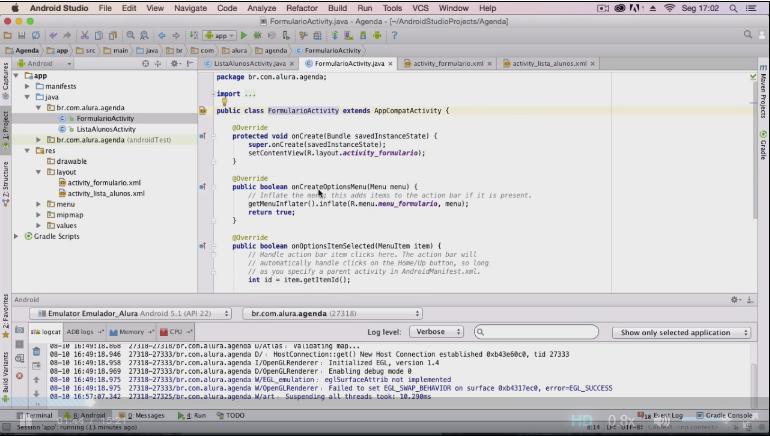
# **Adicionar alunos na agenda**

## **Adicionar alunos na agenda**

Agora, nossa lista possui quatro novos alunos, uma aparência diferenciada, uma animação e um *layout* melhorado. Para continuar inserindo novos alunos é preciso um novo formulário, ou seja, uma nova Activity:

"File> New> Activity> Blank Activity".

Seguindo esse caminho abrirá uma nova janela e nela vários dados serão pedidos como o nome da activity, o título, etc. Primeiro, vamos nomear essa nova Activity de "FormulárioActivity" e ao fazer isso repare que o *Layout Name* também é alterado. Dessa forma, teremos uma classe Activity e uma xml chamadas, respectivamente, de FormularioActivity.java e activity\_formulario.xml. Preenchemos o campo *Title* com "Formulário". Com o restante não nos preocupamos nesse momento, portanto, é só dar "Finish". Abrirá a xml da Activity e para encontrar o FormularioActivity basta ir no lado esquerdo da tela, onde estão as pastas, e clicar duas vezes no "br.com.alura.agenda", escolher o arquivo e a nova Activity será mostrada. Nossa tela ficará assim:



Observe esse arquivo, como não vamos utilizar vários dos métodos que aparecem nele basta apagá-los. Ficaremos 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 organizar também a activity\_formulario.xml. Podemos apagar o TextView e alterar o RelativeLayout por um LinearLayout, lembre-se que essa classe distribui as informações de maneira linear e como desejamos dispor as informações na vertical vamos acrescentar também o atributo orientation="vertical". Além disso, apagaremos 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, nomes, endereços, *sites*pessoais, notas e telefones. Para inserir essas informações vamos precisar de um campo que possa ser editado. Lembra que já vimos alguns? O ListView e o TextView! Como queremos editar o conteúdo, vamos usar o campo EditText e dentro dele o atributo hint, que indica o que vai constar nesse campo, por exemplo, nomes dos alunos:

EditText android:hint"Nome"

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">  
  
 <EditText android:hint"Nome"  
  
</LinearLayout>

***Lembre-se:*** Quando alguma palavra ficar em vermelho é porque falta a largura e altura obrigatórias, portanto, é preciso acrescentar layout\_width e layout\_height.

Mas, dessa vez queremos que o componente da largura acompanhe a mesma medida da tela. Para isso, devemos acrescentar match\_parent na largura:

layout\_width="match\_parent".

Porém, a altura deve ocupar apenas o espaço suficiente para que caibam as informações, portanto, usaremos a wrap\_contentque utiliza apenas o espaço necessário. Digitamos layout\_height="wrap\_content" :

<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! Podemos repetir o que fizemos acrescentando na próxima linha o EditText, com o conteúdo "endereço" e adicionamos também a largura e altura. No caso, android:layout\_width="match\_parent", para acompanhar a tela, e android:layout\_height="wrap\_content"para utilizar apenas o espaço necessário. Adicionaremos:

<EditText android:hint="Endereço"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />

Falta acrescentar mais dois campos e para economizar tempo vamos copiar e colar os EditText que já fizemos. Podemos usar os atalhos do *Mac*, o "Comand + C" e "Comand + V" ou do *Windows* "Ctrl+C" e "Ctrl+V" . Colando isso, falta alterar os campos com "Telefone" e "Site". Teremos:

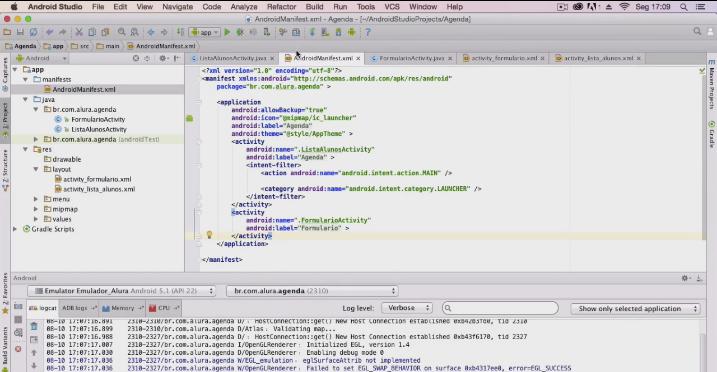
<LinearLayout xmlns:android="http//schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"   
 android:layout\_height="match\_parent"   
 android:orientation="vertical">  
  
 <EditText android:hint="Nome"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content" />  
  
 <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 as notas serão representadas com estrelinhas, como as que são usadas para avaliar restaurantes e hotéis. Podemos usar uma função do *Android* que faz isso por nós, a RatingBar. Na próxima linha, adicionamos RatingBar, damos um enter e adicionamos o layout\_width e layout\_heigthambos seguidos de wrap\_content, pois queremos que a altura e a largura sejam equivalentes apenas ao que é necessário para preencher as estrelinhas. Agora, falta definir também o número de estrelas que queremos que apareçam, para isso, adicionamos um atributo numStars e colocamos a quantidade de estrelas que quisermos, no caso, 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 dez estrelas. Vamos inserir android:max="10". 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>

Agora nossa Activity e o xml estão certinhas! Bom, como passamos a ter duas Activities, isto é, duas telas e nosso *Android só pode mostrar uma de cada vez... É preciso dizer ao* Android\* qual das telas deve ser aberta!

Existe uma pasta chamada manifest e dentro dela um arquivo de nome AndroidManifest.xml que contêm diversas informações acerca do nosso aplicativo, por exemplo, uma *tag* application , o título da aplicação, o tema, etc. Todas as novas telas que criamos estão dentro dessa application, por isso vemos que duas *tags* activity. A primeira activitycorresponde a tela inicial, a que construímos naquele primeiro momento com a lista de alunos e a segunda equivale a que acabamos de criar. Estamos vendo o seguinte:



É preciso informar ao *Android* que a nova activity também faz parte da aplicação. Para fazer isso é necessário editar o arquivo AndroidManifest.xml. Dentro dele conseguimos colocar todas as activity e fazer vínculos entre elas, por exemplo, a activity que aparece acompanhada do nome ListaAlunosActivity advém, justamente da classe ListaAlunosActivity. Como podemos observar:



Repare no código abaixo, a segunda activity está vinculada a outra classe, a FormularioActivity. Vemos que existe a aplicação, com o nome do aplicativo, o tema e duas activities. 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 se repete na activitydo formulário. O intent-filter tem ainda dois itens dentro, um "android.intent.action.MAIN" e um "android.intent.category.LAUNCHER". Esse filtro significa 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, vemos a "Agenda" que criamos.



O *Laucher* busca todas as aplicações do celular e ele só consegue encontrá-las devido ao intent-filter que fica na activity. Se estamos dizendo que a categoria da aplicação é *Launcher*, então, definimos o intent de cima como um *Main*, isto é, como a activityprincipal da aplicação. Por isso, quando clicamos no aplicativo da agenda na tela do celular ele sabe que deve abrir a activity que definimos como *main*, no caso a lista. Para rodar o formulário na tela do celular, vamos cortar ("Ctrl + C") toda a intent e colar ("Ctrl+V" ) ela na activity nova. 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 principal, nossa *Main*. Vamos rodar a aplicação para ver o que acontece? Observe nas imagens abaixo, nossa lista está pronta com nome, endereço, telefone, site e as estrelinhas. Inclusive, podemos digitar na tela as informações que queremos acrescentar:

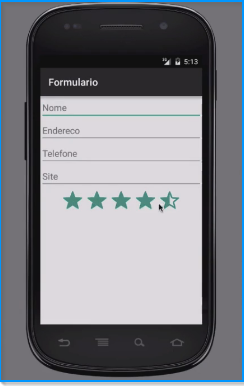




Perceba que as estrelas não estão centralizadas. Para resolver isso é rápido. Basta ir na abaactivity\_formulario.xml e introduzir no Ratingbar um alinhamento, o layout\_gravity e center.

<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 inserir um botão através do Button. É preciso definir 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. Criado o botão é preciso introduzir um texto para ele. Adicionamos um text, no caso "Salvar" e ficaremos comandroi:text="Salvar":

<Button   
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:text="Salvar"/>

Vamos rodar no emulador de novo:



Pronto, agora conseguimos inserir o nome, os dados do aluno, dar uma nota e Salvar! :)