

# Android

## Interfaces Gráficas com Adapters e Dialogs



Softblue  
cursos online

---

---

---


---

---

---

---

Tópicos Abordados



- Adapters
- Listas
  - ListView
  - ListFragment
- GridViews
- Spinners
- AutoCompleteTextView
- Dialogs
  - Criando e exibindo dialogs
  - Dialogs com listas
  - Dialogs customizados
  - TimePicker
  - DatePicker

---

---

---


---

---

---

---

Adapters



- Os adapters fazem uma "ponte" entre os dados que você quer exibir na tela e as views
- Diversas views fazem o uso de adapters para exibir dados
  - *ListView*
  - *Spinner*
  - *GridView*
  - *AutoCompleteTextView*

---

---

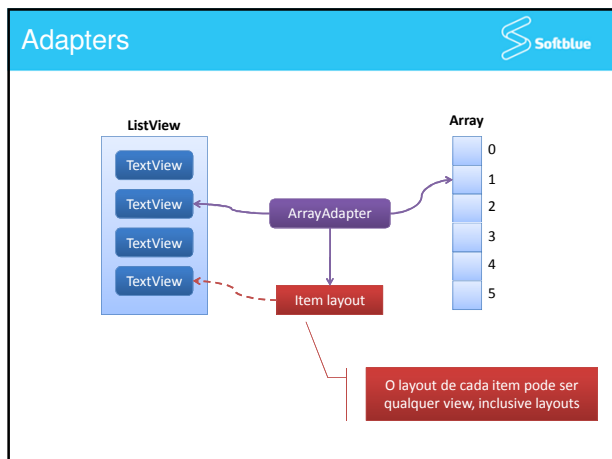
---

---

---

---

---




---

---

---

---

---

---

---

---

- ## Adapters do Android
- O Android possui alguns adapters nativos
  - Dois muito utilizados
    - *ArrayAdapter*
      - Um array é usado como fonte de dados para a view
      - O tipo destes dados pode ser parametrizado via generics
    - *SimpleCursorAdapter*
      - Um cursor é usado como fonte de dados para a view
      - Usado quando os dados vêm de um banco de dados ou de um content provider

---

---

---

---

---

---

---

---

## Adapters do Android

- Você pode criar seus próprios adapters, quando os existentes não atendem a sua necessidade
  - Herança de *BaseAdapter*
  - Herança de um adapter existente
- O método **getView()** deve ser sobrescrito

```

public View getView(
    int position,
    View convertView,
    ViewGroup parent) {
    //...
}
  
```

Retorna a view de cada item do adapter

---

---

---

---


---

---

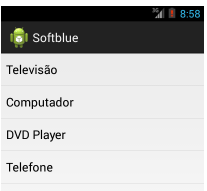
---

---

ListView



- Lista itens na vertical e permite a seleção de itens
- Um adapter pode ser utilizado para customizar a forma como cada item da lista será mostrado



---

---

---

---


---

---

---

---

Criando ListViews



- Quando uma activity é composta apenas pela lista, basta herdá-la de **ListActivity**
- **setListAdapter()** é usado na definição do adapter

```

public class MyActivity extends ListActivity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ArrayAdapter<CharSequence> adapter = ArrayAdapter
            .createFromResource(this, R.array.eletronicos,
                android.R.layout.simple_list_item_1);

        setListAdapter(adapter);
    }
}

```

/res/values/arrays.xml

```

<resources>
<string-array name="eletronicos">
<item>Televisão</item>
<item>Computador</item>
<item>DVD Player</item>
<item>Telefone</item>
</string-array>
</resources>

```

---

---

---

---


---

---

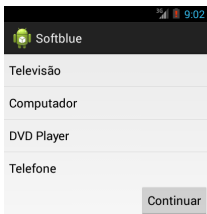
---

---

Criando ListViews



- Se a activity não for composta apenas pela *ListView*, é possível declarar a view num arquivo de layout, juntamente com os outros elementos necessários



---

---

---

---

---

---

---

---

3

## Criando ListViews



/res/layout/lista.xml

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ListView
        android:id="@+id/list_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />

    <Button
        android:id="@+id/continuar"
        android:text="@string/txt_continuar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

</LinearLayout>
```

---

---

---

---

---

---

---

## Criando ListViews



```
public class MyActivity extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.lista);

        ListView list = (ListView) findViewById(R.id.list_view);

        ArrayAdapter<CharSequence> adapter = ArrayAdapter
            .createFromResource(this, R.array.eletronicos,
                android.R.layout.simple_list_item_1);

        list.setAdapter(adapter);
    }
}
```

---

---

---

---

---

---

---

## Escolhendo Itens da ListView



- É possível registrar um listener na *ListView* para ser chamado quando determinado elemento é clicado

```
public class MyActivity extends Activity
    implements OnItemClickListener {

    public void onCreate(Bundle savedInstanceState) {
        //...
        list.setOnItemClickListener(this);
    }

    public void onItemClick(AdapterView<?> parent, View view,
        int position, long id) {
        //...
    }
}
```

---

---

---

---

---

---

---

## Escolhendo Itens da ListView



- No caso da activity herdar de *ListActivity*, o método **onListItemClick()** pode ser sobrescrito

```
public void onListItemClick(ListView l, View v, int position,
    long id) {
    //...
}
```

---

---

---

---

---

---

---

## A Classe *ListFragment*



- Com o surgimento dos fragments, foi criada a classe **ListFragment**
- Seu funcionamento é igual ao *ListActivity*, mas é usada em fragments

```
public class MyFragment extends ListFragment {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ArrayAdapter<CharSequence> adapter = ArrayAdapter
            .createFromResource(this, R.array.eletronicos,
                android.R.layout.simple_list_item_1);

        setListAdapter(adapter);
    }
}
```

---

---

---

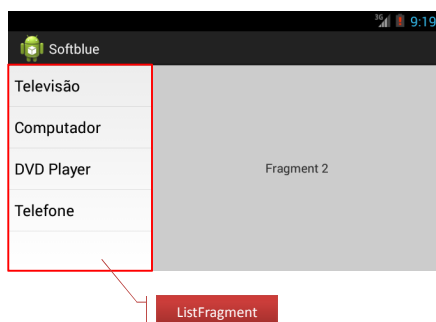
---

---

---

---

## A Classe *ListFragment*



---

---

---

---

---

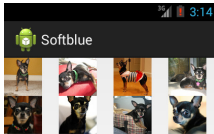
---

---

## GridView



- Funciona como o ListView, mas os itens são exibidos em forma de grade usando linhas e colunas
  - Mais utilizado com exibição de imagens
- Seu funcionamento é muito semelhante ao funcionamento de ListViews



---

---

---

---

---

---

---

## Criando GridViews



/res/layout/grid.xml

```
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/grid"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:columnWidth="40dp"
    android:horizontalSpacing="10dp"
    android:numColumns="4" />
```

---

---

---

---

---

---

---

## Spinner



- Um spinner é uma caixa de seleção com itens pré-definidos que podem ser escolhidos



---

---

---

---

---

---

---

## Criando Spinners



/res/layout/activity\_main.xml

```
<LinearLayout
...

<Spinner
    android:id="@+id/spinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
/>
</LinearLayout>
```

---

---

---

---

---

---

---

## Criando Spinners



```
public class MyActivity extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);
        Spinner spinner = (Spinner) findViewById(R.id.spinner);

        ArrayAdapter<CharSequence> adapter = ArrayAdapter
            .createFromResource(this, R.array.eletronicos,
                android.R.layout.simple_spinner_item);

        adapter.setDropDownViewResource(
            android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);
    }
}
```

---

---

---

---

---

---

---

## Detectando o Elemento Escolhido



- É possível registrar um listener do tipo **AdapterView.OnItemSelectedListener()** para ser chamado quando um item é escolhido

```
public class MainActivity extends Activity implements
    AdapterView.OnItemSelectedListener {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Spinner spinner = (Spinner) findViewById(R.id.spinner);
        spinner.setOnItemSelectedListener(this);
    }

    public void onItemSelected(AdapterView<?> parent, View view,
        int position, long id) {
    }

    public void onNothingSelected(AdapterView<?> parent) {
    }
}
```

---

---

---

---

---

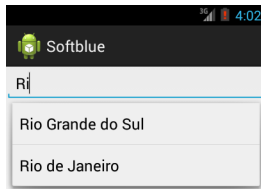
---

---

## AutoCompleteTextView



- É uma caixa de texto que mostra sugestões enquanto o texto está sendo digitado



---

---

---

---

---

---

---

## Criando AutoCompleteTextViews



```
/res/layout/activity_main.xml  
  
<LinearLayout  
...  
    <AutoCompleteTextView  
        android:id="@+id/edt_estado"  
        android:layout_width="match_parent"  
        android:layout_height="wrap_content"  
    />  
</LinearLayout>
```

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    AutoCompleteTextView view =  
        (AutoCompleteTextView) findViewById(R.id.edt_estado);  
  
    ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(  
        this, R.array.estados, android.R.layout.simple_list_item_1);  
    view.setAdapter(adapter);  
}
```

---

---

---

---

---

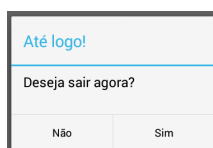
---

---

## Dialogs



- Dialogs são caixas de informações que servem para alertar o usuário sobre algo ou para solicitar algum tipo de informação
- Duas classes principais
  - **AlertDialog**: Representa um dialog
  - **AlertDialog.Builder**: Constrói um dialog



---

---

---

---

---

---

---



## A Classe *DialogFragment*



- A forma recomendada de exibir dialogs é através do uso de um **DialogFragment**
  - Classe que herda de *Fragment*
- Permite o correto gerenciamento do ciclo de vida do dialog
- Possibilita o reaproveitamento do dialog em várias partes da aplicação
- A classe *DialogFragment* surgiu na versão 3.0 do Android (API Level 11)
  - É preciso usar a API de compatibilidade se a aplicação for executada em dispositivos com versão anterior

---

---

---

---

---

---

---

## Criando um Dialog



```
public class MyDialog extends DialogFragment
    implements DialogInterface.OnClickListener {

    public Dialog onCreateDialog(Bundle savedInstanceState) {
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        builder.setTitle("Até logo!")
            .setMessage("Deseja sair agora?")
            .setPositiveButton(R.string.sim, this)
            .setNegativeButton(R.string.nao, this);

        return builder.create();
    }

    public void onClick(DialogInterface dialog, int which) {
        if (which == Dialog.BUTTON_NEGATIVE) {
            //...
        } else if (which == Dialog.BUTTON_POSITIVE) {
            //...
        }
    }
}
```

---

---

---

---

---

---

---

## Exibindo um Dialog



- A exibição é feita pela activity através da instanciação do dialog e da chamada ao método **show()**

```
FragmentManager fm = getFragmentManager();
MyDialog dialog = new MyDialog();
dialog.show(fm, null);
```

FragmentManager

Identificador do  
fragment (tag)

---

---

---

---


---

---


---

## Criando Dialogs com Listas Softblue

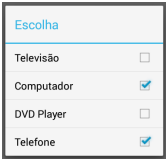
- Um dialog pode ter uma lista associada
- Pode ser de três tipos



Lista Simples



Lista Simples com Radio Buttons



Lista Múltipla com Checkboxes

---

---

---

---

---

---

---

---

## Lista Simples Softblue

```

public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog
        .Builder(getActivity());

    builder.setTitle("Escolha")
        .setItems(R.array.eletronicos, this);

    return builder.create();
}

public void onClick(DialogInterface dialog, int which) {
    //...
}
```

which armazena o índice do elemento escolhido

---

---

---

---

---

---

---

---

## Lista Simples com Radio Buttons Softblue

```

public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog
        .Builder(getActivity());

    builder.setTitle("Escolha")
        .setSingleChoiceItems(R.array.eletronicos, -1, this);

    return builder.create();
}

public void onClick(DialogInterface dialog, int which) {
    //...
}
```

which armazena o índice do elemento escolhido

---

---

---

---


---

---

---

---

Lista Múltipla com Checkboxes



DialogInterface.  
OnMultiChoiceClickListener

```

public Dialog onCreateDialog(Bundle savedInstanceState) {
    AlertDialog.Builder builder = new AlertDialog
        .Builder(getActivity());

    builder.setTitle("Escolha")
        .setMultiChoiceItems(R.array.eletronicos, null, this);

    return builder.create();
}

public void onClick(DialogInterface dialog, int which,
    boolean isChecked) {
    //...
}

```

isChecked indica se o item está selecionado

which armazena o índice do elemento clicado

---

---

---

---


---

---

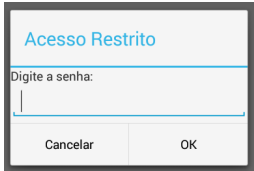
---

---

AlertDialogs Customizados



- Para uma maior flexibilidade sobre a aparência de um *AlertDialog*, é possível definir uma view a ser aplicada ao dialog



---

---

---

---

---

---

---

---

AlertDialogs Customizados



```

/res/layout/pwd_dialog.xml

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

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

    <EditText
        android:id="@+id/senha"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="numberPassword" />

</LinearLayout>

```

---

---

---

---

---

---

---

---

AlertDialogs Customizados


```

AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setTitle("Acesso Restrito");

LayoutInflater inflater = getLayoutInflater();
View dialogView = inflater.inflate(R.layout.pwd_dialog, null);
builder.setView(dialogView);

builder.setPositiveButton(R.string.ok, this);
builder.setNegativeButton(R.string.cancelar, this);
...

```

---

---

---


---

---

---

---

---

Activity como um Dialog


- Uma activity pode assumir a aparência de um dialog
- Basta mudar o tema da activity

AndroidManifest.xml

```

<activity
    android:name="softblue.android.MainActivity"
    android:theme="@android:style/Theme.Dialog">

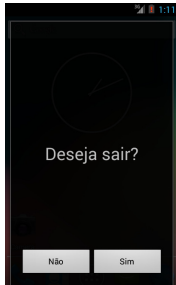
```

- No método `onCreate()` é possível esconder o título da activity

```

requestWindowFeature(Window.FEATURE_NO_TITLE);

```



---

---

---

---

---

---

---

---

TimePickerDialog


- O Android possui um dialog que permite escolher uma hora e minuto do dia
- Classe **TimePickerDialog**



---

---

---

---

---

---

---

---

Criando um TimePickerDialog

- É utilizada a já conhecida classe *DialogFragment*

```

public class MyDialog extends DialogFragment
    implements TimePickerDialog.OnTimeSetListener {

    public Dialog onCreateDialog(Bundle savedInstanceState) {
        return new TimePickerDialog(getActivity(), this, 16, 30, true);
    }

    public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
        //...
    }
}

```

Listener chamado quando a hora é definida

Retorna instância de TimePickerDialog

Método do listener

---

---

---

---

---

---

---

---

Usando um TimePickerDialog

- A activity deve chamar o método **show()**, como funciona para qualquer dialog

```

FragmentManager fm = getFragmentManager();
MyDialog dialog = new MyDialog();
dialog.show(fm, null);

```

---

---

---

---

---

---

---

---

DatePickerDialog

- O Android possui um dialog que permite escolher uma data
- Classe **DatePickerDialog**

---

---

---

---

---

---

---

---

### Criando um DatePickerDialog



- É utilizada a já conhecida classe *DialogFragment*

```
public class MyDialog extends DialogFragment
    implements DatePickerDialog.OnDateSetListener {

    public Dialog onCreateDialog(Bundle savedInstanceState) {
        return new DatePickerDialog(getActivity(), this, 2030, 9, 5);
    }

    public void onDateSet(DatePicker view, int year, int month, int day) {
        //...
    }
}
```

Listener chamado quando a data é definida

Retorna instância de DatePickerDialog

Método do listener

---

---

---


---

---

---

---

### Usando um DatePickerDialog



- A activity deve chamar o método **show()**, como funciona para qualquer dialog

```
FragmentManager fm = getFragmentManager();
MyDialog dialog = new MyDialog();
dialog.show(fm, null);
```

---

---

---

---

---

---

---



---

---

---

---

---

---

---