01

Adicionar alunos na agenda

Transcrição

A lista agora 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".

> Nas versões mais recentes do Android Studio, não existe mais a opção Blank Activity então você pode utilizar a Empty Activity no lugar dela.

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 "FormularioActivity" 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, clicar duas vezes no "br.com.alura.agenda", escolher o arquivo e a nova `Activity` será mostrada.

Como não vamos utilizar vários dos métodos do arquivo, 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);

    }

}
```

Também vamos organizar 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, mas desejamos dispor as informações na vertical, então, acrescentamos o atributo `orientation="vertical"` . Além disso, apagaremos as linhas não utilizadas, como: `xmlns: tools = "http://schemas.android.com/tools"` . Também removeremos 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.

Vamos começar a inserir os campos do formulário? Vamos acrescentar Nome, Endereços, *Sites* pessoais, Notas e Telefones. Para inserir as informações precisaremos 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 inserimos o atributo `hint`, que indica o que constará nesse campo, no caso, 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 uma palavra fica sublinhada em vermelho é porque falta indicar largura e altura obrigatórias. É preciso acrescentar `layout_width` e `layout_height`.

Dessa vez queremos que o componente da largura acompanhe a mesma medida da tela. Para isso, acrescentaremos `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_content` que 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 outros dois campos e para economizar tempo vamos copiar e colar os `EditText` que 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 "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", cujo valor será representado por 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_heigth` ambos seguidos de `wrap_content` , pois queremos que a altura e a largura sejam equivalentes ao espaço necessário para preencher as estrelinhas. Falta definir o número de estrelas que queremos que apareçam, para isso, adicionamos um atributo `numStars` e colocamos a quantidade de estrelas, 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" />
```

`Activity` e `xml` estão corretos!

Passamos a ter duas `Activities`, isto é, duas telas. Mas, o Android só pode mostrar uma de cada vez, portanto, 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 aplicativo, por exemplo, uma *tag* `application`, o título da aplicação, o tema, etc. Todas as novas telas que criarmos estarão dentro dessa `application`, por isso vemos duas *tags* `activity`. A primeira `activity` corresponde a tela inicial, a que construímos no primeiro momento com a lista de alunos e a segunda é a que acabamos de criar.

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`.

Repare no código abaixo, a segunda `activity` está vinculada a outra classe, a `FormularioActivity`. Pela leitura do código sabemos que existe a aplicação, 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 android: name="android. intent.category.LAUNCHER" />
            </intent-filter>
      </activity>
      <activity
                android:name=".FormularioActivity"
                     android:label="@string/title_activity_formulario" >
            </activity>
    </application>
```

O que importa é que o `intent-filter`, que está dentro da `activity` da lista, não se repete na `activity` do 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 `activity` da 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 `activity` principal 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*. Para rodar o formulário na tela do celular, vamos cortar ("Ctrl + C") toda a `intent` e colá-la ("Ctrl+V") na `activity` nova. Ficaremos com o seguinte:

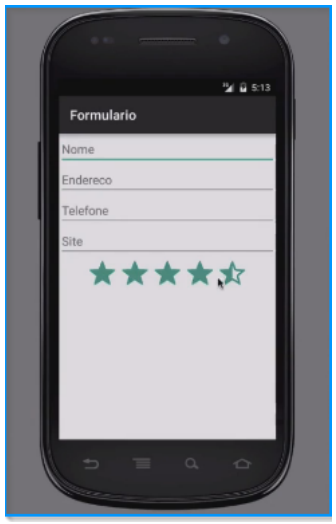
```
<activity
        android:name=".FormularioActivity"
        android:label="@string/title_activity_formulario" >
        <intent-filter>
            <action android:name=" android.intent.action.MAIN" />

            <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 as imagens abaixo, a 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 aba `activity_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 por meio do `Button` . Definiremos a largura para ocupar a tela inteira, `match_parent` , e a altura que será `wrap_content` , para ocupar apenas o necessário. Criado o botão é preciso introduzir o texto "Salvar" nele. Adicionamos o `text` e ficaremos com `androi: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! :)