PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Componentes Visuais Avançados

Professor: Danilo Giacobo

OBJETIVOS DA AULA

- Descobrir os componentes visuais mais sofisticados e utilizar o máximo de seus recursos
- Conhecer e usar os componentes RadioGroup e RadioButton
- Conhecer e usar o componente CheckBox
- Conhecer e usar o componente ToggleButton
- Conhecer e usar os componentes DatePicker e TimePicker
- Conhecer e usar o componente ImageButton
- Conhecer e usar o componente ListView
- Conhecer e usar o componente Spinner
- Conhecer e usar o componente AutoCompleteTextView
- Conhecer e usar o componente ProgressBar

INTRODUÇÃO

- Para quem desenvolvia aplicações para celulares mais antigos, a plataforma Android é um verdadeiro paraíso.
- A plataforma Android possui dezenas de componentes visuais sofisticados, com efeitos e muitas características para serem personalizadas.
- A plataforma Android está se tornando uma referência na quantidade e na qualidade dos componentes visuais existentes.
- Vale lembrar que os componentes visuais do Android são definidos a partir de *tags* no arquivo XML de layout e possuem classes correspondentes para que possam ser utilizadas no código Java (*Activity*).

INTRODUÇÃO

Para um melhor entendimento, podem-se dividir os componentes visuais em seis grupos:

- **Views básicas:** composto de componentes básicos, como caixa de textos e botões.
- **Pickers Views:** componentes especiais que permitem ao usuário selecionar os dados a partir de uma fonte de dados específica.
- **Views de listas:** componentes que mostram uma lista de informações.
- **Views para imagens:** componentes utilizados para o tratamento e a apresentação de imagens.
- **Menus:** formados por opções de menu.
- **Extras:** componentes especiais com funções muito específicas.

INTRODUÇÃO

- Em uma aplicação móvel, na maioria das vezes são utilizados apenas os componentes Views básicos, sendo que grande parte das necessidades em uma interface visual, como, por exemplo, a digitação de textos, caixas de escolha, botões de opções, etc., é suprida com esses componentes.
- Em aulas anteriores já vimos os componentes EditText, TextView e Button. Eles são usualmente utilizados nas aplicações Android para a entrada, saída e processamento de dados.
- Nesta aula serão apresentados outros componentes visuais da plataforma Android, como o RadioButton, RadioGroup, CheckBox, ToggleButton, DatePicker, TimePicker, ImageButton, ListView, Spinner e AutoCompleteTextView.

DESENVOLVENDO O APLICATIVO DE EXEMPLO

- Para fazer uso dos componentes visuais avançados, vamos criar um novo projeto Android. Para este, será utilizado o nome *UsandoComponentesAvancados*.
- Os passos para iniciar um novo projeto Android no IDE Eclipse foram apresentados com detalhes em aulas anteriores. Desta forma, nesta aula, serão apresentados apenas os códigos da interface gráfica (XML) e da Activity (Java).
- Para testar os componentes visuais, será desenvolvido um aplicativo simples, no qual serão mostrados na tela o componente visual estudado e um botão que apresenta as características desse componentes durante a execução.
- O slide seguinte apresenta o arquivo XML referente à interface gráfica do aplicativo.

O ARQUIVO XML DA APLICAÇÃO

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <!--  declaração do componente visual aqui  -->

    <Button
        android:id="@+id/btTestarComponente"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/testar"
        android:onClick="btTestarComponenteOnClick" />

</LinearLayout>
```

EXPLICANDO O ARQUIVO XML DA APLICAÇÃO

- A interface desenvolvida apresentará sempre dois componentes visuais na tela do dispositivo Android, ambos organizados por um gerenciador de layout linear no formato vertical.
- O primeiro componente apresentado será o componente estudado (RadioButton, RadioGroup, Checkbox, etc.), que deve ser declarado no lugar do comentário existente no código.
- O segundo componente é um Button, o qual apresentará, quando pressionado, algumas características físicas do componente declarado.
- O código no slide seguinte apresenta a *Activity* Java responsável pelo processamento do aplicativo.

O CÓDIGO-FONTE DA ACTIVITY

- O código apresentado abaixo será usado ao longo dos exemplos de utilização dos componentes visuais Android desta aula.

```
1  package pm25s.aula07.usandocomponentesavancados;
2
3  import android.os.Bundle;
4  import android.view.View;
5  import android.app.Activity;
6  // importação do componente visual aqui
7
8  public class MainActivity extends Activity {
9
10     // declaração do componente visual aqui
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16
17         // recuperação do componente visual aqui
18     }
19
20     public void btTestarComponenteOnClick(View v) {
21         // código para processamento do componente visual aqui
22     }
23 }
```

COMPONENTE RADIOGROUP E RADIOBUTTON

- Os componentes RadioGroup e RadioButton são usados em uma aplicação para que a pessoa possa selecionar apenas uma opção dentre um grupo de opções disponíveis.
- O componente visual que representa o componente de seleção exclusiva é o RadioButton, assim, para o exemplo do campo Sexo, dois componentes seriam necessários: um para o campo *Masculino* e outro para o campo *Feminino*.
- Em algumas situações, precisamos de grupos de RadioButton, pois eles são componentes de seleção exclusivos, assim, só é possível selecionar um RadioButton na tela.
- Na plataforma Android não é possível utilizar RadioButtons sem que estejam inseridos em um RadioGroup.
- Os códigos de exemplo correspondentes a estes componentes visuais estão disponíveis nos próximos slides.

COMPONENTE RADIOGROUP E RADIOBUTTON

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/sexo" />

<RadioGroup
    android:id="@+id/rgSexo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <RadioButton
        android:id="@+id/rbMasculino"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/masculino"
        android:checked="true" />

    <RadioButton
        android:id="@+id/rbFeminino"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/feminino" />

</RadioGroup>
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/estado_civil" />

<RadioGroup
    android:id="@+id/rgEstadoCivil"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <RadioButton
        android:id="@+id/rbCasado"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/casado"
        android:checked="true" />

    <RadioButton
        android:id="@+id/rbSolteiro"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/solteiro" />

    <RadioButton
        android:id="@+id/rbDivorciado"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/divorciado" />

</RadioGroup>
```

COMPONENTE RADIOGROUP E RADIOBUTTON

COMPONENTE RADIOGROUP E RADIOBUTTON

```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.RadioButton;
import android.widget.Toast;
import android.app.Activity;

public class RadioButtonActivity extends Activity {

    private RadioButton rbMasculino;
    private RadioButton rbSolteiro;
    private RadioButton rbCasado;

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

        rbMasculino = (RadioButton) findViewById(R.id.rbMasculino);
        rbSolteiro = (RadioButton) findViewById(R.id.rbSolteiro);
        rbCasado = (RadioButton) findViewById(R.id.rbCasado);
    }
```

```java
    public void btTestarComponenteOnClick(View v) {
        String texto = "";

        if(rbMasculino.isChecked()) {
            texto = "Masculino selecionado.\n";
        } else {
            texto = "Feminino selecionado.\n";
        }

        if(rbSolteiro.isChecked()) {
            texto += "Solteiro selecionado.";
        } else if(rbCasado.isChecked()) {
            texto += "Casado selecionado.";
        } else {
            texto += "Divorciado selecionado.";
        }

        Toast.makeText(this, texto, Toast.LENGTH_LONG).show();
    }
}
```

COMPONENTE RADIOGROUP E RADIOBUTTON

Dica: Usando RadioGroup na condicional

Pode-se utilizar o componente RadioGroup na condicional para identificar qual de seus RadioButtons foi utilizado e para isso, basta instanciar/recuperar o RadioGroup:

Ex. RadioGroup rgSexo = (RadioGroup) findViewById(R.id.rgSexo);

E após, na condicional, fazer o teste como segue:

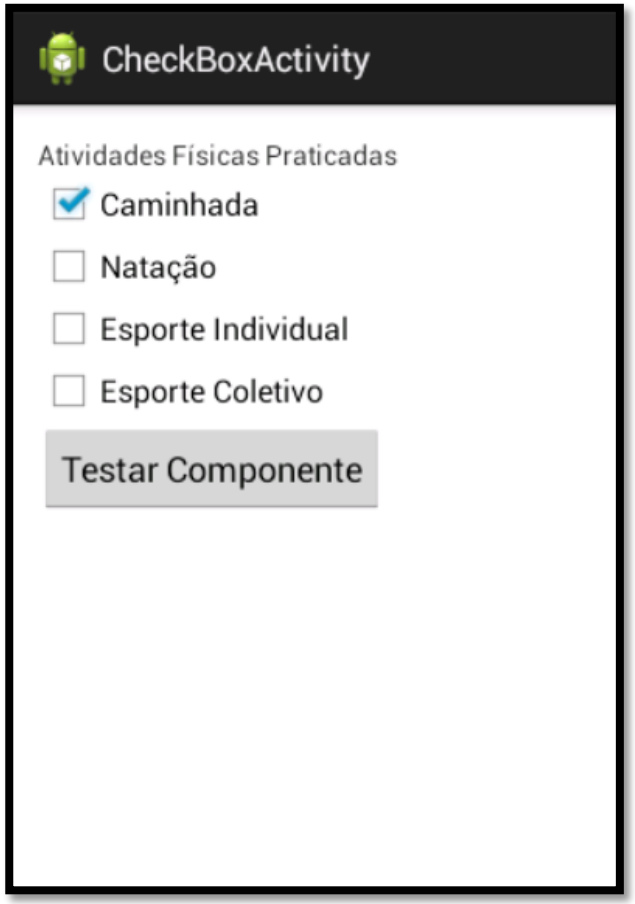
```
if(rgSexo.getCheckedRadioButtonId() == R.id.rbMasculino) {
        // tratamento do RadioButton Masculino
} else {
        // tratamento do RadioButton Feminino
}
```

COMPONENTE CHECKBOX

- O componente CheckBox é similar ao componente RadioButton visto anteriormente, com a diferença de permitir a seleção de qualquer número de opções no aplicativo, ao contrário do RadioButton, que permite apenas uma por RadioGroup. Por esse motivo, o CheckBox não necessita do uso de componentes de grupo.
- Esse componente pode ser utilizado para a seleção, onde existe a opção do usuário selecionar mais de um campo, como, por exemplo, de uma lista de atividades físicas praticadas.
- Os códigos de exemplo correspondentes a este componente visual estão disponíveis nos próximos slides.

COMPONENTE CHECKBOX

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/atividades" />

<CheckBox
    android:id="@+id/cbCaminhada"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/caminhada"
    android:checked="true" />

<CheckBox
    android:id="@+id/cbNatacao"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/natacao" />

<CheckBox
    android:id="@+id/cbEsporteIndividual"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/esporte_individual" />

<CheckBox
    android:id="@+id/cbEsporteColetivo"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/esporte_coletivo" />
```

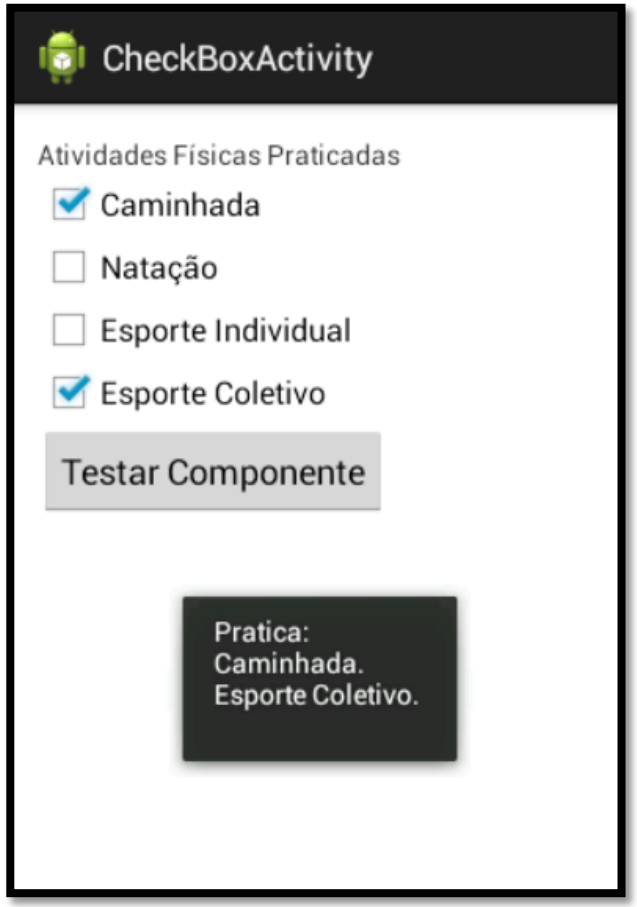


COMPONENTE CHECKBOX

```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.Toast;
import android.app.Activity;

public class CheckBoxActivity extends Activity {

    private CheckBox cbCaminhada;
    private CheckBox cbNatacao;
    private CheckBox cbEsporteIndividual;
    private CheckBox cbEsporteColetivo;

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

        cbCaminhada = (CheckBox) findViewById(R.id.cbCaminhada);
        cbNatacao = (CheckBox) findViewById(R.id.cbNatacao);
        cbEsporteIndividual = (CheckBox) findViewById(R.id.cbEsporteIndividual);
        cbEsporteColetivo = (CheckBox) findViewById(R.id.cbEsporteColetivo);
    }
```

```java
    public void btTestarComponenteOnClick(View v) {
        String texto = "Pratica: \n";

        if(cbCaminhada.isChecked())
            texto += cbCaminhada.getText() + ".\n";

        if(cbNatacao.isChecked())
            texto += cbNatacao.getText() + ".\n";

        if(cbEsporteIndividual.isChecked())
            texto += cbEsporteIndividual.getText() + ".\n";

        if(cbEsporteColetivo.isChecked())
            texto += cbEsporteColetivo.getText() + ".\n";

        Toast.makeText(this, texto, Toast.LENGTH_LONG).show();
    }
}
```

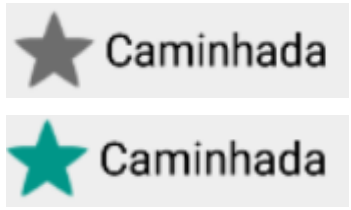
COMPONENTE CHECKBOX



Dica: Mudando o estilo de um CheckBox

É possível mudar visualmente o componente CheckBox. Para isso, basta adicionar a propriedade style à declaração XML do componente, conforme abaixo:

```
<CheckBox
    style="?android:attr/starStyle"
    android:id="@+id/cbCaminhada"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/caminhada"
    android:checked="true" />
```



COMPONENTE TOGGLEBUTTON

- Este componente é uma mistura dos componentes CheckBox e Button.
- É um componente de seleção, no qual o usuário por marcá-lo ou não, entretanto, sua aparência é idêntica a de um botão.
- O ToggleButton possui uma representação visual para indicar se o mesmo está ligado ou não e essa indicação é dada por um retângulo cinza escuro quando desligado ou azul/verde se ligado, dependendo da versão do Android.
- Além da cor, também é apresentado um texto para o usuário indicando se o ToggleButton está ligado ou não.
- Os códigos de exemplo correspondentes a este componente visual estão disponíveis nos próximos slides.

COMPONENTE TOGGLEBUTTON

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".ToggleButtonActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/gravar_log" />

    <ToggleButton
        android:id="@+id/tbLog"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btTestarComponente"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/testar"
        android:onClick="btTestarComponenteOnClick" />

</LinearLayout>
```

COMPONENTE TOGGLEBUTTON

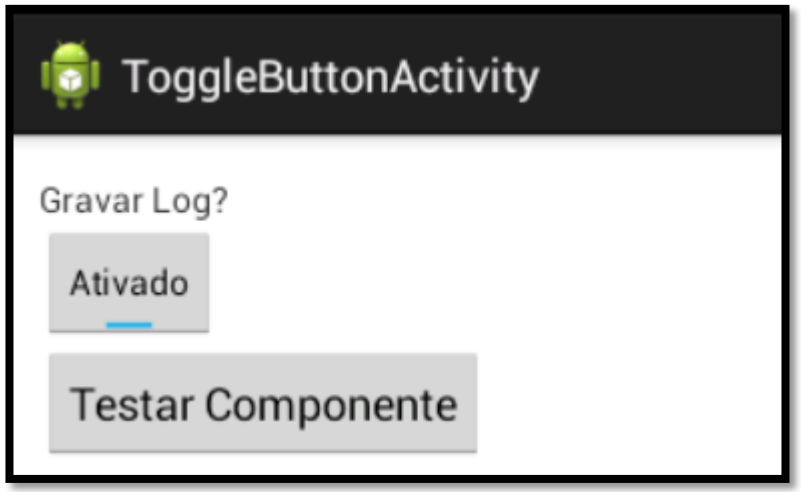
```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.ToggleButton;
import android.widget.Toast;
import android.app.Activity;

public class ToggleButtonActivity extends Activity {

    private ToggleButton tbLog;

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

        tbLog = (ToggleButton) findViewById(R.id.tbLog);
    }

    public void btTestarComponenteOnClick(View v) {
        if(tbLog.isChecked())
            Toast.makeText(this, "Log Habilitado", Toast.LENGTH_LONG).show();
        else
            Toast.makeText(this, "Log Desabilitado", Toast.LENGTH_LONG).show();
    }
}
```

COMPONENTE TOGGLEBUTTON

Dica: Mudando o texto do componente ToggleButton.

Como apresentado, o ToggleButton apresenta, de duas formas para o usuário, se o mesmo está ligado ou não, sendo por meio de sua cor e também através de seu texto. O texto, por padrão em um dispositivo configurado para o idioma inglês, é ON e OFF, porém, isto pode ser personalizado por meio das propriedades XML textOn e textOff, como mostrado a seguir:

```
<ToggleButton
    android:id="@+id/tbLog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Ativado"
    android:textOff="Desativado" />
```



COMPONENTES DATEPICKER E TIMEPICKER

- Esses componentes possuem como característica a riqueza visual e a facilidade na entrada de dados.
- É permitida a entrada segura de informações temporais, evitando inconsistências na digitação (uma data ou hora inválida).
- Dependendo da versão do Android um calendário é apresentado para a escolha da data, facilitando a escolha da informação.
- Os códigos de exemplo correspondentes a estes componentes visuais estão disponíveis nos próximos slides.

COMPONENTES DATEPICKER E TIMEPICKER

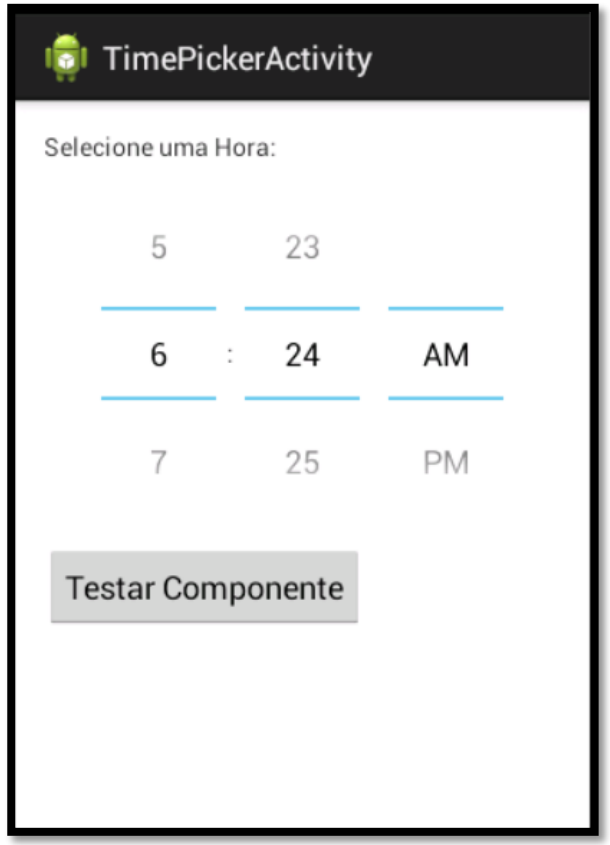
```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/data" />

<DatePicker
    android:id="@+id/dpData"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:calendarViewShown="false" />

<Button
    android:id="@+id/btTestarComponente"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/testar"
    android:onClick="btTestarComponenteOnClick" />
```

COMPONENTES DATEPICKER E TIMEPICKER

```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.DatePicker;
import android.widget.Toast;
import android.app.Activity;

public class DatePickerActivity extends Activity {

    private DatePicker dpData;

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

        dpData = (DatePicker) findViewById(R.id.dpData);
    }

    public void btTestarComponenteOnClick(View v) {
        int dia = dpData.getDayOfMonth();
        int mes = dpData.getMonth();
        int ano = dpData.getYear();

        String data = dia + "/" + (mes + 1) + "/" + ano;

        Toast.makeText(this, "Data selecionada: " + data, Toast.LENGTH_LONG).show();
    }
}
```

COMPONENTES DATEPICKER E TIMEPICKER

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hora" />

<TimePicker
    android:id="@+id/tpHora"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />

<Button
    android:id="@+id/btTestarComponente"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/testar"
    android:onClick="btTestarComponenteOnClick" />
```



COMPONENTES DATEPICKER E TIMEPICKER

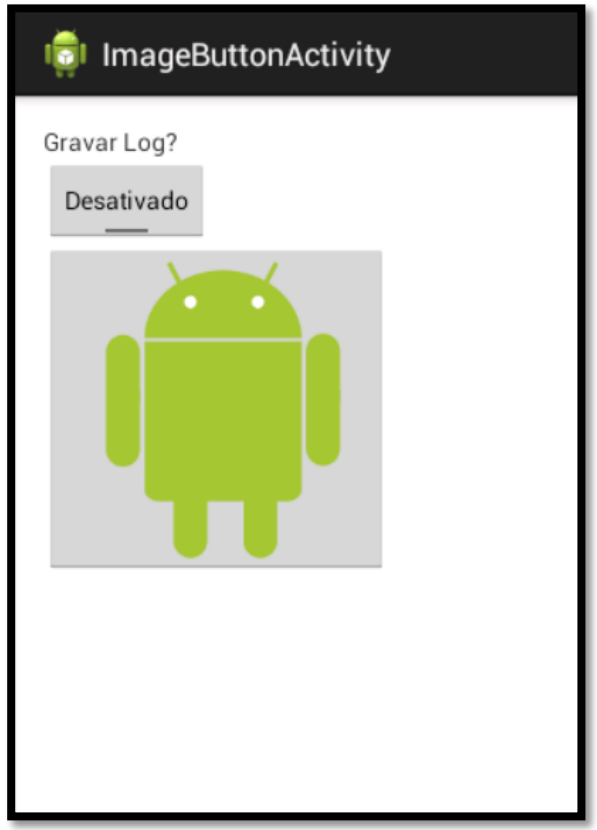
```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.TimePicker;
import android.widget.Toast;
import android.app.Activity;

public class TimePickerActivity extends Activity {

    private TimePicker tpHora;

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

        tpHora = (TimePicker) findViewById(R.id.tpHora);
    }

    public void btTestarComponenteOnClick(View v) {
        int hora = tpHora.getCurrentHour();
        int minuto = tpHora.getCurrentMinute();

        String dados = hora + ":" + minuto;

        Toast.makeText(this, "Hora selecionada: " + dados, Toast.LENGTH_LONG).show();
    }
}
```

COMPONENTE IMAGEBUTTON

- O componente ImageButton tem o funcionamento idêntico a de um Button tradicional.
- A única diferença está em sua representação visual, que substitui o tradicional texto dos botões por uma imagem.
- Algumas dicas importantes: use imagens pequenas no formato PNG com efeito de transparência.
- O nome do arquivo da imagem também deve ser cuidadosamente escolhido conforme regras da linguagem Java.
- Para o exemplo deste componente visual, uma imagem do Android será utilizada. Esta imagem deverá ser copiada e colada em todas as pastas *drawable* do projeto.
- O código da aplicação pode ser visto nos slides seguintes.

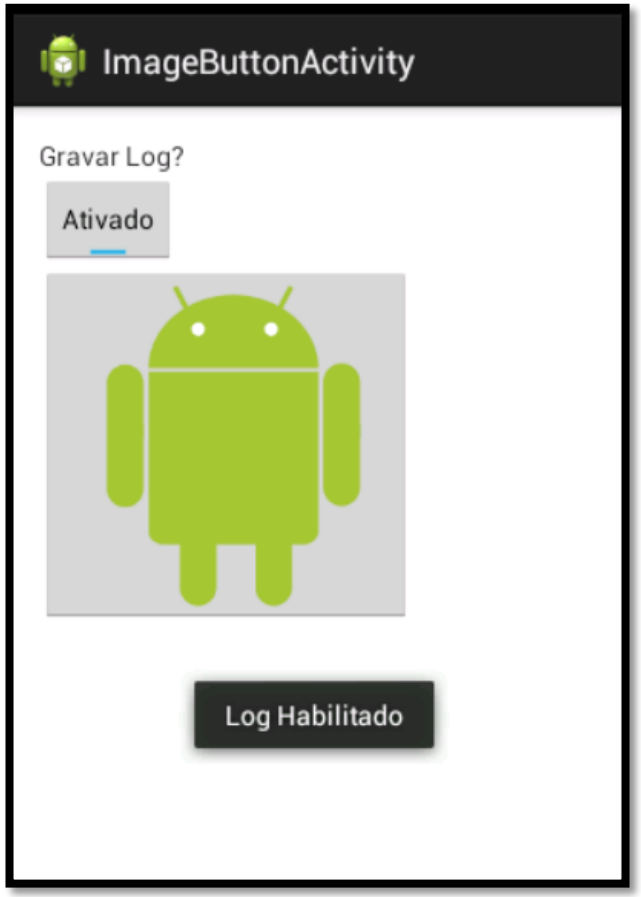
COMPONENTE IMAGEBUTTON

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/gravar_log" />

<ToggleButton
    android:id="@+id/tbLog"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textOn="Ativado"
    android:textOff="Desativado" />

<ImageButton
    android:id="@+id/btTestarComponente"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/android"
    android:contentDescription="@string/testar"
    android:onClick="btTestarComponenteOnClick" />
```



COMPONENTE IMAGEBUTTON

```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.ToggleButton;
import android.widget.Toast;
import android.app.Activity;

public class ImageButtonActivity extends Activity {

    private ToggleButton tbLog;

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

        tbLog = (ToggleButton) findViewById(R.id.tbLog);
    }

    public void btTestarComponenteOnClick(View v) {
        if(tbLog.isChecked())
            Toast.makeText(this, "Log Habilitado", Toast.LENGTH_LONG).show();
        else
            Toast.makeText(this, "Log Desabilitado", Toast.LENGTH_LONG).show();
    }
}
```



COMPONENTES DE LISTA

- Uma categoria de componentes mais sofisticados da plataforma Android são os componentes de lista. Esses componentes trabalham apresentando um conjunto de dados, muitas vezes formados por uma *string* (mas nada impede que sejam imagens ou até mesmo outros componentes visuais) em listas para o usuário.
- Esses dados podem ser recuperados de listas estáticas, como, por exemplo, as definidas em um arquivo XML, ou de forma dinâmica, com o conteúdo oriundo de um banco de dados.
- Esses componentes diferem pela forma de apresentação dos dados. Destacam-se, nesta categoria, os componentes ListView, Spinner e AutoCompleteTextView.
- Para exemplificar o uso desses componentes, serão desenvolvidos novos exemplos, assim, as estruturas utilizadas até o momento para a interface e lógica do negócio poderão ser desconsideradas.

COMPONENTE LISTVIEW

- O componente ListView é utilizado para mostrar uma grande quantidade de dados em formato de lista.
- Você pode tratar eventos de cliques e de seleção, além de poder utilizar o mesmo apenas para apresentação de dados.
- Para apresentar algumas das funcionalidades deste componente, criaremos uma lista com alguns países da copa do mundo de 2014. Ao clicarmos em um dos nomes, mostraremos a posição da seleção no ranking da FIFA em um componente *Toast*.
- Para o desenvolvimento deste exemplo, podem-se utilizar três metodologias:
  - Recuperar os elementos diretamente de um arquivo XML;
  - Montar os elementos da lista em um componente *ArrayAdapter*;
  - Excluir o arquivo *activity_principal.xml* e usar no lugar de *extends Activity*, *extends ListActivity*.

RECUPERAR ELEMENTOS DE UM ARQUIVO XML

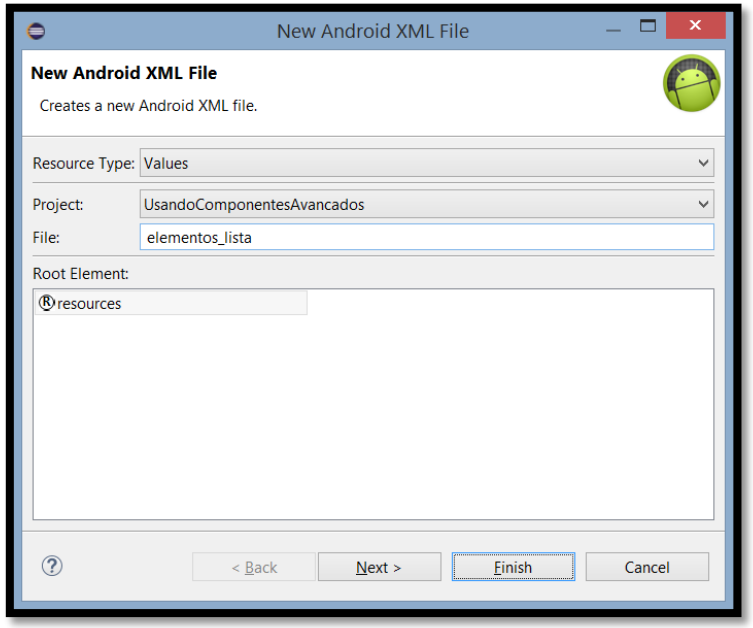
- Como primeiro exemplo, será recuperado o nome das seleções a partir de um arquivo XML, que também será criado no exemplo. Assim primeiramente iremos alterar o arquivo *activity_principal.xml* para inserir o componente ListView.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ListView xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:id="@+id/lvSelecoesCopa"
6     android:entries="@array/paises_copa" >
7 </ListView>
```

- O erro na linha 6 será corrigido posteriormente.
- Como o ListView é o único componente na tela não é preciso usar um gerenciador de layout neste caso.

RECUPERAR ELEMENTOS DE UM ARQUIVO XML

- Para criar um XML com um *array* de *strings*, clique com o botão direito no projeto, escolhendo a opção **New > File > Android XML File**. Na tela apresentada, em **Resource Type**, deve-se escolher **Values** e em **File**, digitar o nome, que pode ser **elementos_lista**, conforme imagem abaixo:



RECUPERAR ELEMENTOS DE UM ARQUIVO XML

- No arquivo criado dentro da pasta *values* do projeto, deve-se editar o XML, colocando o conteúdo conforme imagem abaixo:

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="paises_copa">
        <item>Alemanha</item>
        <item>Argentina</item>
        <item>Holanda</item>
        <item>Colômbia</item>
        <item>Bélgica</item>
        <item>Uruguai</item>
        <item>Brasil</item>
    </string-array>
</resources>
```

- Depois disso é só declarar e recuperar o componente **ListView** na classe *Activity* para um futuro uso da mesma, conforme apresentado no próximo slide.

RECUPERAR ELEMENTOS DE UM ARQUIVO XML

```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.ListView;
import android.widget.Toast;
import android.app.Activity;
import android.widget.AdapterView;

public class ListView1Activity extends Activity {

    private ListView lvSelecoesCopa;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_view1);

        lvSelecoesCopa = (ListView)findViewById(R.id.lvSelecoesCopa);
        lvSelecoesCopa.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
                tratarOpcoesItem(arg2);
            }
        });
    }

    public void tratarOpcoesItem(int posicao) {
        int posFifa = posicao + 1;
        Toast.makeText(this, "Posição no Ranking: " + posFifa, Toast.LENGTH_SHORT).show();
    }
}
```

ELEMENTOS DA LISTA COM O ARRAYADAPTER

- O segundo exemplo de utilização de um ListView é a apresentação de seu conteúdo a partir do código Java, e não mais a partir de uma lista XML.
- A primeira mudança acontece no arquivo XML, onde se deve retirar a propriedade android:entries, deixando o código conforme listagem abaixo:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ListView xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:id="@+id/lvSelecoesCopa" >
6 </ListView>
```

- A maior diferença acontecerá na classe *Activity*, já que, nesta situação, os elementos da lista serão valorizados durante a execução, ou seja, diretamente no código Java. Assim, a classe PrincopalActivity.java modificada é apresentada no slide seguinte.

ELEMENTOS DA LISTA COM O ARRAYADAPTER

```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.ListView;
import android.widget.Toast;
import android.app.Activity;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;

public class ListView2Activity extends Activity {

    private ListView lvSelecoesCopa;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_list_view2);

        lvSelecoesCopa = (ListView)findViewById(R.id.lvSelecoesCopa);
        String itens[] = {"Alemanha", "Argentina", "Holanda", "Colômbia", "Bélgica", "Uruguai", "Brasil"};

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, itens);
        lvSelecoesCopa.setAdapter(adapter);

        lvSelecoesCopa.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
                tratarOpcoesItem(arg2);
            }
        });
    }

    public void tratarOpcoesItem(int posicao) {
        int posFifa = posicao + 1;
        Toast.makeText(this, "Posição no Ranking: " + posFifa, Toast.LENGTH_SHORT).show();
    }
}
```

ELEMENTOS DA LISTA COM O ARRAYADAPTER

Dica: Recuperando os elementos de um XML durante a execução

- É possível também, durante a execução, recuperar o conteúdo de um arquivo XML, apresentando-o em um ListView na execução. Basta criar um ArrayAdapter com o código apresentado abaixo, valorizando a propriedade adapter de um ListView:

```
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this, R.array.paises_copa, android.R.layout.simple_list_item_1);
lvSelecoesCopa.setAdapter(adapter);
```

LISTACTIVITY NO LUGAR DA INTERFACE XML

- Por fim, uma nova forma de usar o ListView, já que comumente é o único componente apresentado na tela, é utilizando a classe ListActivity. Nesta situação, não é necessário o arquivo XML de layout, podendo até ser excluído do projeto.
- A diferença acontece na declaração da Activity Java, substituindo extends Activity por extends ListActivity, e tirando toda a referência ao ListView, conforme código apresentado no próximo slide.
- Ao comparar este código com o do primeiro exemplo, observamos que o número de linhas de código digitada diminui consideravelmente, assim como a complexidade da classe.

LISTACTIVITY NO LUGAR DA INTERFACE XML

```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.ListView;
import android.widget.Toast;
import android.app.ListActivity;
import android.widget.ArrayAdapter;

public class ListView3Activity extends ListActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        String itens[] = {"Alemanha", "Argentina", "Holanda", "Colômbia", "Bélgica", "Uruguai", "Brasil"};
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, itens);
        this.setListAdapter(adapter);
    }

    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        int posFifa = position + 1;
        Toast.makeText(this, "Posição no Ranking: " + posFifa, Toast.LENGTH_SHORT).show();
    }
}
```

LISTACTIVITY NO LUGAR DA INTERFACE XML

Dica: Usando ListView e uma interface gráfica XML

- É possível utilizar a classe ListView e, mesmo assim, utilizar uma interface gráfica XML com a declaração de ListView. Esta técnica é muito utilizada se o programador precisa definir características da lista diretamente no arquivo XML. Neste caso, é obrigatório que o ListView definido na interface gráfica possua o nome android:list, como pode ser observado no código abaixo:

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <ListView xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="fill_parent"
4     android:layout_height="fill_parent"
5     android:id="@+id/android:list" >
6 </ListView>
```

COMPONENTE SPINNER

- Este componente também possui a função de apresentar um conjunto de dados no formato de lista, semelhante ao visto no exemplo anterior, com o ListView.
- O Spinner, ao contrário do ListView, que costuma ser um componente solitário na tela do dispositivo móvel, costuma apresentar suas informações nas telas com outros componentes, como EditText e Button.
- O diferencial do Spinner é a possibilidade de apresentar uma grande quantidade de dados em uma lista, porém, fica resumido a um componente de texto (semelhante ao EditText).
- Nos slides seguintes será apresentado um exemplo de declaração de um componente Spinner no arquivo XML principal e na classe *Activity* Java.

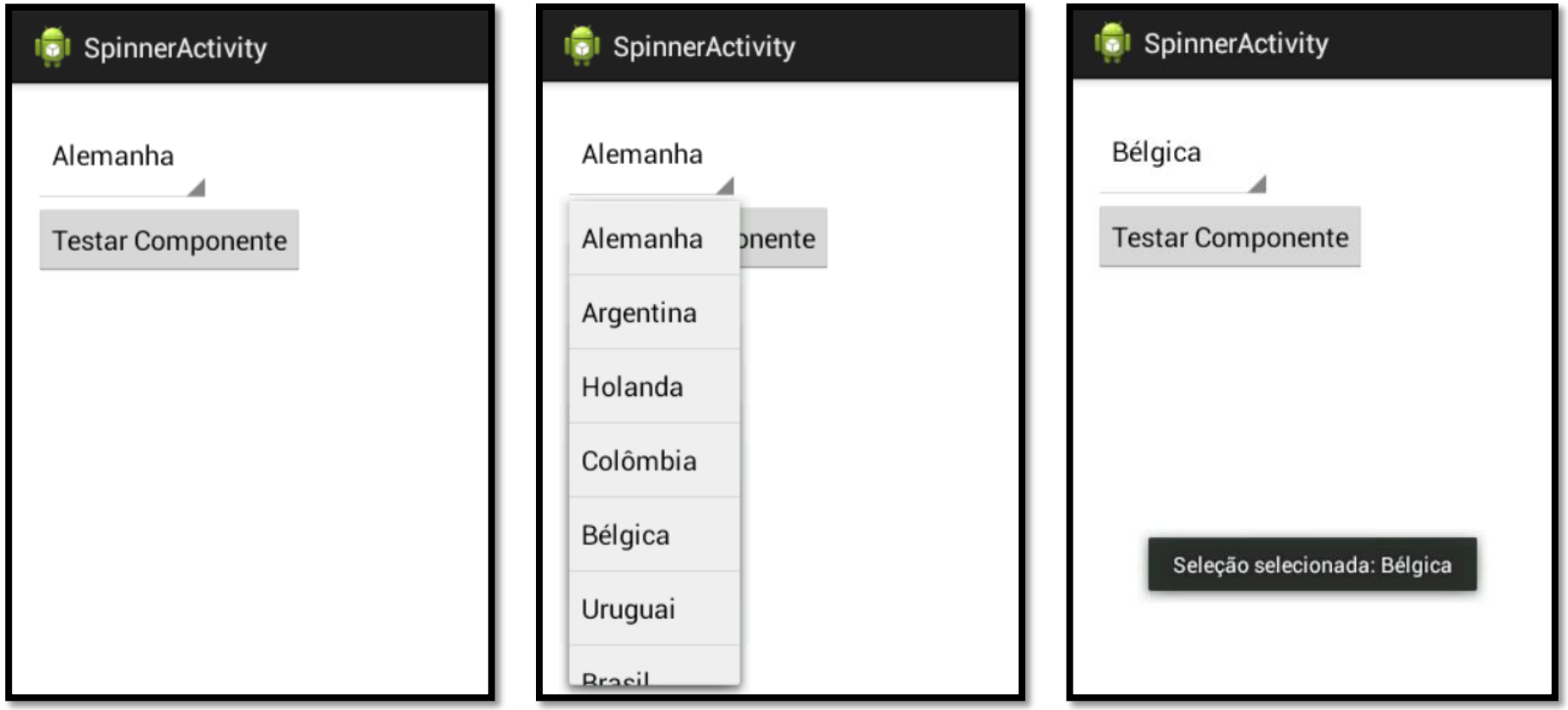
COMPONENTE SPINNER

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".SpinnerActivity" >

    <Spinner
        android:id="@+id/spSelecoesCopa"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/btTestarComponente"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/testar"
        android:onClick="btTestarComponenteOnClick" />

</LinearLayout>
```

COMPONENTE SPINNER

```java
package pm25s.aula07.usandocomponentesavancados;

import android.os.Bundle;
import android.view.View;
import android.widget.Spinner;
import android.widget.Toast;
import android.app.Activity;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;

public class SpinnerActivity extends Activity {

    private Spinner spSelecoesCopa;

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

        spSelecoesCopa = (Spinner)findViewById(R.id.spSelecoesCopa);
        String itens[] = {"Alemanha", "Argentina", "Holanda", "Colômbia", "Bélgica", "Uruguai", "Brasil"};

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, itens);
        spSelecoesCopa.setAdapter(adapter);

        spSelecoesCopa.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
                tratarOpcoesItem(arg2);
            }
            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
            }
        });
    }

    public void tratarOpcoesItem(int posicao) {
        int posFifa = posicao + 1;
        Toast.makeText(this, "Posição no Ranking: " + posFifa, Toast.LENGTH_SHORT).show();
    }

    public void btTestarComponenteOnClick(View v) {
        String texto = spSelecoesCopa.getSelectedItem().toString();
        Toast.makeText(this, "Seleção selecionada: " + texto, Toast.LENGTH_SHORT).show();
    }
}
```

COMPONENTE SPINNER



COMPONENTE SPINNER

Dica: Propriedade android:entries

- Assim como a classe *ListView*, a classe *Spinner* também permite a recuperação de elementos a partir dos dados existentes em um arquivo XML utilizando a propriedade android:entries na sua declaração dentro da *Activity* Principal, assim como no componente ListView.

```
<Spinner
    android:id="@+id/spSelecoesCopa"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:entries="@array/paises_copa" />
```

COMPONENTE AUTOCOMPLETETEXTVIEW

- O AutoCompleteTextView também é uma variação do componente de lista.
- Embora tenha a terminação TextView, esse componente é mais semelhante a um EditText do que ao TextView, uma vez que o usuário pode digitar texto dentro dele. A diferença é que à medida que o usuário vai digitando no componente, vão sendo apresentadas sugestões pré-cadastradas para completar automaticamente o texto, facilitando a entrada de dados e evitando erros de digitação.
- No slide seguinte é apresentado o código do arquivo XML com a utilização do componente AutoCompleteTextView.

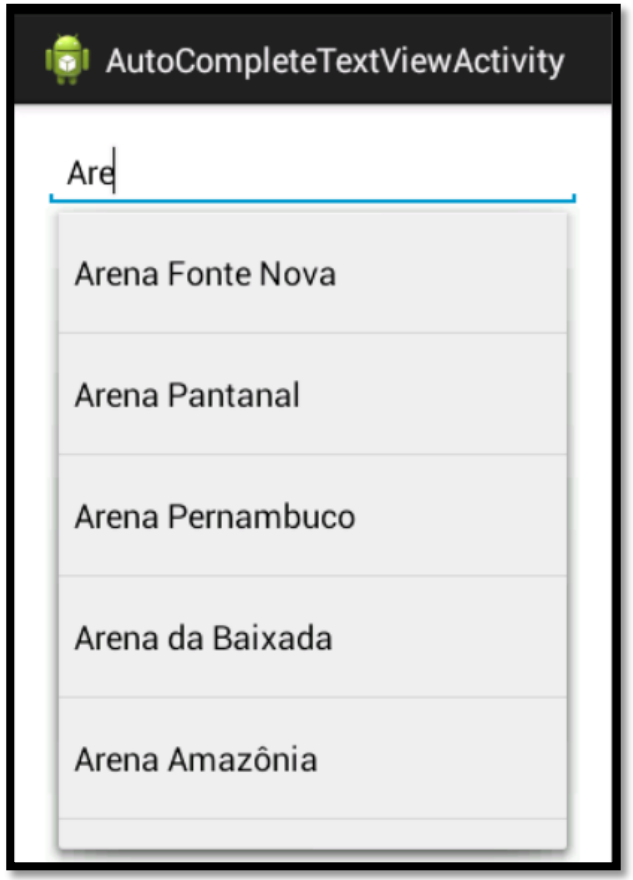
COMPONENTE AUTOCOMPLETETEXTVIEW

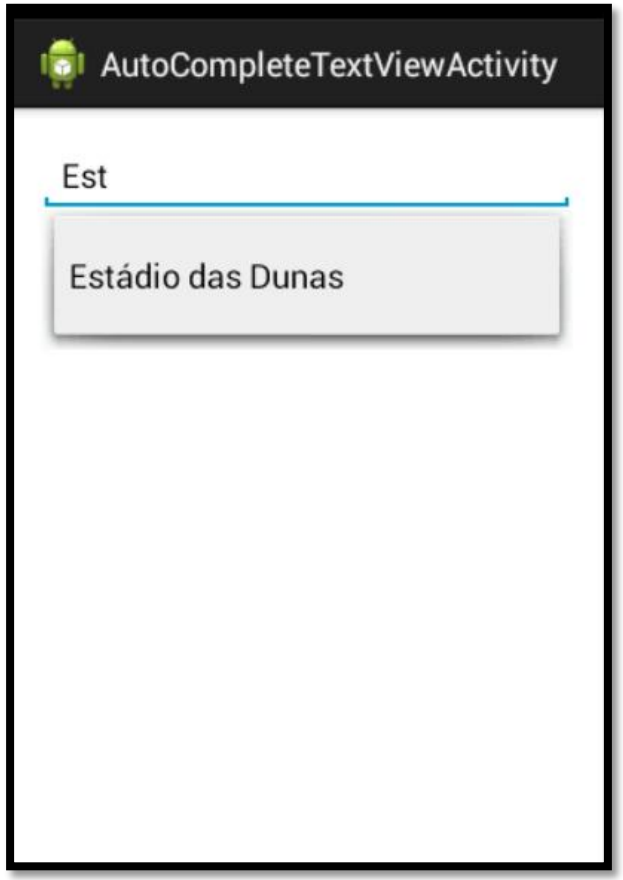
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".AutoCompleteTextViewActivity" >

    <AutoCompleteTextView
        android:id="@+id/txtEstadios"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />

</LinearLayout>
```

COMPONENTE AUTOCOMPLETETEXTVIEW

```
1  package pm25s.aula07.usandocomponentesavancados;
2
3  import android.os.Bundle;
4  import android.app.Activity;
5  import android.widget.ArrayAdapter;
6  import android.widget.AutoCompleteTextView;
7
8  public class AutoCompleteTextViewActivity extends Activity {
9
10     String[] estadios = {"Mineirão", "Nacional", "Arena Fonte Nova", "Arena Pantanal", "Arena Pernambuco",
11                          "Arena da Baixada", "Beira-Rio", "Castelão", "Arena Amazônia", "Estádio das Dunas",
12                          "Maracanã", "Arena São Paulo"};
13
14     @Override
15     protected void onCreate(Bundle savedInstanceState) {
16         super.onCreate(savedInstanceState);
17         setContentView(R.layout.activity_auto_complete_text_view);
18
19         ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_dropdown_item_1line, estadios);
20         AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.txtEstadios);
21         textView.setThreshold(3);
22         textView.setAdapter(adapter);
23     }
24 }
```

COMPONENTE AUTOCOMPLETETEXTVIEW



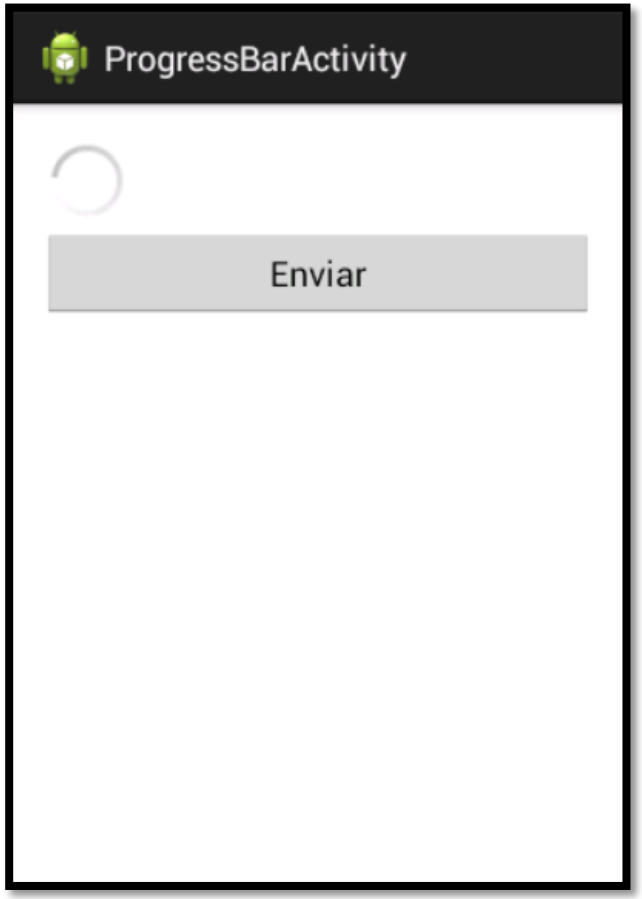


COMPONENTE PROGRESSBAR

- A barra de progresso é um componente muito comum em qualquer interface de usuário.
- Ela pode ser utilizada para acompanhar quantos bytes foram transmitidos na rede, como está o andamento da execução de uma tarefa e, principalmente, para dar um retorno ao usuário nos comandos mais lentos, evitando que este pense que o programa travou.
- Para o exemplo de uso deste componente, foi utilizada a interface visual contida no arquivo XML, cujo conteúdo está no slide seguinte.

COMPONENTE PROGRESSBAR

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".ProgressBarActivity" >

    <ProgressBar
        android:id="@+id/pbBarra"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

     <Button
        android:id="@+id/btEnviar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/enviar" />

</LinearLayout>
```



COMPONENTE PROGRESSBAR

```
1  package pm25s.aula07.usandocomponentesavancados;
2
3  import android.os.Bundle;
6
7  public class ProgressBarActivity extends Activity {
8      ProgressBar pbBarra = null;
9      int progressStatus = 0;
10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_progress_bar);
15
16         pbBarra = (ProgressBar) findViewById(R.id.pbBarra);
17
18         new Thread(new Runnable() {
19             public void run() {
20                 while(progressStatus < 100) {
21                     progressStatus++;
22                     pbBarra.post(new Runnable() {
23                         @Override
24                         public void run() {
25                             pbBarra.setProgress(progressStatus);
26                         }
27                     });
28
29                     try {
30                         Thread.sleep(100);
31                     } catch(InterruptedException e) {
32                         e.printStackTrace();
33                     }
34                 }
35
36                 pbBarra.post(new Runnable() {
37                     @Override
38                     public void run() {
39                         pbBarra.setVisibility(ProgressBar.GONE);
40                     }
41                 });
42             }
43         }).start();
```

COMPONENTE PROGRESSBAR

- É possível mudar o estilo de visualização do componente a partir do arquivo XML. Para isso, altere a declaração de ProgressBar, conforme listagem abaixo:

```
<ProgressBar
    android:id="@+id/pbBarra"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    style="?android:attr/progressBarStyleHorizontal"
    android:max="100" />
```

COMPONENTE PROGRESSBAR

- Altere também o seguinte trecho do arquivo Java para tornar a barra de progressão mais funcional:

```
while(progressStatus < 100) {
    progressStatus++;
    pbBarra.post(new Runnable() {
        @Override
        public void run() {
            pbBarra.setProgress(progressStatus);
        }
    });

    try {
        Thread.sleep(100);
    } catch(InterruptedException e) {
        e.printStackTrace();
    }
}

pbBarra.post(new Runnable() {
    @Override
    public void run() {
        pbBarra.setVisibility(ProgressBar.GONE);
    }
});
```