**Novo projeto de uma Agenda**

**Novo projeto de uma Agenda**

**Um novo projeto**

Para desenvolver um aplicativo para **Android** no ***Android Studio*** é necessário criar um novo projeto.

Então, clica-se em "> Start a new Android Studio project"

Troca-se o nome da aplicação pela que se deseja criar, por exemplo, uma agenda de alunos chama-se simplesmente de Agenda. No campo domínio da empresa se insere a convenção do **Java**, alura.com.br. Logo abaixo vai mostrar onde está sendo gerado o projeto. Após inserir as informações clica-se em "Next".

Em seguida, será pedido qual a versão mínima que a aplicação vai rodar no **Android**. Por exemplo, aqui será sugerido que seja o *Android 4.0*. Pode-se visualizar a quantidade de Androids atingidos por essa opção, no caso do**Android 4.0** é 94% do mercado.

A seguir será pedido o que se quer colocar no projeto. Deve ser selecionada a opção "Blank Activity". E no momento é apenas isso que importa.

Após esse procedimento, vai aparecer um campo pedindo o nome da *Activity*. Troca-se o *Main* pelo que se deseja criar, no caso uma lista de alunos. Assim pode-se inserir no lugar do *Main*, por exemplo: "listadealunosActivity".

Os outros valores nesse momento não precisam ser preenchidos. Clica-se em "Finish".

Irá demorar um pouco, mas o projeto será gerado. Então, seleciona-se "Open".

Será mostrada a relação de pastas. Após terminar de renderizar deve-se primeiro rodar a aplicação criada para visualizá-la. E para executar uma aplicação vai-se no menu e seleciona: "Run > Run app".

Clicando nessa opção ele termina de fazer o *build* do projeto, o que demora um pouco, pois está gerando o código final da aplicação.

Após aguardar um momento vai ser aberta uma nova janela e nela deve ser selecionada a opção de onde se deseja rodar a aplicação. No caso, se houvesse um aplicativo real ele poderia ser escolhido para teste. Aqui, vamos selecionar um emulador alura. Clica-se em "OK".

Na tela aparecerá uma imagem da própria tela de celular. Simulando uma tela de celular de verdade, onde inclusive se pode aumentar volume, som e etc.

***Dica:*** Uma vez iniciado o emulador é melhor deixá-lo aberto e minimizado, pois ele demora para iniciar. Com isso não será necessário esperar toda a vez que se quiser ver a aplicação.

Após fazer o *build* irá aparecer a tela do celular e como ele funciona como um celular de verdade, é necessário destravá-lo. Isso se faz arrastando o cadeado para cima.

Aparecerá a aplicação e na tela do celular estará escrito Agenda e *Hello word!*. Mas, não é este o tipo de texto que queremos, portanto, este último será modificado.

Após testar a aplicação e vê-la rodando volta-se ao **Android Studio** para ver o que foi gerado.

Fecham-se os arquivos já abertos e fica só o projeto. No lado esquerdo da tela está o *app* que ele criou e abrindo esse projeto de *app* verificam-se três pastas: "java", "res" e "manifests".

Dá-se um duplo clique na pasta java e dentro dela abrem-se dois pacotes. Clica-se duas vezes nesse pacote br.alura.agenda e irá aparecer o ListaAlunosActivity que é a nossa Classe.

Clica-se duas vezes na ListaAlunosActivitye aparecerá a Classe da aplicação e códigos já gerados.

A primeira coisa quando se quer desenvolver uma aplicação para desktop, windows, mac e outros, é ter uma janela para poder mostrar ao usuário, ou seja, um *container*. A janela no **Android** não é diferente.

No **Android** é necessário uma tela, a qual é chamada de *Activity*. É importante lembrar que toda vez que se fala em*Activity* no **Android** refere-se a uma tela de uma aplicação.

Ao criar uma nova classe java, faremos

public class ListaAlunosActivity extends AppCompactActivity {

//....

}

Nós teríamos apenas uma classe normal, mas como queremos dizer que nossa classe representa uma *Activity* para o **Android** é utilizada a herança que é feita através do comando extends.

Esse AppCompatActivity é uma classe que já existe no **Android** e a estendendo estamos afirmando que ela terá o comportamento de uma tela de **Android**.

No código pode-se verificar que existem três métodos, mas o importante é apenas o método OnCreate. Os outros dois podem ser apagados e nada será perdido ao fazer isso.

Primeiro, onCreate é um método que está sendo subscrito da classe AppCompactActivity. O **Android** chama esse método onCreate quando ele cria a tela do celular.

Isto é, criou a tela, abre a aplicação e ele chama onCreate.

A primeira coisa que se faz na on Create é chamar a on Create que já existia na classe super.

Pois, se existe algum comportamento específico ele será recuperado ao se chamar o super. Caso contrário, não chamando o super, estaremos subscrevendo todo o comportamento e jogando fora o que já existe. Por isso, é importante lembrar de recuperar o super antes de inserir o comportamento da nossa tela.

O que resta na Activity é apenas uma linha:

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

Nesse código tudo o que já estava na tela, como a barra e as palavras *Hello world!* Não aparecem no código.

O setContentView é de onde vem o conteúdo da tela e isso separa dois itens: a parte visual e o comportamento.

Na nossa *Activity* coloca-se o comportamento, ou seja, como queremos que o usuário interaja nela.

A parte visual, o que está sendo exibida e disposta na tela coloca-se em um arquivo diferente, em um arquivo de*Layout*.

O R.layout.activity\_lista\_alunos é um atalho no **Android** para acessar um arquivo em determinada pasta.

Olhamos a estrutura do projeto e vemos as três pastas: "manifests", "java" e "res". Ao abrir a pasta res vemos que nela existem várias pastas e dentre elas a pasta *Layout*. Abrindo a pasta *Layout* com um duplo clique vemos que tem um arquivo chamado activity\_lista\_alunos. Percebe-se que é o mesmo nome que está escrito no código!.

A classe R que está sendo utilizada é uma classe gerada automaticamente por **Android** para fazer referencia a qualquer recurso que está na pasta "res". O res vem de *resources*, isto é, recursos e essa pasta R faz referência a pasta res.

O que queremos é acessar uma subpasta da pasta "res" chamada *Layout*. e dentro da pasta *Layout* quero acessar o arquivo "activity\_lista\_alunos.xml".

Ao clicar duas vezes nesse arquivo iremos visualizar uma perspectiva mais visual. Mas, vamos deixar isso de lado por um momento, pois queremos saber exatamente o que está escrito no arquivo.

Clicamos na aba "Text", do lado esquerdo em baixo.

O que abre é o conteúdo do arquivo xml. e nela está descrito o conteúdo da tela, isto é, o seu *layout*.

Vemos um arquivo xml e algumas tags. A tela abre em cima com <RelativeLayout> e fecha com</RelativeLayout>.

Dentro desse RelativeLayout tem um texto. Por enquanto vamos deixá-los de lado e vamos prestar atencao noTextView.

Quando se fala em View no **Android** está se fazendo uma referência ao que o usuário pode ver e as coisas com as quais ele interage. Por exemplo: um texto, um botão, uma imagem, etc.

Tudo isso são as *views*. Se você quiser colocar algum componente visual na tela para mostrar um texto é necessário utilizar um <TextView ...>

Após fazer isso é necessário especificar algumas propriedades. Por enquanto vamos nos atentar ao atributo text. Nele está descrito qual o texto que desejamos que se exiba na tela, no caso já estava escrita a expressão *"Hello World!"*.

Como queremos mostrar o nome dos alunos no aplicativo apaga-se o texto já escrito e coloca-se o texto desejado, por exemplo, o nome de um primeiro aluno: "Daniel":

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

Após ter feito essa modificação pode-se rodar novamente a aplicação. Para isso tem que voltar ao Menu e

" Run > Run app "

Como a aplicação já foi rodada no início existe também uma outra alternativa mais rápida que é clicar no botao verde de *play*, no menu superior. Assim, não é preciso aguardar de novo o emulador pois esse procedimento já foi realizado.

Ao clicar nesse botão ele vai fazer novamente o *building* e vai mostrar novamente a janela "Choose Device".

Como o emulador já foi rodado ele mostra no Device que já existe um rodando e, em vez de se lancar um novo seleciona-se o EmuladorAlura Android 5.1e clica-se no "OK"

Aguarda-se um pouco a instalação e ao subir de novo a modificação feita no TextView é mostrada.

No caso aparece o nome digitado, o nome do Daniel.

**Mostrando vários alunos**

Voltando para o emulador podemos visualizar como está a aplicação até o momento. Já inserimos na tela uma barra inicial com o título da aplicação e também o nome do primeiro aluno.

Agora, para avançar e inserir o nome de mais alunos temos que relembrar alguns passos!

Lembrando que existem duas classes:

* A *Activity*, que é a tela no **Android**, e nela está representado também o seu comportamento. A *Activity* aparece na tela na aba ListaAlunosActivity.java.
* O .xml, que contem os componentes da tela, isto é, o seu conteúdo. A classe .xml está dentro da "pasta"*Layout*. E aparece na tela na aba activity\_lista\_alunos.xml

Para inserir mais nomes de alunos é preciso modificar o .xml inserindo novas *tags*. Já temos no .xml uma *tag* que inseriu o nome do primeiro aluno, o Daniel. Conforme indicado logo abaixo:

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

Para inserir mais nomes de alunos vamos digitar outra *tag*. Vamos dar dois enter após terminar a TextView do "Daniel".

Nessa *tag* vamos inserir um conteúdo diferente pois, queremos deixá-la vazia. Inserimos um atributo text, igual como está escrito na primeira *tag*, portanto, <TextView android:text....

Perceba que a medida que escrevemos a palavra "text" ela vai sendo completada automaticamente. Só é preciso selecionar a opção android:text e pressionar *enter* para poupar a digitação.

Agora, vamos inserir o conteúdo do TextView. Queremos acrescentar o nome do próximo aluno e que ele seja "Ronaldo". Só precisamos completar com seu nome: <TextView android:text="Ronaldo"...

Note que a TextView vai aparecer sublinhada de vermelho chamando atenção para um fato. Isso ocorre pois qualquer View ou qualquer componente que acrescentarmos na tela tem que ter obrigatoriamente dois atributos.

Os atributos que são pedidos já estão no primeiro TextView e são eles o layout\_width e o layout\_height.

<TextView android:text="Daniel" android:layout\_width="wrap\_content"

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

Esses dois atributos servem para definir o tamanho do componente na tela, isto é, quanto ele vai ocupar de espaço.

Não sabemos quanto esse componente ocupa em termos de medida, isto é, em pixels. Vamos usar, portanto, a constante do *Android*, a wrap\_content, que significa envolver o conteúdo. A largura do componente wrap\_contentpara layout\_width é o suficiente para envolver apenas o conteúdo do TextView que é o nome "Daniel".

A mesma coisa vai acontecer no layout\_height, o *height* significa altura em inglês e, assim, acompanhada da constante wrap\_content o layout\_height="wrap\_content" indica a altura necessária para envolver esse conteúdo. Teremos:

<TextView android: text="Daniel'

android:layout\_width="wrap\_content"

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

Isso é importante para que não seja preciso ficar se preocupando com essas medidas, uma vez que desconhecemos a quantidade de *pixels* do nome Daniel.

Obrigatoriamente, abaixo do nome Ronaldo, é preciso preencher digitando a mesma constante. Digitando apenas o "wid..." já o mostra o android:layout\_width seguido de *required* entre parenteses que em inglês significa obrigatório. Pressione o enter e preencha com a constante seguida por wrap\_content.

Em seguida pressione mais um enter para trocar de linha e faça o mesmo procedimento para a altura, trocando apenas por android:layout\_height.

Teremos o seguinte:

<TextView android:text="Ronaldo"

android:layout\_width="wrap\_content"

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

Agora que a *tag* está completa é preciso fechá-la, para isso coloque apenas a barra /> no final como demostrado a cima no final da linha.

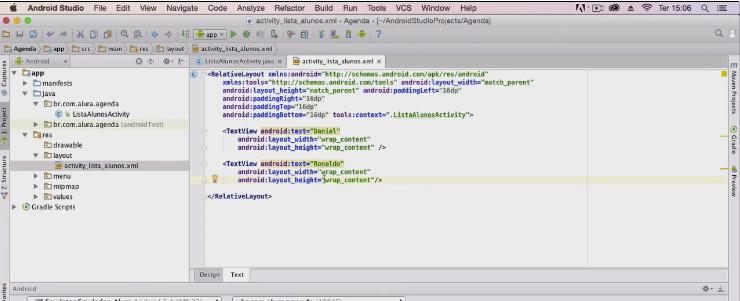
Retomando:

Para cada uma das Views colocamos conteúdo e especificamos a largura e a altura que vão seguidos dewrap\_content. Fizemos isso tanto para "Ronaldo" quanto para "Daniel". Mas, não especificamos ainda onde queremos que nossos nomes apareçam na tela.

A responsabilidade de onde esses nomes serão posicionados fica a cargo da <RelativeLayout.

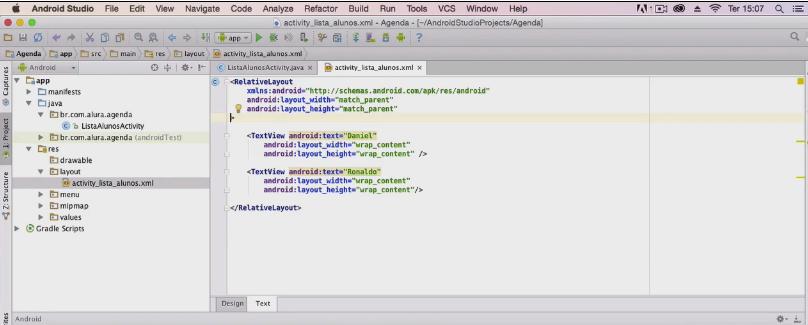
Repare que a RelativeLayout começa em cima, e termina com ListaAlunosActivity que está aberto pois, não tem o />. Dentro disso inserimos o TextView "Daniel" que está fechado e o TextView "Ronaldo", também fechado pois, ambos terminam com />.

Dentro do RelativeLayout temos dois componentes de View e no final temos o RelativeLayout que está fechado com a barra.



O primeiro RelativeLayout, aquele que está no início de tudo, bem na parte superior é um Layoute vai dispor dentro de si os componentes que são como se fossem seus "filhos". Vamos deixar o RelativeLayout de lado por um momento, falaremos dele mais adiante.

Vamos trocar o RelativeLayout por um outro tipo de layout. Primeiro, vamos simplificar um pouco as coisas. Percebemos que no RelativeLayout temos vários atributos que nem chegamos a utilizar. Vamos dar um enter ao final do RelativeLayout na primeira linha e depois, dos **Android** e tools nas próximas. Vai ficar assim:

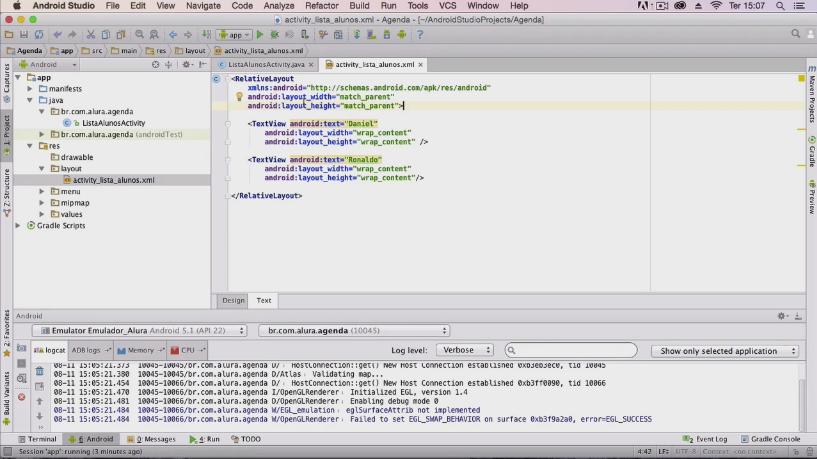


Agora, vamos deixar apenas o necessário removendo aquilo que não é útil nesse momento, como a linha:

xmlns:tools="http://....

As linhas de largura e altura são necessárias ***obrigatoriamente*** para todos os componentes e não devem ser removidas. Todas as linhas seguintes até o ListaActivitypodem ser apagadas até o sinal > da *tag*RelativeLayout. O sinal de maior deve permanecer.

Bom, ficamos apenas com três atributos dentro da RelativeLayout que é justamente apenas o necessário para faz as coisas funcionarem.



Agora, vamos trocar esse RelativeLayout por um Layout diferente.

Como queremos dispor esses componentes?

Queremos dispor os nomes de maneira linear, um em cima do outro de cima para baixo, como uma lista. Ficaremos com a aparência das listas de e-mail ou como nas de aplicativos de mensagens.

No **Android** temos uma *tag* chamada LinearLayout que faz justamente a disposição que queremos, ou seja, de maneira linear.

Um detalhe importante é que sempre que mudamos algo na parte de cima também é alterado lá embaixo. Essa alteração automática ajuda e muito, pois às vezes o arquivo é muito grande e acabamos esquecendo de trocar em outros lugares. Então, uma vez que trocou em cima vai alterar a tag embaixo automaticamente.

<LinearLayout

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

//...

</LinearLayout>

Note que embaixo tem a barrinha no </ indicando que o LinerLayout foi fechado.

Após acrescentar o LinearLayout ainda temos que fazer algumas alterações, pois, caso contrário, o que vai aparecer em nossa tela é que os nomes "Daniel" e "Ronaldo" aparecerão colados e ficarão "DanielRonaldo" e não na disposição linear que desejamos.

Como o responsável por posicionar os componentes é o LinearLayout é nele que vamos introduzir as modificações para alterar a distribuição dos componentes.

Damos um *enter*, criando uma nova linha abaixo do android:layout\_height="match\_parent" e nessa nova linha inserimos um atributo chamado orientation que vai auxiliar, como indica o próprio nome, na orientação que queremos dar aos nossos componentes.

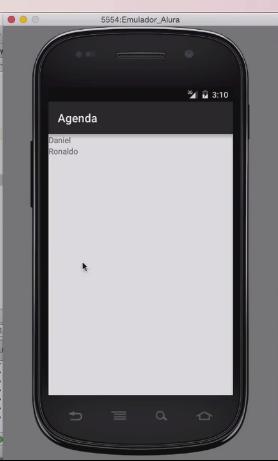
Ao começar a digitar a palavra *orientation* ela já vai aparecer nas opções e aí é só pressionar um enter. Depois de dar enter ele vai perguntar qual a disposição que queremos dar, ou vertical ou horizontal. No nosso caso vamos selecionar a opção vertical e clicar nela. Vai ficar da seguinte maneira:

android:orientation="vertical">

Vamos visualizar se o que fizemos funciona, para isso salvamos o que já fizemos clicando no item do disquete no lado esquerdo do menu e executamos a apresentação no *play* que é a seta verde também no menu.

Aguardamos um pouco e vai abrir uma janela perguntando onde queremos rodar. Como o emulador já está aberto lembre-se de selecioná-lo e pronto, aguarde ele subir.

Vai aparecer na tela o nome "Daniel" e embaixo dele o nome "Ronaldo", como se fosse uma lista mesmo.



Lembre-se que se acontecer alguma coisa na parte visual, por exemplo, uma sobreposição de nomes, isso indica problemas no .xml.

Vamos voltar rapidamente para mais um comentário. Lembra que os atributos que colocamos junto doLinearLayout são obrigatórios? Todas as *tags* devem ter o layout\_widhte o layout\_height que indicam largura e altura respectivamente.

Bom, o LinearLayout irá mostrar a lista completa dos nomes dos alunos que iremos inserir na tela, mas e se tivéssemos definido uma altura fixa para o LinearLayout, por exemplo, a metade da tela?

O que ocorreria é que conforme acrescentássemos mais nomes de alunos, menor iria ficando o espaço até que ele se tornasse insuficiente.

Então, vamos inserir no LinearLayout uma constante diferente. Vou acrescentar nele a constante match\_parentcomo largura e como altura. O match\_parent significa, mais ou menos, que você tem a largura igual a do seu pai.

No caso da TextView se nós apagarmos o wrap\_content e digitarmos o match\_parent no lugar, estaríamos dizendo que sua largura tem que ser igual a do seu pai, ou seja, igual a largura do LinearLayout que é a *tag* "pai". No caso do TextView isso não acarreta nenhum problema.

Se alterarmos o LinearLayout inserindo um match\_parent na largura, teremos o seguinte:

android:layout\_width="match\_parent"

Entretanto, o LinearLayout não irá se guiar em nenhuma outra *tag* pois, ele próprio é a *tag* raiz. Ele vai se guiar a partir da largura da própria tela do celular.

Isso é justamente o que queremos, que a lista ocupe todo espaço da tela e a mesma coisa para a altura da nossa lista, por isso acrescentamos também o match\_parent na altura. E ficaremos no `LinearLayout com:

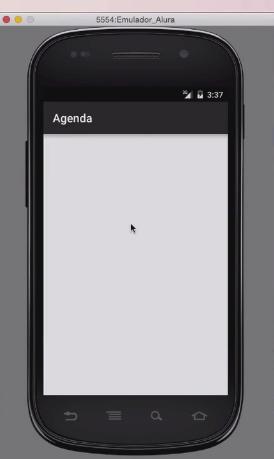
android:layout\_width= "match\_parent"

android:layout\_height="match\_parent"

E, com isso, vamos conseguir mostrar quantos alunos quisermos.

**Melhorando a lista**

Até agora nossa lista possui dois alunos "Ronaldo" e "Daniel". E nossa tela está assim:



***Relembrando:***

Para inseri-los fizemos alterações no .xmle, agora, para seguir acrescentando outros nomes devemos retornar no mesmo local, isto é, no .xml.

Só que existe um caminho mais fácil para isso. É só selecionar com o mouse as linhas do TextView do "Ronaldo" até o fim. Selecionamos o seguinte com o mouse:

<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 utilizamos 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. Vamos ver como ficou!

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 a partir de um 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 mexer no xml. e também não quebraremos a separação já feita no começo, aquela do comportamento na Activity x 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 LinearView 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 owrap\_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"/>

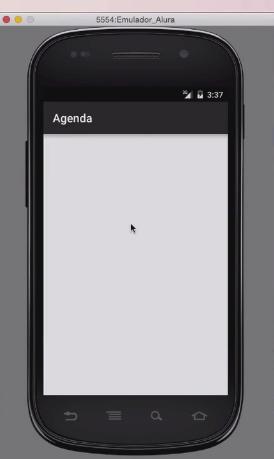
</LinearLayout>

***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 nossa lista é só 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, a tela do nosso celular não terá nada. 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 ListaAlunos.Activity.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 devemos inserir 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 naActivity, 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 daAppCompatActivity. 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 vamos no .xml e encontramos ele 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

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"

android: id="@+id/lista\_alunos"/>

</LinearLayout>

Voltando na Activity vamos, finalmente, referenciar nossa lista no findViewbyId que haviamos 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 instancia da View que foi gerado no SetContentView . Agora falta guardar o findViewbyId.Vamos inserir no .xml uma variável, ListView e vamos nomeá-laListaAluno, assim, ListView ListaAluno vai ficar assim:

protected void onCreate(Bundle saverInstanceStatel {

super.onCreate (savedInstanceState);

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

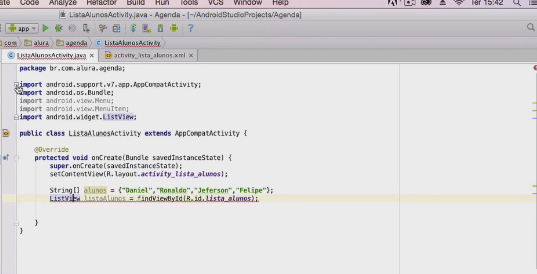
String [] alunos= {"Daniel","Ronaldo", "Jeferson", "Felipe"};

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 ListVie 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 pois, o que está sendo informado é que umaView foi encontrada e está sendo devolvida mas, que isso não pode ser atribuída em uma ListView, são coisas distintas.

Para resolver isso basta converter nossa referencia de View para ListView através de uma referencia de *casting*.

Vamos introduzir entre parenteses que iremos converter essa referencia para uma ListView, 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 colocar os alunos dentro de nossa lista. Para fazer isso precisamos converter os alunos que são String para as View que são pedidas.

A classe ArrayAdapter faz exatamente isso no *Android* e para isso precisamos instanciá-la.

Vamos dar um enter no final da linha ListView e na próxima próxima linha vamos inserir uma new ArrayAdapteradaptando com o que queremos converter em View. Então, teremos 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 o 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 ArrayAdapted.

Agora falta especificar quais são os parâmetros que precisamos para a Array Adapted.

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 referencia para a própria Activity. Para passar uma referencia basta inserir o this. Então, acrescentamos ao new ArrayAdapter<String>, entre parenteses, 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 classeR, mas não a classe do nosso projeto, vamos utilizar a classe R do próprio *Android*. Escrevendo,android.R.layout.simple\_list\_item\_1. Isso, é simplesmente um TextView que auxilia a mostrar o nome do aluno.

Assim, teremos:

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

Falta dizer quais são os dados que quero 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. Assim: new ArrayAdapter <String> (this, android.R.layout.simple\_list\_item\_1, alunos). Ficamos com:

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

Agora a ArrayAdapted está estanciada e para que a referencia não se perca vamos digitar na frente dela uma variável do tipo ArrayAdapted de String, que nós chamaremos de adapter.

Com isso acrescentaremos o seguinte, ArrayAdapeter<String> adapter, e, por fim, teremos:

ArrayAdapeter<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.

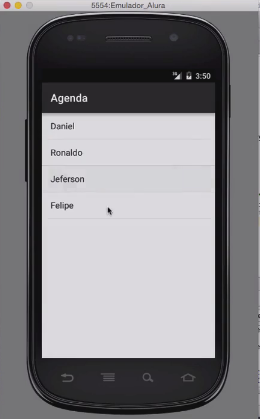
Vamos dizer para a própria lista para utilizar o adapter como sendo seu adapter. Isso pode soar um pouco estranho, mas na verdade, o que quero dizer com isso é que a lista deverá pedir ao adapter para converter os alunos em View. Assim, na linha de baixo do ArrayAdapter vamos digitar ListaAlunos.setAdapter(adapter);.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

ArrayAdapeter<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, estarão com a cor preta para demostrar que estão sendo utilizadas.

Vamos rodar o emulador para ver o que acontece:



Esse é o resultado, temos uma lista bem mais completa, com todos os nomes dos alunos, com um comportamento diferente, espaçamento e até um animação.