01

Floating button

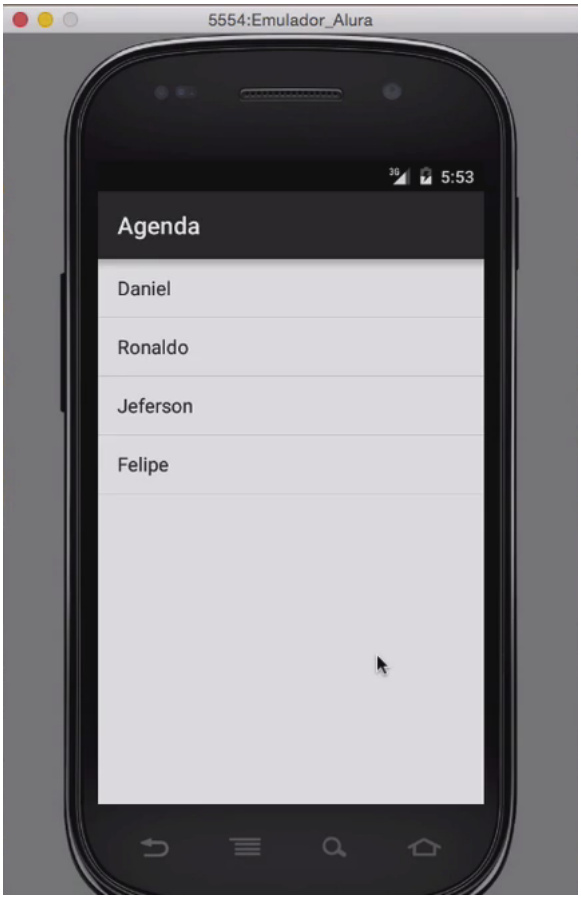
Transcrição

A aplicação tem campos, botão de salvar e uma mensagem automática. Vamos retornar a lista de alunos? Vamos acessar a aba `AndroidManifest.xml` .

Para que a lista de alunos reapareça na tela do emulador é preciso retornar a `intent` na `activity` da `ListaAlunosActivity` . Portanto, 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! Agora, ao rodar a aplicação o que encontramos é a lista de alunos.



O próximo passo é criar uma forma de sair da lista para chegar ao formulário. Para solucionar a questão, vamos inserir um botão "novo aluno" na `activity_lista_alunos.xml` . Ao acessar essa aba encontramos o seguinte:

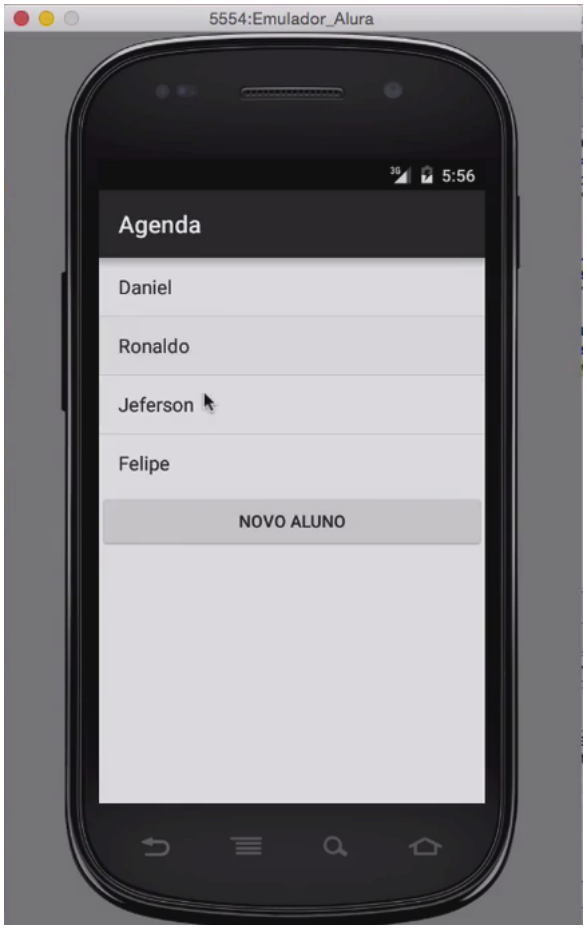
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Listview
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/lista_alunos"/>

</LinearLayout>
```

Embaixo do `LinearLayout` vamos acrescentar uma `View` que representa um botão, a `Button` . Nesse `Button` definiremos os parâmetros de altura e largura. A `layout_width` e `match_parent` como largura e o `layout_height` e `wrap_content` como altura. Por fim, definimos também o texto do botão através da `text` de conteúdo: "Novo aluno". Ficaremos com o seguinte:

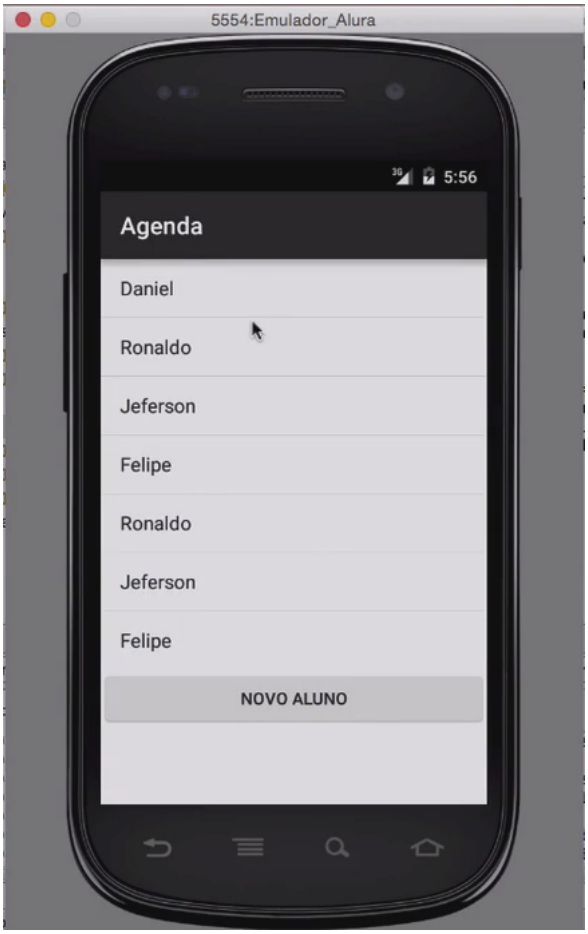
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <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>
```

É preciso pensar nas dimensões da tela para encaixar o botão nela. A `ListView` tem altura e largura definidas como `match_parent` e por isso ocupa a tela inteira e empurra o botão para fora. Isso ocorre porque também utilizamos o `LinearLayout` que possui 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:



Para acrescentar novos alunos vamos na aba 'ListaAlunosActivity.java' na `String` e acrescentamos novos nomes. Rodando o emulador veremos que os nomes foram adicionados:



Ao adicionar mais nomes, a lista mostra um *scroll* automático. Mas, continuaremos com o problema do botão não aparecer. O `ListView` possui internamente um `Scroll` e por isso a lista rola automaticamente.

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

Perceba que já fizemos isso antes! Distribuímos os componentes uns em cima dos outros de forma involuntária. Lembre-se do início, quando nossa `activity_lista_alunos.xml` continha um `RelativeLayout`. A `RelativeLayout` organiza os componentes dispondo-os de maneira relativa, isto é, distribuídos da maneira a ficarem uns em cima dos outros.

Poderíamos acrescentar o atributo `below` no `Button`. Adicionaríamos o `android:layout_below` seguido do `"@id/Lista_alunos"` que indica o local onde o botão deve ficar. Essas alterações seriam feitas na `activity_lista_alunos.xml`. 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:

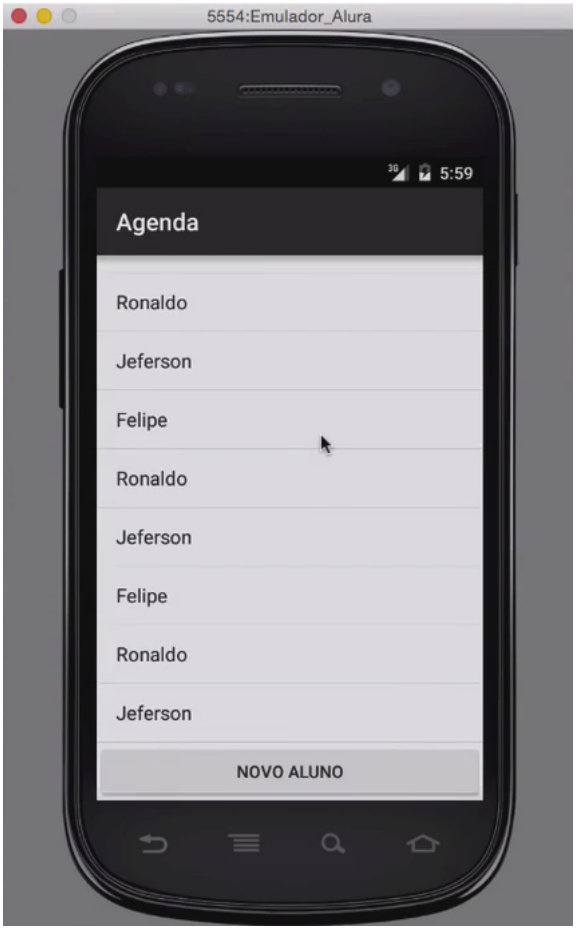
```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout height="match_parent"
    android:orientation="“vertical">

    <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 `alignParentBottom` , que alinha na parte de baixo do "pai". Acrescentaremos, ainda, o `true` , para dizer que é verdadeiro. Ficaremos com:

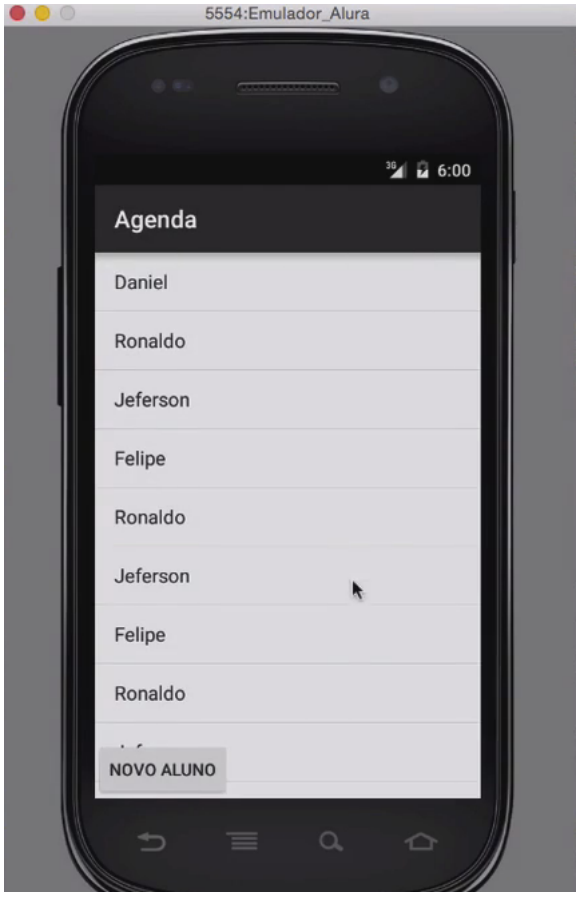
```
    <Button android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Novo aluno"
        android:layout_alignParentBottom="true"/>
```

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:



O botão ficou no lado esquerdo e ainda está em cima dos nomes dos demais alunos.

Vamos jogar o botão para o lado direito para não atrapalhar a leitura. 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:

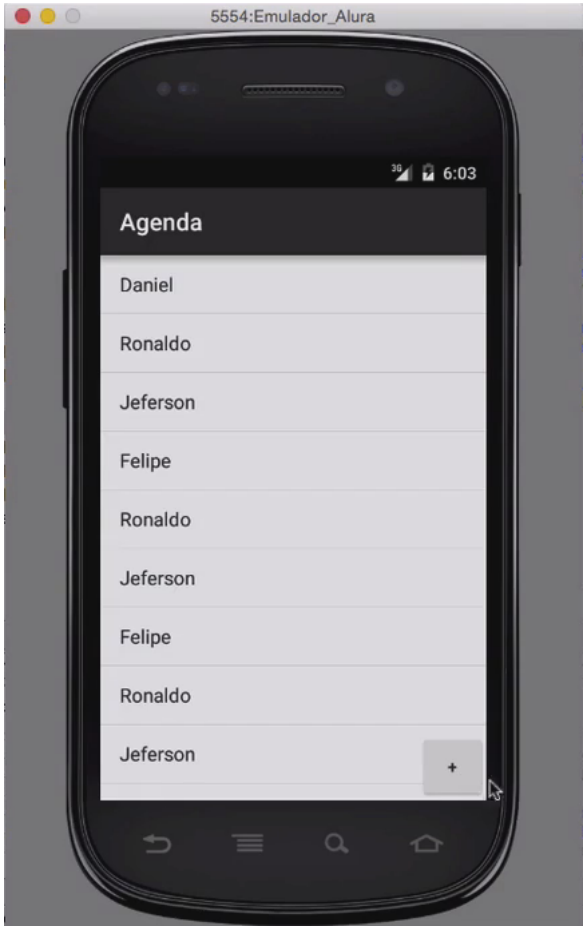
O Google recomenda um padrão para os botões dos aplicativos, que eles apareçam destacando a ação mais importante que realizam. Por exemplo, o aplicativo do Gmail utiliza um botão redondo e vermelho com sinal de "mais" para a ação de adicionar. Vamos alterar nosso botão para que ele fique dentro desse padrão e dando a ele um destaque maior.

Vamos retornar a `activity_lista_alunos.xml` e alterar o texto do botão. Apagaremos o "novos alunos" e acrescentaremos o símbolo de positivo "+" para ficar no padrão Google. Ficaremos com `android:text="+"`. Também vamos alterar o tamanho do botão e escreveremos "na mão", isto é, digitando as medidas que desejamos. Utilizaremos `56dp` e ficaremos com `<android:layout_width="56dp"` e `<android:layout_height="56dp"`. O "56dp" indica que o botão terá essa medida na tela de um celular de quatro polegadas ou em um *tablet* de 10 polegadas. Essa medida é convertida para a realidade e o "56dp" 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 está quadrado e grudado ao canto da tela.

Para alterar isso voltamos na `activity_lista_alunos.xml` e acrescentamos uma margem, com `marginBottom`. Vamos adicionar na margem uma medida, o valor "16 dp" e ficaremos com `android:layout_marginBottom="16dp"`. Acrescentaremos que a margem deve ficar a direita, 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 não resolve o problema da forma do botão. Para solucionar a questão teremos que criar um novo `.xml` que representará a forma do botão.

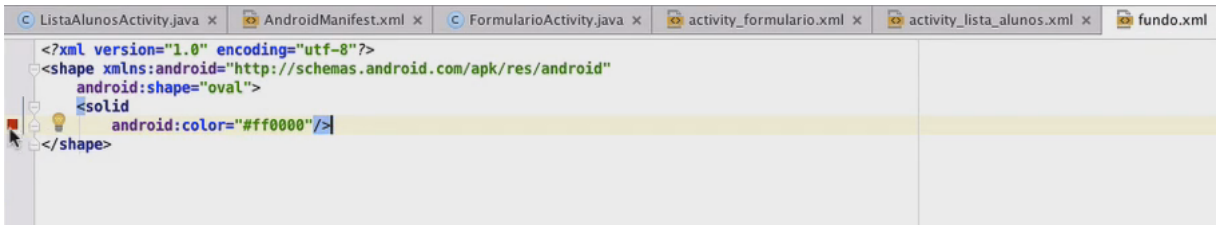
Vamos na pasta "res" e clicamos com o botão direito do mouse em "drawable" e "New > Drawable resource file". O *Drawable* é um arquivo que é desenhável. Seguindo esse caminho abrirá uma janela que pedirá o "File Name", que denominaremos de "Fundo", pois equivale ao fundo do botão. Damos um "Ok" e o `.xml` será criado.

Nesse `.xml` vamos preencher a *tag* `shape` e nele definiremos 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 forma, que será 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 desejamos. Na próxima linha digitamos o atributo `color` e como queremos que seja vermelha, adicionaremos o padrão web, `#ff0000`. Teremos:

```
<?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 no código de cores e fechar do lado esquerdo. Isso fará com que um quadradinho vermelho apareça e clicando nele abre-se uma paleta de cores. Conforme mostram as imagens na sequência:



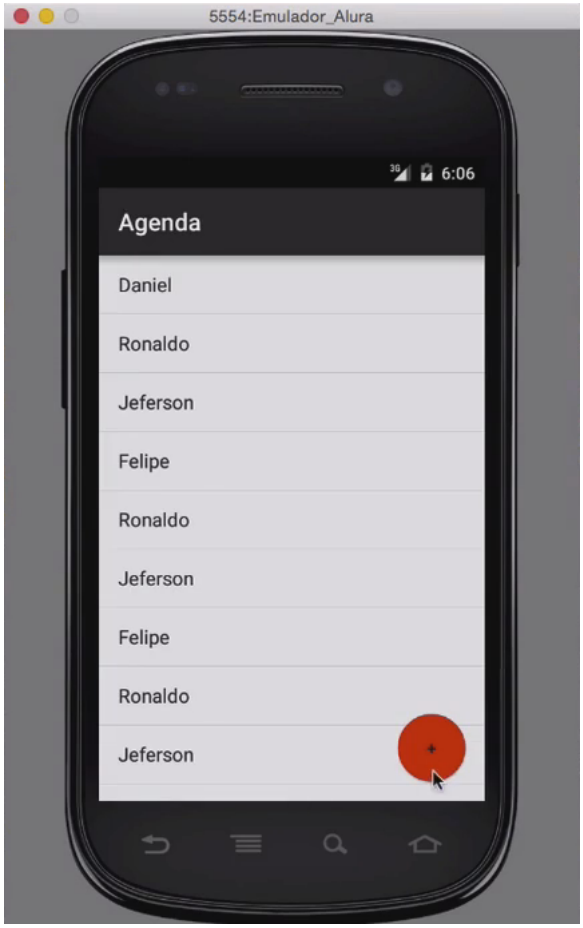
Na paleta basta escolher a cor que deseja, clicar nela e dando um "Choose" o código será preenchido automaticamente. No caso, usaremos o `#ff000"` do padrão da web.

Definimos que será oval e vermelho, então, 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 `@` seguido de `drawable` e `/`. Vamos 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:



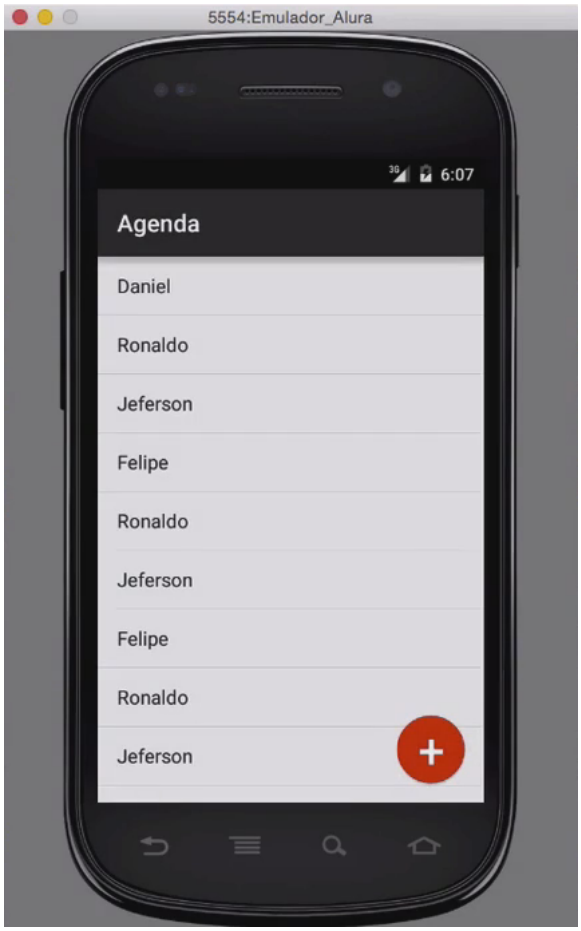
Agora, vamos aumentar o botão e alterar a cor do "+" por branco.

Voltamos na `activity_lista_alunos.xml` , acrescentamos `textColor` e segundo o padrão da *web*, como usaremos a cor branca, adicionamos `#ffffff` . Na linha de baixo inserimos o atributo `textSize` e outro tipo de medida, um `sp` e 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 redimensionada 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 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"/>
```

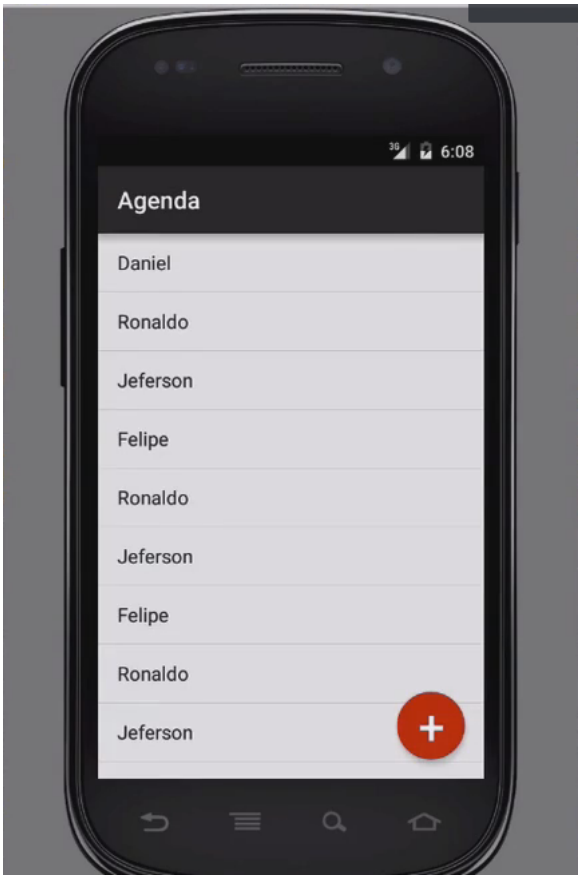
Se você rodar a aplicação verá que a elevação não vai aparecer, isso ocorre devido a um problema de compatibilidade entre os *Android* mais novos em relação aos mais antigos. Para fazer com que a sombra apareça falta um atributo que acrescentaremos depois do `background` que é o `stateListAnimator` atribuindo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

Nos *Androids* mais antigos teremos que inserir esse atributo de `stateListAnimator` com 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 no padrão **Google**!