

# Android GUI

Visão geral da interface gráfica do usuário

Alisson G. Chiquitto

<sup>1</sup>Instituto Federal do Mato Grosso do Sul

Naviraí, 2018

# Outline - Seção

- 1 Introdução
- 2 Controles de entrada
- 3 Eventos de entrada
- 4 Referências

Todos os elementos da interface do usuário em um aplicativo para Android são criados usando objetos `View` e `ViewGroup`.

Uma `View` é um objeto que desenha algo na tela com o qual o usuário pode interagir.

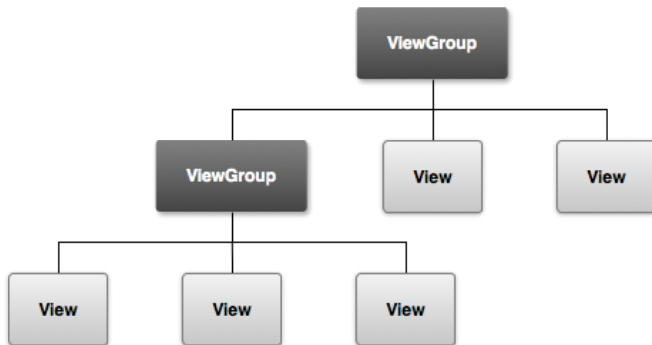
Exemplos: `Button`, `ImageView` e `TextView`.

# ViewGroup

- ViewGroup herda View;
- é um *container* para outros objetos View (e ViewGroup);
- organiza os elementos filhos;

# Árvore de componentes

A UI é definida usando uma hierarquia de objetos `View` e `ViewGroup`.



# Declaração de layout

Para declarar o layout, é possível instanciar objetos View no código Java, mas a forma mais fácil e efetiva de definir o layout é com um arquivo XML.

# Exemplo de XML de layout

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:layout_width="fill_parent"
4   android:layout_height="fill_parent"
5   android:orientation="vertical">
6   <TextView android:id="@+id/text"
7     android:layout_width="wrap_content"
8     android:layout_height="wrap_content"
9     android:text="I am a TextView" />
10  <Button android:id="@+id/button"
11    android:layout_width="wrap_content"
12    android:layout_height="wrap_content"
13    android:text="I am a Button" />
14 </LinearLayout>
```

# Componentes de interface do usuário

- o Android fornece componentes que oferecem um layout de UI;
- cada componente possui uma API de acesso;



## 1 Introdução

## 2 Controles de entrada

- Botão
- Campo de texto
- Caixa de seleção
- Botão de opção
- Botão de alternar
- Controle giratório
- Seletores

## 3 Eventos de entrada

## 4 Referências

Controles de entrada são os componentes interativos na interface do usuário.

O Android oferece vários controles que podem ser usados na IU, como botões, campos de texto, barras de busca, caixas de seleção, botões de zoom, entre outros.

## 2 Controles de entrada

- Botão
- Campo de texto
- Caixa de seleção
- Botão de opção
- Botão de alternar
- Controle giratório
- Seletores

# Controle: Botão



Um botão consiste de um texto ou um ícone (ou ambos) que comunica que uma ação ocorreu que o usuário tocou ele.

Classes relacionadas:

- [Button Reference](#)
- [Button Guide](#)

# Controle: Botão

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:layout_width="fill_parent" android:layout_height="fill_parent"
4   android:orientation="horizontal">
5
6   <Button
7     android:layout_width="wrap_content" android:layout_height="wrap_content"
8     android:text="@string/button_text" />
9
10  <ImageButton
11    android:layout_width="wrap_content" android:layout_height="wrap_content"
12    android:src="@drawable/button_icon" />
13
14  <Button
15    android:layout_width="wrap_content" android:layout_height="wrap_content"
16    android:text="@string/button_text"
17    android:drawableLeft="@drawable/button_icon" />
18
19 </LinearLayout>
```

## 2 Controles de entrada

- Botão
- Campo de texto
- Caixa de seleção
- Botão de opção
- Botão de alternar
- Controle giratório
- Seletores

# Controle: Campo de texto

Informe um número

Campo de texto editável. É possível usar o widget `AutoCompleteTextView` para criar um widget de entrada de texto que forneça sugestões para preenchimento automático.

Classes relacionadas:

- `EditText`
- `AutoCompleteTextView`
- `Provide Auto-complete Suggestions`

# Controle: Campo de Texto

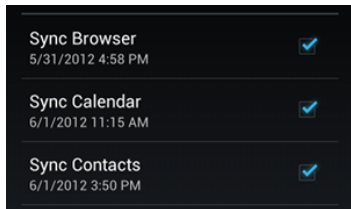
```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:layout_width="fill_parent" android:layout_height="fill_parent"
4   android:orientation="horizontal">
5
6   <EditText
7     android:id="@+id/phone"
8     android:layout_width="fill_parent" android:layout_height="wrap_content"
9     android:hint="@string/phone_hint"
10    android:inputType="phone" />
11
12   <AutoCompleteTextView
13     android:id="@+id/autocomplete_country"
14     android:layout_width="fill_parent"
15     android:layout_height="wrap_content" />
16
17 </LinearLayout>
```



## 2 Controles de entrada

- Botão
- Campo de texto
- Caixa de seleção
- Botão de opção
- Botão de alternar
- Controle giratório
- Seletores

# Controle: Caixa de seleção



Chave liga/desliga que pode ser alternada pelo usuário. Use caixas de seleção ao apresentar aos usuários um grupo de opções selecionáveis que não sejam mutuamente exclusivas.

Geralmente utilizada em uma lista vertical de opções.

Classes relacionadas:

- [Checkbox Class Reference](#)
- [Checkboxes Guide](#)

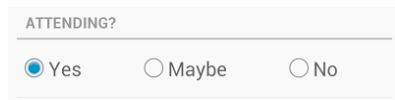
# Controle: Caixa de seleção

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:orientation="vertical"
4   android:layout_width="fill_parent" android:layout_height="fill_parent">
5
6   <CheckBox android:id="@+id/checkbox_meat"
7     android:layout_width="wrap_content" android:layout_height="wrap_content"
8     android:text="@string/meat" />
9
10  <CheckBox android:id="@+id/checkbox_cheese"
11    android:layout_width="wrap_content" android:layout_height="wrap_content"
12    android:text="@string/cheese" />
13
14 </LinearLayout>
```

## 2 Controles de entrada

- Botão
- Campo de texto
- Caixa de seleção
- **Botão de opção**
- Botão de alternar
- Controle giratório
- Seletores

# Controle: Botão de opção



ATTENDING?

☒ Yes ☐ Maybe ☐ No

Similar às caixas de seleção, exceto que somente uma opção pode ser selecionada no grupo.

Classes relacionadas:

- [RadioGroup Class Reference](#)
- [RadioButton Class Reference](#)
- [Radio Buttons Guide](#)

# Controle: Botão de opção

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <RadioGroup xmlns:android="http://schemas.android.com/apk/res/android"
3   android:layout_width="fill_parent" android:layout_height="wrap_content"
4   android:orientation="vertical">
5
6   <RadioButton android:id="@+id/radio_pirates"
7     android:layout_width="wrap_content" android:layout_height="wrap_content"
8     android:text="@string/pirates" />
9
10  <RadioButton android:id="@+id/radio_ninjas"
11    android:layout_width="wrap_content" android:layout_height="wrap_content"
12    android:text="@string/ninjas" />
13
14 </RadioGroup>
```

## 2 Controles de entrada

- Botão
- Campo de texto
- Caixa de seleção
- Botão de opção
- Botão de alternar
- Controle giratório
- Seletores

# Controle: Botão de alternar



Botão liga/desliga que permite o usuário que altere uma configuração entre 2 estados.

Classes relacionadas:

- [ToggleButton Class Reference](#)
- [Toggle Buttons Guide](#)



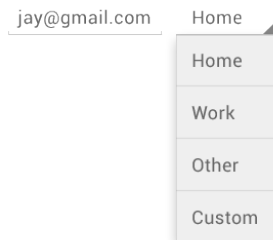
# Controle: Botão de alternar

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:orientation="vertical"
4   android:layout_width="fill_parent" android:layout_height="fill_parent">
5
6   <ToggleButton
7     android:id="@+id/toggleButton"
8     android:layout_width="match_parent" android:layout_height="wrap_content"
9     android:checked="true" />
10
11 </LinearLayout>
```

## 2 Controles de entrada

- Botão
- Campo de texto
- Caixa de seleção
- Botão de opção
- Botão de alternar
- Controle giratório
- Seletores

# Controle: Controle giratório



Lista suspensa que permite aos usuários selecionar um valor de um conjunto.

Classes relacionadas:

- [Spinner Class Reference](#)
- [Spinner Guide](#)

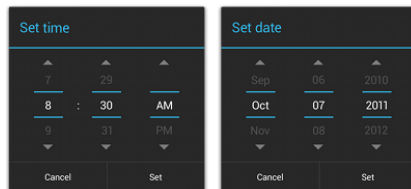
# Controle: Botão de alternar

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:orientation="vertical"
4   android:layout_width="fill_parent" android:layout_height="fill_parent">
5
6   <Spinner
7     android:id="@+id/planets_spinner"
8     android:layout_width="fill_parent"
9     android:layout_height="wrap_content" />
10
11 </LinearLayout>
```

## 2 Controles de entrada

- Botão
- Campo de texto
- Caixa de seleção
- Botão de opção
- Botão de alternar
- Controle giratório
- Seletores

# Controle: Seletores



Controles para o usuário selecionar um horário/data válido. Cada seletor fornece controles para selecionar cada parte do horário/data.

Classes relacionadas:

- [DatePicker Class Reference](#)
- [TimePicker Class Reference](#)
- [Pickers Guide](#)

# Controle: Seletor de data

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:orientation="vertical"
4   android:layout_width="fill_parent"
5   android:layout_height="fill_parent">
6
7   <DatePicker
8     android:layout_width="wrap_content"
9     android:layout_height="wrap_content"
10    android:endYear="2100"
11    android:startYear="1900" />
12
13 </LinearLayout>
```

# Controle: Seletor de horário

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   android:orientation="vertical"
4   android:layout_width="fill_parent"
5   android:layout_height="fill_parent">
6
7   <TimePicker
8     android:id="@+id/timePicker"
9     android:layout_width="match_parent"
10    android:layout_height="wrap_content" />
11
12 </LinearLayout>
```



# Outline - Seção

- 1 Introdução
- 2 Controles de entrada
- 3 Eventos de entrada**
  - Listeners
  - Formas de interceptação
- 4 Referências

# Eventos de entrada

- há mais de uma maneira de interceptar os eventos da interação de um usuário;
- a classe `View` fornece os meios para fazer isso.

Mais em [Eventos de entrada](#)

## 3 Eventos de entrada

- Listeners

- OnClickListener
- OnLongClickListener
- onTouchListener
- Formas de interceptação

Um *listener* é uma *interface* que contém um único método de retorno de chamada.

Esse método será chamado quando uma ação da *View* (no qual o *listener* estiver registrada) for ativada pela interação do usuário com a UI.

## 3 Eventos de entrada

- Listeners
  - OnClickListener
  - OnLongClickListener
  - onTouchListener

# Listener: View.OnClickListener

Chamado quando o usuário clica no componente (ou atribui foco ao item com as teclas de navegação e pressiona a tecla “enter”).

Método: `onClick(View v)`

[View.OnClickListener Reference](#)

# Listener: View.OnClickListener

```
1 OnClickListener onClickListener = new OnClickListener() {  
2     public void onClick(View v) {  
3         // item clicado  
4     }  
5 };
```

## 3 Eventos de entrada

- Listeners

- OnClickListener
- **OnLongClickListener**
- onTouchListener



# Listener: View.OnLongClickListener

Chamado quando o usuário mantém clicado um componente.

Método: `onLongClick(View v)`

[View.OnLongClickListener Reference](#)

# Listener: View.OnLongClickListener

```
1 OnLongClickListener onLongClickListener = new OnLongClickListener() {  
2     public void onLongClick(View v) {  
3         // item clicado  
4     }  
5 };
```

## 3 Eventos de entrada

- Listeners

- OnClickListener
- OnLongClickListener
- onTouchListener

# Listener: View.OnTouchListener

Chamado quando o usuário realiza uma ação com o toque, incluindo o pressionamento a liberação ou qualquer outro ação (dentro dos limites do componente).

Método: `onTouch(View v)`

[View.OnTouchListener Reference](#)

# Listener: View.OnTouchListener

```
1 OnTouchListener onTouchListener = new OnTouchListener() {  
2     public void onTouch(View v) {  
3         // item clicado  
4     }  
5 };
```

## 3 Eventos de entrada

- Listeners
- Formas de interceptação
  - Instância de um listener
  - Implementar um listener na Activity
  - Listener via XML

# Formas de interceptação

Há 3 formas práticas de interceptação de um evento:

- instanciar um *listener* e passa-lo para a *view*;
- implementar um *listener* na *activity*;
- *listener* via XML

## 3 Eventos de entrada

- Formas de interceptação
  - Instância de um listener
  - Implementar um listener na Activity
  - Listener via XML



# Instanciar um listener

Esta forma de interceptar um evento é preciso instanciar uma classe anônima de uma interface *listener*.

Depois é necessário adicionar a instância do *listener* ao objeto View.

# Instanciar um listener

```
1 class MainActivity {  
2  
3     // Criar uma implementação anonima de OnClickListener  
4     private OnClickListener clickListener = new OnClickListener() {  
5         public void onClick(View v) {  
6             // fazer algumas coisa quando o botão for clicado  
7         }  
8     };  
9  
10    protected void onCreate(Bundle savedInstanceState) {  
11        // Capturar o botão do layout  
12        Button button = (Button)findViewById(R.id.botao);  
13  
14        // Registrar o listener onClick no botão  
15        button.setOnClickListener(clickListener);  
16    }  
17  
18 }
```

## 3 Eventos de entrada

- Formas de interceptação
  - Instância de um listener
  - Implementar um listener na Activity
  - Listener via XML

# Activity implementando um listener

Nesta forma, é preciso que a *activity* implemente uma interface de um *listener*.

Os métodos definidos na interface deverão ser implementados na classe *activity*.

Este método receberá todas as interceptações dos eventos que utilizam a *activity* como *listener*.

# Activity implementando um listener

```
1 // Implementar interface do listener
2 public class ExampleActivity extends Activity
3 implements OnClickListener {
4
5     protected void onCreate(Bundle savedInstanceState) {
6         Button button = (Button)findViewById(R.id.botao);
7         button.setOnClickListener(this);
8     }
9
10    // Implementar o callback (método) do listener
11    public void onClick(View v) {
12        // filtrar pelo ID dos componentes
13        switch (v.getId()) {
14            case R.id.botao:
15                // faça alguma coisa aqui
16                break;
17        }
18    }
19 }
```

## 3 Eventos de entrada

- Formas de interceptação
  - Instância de um listener
  - Implementar um listener na Activity
  - Listener via XML

# Implementando um listener pelo XML

Nesta forma, o *listener* será implementado no layout (XML).

O método correspondente deve existir na *activity*.

# Implementando um listener pelo XML

## Layout:

```
1 <Button android:id="@+id/mybutton"  
2   android:onClick="botaoClicado" />
```

## Activity:

```
1 public class ExampleActivity extends Activity {  
2  
3   public void botaoClicado(View v) {  
4     // fazer alguma coisa interessante  
5   }  
6  
7 }
```



# Outline - Seção

- 1 Introdução
- 2 Controles de entrada
- 3 Eventos de entrada
- 4 Referências**

-  Visão geral da IU | Android Developers  
<https://developer.android.com/guide/topics/ui/overview.html>
-  Controles de entrada | Android Developers  
<https://developer.android.com/guide/topics/ui/controls.html>
-  Eventos de entrada | Android Developers  
<https://developer.android.com/guide/topics/ui/ui-events.html>
-  Basic Event Listeners  
[https://github.com/codepath/android\\_guides/wiki/Basic-Event-Listeners](https://github.com/codepath/android_guides/wiki/Basic-Event-Listeners)