**Utilizando menus**

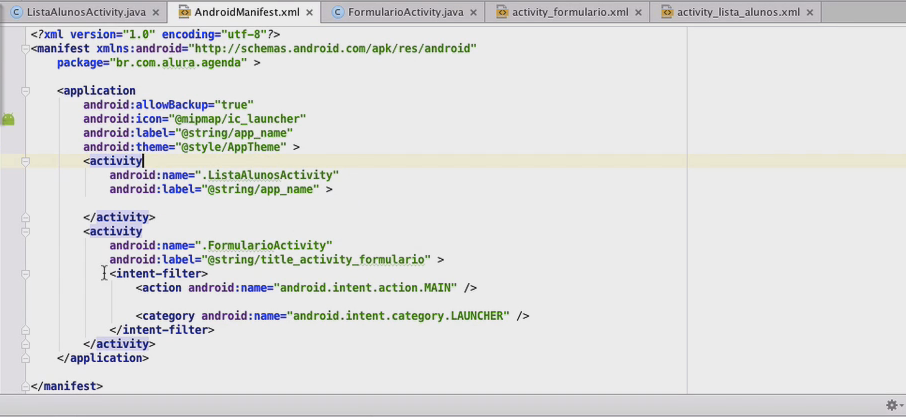
**Utilizando Menus**

**Floating button**

Até agora nossa aplicação tem campos para preencher, o botão de salvar e uma mensagem que será gerada quando apertarmos ele.

Vamos retornar a nossa lista de alunos? Para que a lista de alunos volte a apareça em nossa tela do celular vamos na aba AndroidManifest.xml onde encontraremos isso:

Temos a seguinte tela:



Para que a lista de alunos volte e apareça na tela quando rodarmos o emulador vamos retornar a intent para aactivity da ListaAlunosActivity. Para isso vamos selecionar com o mouse apenas a intent e deslocá-la para aactivity da lista de alunos. Damos um "Comand+X" ou "Ctrl+X" e um "Comand+V" ou "Ctrl+V" dentro daactivity da lista de alunos. Vai ficar 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

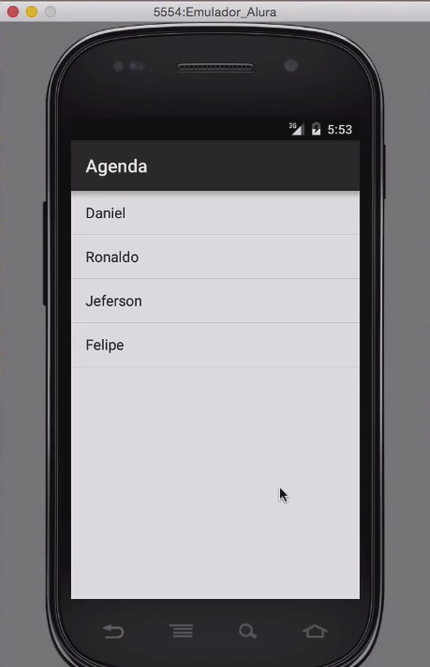
android:name=".FormularioActivity"

android:label="@string/titIe\_activity\_formulario" >

</activity>

</application>

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



Agora, vamos criar uma forma de sair da lista e chegar no formulário. Para fazer isso podemos colocar um botão de "novo aluno" para chegar até o formulário.

Para modificar a tela da lista alunos vamos na pasta "layout", na activity\_lista\_alunos.xml e 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 representa um botão, através do Button. NesseButton definiremos os parâmetros de altura e largura. A layout\_width como largura, acompanhada de ummatch\_parent e layout\_height de altura, acompanhada de um wrap\_content e, por fim, definimos também o texto do botão através da text. Vamos inserir o texto "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

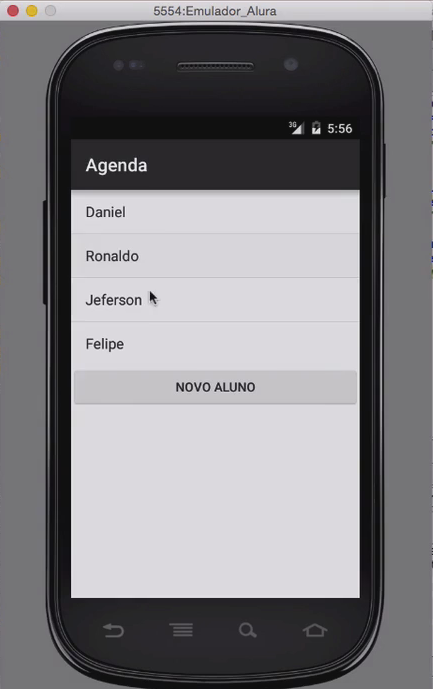
android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

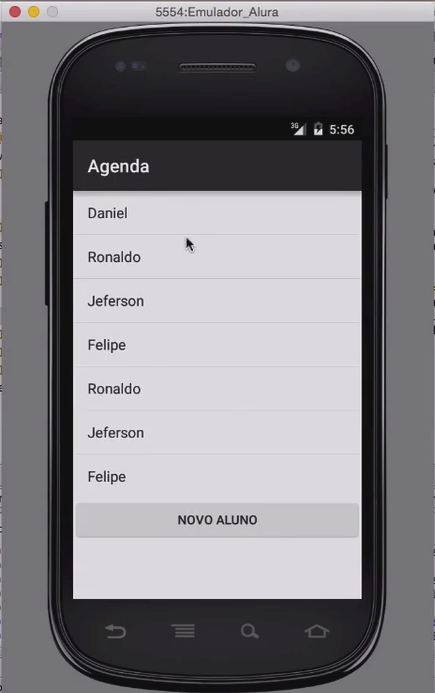
android:text="Novo aluno"/>

</LinearLayout>

Para que o botão apareça na tela temos que atentar para a ListView que tem sua altura e largura definida no parâmetro match\_parent, isto significa que ocupará a tela inteira, pois, ela empurra nosso botão para fora da tela. Isso ocorre também porque estamos utilizando um LinearLayoutque indica que a orientação deve ocorrer na vertical. Para resolver essa situação trocamos o match\_parent da altura da ListView por um wrap\_content, para que a lista passe a ocupar apenas o necessário. Vamos salvar e rodar a aplicação. Teremos o seguinte:



Se quisermos acrescentar novos alunos vamos até a aba 'ListaAlunosActivity.java' no String, onde já estão os nomes dos alunos e acrescentamos novos nomes. Rodando o emulador mais uma vez veremos o que temos:



Se quisermos acrescentar ainda mais nomes é possível e inclusive a lista vai rolar mas, continuaremos com o problema de que o botão não vai mais estar aparente, ele estará fora da tela. O ListView tem internamente umScroll, por isso ela já rola automaticamente. Queremos que o botão fique fixo e que ele apareça sempre na tela, na parte inferior e em cima da lista, mesmo quando role a barra.

Perceba que já colocamos um componente em cima do outro de forma involuntária. Lembre-se do ínicio, quando nossa activity\_lista\_alunos.xml continha um RelativeLayout, lá no início quando inserimos os dois primeiros alunos. A RelativeLayout organiza os componentes dispondo-os de maneira relativa, isto é, perto de quem os componentes estão, então eles ficam distribuídos da maneira a ficarem uns a cima dos outros.

Poderíamos acrescentar o atributo bellow no Button. Adicionaríamos o android:layout\_bellow seguido do idque é "@id/Lista\_alunos" para indicar abaixo de quem gostaríamos que o botão ficasse. 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:

<RelativLayout 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"/>

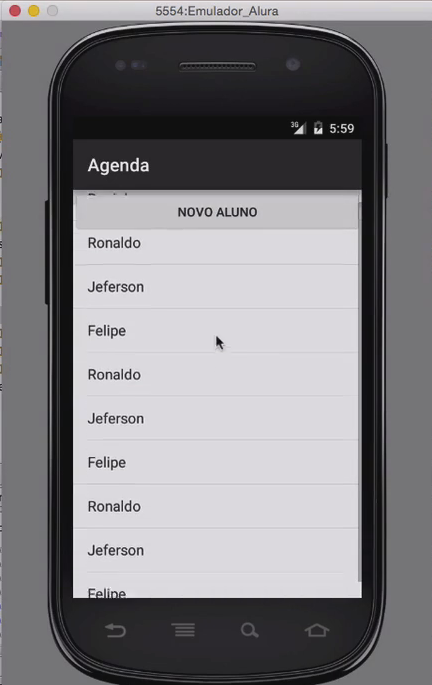
<Button android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Novo aluno” />

</RelativeLayout>

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



O botão aparece em cima da lista e quando rolamos ele não desaparece mas, ele está no topo da lista e não embaixo como queremos. Para colocar o botão embaixo, utilizaremos um atributo do RelativeLayout que é oalignParentBottom, isso significa, alinhar na parte de baixo do meu "pai", que nesse caso é o RelativeLayout. Acrescentaremos ainda o true, para dizer que é verdadeiro. Ficaremos com:

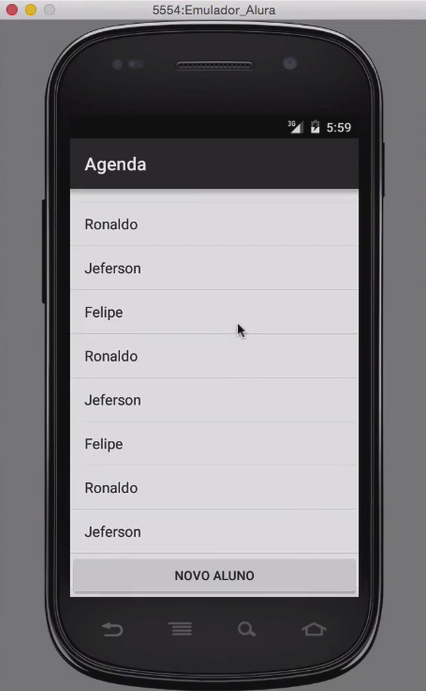
<Button android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Novo aluno"

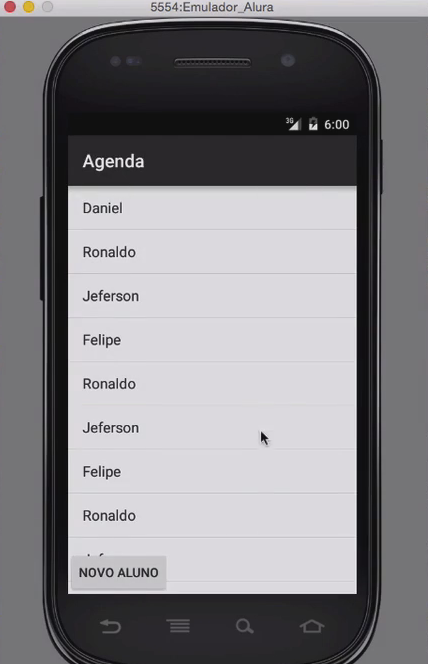
android:layout\_alignParentBottom="true"/>

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



Agora ele está alinhado como queremos mas, repare que ele ainda está atrapalhando a leitura dos nomes, vamos diminuir esse botão. Para isso basta alterar no Button a largura do botão, adicionando wrap\_content e teremosandroid: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 outroalignParent, mas dessa vez diremos que ele se alinha com o "pai" à direita, teremos um layout\_alignParentRighte 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 comandroid: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"

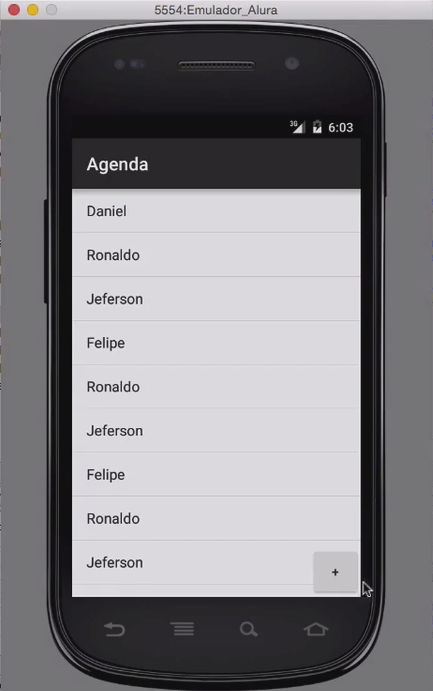
android:layout\_height="56dp"

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 comandroid: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"

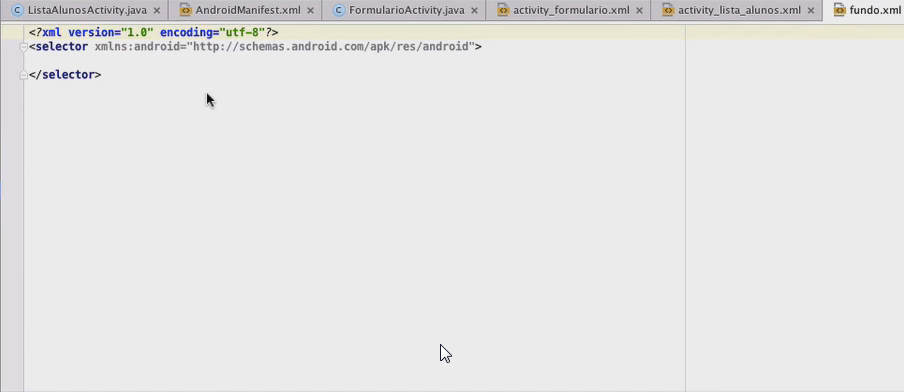
android:layout\_marginBottom="16dp"

android:layout\_marginRight="16dp" />

Entretanto, isso ainda não vai resolver o problema da forma do botão. Para isso teremos que criar um novo xmlque 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">

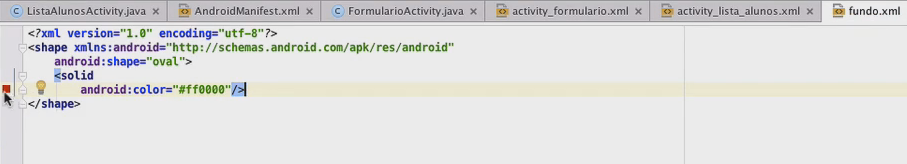
<solid

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 seguinteandroid:background="@drawable/fundo" e o código ficará assim:

<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="56dp"

android:layout\_height="56dp"

android:text="+"

android:textColor="#ffffff"

android:layout\_alignParentBottom="true"

android:layout\_alignParentright="true"

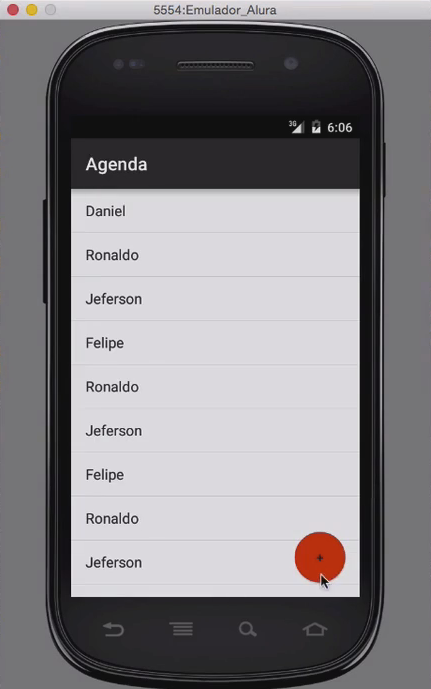
android:layout\_marginBottom="16dp"

android:layout\_marginRight="16dp"

android:background="@drawable/fundo" />

</LinearLayout>

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 text\_Size="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"

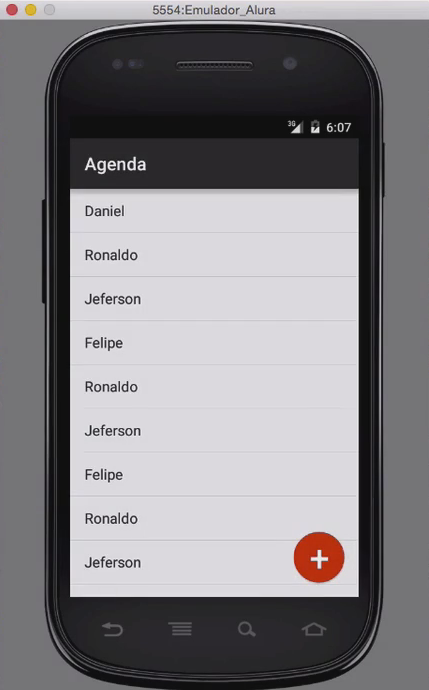
android:layout\_alignParentright="true"

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 stateListAnimator com essa classificação de nulo para desenhar o que queremos.

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

<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="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"

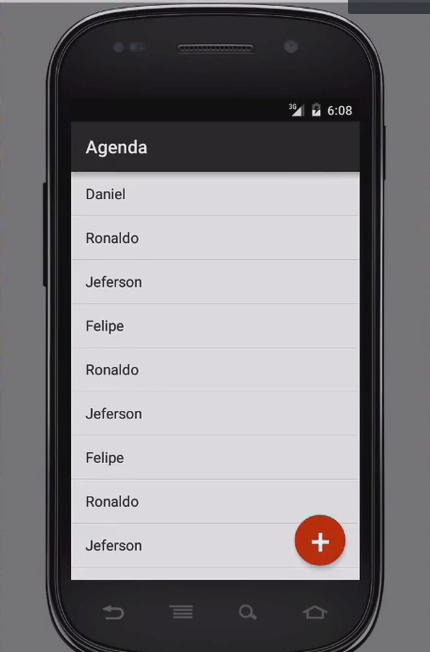
android:layout\_marginRight="16dp"

android:background="@drawable/fundo"

android:stateListAnimator="@null" />

</LinearLayout>

Para ver como ficou, vamos salvar e rodar:



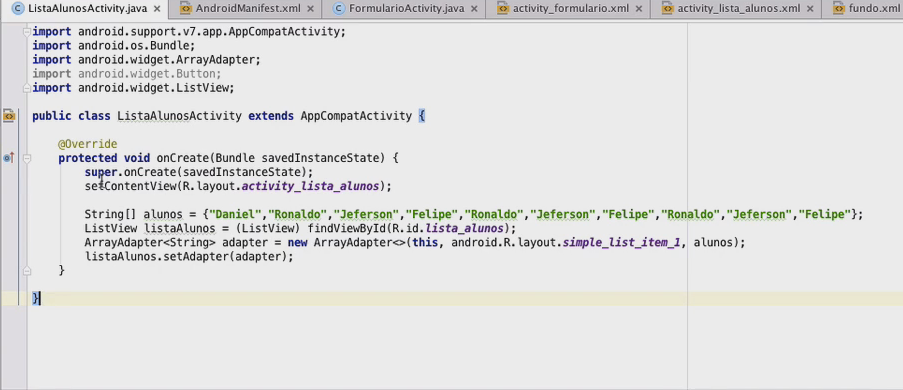
Nossa sombra agora está aparecendo e ela está bem bem sútil como podemos reparar!

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

**Navegando entre as telas**

Agora nós temos um botão flutuante mas, queremos que ele tenha uma ação real. Queremos que ele nós leve até o formulário e que nós possamos salvar os nomes dos alunos!

Nossa tela está da seguinte maneira:



Se estamos falando de um comportamento que queremos introduzir no botão, vamos alterar oListaAlunosActivity que está dentro da pasta "br.com.alura.agenda".

Ficaremos na aba ListaAlunosActivity. Nela encontramos o seguinte:

import android.support.v7.app.AppCompatActivity; import android.os.Bundle;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.ListView;

public class ListaAlunosActivity extends AppCompatActivity {

@Override protected void onCreate(Bundle savedInstanceState);

set.onCreate(savedInstanceState);

setContentView(R.layout.activity\_lista\_alunos);

String [] alunos = {"Daniel","Ronaldo","Jeferson","Felipe","Ronaldo","Jeferson","Felipe","Ronaldo","Jeferson , Felipe"};

ListView listaAlunos = (ListView) findViewById(R.id.lista\_alunos);

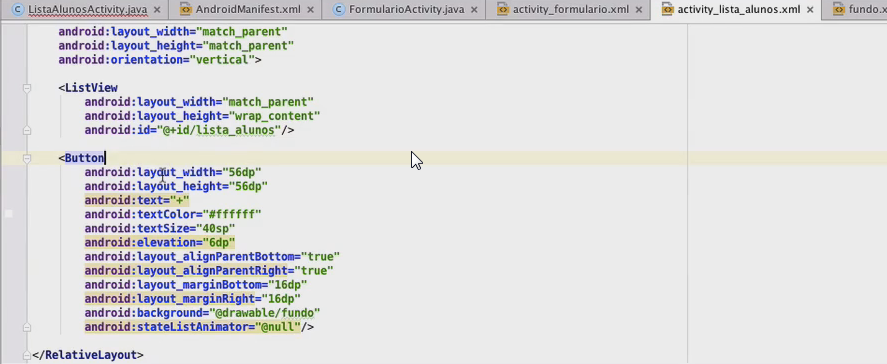
ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1, alunos);

listaAlunos.setAdapter(adapter);

}

Dentro vamos precisar de uma referência para o botão de "Novo aluno" que criamos anteriormente. Assim, poderemos associar algum comportamento a ele. Após o listaAlunos.setAdapter(adapter) damos dois enter e vamos adicionar Button, damos um espaço, escrevemos novoAluno, novo espaço e =, outro espaço e digitamos afindViewById e definiremos na sequência o Id desse botão que preencheremos dentro do parênteses. Teremos:Button novoAluno = findViewById ().

Para terminar de preencher precisamos criar um Id para o botão. Para isso voltamos naactivity.\_lista\_alunos.xml e encontraremos a seguinte tela:



O id pode ser adicionado em qualquer lugar do Button. Acrescentaremos o id e também o nome do botão, "novo aluno", isto é, android:id="@+id/novo\_aluno" Teremos o seguinte:

<Button android:id="@+id/novo\_aluno"

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"/>

Voltamos no ListaAlunosActivity.java e inserimos entre os parênteses a Id que acabamos de criar no .xml. Ficaremos com: Button novoAluno = findViewById (R.Id.novo\_aluno). Ainda temos que definir um *cast* para que ele devolva uma view, então, digitamos o atalho "Alt+Enter" em cima da findViewById e escolhemos o "Cast to android.widget.Button". Teremos:

@Override protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState); {

setContentView(R.layout.activity\_lista\_alunos);

String [] alunos ={"Daniel","Renaldo","Jeferson","Felipe","Renaldo","Jeferson","Felipe","Ronaldo","Jeferson","Felipe"};

ListView listaAlunos = (ListView) findViewById(R.id.lista\_alunos);

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1, alunos);

listaAlunos.setAdapter(adapter);

Button novoAluno = (Button) findViewById (R.Id.novo\_aluno);

}

}

Para acrescentar um botão com clique no *Android* acrescentaremos na linha seguinte da findViewById oNovoAluno.setOnClickListener(), pois o que queremos é "ouvir" o evento de clique. Dentro do parênteses criamos de novo uma classe anônima, new\_OnClickListener, damos um enter e o *Android* cria a estrutura e agora é só preencher o comportamento que queremos.

novoAluno.setOnClickListener(new View.OnClickListener() {

@Override public void onClickView v) public void onClickListener (new View.OnClickListener() {

}

}

O que poderíamos fazer agora para indicar o comportamento que queremos é instanciar a activity.

Mas, o que queremos é colocar a activity em primeiro plano e isso é uma responsabilidade do próprio *Android*. É o mesmo caso, por exemplo, de quando estamos com uma aplicação aberta e recebemos uma chamada telefônica, a prioridade é para a chamada telefônica, então, ela vai se sobrepor a aplicação aberta e isso quem faz é o *Android*. Para colocar a activity na frente temos que dialogar justamente com ele, o *Android*. Por isso, não vamos instanciar a activity.

Vamos declarar qual a nossa intenção para o *Android*, então, nessa nova estrutura vamos criar um objeto de tipoIntent. Vamos nomear esse novo objeto de VaiProFormulario, ignore a gramática de "pro". E para criar ele vamos pedir 'new Intent ' e decidir alguns parâmetros, que são bem fáceis de lembrar. Teremos intent intentVaiProFormulario = new Intent().

O primeiro parâmetro é a nossa identificação, isto é, quem somos. Apenas através da identificação é que conseguimos pedir para o *Android* trocar de tela. Então, vou dizer que sou o ListaAlunosActivity e que gostaria de ir no formulário. Para isso, colocaremos entre os parênteses da Intent um this. Lembre-se que o this significa uma referência a classe anônima que criamos anteriormente. Para fazer uma referência ao ListaAlunosActivitytemos que introduzir no parênteses, antes do 'this', ListaAlunoActivity, um ponto e this. Agora vamos acrescentar o segundo parâmetro, para onde estamos indo. Acrescentamos uma vírgula e digitamosFormularioActivity, um ponto e class. Teremos

intent intentVaiProFormulario = new Intent (ListaAlunosActivity.this, FormularioActivity.class)

E, por fim, teremos o seguinte:

@Override protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState); {

setContentView(R.layout.activity\_lista\_alunos);

String [] alunos ={"Daniel","Ronaldo","Jeferson","Felipe","Ronaldo","Jeferson","Felipe","Ronaldo","Jeferson","Felipe"};

ListView listaAlunos = (ListView) findViewById(R.id.lista\_alunos);

ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple\_list\_item\_1, alunos);

listaAlunos.setAdapter(adapter);

Button novoAluno = (Button) findViewById (R.Id.novo\_aluno);

novoAluno.setOnClickListener(new View.OnClickListener() {

@Override public void onClickView v) public void onClickListener (new View.OnClickListener() {

intent intentVaiProFormulario = new Intent (ListaAlunosActivity.this, FormularioActivity.class)

}

});

}

O que fizemos por último foi avisar para o *Android* o que ele terá de instanciar através da classe, class, que indica, no caso, que deve ser o FormularioActivity.

A intent que criamos é uma variável que o *Android* desconhece e para passar o conhecimento ao *Android*, avisar que essa é nossa intenção, vamos adicionar na próxima linha da intent um método, o startActivity e entre os parênteses vamos acrescentar intentVaiproFormulario e fechar com o ;. TeremosstartActivity(intentVaiproFormulario). Ficaremos com:

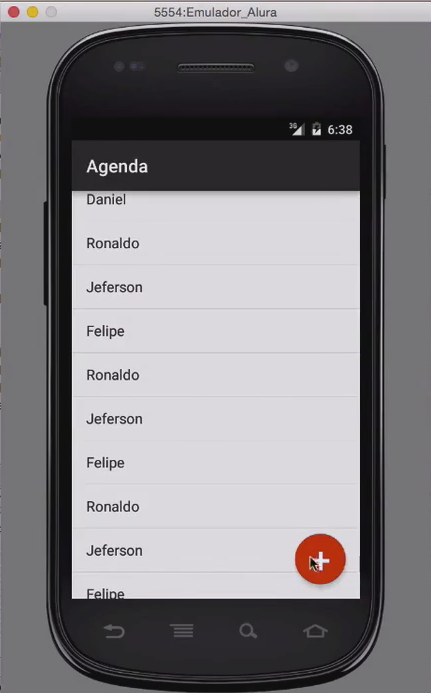
@Override public void onClickView v) public void onClickListener (new View.OnClickListener() {

intent intentVaiProFormulario = new Intent (ListaAlunosActivity.this, FormularioActivity.class)

startActivity(intentVaiproFormulario);

}

Vamos salvar e dar play para ver como ficou no emulador. Bom, depois que apertamos o botão de "+" somos redirecionados para o formulário, mas ao chegar no formulário e selecionarmos o salvar permanecemos na tela do formulário.





Conseguimos agora apertar o botão de "+" e ele vai nos direcionar para o formulário, mas quando chegamos no formulário e preenchemos o que queremos ele não volta para a lista.

Vamos voltar na aba FormulárioActivity.java e usar a mesma ideia. Encontramos ela assim:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState); setContentView(R.layout.activity\_formulario);

Button botaoSalvar = (Button) findViewById(R.id.formulario\_salvar);

botaoSalvar.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH\_SHORT).show();

}

});

}

Vamos utilizar uma intente a essa intent nomearemos vaiPralistae "=" a new intent. Vai ficar em vermelho pedindo para ser importada, para importar basta usar o atalho "Alt+Enter". Ficamos com, `intent vaiPralista = new intent()'.

Agora vamos acrescentar no parênteses o contexto, isto é, o this mas, lembrando que como estamos em uma classe anônima temos que adicionar os parâmetros. Adicionaremos: quem sou e para onde vou. Então, quem somos, preenchemos com o FormularioActivity antes do this e, para onde vou, digitamos uma vírgula e após othis acrescentamos ListaAlunosActivity e informamos a classe, então digitamos um ponto e class. Teremos o seguinte

intent vaiPralista = new intent(FormularioActivity, this, ListaAlunosActivity.class)

Mas, o *Android* não conhece essa intenção, então temos que avisar ele através do método startActivity que digitaremos na próxima linha. Acrescentamos também a indicação vaiPraLista.

Ficaremos com:

@Override

public void onClick(View v) {

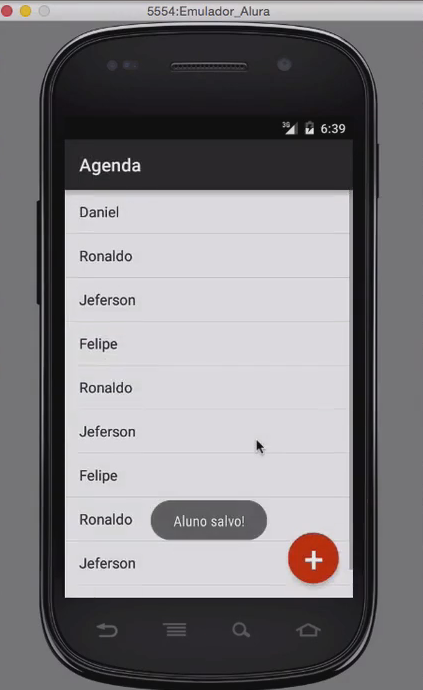
Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH\_SHORT).show();

intent vaiPralista = new intent(FormularioActivity, this, ListaAlunosActivity.class);

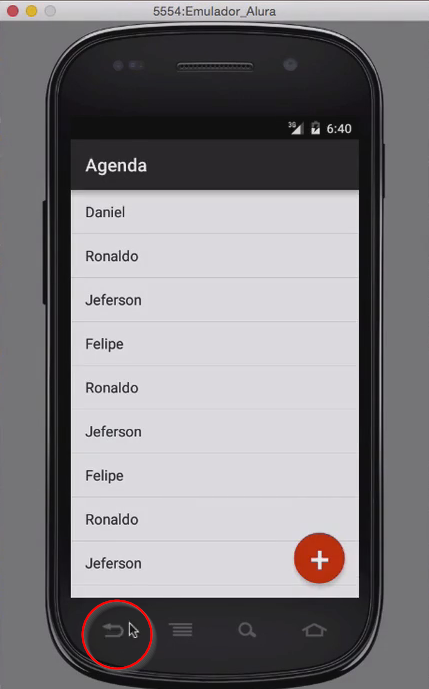
startActivity(vaiPralista);

}

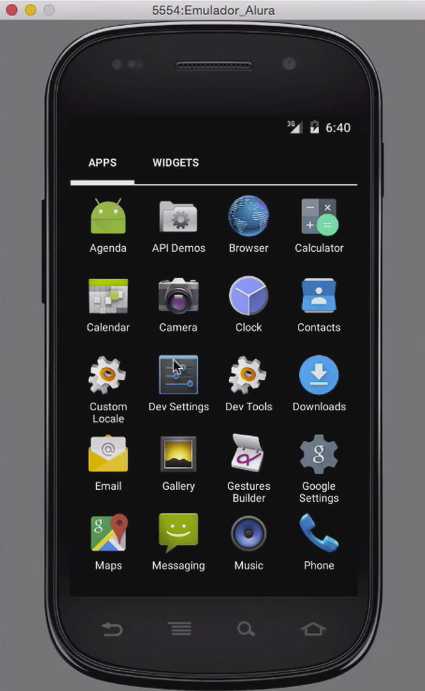
Vamos rodar o emulador e ver como ficou!



Pronto! Agora conseguimos voltar na lista de alunos! Mas, e se selecionarmos a opção de voltar do próprio celular? Aquela flecha que fica no teclado do próprio celular. Aqui para enteder melhor:



Voltamos para o formulário e não para o início da aplicação, o que é um pouco estranho. Se clicamos em salvar iremos duplicar a informação e se clicamos em salvar de novo voltamos para uma outra agenda. O ideal é que quando estivessemos na lista voltassemos para a tela do celular, a do Launcher, a home, aquela que mostra todas as aplicações do meu celular. Esta aqui:



Vamos entender um pouco o que está acontecendo! Vamos na ListaAlunosActivity.java. Toda vez que chamamos o intent e preenchemos com o startActivity o *Android* de fato inicia uma nova activity do formulário, o que está ok. Mas, vamos rapidamente na FormularioActivity.java, qundo pedimos para o *Android* ir para lista e damos um startActivity para a ListadeAlunos o *Android* entende que é para iniciar uma novaactivity e não que queremos apenas voltar apenas para a lista. Ele irá criar uma nova lista e vamos de fato para essa outra lista, e quando apertarmos para voltar ele vai destruir a lista que acabamos de criar. Isto é, algo bastante estranho acontece!

Vamos alterar esse comportamento para que quando estivermos na lista e apertamos o botão de voltar do teclado, sejamos direcionados para o *Launcher*.

Vamos na aba FormularioActivity.java e ao em vez de utilizarmos uma intent vamos apagar ela e vamos dizer para o *Android* que queremos apenas finalizar, ou seja, terminar a Activity do formulário.

Então, após a linha onde especificamos a mensagem que queremos que apareça após clicar o botão de "Salvar", acrescentamos na próxima linha um finish. Ficaremos com:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_formulario) ;

Button botaoSaLvar = (Button) findViewById(R.id.formulario\_salvar);

botaoSalvar.setOnClickListener(new View.OnClickListener() {

@Override public void onClick(View v) {

Toast.maketext(FormularioActivity.this, “Aluno salvol“, Toast.LENGTH\_SHORT).show();

finish();

}

});

}

Vamos rodar o emulador para ver se deu certo. Se apertamos o botão de "+", preenchermos o formulário, salvarmos, voltaremos para a lista. Se selecionarmos o botão de voltar... Voltaremos ao home!

Pronto! Essa parte está finalizada!

**Utilizando menus**

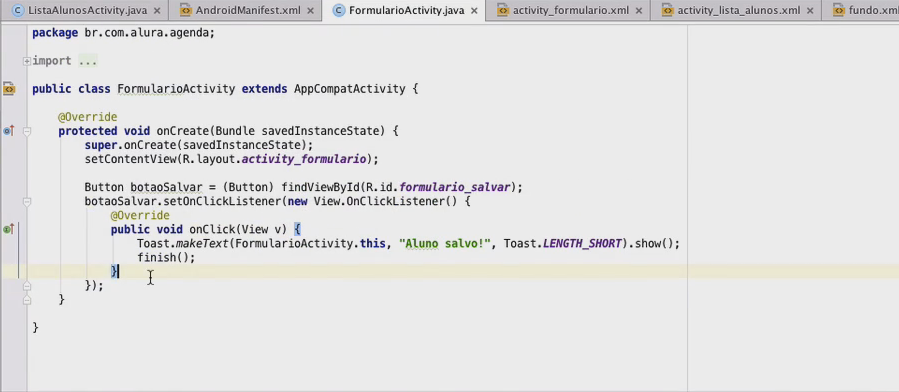
Apenas lembrando que no último vídeo conseguimos introduzir no botão "+" a função de ir até o formulário e no botão "Salvar" a de voltar para a lista. Também conseguimos apertar o botão de voltar da tela do celular e fazer com que ele retorne ao *Launcher*.

Agora, vamos alterar o botão de "Salvar" por um com um com um design mais bacana. Para que ele combine com o nosso botão "+" com padrão *google*. Lembrando como ele está no momento:



Vamos introduzir o botão de salvar junto a nossa barra de "Formulário". Desse modo, se o formulário for muito grande e a barra tiver que rolar para ver tudo será mais fácil se o botão ficar em cima.

Vamos para a aba FormularioActivity.java. Essa é nossa tela:



A barra, entretanto, é controlada pelo próprio *Android*. Quando extendemos a activity com oAppCompatActivity estamos dizendo que queremos uma aplicação que tem uma *action bar*. Então, precisamos comunicar ao *Android* que queremos introduzir na barra um botão.

Podemos fazer isso introduzindo um método na activity. Para isso, damos alguns enter após o método e escrevemos onCreateOptionsMenu ele vai sugerir um método. Dando um enter, automaticamente ele preencher o*overreide* desse método.

Esse método é no *Android* cria a barra do Formulário e define quais os itens do menu que constarão nela. Os botões que queremos inserir fazem parte do menu da aplicação. A diferença é que algumas opções serão mostradas dentro da barra e as outras como menu.

Ficaremos com:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_formulario);

Button botaoSaLvar = (Button) findViewById(R.id.formulario\_salvar);

botaoSalvar.setOnCIickListener(new View.0nCIickListener() {

@Override

public void onClick(View v) {

Toast.makeText(FormularioActivity.this. "Aluno salvo!", Toast.LENGTH\_SHORT).show();

finish();

}

});

}

@Override

public boolean onCreateOptionsMenu(Menu menu) {

return super.onCreateOptionsMenu(menu);

}

Inicialmente, não temos nada dentro do método e para criar um menu vamos na pasta "res". Clicando duas vezes nela vemos que já existem dois menus criados: "menu\_formulário.xml" e o "menu\_lista\_alunos.xml". Eles foram criados automaticamente pelo *Android Studio*, pois, toda vez que criamos uma activity criam-se esse menus.

Para ver isso, basta ir em menu, "File > New File > Blanky Activity". Quando abrir a janela pedindo os dados da nova*Activity* repare que existe um campo referente ao menu, o *Menu Resources Name*. Esse campo diz respeito ao menu que será criado. automaticamente, junto com a activity.

Se abrirmos o "menu\_formulário.xml" veremos que já aparece um menu mais ou menos pronto. Nele já contêm uma *tag* menu que diz, justamente, que isso representa um menu. Nós temos a seguinte tela:

<menu 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"

tools:context="br.com.alura.agenda.FormularioActivity">

<item android:id="@+id/action\_settings"

android:title="Settings"

android:orderInCategory="100" app:showAsAction="never" />

</menu>

Dentro do menu temos um item, o Settings. Bom, esse item não será necessário e vamos simplesmente apagá-lo.

Vamos começar a descrever os itens que queremos no menu. Damos um enter e inserimos a *tag* item, introduzindo também um id que chamaremos de menu\_formulário\_ok. Já faremos isso pois iremos manipulá-lo mais adiante. Vamos inserir o android:id="@+id/menu\_formulario\_ok".

Também iremos introduzir um texto, para indicar o que significa essa opção, vamos colocar que o título dele será "ok". Damos um enter e acrescentamos 'android:title="Ok"'.

Outra coisa que acrescentaremos é um ícone. Usaremos, para tanto, o icon,assim, o botão aparecerá em forma de desenho na nossa barra. Para fazer isso utilizaremos, na próxima linha, icone também um recurso desenhável, odrawable, ficando android:icon="@drawable" . Teremos:

<menu 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"

tools:context="br.com.alura.agenda.FormularioActivity">

<item android:id="@+id/action\_settings"

android:title="Settings"

android:orderInCategory="100"

app:showAsAction="never" />

<item android:id="@+id/menu\_formulario\_ok"

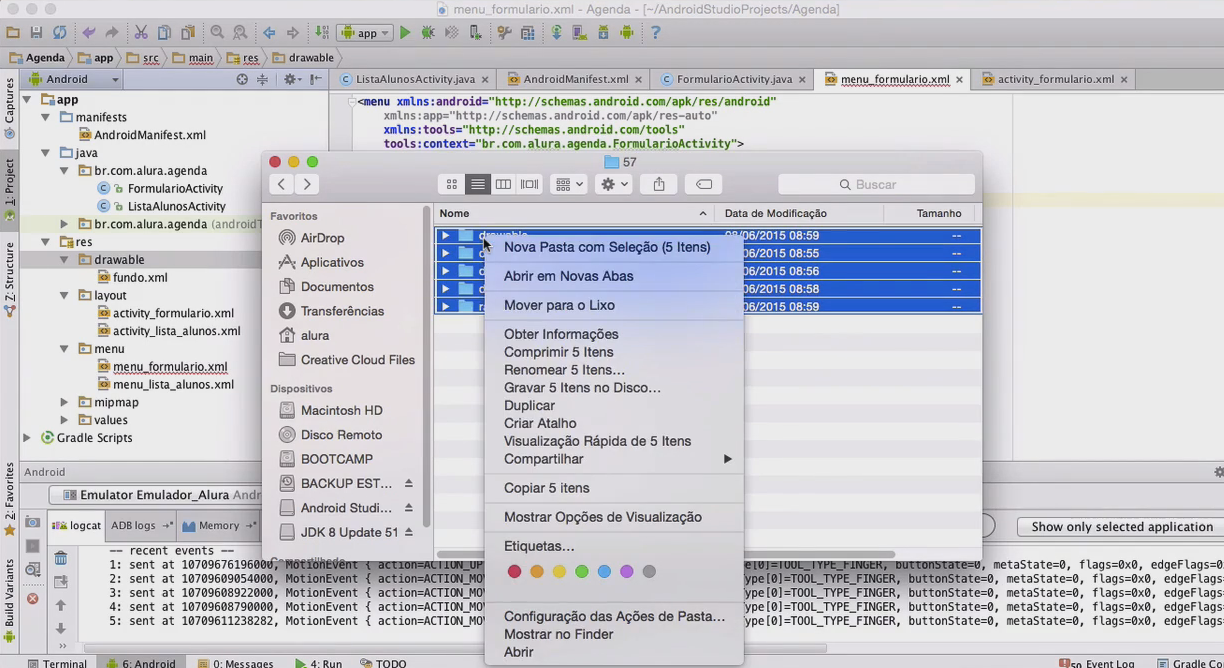
android:title="Ok"

android:icon="@drawable" />

</menu>

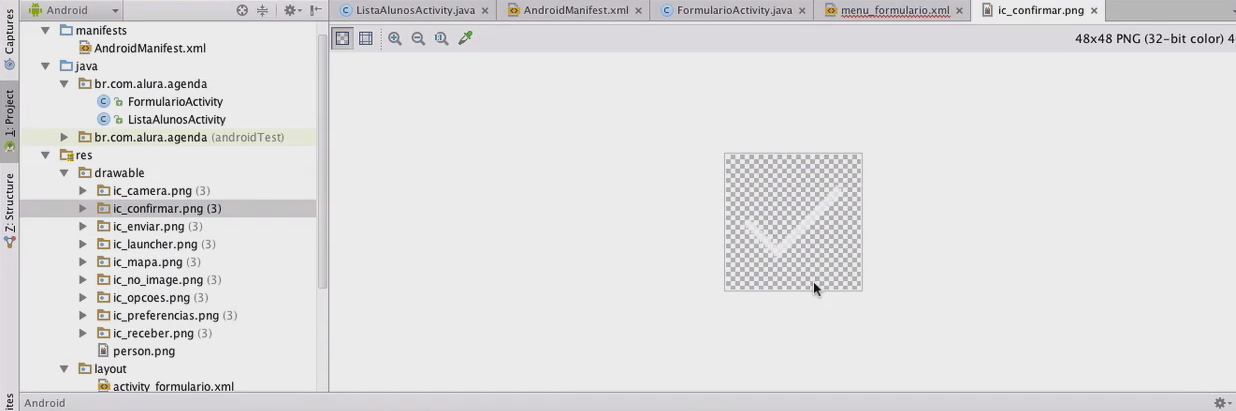
Para completar o icon temos que utilizar um recurso, por enquanto, se olharmos na pasta "drawable" temos apenas o arquivo fundo.xml.

Não tem problemas, vamos no arquivo e copiamos todas as pastas pois nelas estão contidas todas as imagens que utilizaremos ao longo do curso.



Selecionamos com o mouse todas as pastas, clicamos com o botão direito e selecionamos "Copiar 5 itens" e levamos para a pasta do nosso projeto. Fazemos o seguinte caminho, "AndroidStudioProjects > Agenda > app > src > main > res ". Chegando no "res" é só pedir para colar os itens, clicando com o botão direito e selecionando "Colar 5 itens". Se abrir uma janela perguntando se desejamos substituir é só escolher o "Substituir" . Agora temos novos recursos na aplicação.

Assim, voltando para o *Android Studio* se abrirmos a pasta "res", veremos que ela está cheia de novos itens. O que utilizaremos é o item de confirmar, clicando duas vezes ele irá aparecer,



O nome desse item é ic\_confirmar.png.

Vamos retornar na nossa aba menu\_formulario.xml e completamos o nosso icon com o nome do item que vamos adicionar. Ficando android:icon="@drawable/ic\_confirmar". Lembre-se de fechar com o />. Teremos:

<menu 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"

tools:context="br.com.alura.agenda.FormularioActivity">

<item android:id="@+id/action\_settings"

android:title="Settings"

android:orderInCategory="100"

app:showAsAction="never" />

<item android:id="@+id/menu\_formulario\_ok"

android:title="Ok"

android:icon="@drawable/ic\_confirmar"/>

</menu>

Agora que criamos o .xml precisamos utilizar ele no nosso código. Se voltarmos no FormularioActivity.javavemos o método que o *Android* criou para chamar a barrinha:

@Override public boolean onCreateOptionsMenu(Menu menu) {

return super.onCreateOptionsMenu(menu);

}

Nesse método vamos pedir que o *Android* pegue o menu\_formulario.xml e transforme em View. Isto é, transformar em coisas com as quais o usuário possa interagir.

Para pegar um .xml qualquer em um conjunto de Views a gente usa um conceito do *Android* chamado 'inflaiter'

Para isso, ficamos no FormulárioActivity.java e no método menu, damos um enter e na próxima linha, acrescentamos um conceito especialista em fazer menus, que é o getMenuInflater.

Esse método getMenuInflater está vindo da AppCompatActivity e serve para devolver uma instância de um menu inflater. Então, acrescentaremos antes do que já escrevemos, MenuInflater inflater e um "=" agetMenuInflater. Teremos, MenuInflater inflater = getMenuInflater. Se ficar em vermelho lembre de importar dando "Alt+Enter".

Temos que lembrar que o inflater tem que inflar algo, no caso ele é especialista em inflar uma .xml. Temos que informar para ele qual a xmlque ele vai inflar.

Então, digitamos a classe R, um ".", e como queremos inflar um menu, vamos buscar ele dentre as opções apresentadas, escolhemos "menu", digitamos "." de novo e vamos escolher o "menu\_formulario", pois é justamente isso que queremos inflar. Teremos, inflater(R.menu.menuformulário).

Precisamos dizer, ainda, onde ele vai colocar essas informações, no caso, no menu da *action bar* que está vazio. Ele é aquele "menu" da *action bar* que está vazio.



Completaremos com mais um "." e menu, para indicar que queremos que todos os itens estejam nele. Completando, ficaremos com, inflater(R.menu.menuformulário.menu).

Não podemos esquecer de chamar o método inflate. Ele será adicionado depois do `inflater', damos um "." digitamos 'inflate'.

Ficaremos com:

@Override

public boolean onCreateOptionsMenu(Menu menu) {

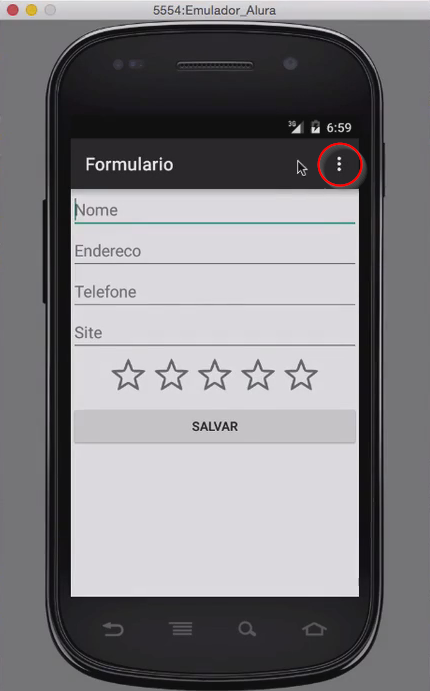
MenuInflater inflater = getMenuInflater();

inflater.inflate(R.menu.menuformulario, menu)

return super.onCreateOptionsMenu(menu);

}

Vamos ver o que acontece se fizermos só isso:



Nossa barra não está mais vazia! Ela está com esses três pontinhos no canto direito. Se clicarmos em cima das três bolinhas o que aparece é um "ok" escrito. Nosso ícone do menu veio parar aqui e de maneira textual.



Mas, nós tínhamos pedido para ele colocar o item de *check mark*, aquele que aparece na imagem acima. Lembra?!

Para mostrar o item do menu, não em forma de texto, mas em forma de ícone vamos voltar na aba 'menu\_formulario.xml'. Vamos usar a propriedade showAsaction Quando digitarmos ele vai dar algumas opções de escolha, vamos usar o always, pois queremos que nosso ícone seja sempre mostrado na *action bar*. Acrescentaremos, android:showAsaction="always".

O showAsaction precisa vir de um outro *name space*. Aqui temos dois *name space*, um 'android' e um 'app'. Usaremos o app. Apagamos o android e digitaremos app no lugar e ":". Teremos app:showAsaction="always".

Ficaremos com:

<menu 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"

tools:context="br.com.alura.agenda.FormularioActivity">

<item android:id="@+id/action\_settings"

android:title="Settings"

android:orderInCategory="100"

app:showAsAction="never" />

<item android:id="@+id/menu\_formulario\_ok"

android:title="Ok"

android:icon="@drawable/ic\_confirmar"

app:showAsaction="always"/>

</menu>

Com isso estamos dizendo que esse item do menu deve ser mostrado com uma das ações da *action bar*.

Vamos rodar para ver como fica.



Agora temos o botão de confirmar, que inclusive já tem uma animação quando clicamos nele!

Falta colocar o comportamento do botão e excluir o botão de "Salvar" que está embaixo do formulário.

Vamos na aba activity\_formulario.xml para tirar o botão antigo. Selecionamos com o mouse o seguinte e depois apagamos:

<Button android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Salvar"

android:id="@+id/formulario\_salvar"/>

Pronto! Se voltarmos agora no FormularioActivity.java não encontraremos mais o R.id.formulario\_salvar.

Só que agora precisamos de uma forma de definir qual o comportamento do clique no menu. Poderíamos definir um Listener, mas existe uma forma mais elegante de fazer isso no *Android*. De novo vamos subscrever um método que já existe, o onOptionsItemSelected e acrescentaremos isso abaixo do seguinte código:

//...

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater;

inflater(R.menu.menuformulario.menu);

return super.onCreateOptionsMenu(menu);

}

Ao digitar o método onOptionsItemSelected o *Android* já vai sinalizar as opções logo abaixo. Escolhemos o método que queremos e é só dar um "Enter". Pronto! Está feito o *override* do método. Teremos:

//...

@Override

public boolean onCreateOptionsMenu(Menu menu) {

MenuInflater inflater = getMenuInflater;

inflater(R.menu.menuformulário.menu);

return super.OnCreateOptionMenu(menu);

}

@Override

public boolean onOptionsItemSelected(MenuItem item) {

return super.onOptionsItemSelected(item);

}

Repare que recebemos nesse método o item que foi selecionado, o MenuItem item. Assim, toda vez que clicarmos no item no menu o *Android* vai evocar esse método na sua activity e vai informar qual foi o item no menu que foi clicado.

Agora, vamos pegar o comportamento do botão salvar, o Toast e selecionar com o mouse até o finish.

@Override

public void onClick(View v) {

Toast.makeText(FormularioActivity.this. "Aluno salvo!", Toast.LENGTH\_SHORT).show();

finish();

}

Podemos recortar o 'Toast', através do "Comand+X" ou "Ctrl+X" e colar no Override do onOptionsItemSelectedcom "Command+V" ou "Ctrl+V".

Temos que tomar cuidado que esse comportamento que acabamos de inserir será extendido para os demais itens do menu. Assim, se tivessemos 50 itens no menu, todos eles seriam englobados nesse comportamento. Como não queremos que isso ocorra, precisamos fazer uma distinção. Para isso, podemos fazer um switch, que acrescentaremos na linha de baixo de public void. Digitaremos switch e entre parênteses colocaremos "item", um ".", e acrescentaremos o Id do item. Teremos, switch (item.getItemId()).

Dependendo do Id que tivermos teremos casos distintos. Na linha de baixo do switch teremos case R.id.menu\_formulario\_ok.

Caso o id seja o menu\_formulário\_ok queremos ter o comportamento que inserimos anteriormente, selecionamos, mais uma vez, o Toast até o finish, damos um "Comand+X" ou "Ctrl+X" e "Comand+V" ou "Ctrl+V" e deslocamento ele para baixo do case que acabamos de introduzir.

Lembre-se de colocar no final o break, seguido de um ;. Caso contrário, ele ficará executando todos os cases que vierem depois dele. Se não quisermos esse comportamento, acrescentamos o breakque ele "tranca" o switch.

@Override

public boolean onOptionsItemSelected(MenuItem item) {

switch (item.getItemId()) {

case R.id.menu\_formulario\_ok;

Toast.makeText(FormularioActivity.this. "Aluno salvo!", Toast.LENGTH\_SHORT).show();

finish();

break;

return super.onOptionsItemSelected(item);

}

Vamos deletar também o botão de salvar, pois ele não existe mais na nossa tela. Deletamos o seguinte:

Button botaoSaLvar = (Button) findViewById(R.id.formulario\_salvar);

botaoSalvar.setOnCIickListener(new View.0nCIickListener() {

@Override

public void onClick(View v) {

}

});

E agora quando clicarmos no botão ele vai chamar o método onOptionsItemSelected passando o item que foi selecionado. Se esse item que foi selecionado for o menu\_formulario\_ok que é o item que definimos nomenu\_formulario.xml, ele mostrará o Toast e termina a activity como queremos.

Vamos salvar e ver como ficou?



Agora, quando quisermos adicionar mais um aluno, vamos preencher o formulário e dar um "Ok", ele vai nos avisar que o aluno foi salvo e vai voltar para lista!