**Adicionando alunos na agenda**

**Adicionando alunos na seção**

**Criando um formulário**

Até o momento conseguimos inserir quatro alunos em nossa lista e, agora, ela possui uma aparência diferenciada, com animação e um *layout* melhorado.

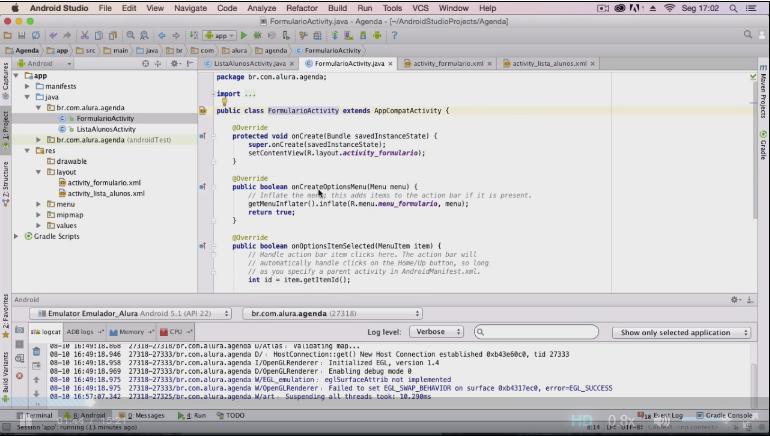
Para continuar inserindo novos nomes de alunos precisamos agora criar um formulário. E para isso criaremos uma nova Activity. Seguindo o caminho "File> New> Activity> Blank Activity".

Lembra que esse é o caminho que fizemos no início do projeto? Pois então, vai abrir uma nova janela e nela várias dados serão pedidos, como o nome da activity, o título e etc.

Nessa janela vamos, primeiro, nomear essa nova Activity de "FormulárioActivity" pois, queremos, justamente, criar um formulário. Repare que o *Layout Name* foi alterado. É por isso que teremos a classe Activity chamadaFormularioActivity.java e uma .xml chamada activity\_formulario.xml. Falta preencher o *Title* da activity, vamos chamá-la de "Formulário". O resto, por enquanto, não precisamos nos preocupar. Agora é só dar um "Finish".

Com isso vai abrir direto o xml da Activity. Para encontrar o FormularioActivity basta ir no lado esquerdo da tela, onde estão as pastas, e clicar duas vezes no "br.com.alura.agenda" que vai abrir a nova Activity.

A nossa tela vai ficar assim:



Como podemos observar pela imagem a cima, a Activity já vai estar "extendendo" a AppCompatActivity, já colocou o método onCreate e também já associou o novo .xml, o activity\_formulario, a Activity.

Como não vamos utilizar vários dos métodos que aparecem no FormularioActivity.java basta apagá-los. Vamos ficar apenas com:

package br.com.alura.agenda;

//import...

public classe FormularioActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate (savedInstanceState);

setContentView(R.layout.activity\_formulario);

}

}

Vamos fazer o mesmo na activity\_formulario.xml. Vamos apagar o TextView e vamos alterar o RelativLayoutpor um LinearLayout, lembre-se que essa classe distribui as informações de maneira linear. Também desejamos dispor as informações na vertical e para isso acrescentamos o atributo orientation e selecionamos "vertical".

Agora, vamos organizar nossa tela e apagar as linhas que não utilizaremos, como, xmlns: tools = "http//schemas.android.com/tools". Também apagaremos as linhas que constam após o atributo orientation.

Ficaremos com:

<LinearLayout xmlns:android="http//schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

</LinearLayout>

Não esqueça que basta colocar o sinal de maior, >, após o vertical para fechar.

Bom, vamos começar a inserir dentro disso os campos do formulário? Por exemplo, podemos ter nomes dos alunos, seus endereços, seus *sites* pessoais, notas e telefones.

Para inserir essas informações vamos precisar de um campo que possa ser editado. Lembra que já vimos alguns? OListView e o TextView. Como queremos editar o conteúdo, vamos usar o campo EditText e dentro dele vamos escrever o atributo hint que indica o que vai constar nesse campo, por exemplos, os nomes dos alunos. Teremos o seguinte:

EditText android:hint"Nome"

Vai ficar assim:

<LinearLayout xmlns:android="http//schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<EditText android:hint"Nome"

</LinearLayout>

***Lembre-se:*** Quando fica em vermelho é por que está faltando a largura e altura obrigatórias. Então, dá um enter e acrescenta layout\_width e layout\_height.

Só que dessa vez queremos que o componente da largura acompanhe apenas a largura da tela, para que ocupe a linha inteira onde nós estaremos acrescentando as informações. Para isso, vamos colocar o match\_parent na largura, ficando assim layout\_width="match\_parent".

Mas, para a altura queremos apenas o espaço suficiente para poder digitar as informações, para isso, lembre-se de usar a wrap\_contentque utiliza apenas aquilo que é suficiente para o que queremos. Digitamos então,layout\_height="wrap\_content" e fechamos esse componente com />. Vai ficar assim:

<LinearLayout xmlns:android="http//schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<EditText android:hint"Nome"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

</LinearLayout>

Feito! Agora, vamos introduzir mais informações. Para isso repetimos o que já fizemos, acrescentando na próxima linha o EditText, com o conte[udo "endereço" e adicionamos também a largura e altura que quisermos, aqui no caso o android:layout\_width="match\_parent", para acompanhar a tela e oandroid:layout\_height="wrap\_content" para utilizar apenas o necessário. Adicionaremos:

<EditText android:hint"Endereço"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

Agora, vamos acrescentar mais dois campos e para economizar tempo, vamos selecionar os EditText que já fizemos e usando os atalhos do *Mac*, o "Comand+C" e "Comand+V" ou do *Windows* "Ctrl+C" e "Ctrl+V" e pronto, só alteramos os campos que queremos, acrescentando "Telefone" e "Site". Teremos:

<LinearLayout xmlns:android="http//schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<EditText android:hint"Nome"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

<EditText android:hint"Endereço"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

<EditText android:hint"Telefone"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

<EditText android:hint"Site"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

</LinearLayout>

Vamos introduzir também uma "nota", só que representaremos as notas com estrelinhas, como aquelas que são usadas para avaliar restaurantes e hotéis. Para isso vamos usar uma função do *Android* que faz isso por nós, oRatingBar.

Na próxima linha, adicionamos o RatingBar e fazemos a mesma coisa que nos outros, é só dar um enter e adicionar o layout\_widht e o layout\_heigth ambos seguidos de wrap\_content, pois queremos que a altura e a largura acompanhem só o necessário para preencher as estrelinhas.

Agora falta definir também o número de estrelas que queremos que apareça, para isso vamos dar mais um enter depois da altura e adicionar um atributo numStars e colocaremos a quantidade de estrelas que quisermos, por exemplo, cinco. Teremos o android:numStars="5".

Uma última propriedade é o max, que designa o máximo de nota que um aluno pode ter. Nosso teto será de dez estrelas. Vamos inserir android:max="10".

Essa propriedade faz uma espécie de escala, onde cinco estrelinhas equivalem a nota máxima, dez, e zero estrelas a nota mais baixa, zero.

Ficaremos com:

//...

<EditText

android:hint"Site"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" />

<RatingBar

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:numStars="5" />

</LinearLayout>

Conferindo, estamos com a Activity e o xml certinhos, agora, para ver como ficou, basta salvar, dar um play para rodar, selecionar o aplicativo que já estamos utilizando e ver o resultado.

Faltam, ainda, umas coisinhas, se não nossa tela continuará apenas com o nome dos alunos.



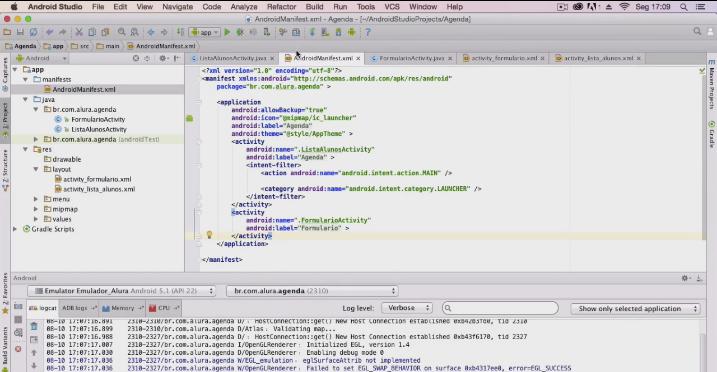
Bom, agora nós temos duas Activity, isto é, temos duas telas. Nosso *Android* terá que escolher qual das telas irá utilizar. O *Android* só pode mostrar uma tela de cada vez.

Precisamos dizer para o *Android* qual das telas queremos que ele abra. Para isso vamos no lado esquerdo nas pastas. Existe uma chamada manifests e dentro dela existe um arquivo chamado AndroidManifest.xml, damos um duplo clique e dentro desse arquivo encontramos várias informações do nosso aplicativo, uma *tag* applicatione nela estão contidas o título da aplicação, o tema e etc.

Dentro da application existem várias *tags*. Todas as novas telas que criamos estão dentro da application, por isso vemos que tem duas *tags* activity.

A primeira activitycorresponde a inicial, a que construímos naquele primeiro momento com a lista de alunos e a segunda que aparece é a mais nova, a que acabamos de criar.

Estamos vendo essa tela:



Para arrumar a tela do celular e para que ela não apareça apenas com o nome dos alunos vamos informar para o*Android* que a nova activity também faz parte da aplicação. E para fazer isso temos que editar o arquivoAndroidManifest.xml. Dentro desse arquivo consigo colocar todas as activity e posso fazer vínculos, por exemplo, a activity que aparece acompanhada do nome ListaAlunosActivity advém, justamente da classeListaAlunosActivity. Como podemos ver:



Logo abaixo, tem a segunda activity que está vinculada a outra classe, o FormularioActivity. Vemos que existe a aplicação, com o nome do aplicativo, o tema e duas activity.

Temos o seguinte:

<application

android:allowBackup="true"

android:icon=@mipmap/ic\_launcher"

android:label="Agenda"

android:theme="@style/AppTheme" >

<activity

android:name=".ListaAlunosActivity"

android:label="@string/app\_name" >

<intent-filter>

<action android name=" android.intent.action.MAIN" />

<category andoid: name="android. intent.category.LAUNCHER" />

</intent-filter>

</activity>

<activity

android:name=".FormularioActivity"

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

</activity>

</application>

O que importa aqui é o intent-filter, que está dentro da activity da lista, mas não está se repetindo naactivitydo formulário.

O intent-filter tem ainda duas coisas dentro, um "android.intent.action.MAIN" e um"android.intent.category.LAUNCHER". Esse filtro quer dizer que qualquer aplicação que possa lançar outras aplicações no *Android* vai encontrar a activityda lista.

Quando olhamos a tela do celular, onde estão dispostos todos os aplicativos do celular, vamos ver a "Agenda" que criamos.



O *Laucher* é o que busca todas as aplicações do celular e ele só consegue encontrar as aplicações devido ao intent-filter dentro da activity.

Se estamos dizendo que a categoria da aplicação é *Launcher* então estou definindo o intent de cima como um*Main*, isto é, que está é a activityprincipal da aplicação. Por isso, quando clicamos no aplicativo da agenda na tela do celular ele sabe que ele tem que abrir a activity que definimos como *main*, no caso a da lista.

Para poder rodar o formulário na tela do celular, vamos cortar toda a intent. Vamos no menu e "Cut" e agora vamos colocar isso dentro da activity nova com o "Ctrl+V" mesmo. Ficaremos com o seguinte:

<activity

android:name=".FormularioActivity"

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

<intent-filter>

<action android:name=" android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

Com isso transformamos o FormularioActivity em nossa lista *main*. Vamos rodar a aplicação para ver o que acontece? Salva, dá um play e escolhe o emulador que já está aberto.

Pronto! Nossa lista está pronta com nome, endereço, telefone, site e as estrelinhas. Inclusive, a gente pode digitar na tela as informações que queremos acrescentar.





Perceba que as estrelas não estão centralizadas. Para resolver isso é rápido. Basta ir na abaactivity\_formulario.xml e introduzir no Ratingbar, aquele lá do início, um alinhamento, o layout\_gravity e quando ele pedir vamos falar que queremos ele no centro, clicando em center.

<RatingBar

android:layout\_width="wrap\_content"

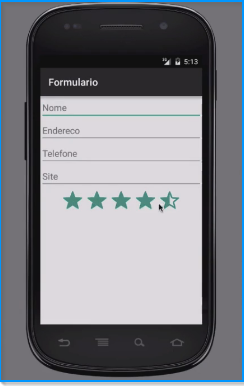
android:layout\_height="wrap\_content"

android:numStars="5"

android:max="10"

android:layout\_gravity"center" />

Vamos rodar de novo e ver o que aconteceu:



Por último, vamos dar um enter e ir para a próxima linha, vamos inserir agora um botão, através do Button. Aqui definimos de novo a largura, que será seguida de match\_parent para ocupar a tela inteira e a altura que será seguida de wrap\_content para ocupar só o necessário. Como criamos o botão vamos também introduzir um texto nele. Para isso inserimor um text e inserimos o texto, no caso "Salvar", ficará assim, androi:text="Salvar". E teremos:

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Salvar"/>

Vamos rodar no emulador de novo:



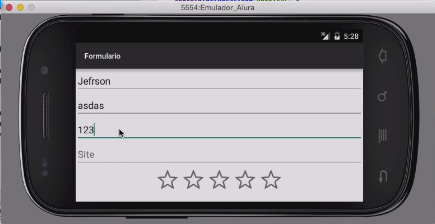
Pronto, agora a gente consegue inserir o nome e dos dados do aluno, dar uma nota e salvar!

**Criando um formulário**

Nossa aplicação agora tem um formulário bonito, com nome, endereço, telefone, site, estrelinhas de avaliação e um botão de salvar e podemos inclusive inserir os dados que quisermos nele! Para ver como ele está no emulador basta salvar, apertar o play e escolher o emulador que já está rodando. Ele está assim:

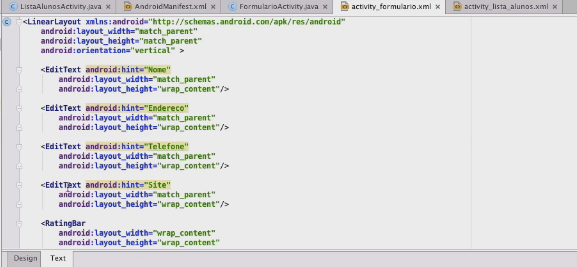


As vezes gostamos de virar o celular para escrever melhor, no emulador também conseguimos fazer isso. Seguindo o atalho do *Mac* "Function+Ctrl+F11" e no *Windows* é "Ctrl+F11" e para voltar a tela ao normal basta usar o mesmo atalho. A tela vai virar assim:



Mas reparem que o botão que inserimos anteriormente, o "Salvar" desapareceu. Vamos arrumar isso! Naactivity\_formulario.xml temos o Button com seus atributos de altura e largura. Normalmente quando algo não aparece na tela do celular, nós rolamos a barra lateral e vamos para baixo, nesse caso nós não conseguimos fazer isso, o que indica um problema.

Vamos voltar na aba activity\_formulario.xml. Encontramos ela assim:



Para resolver o problema basta introduzir uma "rolagem" na tela, ou seja, uma barra que irá rolar e com isso nós poderemos visualizar o que falta.

Voltamos no activity\_formulario.xml. Antes do LinearLayout, fora de tudo, vamos introduzir uma outra *tag* oScrollView que faz com que o conteúdo seja rolado para cima e para baixo. Vamos precisar definir a largura e altura de novo, layout\_widht para largura e layout\_height para altura, que serão seguidos de "match\_parent"pois, queremos que ocupe a tela inteira. Vamos ficar com:

<ScrollView

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

xmlns:android="http://schemas. android.com/apk/res/android" >

//...

</LinearLayout>

</ScrollView>

Lembrem que a 'ScrollView' precisa ficar aberta e não fechada, para englobar tudo o que segue, então, deixamos apenas a >. E lá no final, depois do LinearLayout é que vamos fechar a *tag*.

Reparem que no xmlns :android="http://schemas. android.com/apk/res/android" que aparece tanto na *tag*ScrollView e na LinearLayout. Esse *name* só é necessário na primeira *tag* do arquivo porque é a partir dela que ficarão definidas as demais tags do *Android*.

O *name space* xmlns que vem do endereço http://schemas. android.com/apk/res/android chamamos deandroid então, todos os atributos que seguem, como o android:orientation advém do *name space* que só precisa ser definido uma vez.

Então, pode apagar a linha do *name space* no LinearLayout. Vai ficar:

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

//...

</LinearLayout>

Vamos ajeitar o código pois, ele está um pouco bagunçado. Vamos no Menu "Code> Rearrange Code". Ao fazer isso ele tenta formatar um pouco o código, se não funcionar tem que ser feito na mão. Selecionamos com o mouse deLinearLayout até o último LinearLayout e damos um "Tab", ele vai jogar tudo para frente.

É bom dar uma ajeitada de vez em quando pois, assim, podemos visualizar quem está dentro do quem.

Vamos rodar o emulador para ver se deu certo. Para virar a tela é só dar o "Function+Ctrl+F11" ou "Ctrl+F11" e o que temos é:



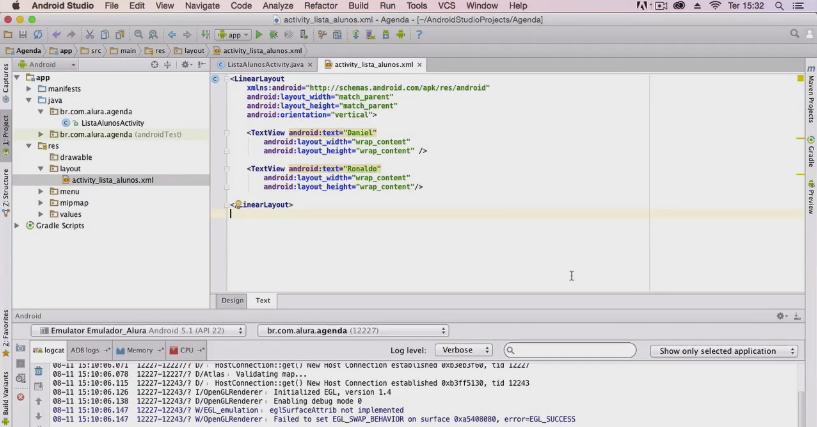
Maravilha! Agora podemos preencher os campos, dar a nota, rolar a tela e salvar!

**Tratando eventos de clique**

O formulário agora tem campos para preencher, estrelinhas para dar notas, um botão de salvar e a possibilidade de rolar a tela se necessário.

O próximo passo é clicar no botão de "Salvar" e acrescentar alguma reação.

Para modificar um comportamento vamos na classe *java* na activity. Isto é, voltamos lá na abaFormularioActivity.java e encontramos:



Primeiro, precisamos do próprio botão para depois construir o *feedback* que ele vai dar.

O setContentView da activityserve para pegar o 'layout' e gerar as instâncias que estão lá na .xml. Para trazer essas instâncias para a activity temos que usar aquele findViewByIdque já conhecemos. Depois do final dosetContentView damos um enter e acrescentamos esse Id. Acrescentando o findViewById

@0verride

protected void onCreate(Bundle savedInstanceState) {

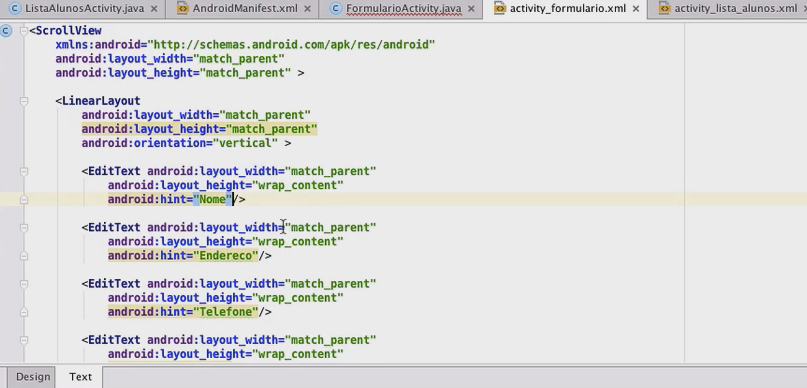
super.onCreate (savedInstanceState);

setContentView(R.layout.activity\_formularios);

findViewById()

}

Esse é um ótimo momento para voltar na activity\_formulario.xml e colocarmos um Id nos campos, pois precisaremos mais para frente deles. Vamos voltar rápido na aba activity\_formulario.xml. Encontramos assim:



Vamos começar pelo primeiro EditText com o campo "Nome" acrescentando um id. Na próxima linha adicionamos o id, seguido de um @, +id, / e para organizar melhor esses id podemos acrescentar onde ele está, no caso, está no arquivo activity\_formulario então nomearemos "formulário" e como ele traz o campo "nome" também acrescentaremos isso. Assim temos,

android:id="@=id/formulário\_nome"

Fazemos isso para que nos mesmos não confundamos e também para deixar claro para a classe R.

E ficamos com o EditTextassim:

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

android:layout\_height="wrap\_content”

android:hint="Nome"

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

Esse procedimento vai ser repetido nos outros campos do Editext, inclusive com a RatingBar e, por fim, faremos o mesmo com o Button que é o que nos motivou a introduzir os Id. Ficaremos com isso:

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

android:layout\_height="wrap\_content”

android:hint="Endereco"

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

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

android:layout\_height="wrap\_content”

android:hint="Telefone"

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

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

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

android:layout\_height="wrap\_content”

android:hint="Telefone"

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

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

<RatingBar

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center"

android:max="10“

android:numStars="5"

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

<Button

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Salvar"/>

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

Agora temos todos os componentes identificados! Vamos voltar para a aba FormulárioActivity e agora sim conseguimos buscar o que queremos.

No caso queremos buscar o botão, então, vamos digitar o findViewById(R.id.formulario\_salvar). Não esqueça de colocar o ;no final. Bom, se fizemos aqui o findView temos que guardar o que queremos e para isso acrescentaremos na frente de tudo o Button que vamos chamar de "botão salvar" para ficar bem claro o que é. Teremenos

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

E o código ficará assim:

@0verride

protected void onCreate(Bundle savedInstanceState) {

super.onCreate (savedInstanceState);

setContentView(R.layout.activity\_formulario);

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

}

Se o 'Button' ficar em vermelho é por que falta importar, então, dá um "Alt+Enter" e deu. Mas, provalvelmente, vai continuar em vermelho pois, o findView devolve uma View. Portanto, temos que fazer o *cast*. Para isso podermos usar o atalho "Alt+Enter" em cima do que ainda estiver em vermelho e ali vai sugerir automaticamente "Cast toandroid.widget.Button". É só clicar e pronto, agora o botão tem uma referência.

Olhando no emulador vemos que o *Android* já traz uma animação automática, como percebemos pela imagem abaixo.

Agora, vamos tratar o clique do botão! Isto é, dar alguma reposta quando o botão for clicado, uma espécie de*feedback*.

Para isso vamos utilizar um Listener que, explicando de maneira grosseira, é como se fosse um "ouvinte". Com isso o *Android* vai avisar a quem estiver "escutando" que algo foi salvo.

Para cadastrar um *listener* nesse botão vamos dar um enter e na linha de baixo adicionaremos Click e o Listenerpois, é justamente o que vai nos avisar que aconteceu um clique. Teremos botaoSalvar.setOnClickListener

Depois que acrescentamos isso ele vai abrir chaves [] e podemos usar o atalho "Comand+P" ou "Ctrl+P". Vamos utilizar aqui a ideia de classe anônima e adicionar new\_onClickListener. O próprio *Android Studio* já vai sugerir com o que temos que completar, basta dar um "Enter" e o código vai ser gerado automaticamente. Só para fechar a instrução do botaoSalvar colocaremos o ; no final. Vai ficar assim:

`botaoSalvar.setOnClickListener[new\_onClickListener]`

Precisamos, agora, completar o método onClick, que será o método executado quando o botão for clicado. E dentro do public void preenchemos com a ação que queremos que se encadeie quando o botão for clicado. O que vamos inserir é uma mensagem que aparece e desaparece sempre que o botão é clicado, como aqueles avisos "seu e-mail foi enviado".

No *Android* o nome disso é *Toast* como torrada em inglês. Vamos criar uma classe Toast e o método makeText que por sua vez vai pedir alguns parâmetros, primeiro, um contexto.

O contexto serve para dizer ao *Android* quem somos, isto é, fornecer nossa identificação. Poderíamos preencher com o this, mas ele estaria se referenciado a nova classe que criamos, a classe anônima. No contexto, entretanto, a referência que queremos passar é da FormularioActivity. Então, vamos adicionar junto ao thisFormularioActivity.this Agora temos o contexto:

Toast.makeText(FormularioActivity.this)

Mas, além disso também queremos acrescentar uma mensagem então, ainda dentro do parentêses, colocamos uma vírgula, damos um espaço, e adicionamos o texto que queremos que apareça quando o botão for clicado. Vamos escrever "Botao clicado!".

Precisamos dizer também qual será a duração da mensagem, para isso usaremos, mais uma vírgula, um espaço e escreveremos a constante Toast. Quando você tiver terminado de digitar vão aparecer duas sugestões automática, a LENGHT\_SHORTe a LENGTH\_LONG. Vamos escolher a LENGHT\_SHORT para que a duração seja curta.

Teremos:

@Override

public void onClick(View v) {

Toast.makeText(FormularioActivity.this, "Botao clicado!", Toast.LENGHT\_SHORT);

Bom, até aqui pedimos que a instância Toast fosse criada, agora falta só pedir que ela seja mostrada também, então acrescentamos o "show" após o parentêses, na mesma linha, .show() e terminamos com o ponto e vírgula.

Vamos ter isso:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate (savedInstanceState);

setContentView(R.layout.activity\_formularios);

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

botaoSalvar.setOnClickListener[new\_OnClickListener]() {

@0verride

public void onClick (View v) {

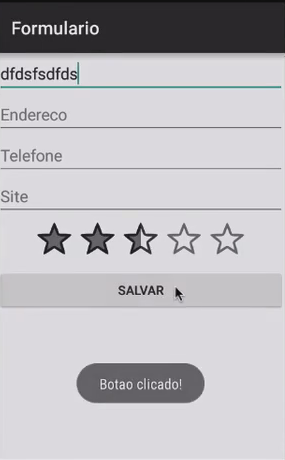
Toast.makeText(FormularioActivity.this, "Botao clicado!", Toast.LENGHT\_SHORT).show();

}

});

}

Vamos rodar o emulador para ver como ficou, para isso, salvar, play, seleciona o que já está rodando e teremos o seguinte:



É isso aí, agora temos também um feedback inclusive do botão!