## **Melhorando a lista**

Até agora nossa lista possui dois alunos "Ronaldo" e "Daniel"! Relembrando que para inseri-los fizemos alterações no .xml. Agora, para seguir acrescentando outros nomes devemos retornar ao mesmo local, isto é, no .xml. Mas, existe um caminho mais fácil para isso, selecione com o mouse as linhas do TextView do "Ronaldo" até o fim:

<TextView android:text="Ronaldo"   
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content" />

Podemos seguir o caminho: "Menu > Edit > Paste" e as linhas selecionadas com o mouse serão reproduzidas na tela ou utilizamos os atalhos "Ctrl+C" e "Ctrl+V" no Windows ou "Comand C" e "Comand V" no *Mac* para copiar e colar, respectivamente. Utilizamos a linha que foi repetida e inserimos os novos nomes na lista. Basta modificar o nome "Ronaldo" por algum outro da nossa escolha, por exemplo, "Jeferson":

<TextView android:text="Jeferson"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />

Agora "Jeferson" é o terceiro aluno!

Para acrescentar outros alunos o procedimento é o mesmo. Selecionamos as linhas de um TextView até o final e, "Menu > Edit > Paste", ou os atalhos. Após coladas as linhas, trocamos o nome do aluno pelo que quisermos, por exemplo, por "Felipe".

<TextView android:text="Felipe"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />

Agora temos quatro novos alunos!

<TextView android:text="Daniel"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />  
  
<TextView android:text="Ronaldo"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />  
  
<TextView android:text="Jeferson"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />  
  
<TextView android:text="Felipe"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />

***Lembrando***: Para visualizar a tela basta salvar, apertar o *play*, escolher o emulador que já está rodando e aguardar iniciar. Para trazer o emulador para frente existe o atalho "Alt + Tab".

Todos os alunos novos que inserimos vão aparecer na tela do celular:



Pronto, agora que aprendemos a introduzir os nomes de maneira simples. Vamos introduzir algumas novidades!?

Vamos trazer esses alunos a partir de um arquivo, ou banco de dados, ou de um *webservice* ou de outro lugar que não o xml.?! Vamos começar pelo arquivo! Essa alteração será feita no Activity. Afinal, é na Activityque fica o comportamento da tela e onde conseguimos escrever o código java. Bom, queremos colocar um arquivo do xml.na Activity e para isso vamos na aba ListaALunosActivity.java.

Existe uma forma simples de fazer esse procedimento. Para isso, o que vamos fazer é dizer para o *Android* que queremos exibir na tela uma lista e no código adicionaremos os elementos dessa lista. Os componentes que desejamos mostrar na tela estão no .xml mas, de onde eles serão trazidos, isto é, a parte da lógica deverá ficar na Activity. Faz sentido, então, que os alunos que estão no .xml, ou seja, os Views sejam trazidos de alguma outra fonte, arquivo, webservice ou banco de dados na Activity. O bom disso é que não será preciso alterar o xml. e também não quebraremos a separação já feita no começo, aquela do comportamento na Activity *vs.*conteúdo na xml.

Para fazer isso apagaremos todos aqueles TextView da xml. Isto é, apagaremos isso:

<TextView android:text="Daniel"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />  
  
<TextView android:text="Ronaldo"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />  
  
<TextView android:text="Jeferson"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />  
  
<TextView android:text="Felipe"   
 android:layout\_width="wrap\_content"   
 android:layout\_height="wrap\_content" />

Apagaremos os View para justamente não precisar recriá-los todos novamente. Ficaremos apenas com isso na .xml:

<LinearLayout  
 xmlms:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:orientation="vertical">

Depois de apagar os TextView na .xml vamos acrescentar um componente diferente do *Android*, a ListView. Esse componente representa uma lista. Damos um *Enter* na última linha da LinearLayout e ao digitar as primeiras letras da palavra ListView, ela já irá aparecer, agora é só selecionar e dar *enter*.

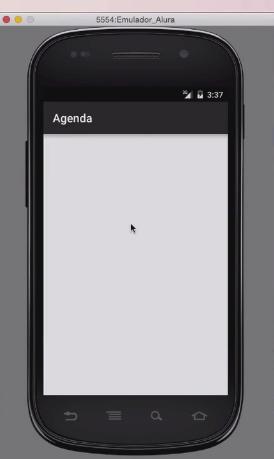
***Lembrando:*** Qualquer tipo de View criada na .xml deve ser seguida pela altura e largura. Mas, ao em vez inserir o wrap\_content, que faz com que nossa lista acompanhe as medidas da tela, vamos pedir para que ela ocupe toda a tela inicial. Inserimos, então, um match\_parent após o layout\_widht e o layout\_height.

Vamos ter:

<LinearLayout  
 xmlms: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"/>   
  
</LinearLayout>

***Relembrando:*** O match\_parent indica que a largura e a altura serão iguais a do "pai", nesse caso as medidas da ListView serão iguais a do LinearLayout, o pai. Esta última, é a classe raiz, que por sua vez não acompanha as medidas de ninguém, ela acompanha as medidas da tela inteira.

Na lista é apenas isso que precisamos especificar. Não esqueça de fechar a *tag* com o />! Só que, se rodarmos o emulador, veremos que a nossa lista estará vazia. Mesmo que tenhamos introduzido uma lista no formulário falta nela o seu conteúdo e, por isso, ela aparece vazia.



Para alterar o conteúdo da lista temos que voltar na Activity e escrever um código para populá-la. Vamos ficar, agora, na aba da ListaAlunosActivity.java e nela encontraremos:

package br.com.alura.agenda;  
  
import ...  
  
public class ListaAlunosActivity extends AppCompatActivity {  
  
 @0verride  
 protected void onCreate(Bundle saverInstanceStatel {   
 super.onCreate (savedInstanceState);  
 setContentView(R.layout.activity\_lista\_alunos);  
  
 }  
}

Poderíamos, primeiro, abrir um arquivo ou um banco de dados, mas por enquanto esse não é o nosso foco. Para introduzir um conteúdo na Activity, isto é, uma "lista de alunos", vamos declarar apenas uma *Array* simples, isto é, uma String e dentro dela vamos inserir os alunos. Então, temos a seguinte *Array*: String[] = {}. Dentro das chaves inserimos os nomes dos alunos. Não podemos esquecer de acrescentar também o nome da variável, no nosso caso, "alunos". Vamos ter o seguinte:

package br.com.alura.agenda;  
  
import ...  
  
public class ListaAlunosActivity extends AppCompatActivity {  
  
 @0verride  
 protected void onCreate(Bundle saverInstanceStatel {   
 super.onCreate (savedInstanceState);  
 setContentView(R.layout.activity\_lista\_alunos);  
  
 String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};  
  
 }  
}

Queremos também trazer a lista que está no .xml para a Activity. Abaixo do que acabamos de escrever na Activity, na próxima linha, acrescentamos listadealuno, um ponto e add. Isto é: listaAluno.add(). Ficamos com:

protected void onCreate(Bundle saverInstanceStatel {   
 super.onCreate (savedInstanceState) {  
 setContentView(R.layout.activity\_lista\_alunos);  
  
 String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};  
 listaAluno.add()  
  
 }

O *add*, de adicionar vem justamente para acrescentar algo, isto é, cada um dos alunos que estão descritos na *Array*. Precisamos também referenciar essa listaAlunos.add(). Para isso vamos usar um método simples da AppCompatActivity. Vamos apagar por um momento o listaAlunos.add() e escrever a palavra *find*. Logo abaixo já vai completar o que queremos, que é findViewById.

protected void onCreate(Bundle saverInstanceStatel {   
 super.onCreate (savedInstanceState) {  
 setContentView(R.layout.activity\_lista\_alunos);  
  
 String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};  
 findViewById  
  
 }

O que estamos fazendo é localizando uma View através da especificação de um Id. Precisamos dizer para o *Android* qual a Viewque gostaríamos de localizar. Para localizar a View voltamos ao .xml e o encontramos assim:

<LinearLayout  
 xmlms: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"/>   
  
</LinearLayout>

Vamos especificar na .xml uma identificação para encontrarmos a View que queremos. Para isso iremos acrescentar um atributo na ListView, o id.

O id é seguido de um @ que serve para referenciar algum recurso da pasta "res", acrescentamos na sequencia o sinal de "+" que é usado para falar que estamos criando um recurso, ou seja, uma identificação para a lista, também digitamos logo em seguida o tipo de recurso que estamos manipulando, nesse caso, um id. Depois do id colocamos, ainda, uma barra só para separar essa chave do id que especificamos na frente. Depois da barra inserimos o nome que queremos dar a ListView, no caso, "lista\_alunos". O id é id="@+id/lista\_alunos"/>. E 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"/>  
  
  
</LinearLayout>

Voltando na Activity vamos, finalmente, referenciar nossa lista no findViewById que havíamos digitado. Ele vai pedir o id que definimos no .xml. Podemos utilizar a classe R e especificaremos apenas o id que queremos, que é lista\_alunos. Agora a chamada do nosso método está completa:findViewById(R.id.lista\_alunos).

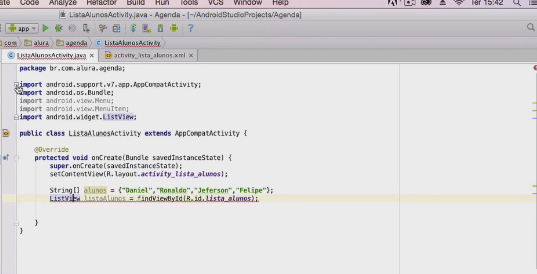
O todo ficará assim:

protected void onCreate(Bundle saverInstanceStatel {   
 super.onCreate (savedInstanceState);  
 setContentView(R.layout.activity\_lista\_alunos);  
  
 String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};  
 findViewById(R.id.lista\_alunos);  
  
 }

Quando inserimos o findViewById ele vai devolver uma instância da View que foi gerado no SetContentView . Agora falta guardar o findViewById. Vamos inserir na Activity uma variável, ListView e vamos nomeá-la ListaAluno, assim, ListView ListaAluno vai ficar da seguinte forma:

protected void onCreate(Bundle saverInstanceStatel {   
 super.onCreate (savedInstanceState);  
 setContentView(R.layout.activity\_lista\_alunos);  
  
 String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};  
 ListView ListaAlunos = findViewById(R.id.lista\_alunos);  
  
 }

A ListView vai ser sublinhada de vermelho pois, necessitamos fazer algumas modificações, vamos fazer um *import*. Vai aparecer em cima da palavra ListView um balão em azul, basta pressionar "Alt+Enter" e ele fará o *import* automaticamente. Para ver o *import* basta clicar no mais, como mostra a figura:



Como percebemos pela imagem, a ListView ainda está em vermelho. Para resolver isso basta converter nossa referência de View para ListView através de um *casting*. Vamos introduzir entre parênteses que iremos converter essa referência, assim, (ListView). Ficamos com:

protected void onCreate(Bundle saverInstanceStatel {   
 super.onCreate (savedInstanceState);  
 setContentView(R.layout.activity\_lista\_alunos);  
  
 String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"} ;  
 ListView ListaAlunos = (ListView) findViewById(R.id.lista\_alunos);  
  
 }

Agora temos,

* uma lista de alunos: String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};
* o componente, a View de alunos: ListView ListaAlunos = (ListView) findViewById(R.id.lista\_alunos);

Falta inserir os alunos dentro de nossa lista. Para fazer isso precisamos converter os alunos String para View. A classe ArrayAdapter faz exatamente isso! Para instanciar ela , pulamos para a próxima linha e inserimos new ArrayAdapter <>:

protected void onCreate(Bundle saverInstanceStatel {   
 super.onCreate (savedInstanceState);  
 setContentView(R.layout.activity\_lista\_alunos);  
  
 String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"} ;  
 ListView ListaAlunos = (ListView) findViewById(R.id.lista\_alunos);  
 new ArrayAdapter <>  
  
 }

Entre os argumentos <> do *generics* vamos preencher com a String, pois é isso que desejamos importar. Então, teremos: new ArrayAdapter<String>.

protected void onCreate(Bundle saverInstanceStatel {   
 super.onCreate (savedInstanceState);  
 setContentView(R.layout.activity\_lista\_alunos);  
  
 String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"} ;  
 ListView ListaAlunos = (ListView) findViewById(R.id.lista\_alunos);  
 new ArrayAdapter <String>;  
  
 }

Se aparecer um balão azul isso significa que a nossa classe ainda não foi importada, basta dar "Alt+Enter" e vai trazer logo acima a classe ArrayAdapter. Agora, falta especificar quais são os parâmetros que precisamos para a Array Adapter. Ele vai pedir o contexto e também o *layout* que iremos utilizar. O contexto serve para nos identificar, então, o que faremos é passar uma referência para a própria Activity. Para passar uma referência basta inserir o this. Então, acrescentamos ao new ArrayAdapter<String>, entre parênteses, o this. Ficaremos com new ArrayAdapter<String>(this, ).

A segunda coisa que ele vai pedir é o *Layout*. Aqui vamos utilizar um *layout* padrão e para isso vamos utilizar a classe R do próprio *Android*. Escrevendo, android.R.layout.simple\_list\_item\_1, que é simplesmente um TextViewque auxilia a mostrar o nome do aluno. Teremos:

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

Falta dizer quais são os dados que queremos converter em 'Views', que nesse caso são os alunos. Assim, só completamos depois da palavra "item", com uma vírgula, um espaço e a palavra "alunos", que é a fonte de dados. Teremos:

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

Agora a ArrayAdapter está instanciada e para que a referência não se perca vamos digitar na frente dela uma variável do tipo ArrayAdapter de String, que nós chamaremos de adapter. Com isso acrescentaremos o seguinte, ArrayAdapter<String> adapter, e, por fim, teremos:

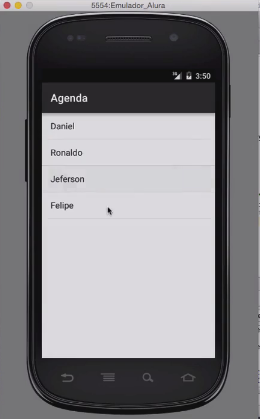
ArrayAdapter<String> adapter = new ArrayAdapter <String> (this, android.R.layout.simple\_list\_item\_1, alunos);

Retomando, temos dois objetos separados, uma Lista de alunos, que é a ListView e também, um adapterque irá converter os alunos que são String em View para serem introduzidos na lista. Precisamos ,agora, juntar as duas coisas! Atenção para as cores, se só a variável está cinza é por que ainda não a utilizamos em nenhum lugar. Mas, agora, nós vamos utilizar!

Vamos dizer para a própria lista para utilizar o adapter como sendo seu adapter. Isso pode soar um pouco estranho, mas na verdade o que queremos dizer é que a lista deverá pedir ao adapter para converter os alunos em View. Assim, na linha de baixo do ArrayAdapter, digitamos ListaAlunos.setAdapter(adapter);:

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

Você pode reparar que as palavras adapter terão mudado de cor, elas estarão com a cor preta para demostrar que estão sendo utilizadas. Vamos rodar o emulador para ver o que acontece:



Temos uma lista bem mais completa, com todos os nomes dos alunos, com um comportamento diferente, espaçamento e até um animação. :)