02

Navegando entre telas

Transcrição

Agora temos um botão flutuante mas, queremos que ele tenha uma ação real. Queremos que ele nós leve até o formulário e que nós possamos salvar os nomes dos alunos!

Se estamos falando de um comportamento que queremos introduzir no botão, vamos alterar o `ListaAlunosActivity` que está dentro da pasta "br.com.alura.agenda".

Ficaremos na aba `ListaAlunosActivity` . Nela encontramos o seguinte:

```
import android.support.v7.app.AppCompatActivity; import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;

public class ListaAlunosActivity extends AppCompatActivity {

    @Override  protected void onCreate(Bundle savedInstanceState);
    set.onCreate(savedInstanceState);
    setContentView(R.layout.activity_lista_alunos);

        String [] alunos = {"Daniel","Ronaldo","Jeferson","Felipe","Ronaldo","Jeferson","Felipe'
        ListView listaAlunos = (ListView) findViewById(R.id.lista_alunos);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_ite
        listaAlunos.setAdapter(adapter);

}
```

Dentro vamos precisar de uma referência para o botão de "Novo aluno" que criamos anteriormente. Assim, poderemos associar algum comportamento a ele. Após o `listaAlunos.setAdapter(adapter)` damos dois enter e vamos adicionar `Button` , damos um espaço, escrevemos `novoAluno` , novo espaço e `=` , outro espaço e digitamos a `findViewById` e posteriormente definimos o `Id` desse botão que preencheremos dentro do parênteses. Teremos: `Button novoAluno = findViewById ()` .

Para terminar de preencher precisamos criar um `Id` para o botão. Para isso voltamos na `activity_lista_alunos.xml` .

O `id` pode ser adicionado em qualquer lugar do `Button` . Além do `id` também vamos inserir o nome do botão, "novo aluno" e teremos `android:id="@+id/novo_aluno"` . O resultado será o seguinte:

```
<Button android:id="@+id/novo_aluno"
    android:layout_width="56dp“
    android:layout_height="56dp"
    android:text="+"
    android:textColor="#ffffff"
    android:textSize="40sp"
    android:elevation="6dp"
    android:layout_alignParentBottom="true“
    android:layout_alignParentRight="true“
```

```
        android:layout_marginBottom="16dp"
        android:layout_marginRight="16dp"
        android:background="@drawable/fundo"
        android:stateListAnimator="@null"/>
```

Voltamos no `ListaAlunosActivity.java` e inserimos entre os parênteses a `Id` que acabamos de criar no `.xml`. Ficaremos com: `Button novoAluno = findViewById (R.id.novo_aluno)`. Ainda, é preciso definir um *cast* para que ele devolva uma view, portanto, digitamos o atalho "Alt+Enter" em cima da `findViewById` e escolhemos o "Cast to android.widget.Button". Teremos:

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); {
    setContentView(R.layout.activity_lista_alunos);

    String [] alunos ={"Daniel","Renaldo","Jeferson","Felipe","Renaldo","Jeferson","Felipe","Ro
    ListView listaAlunos = (ListView) findViewById(R.id.lista_alunos);
    ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_item_1
    listaAlunos.setAdapter(adapter);

    Button novoAluno = (Button) findViewById (R.Id.novo_aluno);
    }
}
```

Para acrescentar um botão com clique no *Android* acrescentaremos na linha seguinte da `findViewById` o `NovoAluno.setOnClickListener()`, pois o que queremos é "ouvir" o evento de clique. Dentro do parênteses criamos de novo uma classe anônima, `new_OnClickListener`, damos um enter e o *Android* cria a estrutura e agora é só preencher o comportamento que queremos.

```
    novoAluno.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick (View v)  public void onClickListener (new View.OnClickListener() {

        }
    }
```

O que poderíamos fazer agora para indicar o comportamento que queremos é instanciar a `activity`.

Mas, o que queremos é colocar a `activity` em primeiro plano e isso é uma responsabilidade do próprio *Android*. É o mesmo caso, por exemplo, de quando estamos com uma aplicação aberta e recebemos uma chamada telefônica, a prioridade é para a chamada telefônica, então, ela vai se sobrepor a aplicação aberta e isso quem faz é o *Android*. Para colocar a `activity` na frente temos que dialogar justamente com o *Android*. Por isso, não vamos instanciar a activity.

Vamos declarar qual a nossa intenção para o *Android*, então, nessa nova estrutura vamos criar um objeto de tipo `Intent`. Vamos nomear esse novo objeto de `VaiProFormulario`, ignore a gramática de "pro". E para criar ele vamos pedir 'new Intent' e decidir alguns parâmetros que são bem fáceis de lembrar. Teremos `intent intentVaiProFormulario = new Intent()`.

O primeiro parâmetro é nossa identificação, isto é, quem somos. Apenas através da identificação é que conseguimos pedir para o *Android* trocar de tela. Então, vou dizer que sou o `ListaAlunosActivity` e que gostaria de ir para o

formulário. Para isso, colocaremos entre os parênteses da `Intent` um `this`. Lembre-se que o `this` significa uma referência a classe anônima que criamos anteriormente. Para fazer uma referência ao `ListaAlunosActivity` temos que introduzir no parênteses, `ListaAlunoActivity`, um ponto e `this`. Agora, vamos acrescentar o segundo parâmetro,que indica para onde vamos. Acrescentamos uma vírgula e digitamos `FormularioActivity`, um ponto e `class`. Teremos

```
intent intentVaiProFormulario = new Intent (ListaAlunosActivity.this, FormularioActivity.class)
```

E, por fim, teremos:

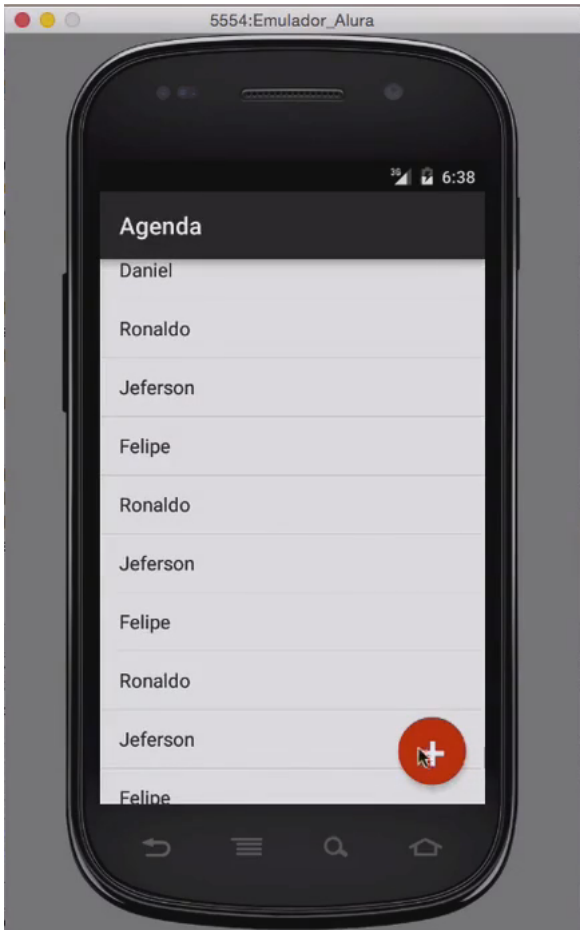
```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); {
    setContentView(R.layout.activity_lista_alunos);

    String [] alunos ={"Daniel","Ronaldo","Jeferson","Felipe","Ronaldo","Jeferson","Felipe"
    ListView listaAlunos = (ListView) findViewById(R.id.lista_alunos);
    ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_list_ite
    listaAlunos.setAdapter(adapter);

    Button novoAluno = (Button) findViewById (R.Id.novo_aluno);
    novoAluno.setOnClickListener(new View.OnClickListener() {
        @Override
         public void onClick (View v)  {
        intent intentVaiProFormulario = new Intent (ListaAlunosActivity.this, FormularioAct

        }
    });

}
```

O que fizemos por último foi avisar para o *Android* o que ele terá de instanciar através da classe, `class`.

A `intent` que criamos é uma variável que o *Android* desconhece. Para avisar ao *Android* que é nossa intenção, vamos adicionar na linha depois da `intent` um método, o `startActivity`, e entre os parênteses vamos acrescentar `intentVaiproFormulario` e fechar com o `;`. Teremos `startActivity(intentVaiproFormulario)`. Ficaremos com:

```
@Override
public void onClick (View v)  {
            intent intentVaiProFormulario = new Intent (ListaAlunosActivity.this, Formu
            startActivity(intentVaiProFormulario);
}
```

Vamos salvar e dar play para ver como ficou no emulador. Bom, depois que apertamos o botão de "+" somos redirecionados para o formulário, mas ao chegar no formulário e selecionarmos o salvar permanecemos na tela do formulário.



Conseguimos agora apertar o botão de "+" e ele vai nos direcionar para o formulário, mas quando chegamos no formulário e preenchemos o que queremos ele não volta para a lista.

Vamos voltar na aba `FormulárioActivity.java` e usar a mesma ideia. Encontramos ela assim:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); setContentView(R.layout.activity_formulario);

        Button botaoSalvar = (Button) findViewById(R.id.formulario_salvar);
        botaoSalvar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
            Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH_SHORT).shou
            }
    });
}
```

Vamos utilizar uma `intent` e a essa `intent` nomeamos `vaiPraLista` e `=` a `new intent` . O código ficará em vermelho pedindo para ser importado, para importar basta usar o atalho "Alt+Enter" e ficamos com, `intent vaiPraLista = new intent()` .

Agora, vamos acrescentar no parênteses o contexto, isto é, o `this` . Lembrando que como estamos em uma classe anônima temos que adicionar os parâmetros. Adicionaremos: quem sou e para onde vou. "Quem somos" preenchemos com o `FormularioActivity` antes do `this` e "Para onde vou" adicionamos uma vírgula após o `this` e acrescentamos `ListaAlunosActivity` informando a classe, então, digitamos um ponto e `class` . Teremos o seguinte:

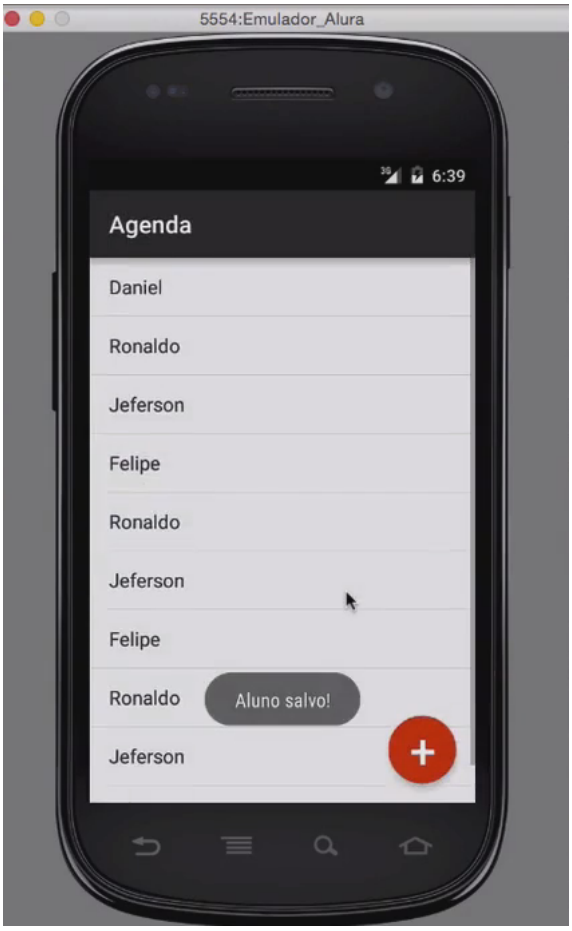
```
Intent vaiPralista = new Intent(FormularioActivity.this,  ListaAlunosActivity.class)
```

Mas, o *Android* não conhece essa intenção, então, temos que avisa-lo através do método `startActivity` que digitaremos na próxima linha. Acrescentamos também a indicação `vaiPraLista` .

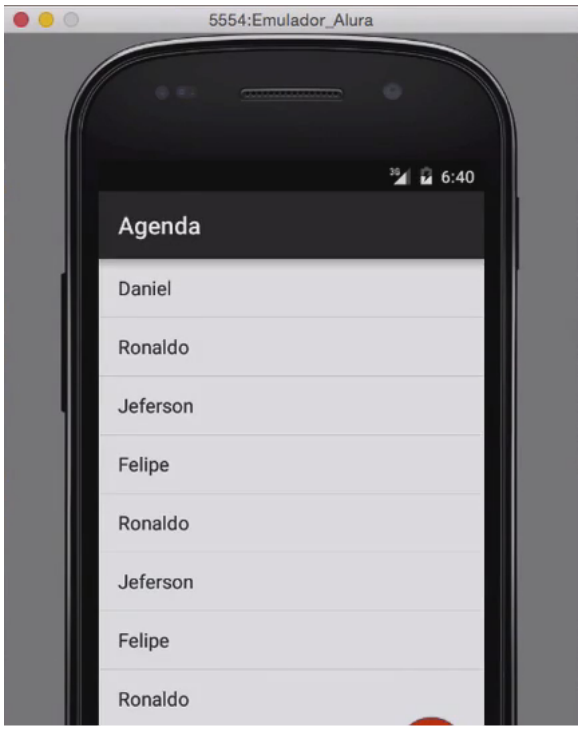
Ficaremos com:

```
        @Override
        public void onClick(View v) {
            Toast.makeText(FormularioActivity.this, "Aluno salvo!", Toast.LENGTH_SHORT).shou
            Intent vaiPralista = new Intent(FormularioActivity.this,  ListaAlunosActivity.cl
            startActivity(vaiPralista);
        }
```

Vamos rodar o emulador e ver como ficou!



Pronto! Agora conseguimos voltar na lista de alunos! Mas, e se selecionarmos a opção de voltar do próprio celular? Aquela flecha que fica normalmente abaixo da tela, no teclado do próprio celular.



Voltamos para o formulário e não para o início da aplicação, o que é um pouco estranho. Se clicamos em salvar iremos duplicar a informação e se clicamos em salvar de novo voltamos para uma outra agenda. O ideal é quando estivermos na lista voltar para a tela do celular, a do *Launcher*(home), aquela que mostra todas as aplicações do celular.

Vamos entender um pouco o que está acontecendo!

Vamos abrir a `ListaAlunosActivity.java` . Toda vez que chamamos o `intent` e preenchemos com o `startActivity` o *Android* de fato inicia uma nova `activity` do formulário, o que está ok. Vamos acessar a `FormularioActivity.java` , quando pedimos para o *Android* ir para a lista e damos um `startActivity` para a `ListadeAlunos` o *Android* entende que é para iniciar uma nova `activity` e não que queremos apenas voltar para a lista. Ele cria uma nova lista e vamos de fato para essa outra lista e quando apertarmos para voltar ele vai destruir a lista que acabamos de criar. Isto é, algo bastante estranho acontece!

Vamos alterar esse comportamento para que quando apertarmos o botão de voltar do teclado, sejamos direcionados para o *Launcher*.

Vamos na aba `FormularioActivity.java` e ao em vez de utilizarmos uma `intent` vamos apagá-la e diremos para o *Android* que queremos apenas finalizar, ou seja, terminar a `Activity` do formulário. Então, após a linha onde especificamos a mensagem que queremos que apareça após clicar o botão de "Salvar", acrescentamos na próxima linha um `finish` . Ficaremos com:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_formulario) ;

    Button botaoSalvar = (Button) findViewById(R.id.formulario_salvar);
    botaoSalvar.setOnClickListener(new View.OnClickListener() {
```

```
            @Override public void onClick(View v) {
            Toast.maketext(FormularioActivity.this, “Aluno salvo!“, Toast.LENGTH_SHORT).show();
            finish();
            }
        });
    }
```

Vamos rodar o emulador para ver se deu certo! Se apertamos o botão de "+", preencheremos o formulário, salvaremos e voltaremos para a lista. E se selecionarmos o botão de voltar... Voltaremos a home!

Pronto! Essa parte está finalizada!