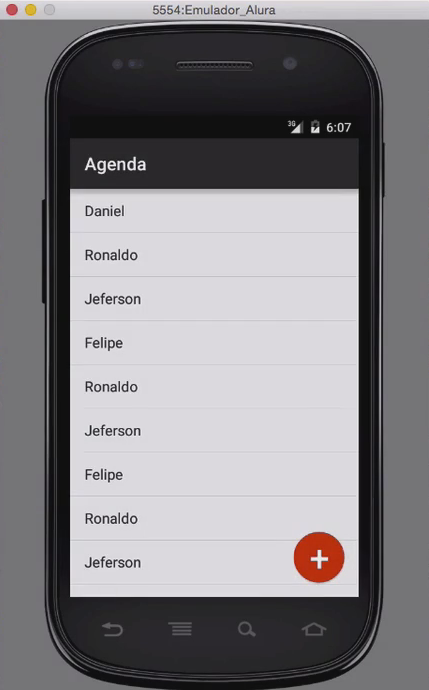


Agora, vamos aumentar o botão e alterar a cor do "+" por branco.

Voltamos lá activity\_lista\_alunos.xml e acrescentamos textColor e acrescentamos o padrão da *web*, usaremos a cor branca #ffffff e na linha de baixo acrescentamos um atributo textSize e iremos acrescentar um outro tipo de medida, um sp. Teremos textSize="40sp". Quando utilizamos texto isso pode indicar que o usuário tenha configurado para deixar o texto maior ou menor e o sp indica que a medida será redimencionada através das opções de acessibilidade. Ficaremos com:

<Button android:layout\_width="56dp"   
 android:layout\_height="56dp"   
 android:text="+"   
 android:textColor="#ffffff"   
 android:textSize="40sp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentright="true"  
 android:layout\_marginBottom="16dp"  
 android:layout\_marginRight="16dp"   
 android:background="@drawable/fundo" />

Vamos rodar e ver o que acontece:



Falta uma sombra que costuma existir nesses componentes, se queremos um botão flutuante vamos indicar qual a elevação do botão através do atributo elevation e acrescentamos que estará a 6dp de distância do fundo da tela. Teremos o seguinte código, android:elevation="6dp" que vamos inserir logo abaixo do textSize. Teremos:

<Button android:layout\_width="56dp"   
 android:layout\_height="56dp"   
 android:text="+"   
 android:textColor="#ffffff"   
 android:textSize="40sp"  
 android:elevation="6dp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentRight="true"  
 android:layout\_marginBottom="16dp"  
 android:layout\_marginRight="16dp"   
 android:background="@drawable/fundo" />

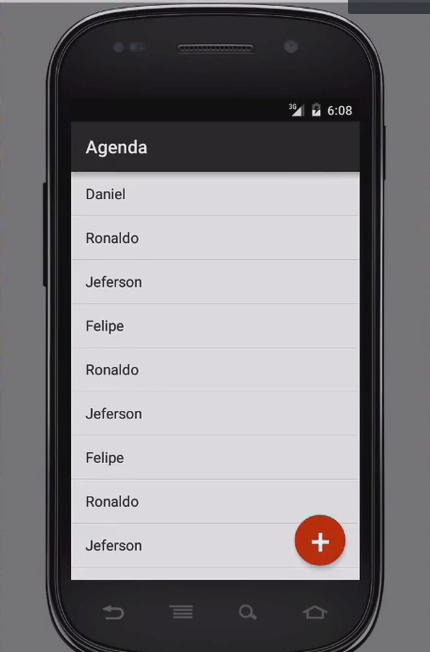
Se você rodar a aplicação verá que a nossa elevação não vai aparecer, isso ocorre devido a um problema de compatibilidade entre os *Android* mais novos em relação aos mais antigos. Para poder fazer com que a sombra apareça falta um atributo que acrescentaremos depois do background que é o stateListAnimator atribuindo nele o @null. O stateListAnimator vai subscrever o comportamento da sombra. Ficaremos com o seguinte, android:stateListAnimator="@null".

Nos *Androids* mais antigos teremos que inserir esse atributo de stateListAnimatorcom essa classificação de nulo para desenhar o que queremos.

Ficaremos com o seguinte código na nossa activity\_lista\_alunos.xml:

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"   
 android:layout\_height="match\_parent"   
 android:orientation="vertical">  
  
 <Listview   
 android:layout\_width="match\_parent"   
 android:layout\_height=“match\_parent"   
 android:id="@+id/lista\_alunos“/>   
  
 <Button android:layout\_width="56dp"   
 android:layout\_height="56dp"   
 android:text="+"   
 android:textColor="#ffffff"   
 android:textSize="40sp"  
 android:elevation="6dp"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentRight="true"  
 android:layout\_marginBottom="16dp"  
 android:layout\_marginRight="16dp"   
 android:background="@drawable/fundo"   
 android:stateListAnimator="@null" />  
  
  
</RelativeLayout>

Para ver como ficou, vamos salvar e rodar:



Nossa sombra agora está aparecendo e ela está bem bem sútil como podemos reparar!

Agora já temos um botão com um destaque no padrão *google*!