



Programação para Android

Aula 02 - Parte 01: Tipos de Layouts: RelativeLayout, LinearLayout, TableLayout

Widgets: TextView, EditText, ImageView, Button

Na aula anterior...

- ▶ Visão geral do Android
- ▶ Kit de desenvolvimento: Eclipse e SDK Manager
- ▶ Configuração do ambiente de desenvolvimento
- ▶ Criação de projetos e execução em dispositivo físico e virtual



Objetivos

- ▶ Finalizar a configuração do ambiente de trabalho
- ▶ Configuração do dispositivo físico de testes
- ▶ Conhecer a estrutura de um projeto de aplicação Android
- ▶ Diferenciar os tipos de layout para aplicativos Android
- ▶ Adicionar e configurar componentes visuais (widgets) em uma Activity



Parte 01: Layout

Layout

- ▶ No Android existem diversos tipos de componentes gráficos:
 - ▶ **TextView:** Mostra um texto na tela. É com certeza o componente gráfico mais usado em Android.
 - ▶ **ImageView:** Mostra uma imagem ou simplesmente uma janela na tela. Toda vez que você for mostrar uma imagem ou mostrar ao usuário uma janela colorida, por exemplo, esse componente será usado.
 - ▶ **EditTex:** Os componentes acima apenas mostram informações na tela. Já o *EditText* obtém um texto digitado pelo usuário, que poderá ser usado para interagir com a aplicação Android.
 - ▶ **Button:** Este componente é um dos mais comuns em qualquer sistema de layout. Neste componente, uma ação é executada após um clique ser dado nele.
 - ▶ **CheckBox:** Um componente que basicamente possui dois valores: verdadeiro ou falso. Muito usado para representar configurações do sistema.
 - ▶ Outros: RadioGroup, ListView, GridView, Spinner, SeekBart, etc...
- ▶ Para que estes esteja organizados e apareçam na tela do dispositivo eles precisam está inseridos em um Layout.

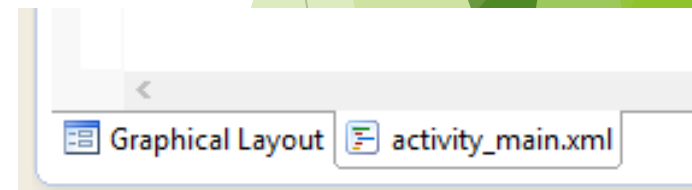
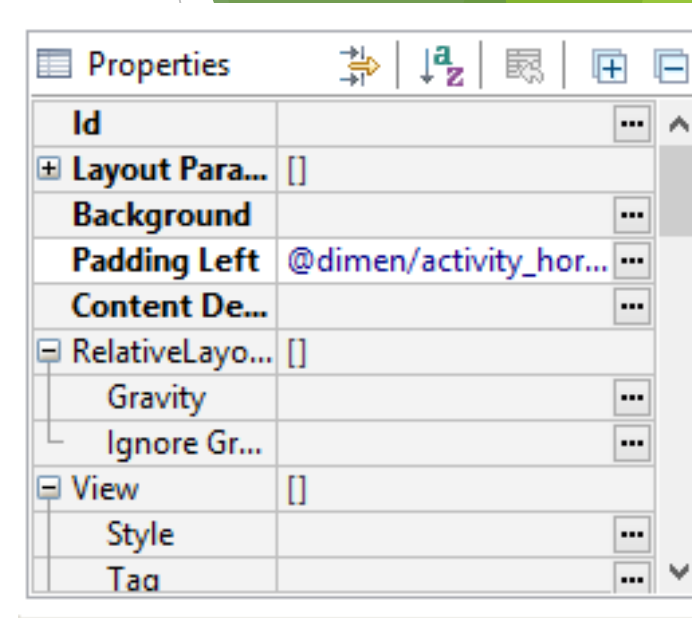
Layout

- ▶ Um layout define a estrutura visual e organiza os componentes gráficos para compor a interface do usuário.
- ▶ Existem diversos tipos de layout em Android. Em geral **3 tipos de layouts são os mais utilizados:**
 - ▶ LinearLayout
 - ▶ RelativeLayout
 - ▶ TableLayout



LinearLayout

- ▶ Este layout é utilizado para dispor seus componentes em uma única direção (horizontal ou vertical)
- ▶ Devemos configurar um layout ou qualquer componente gráficos através das propriedades de cada elemento que pode ser configurada de forma visual ou via código.
- ▶ **OBS: Não são todas as propriedades que podem ser configuradas visualmente. Algumas só podem ser definidas via código.**
- ▶ Para configurar as propriedades de forma visual, clicamos no elemento visual a ser configurado (layout ou componente gráfico) e realizamos as alterações através da janela **PROPERTIES**.
- ▶ Para manipular as propriedades via código XML, clicamos na opção com o nome da activity.xml. Para visualizar de forma gráfica clicamos em Graphical Layout.



LinearLayout

- ▶ Na criação de layouts em Android, sempre chamamos um parâmetro usando o prefixo "*android:*" e em seguida um pósfixo que define qual propriedades será manipulada.
- ▶ Exemplo: `android:layout_width="match_parent"`
- ▶ Este parâmetro define o tamanho da largura da tela que estamos criando. Este tamanho pode ser um tamanho fixo (em pixels, density pixels, ou outras unidades de formatação) ou em tamanhos expansíveis.
- ▶ Existem 3 tamanhos expansíveis em Android:
 - ▶ **fill_parent:** Com esta opção, o tamanho do parâmetro será máximo (ou seja, o tamanho da tela corrente)
 - ▶ **wrap_content:** Com esta opção, o tamanho do parâmetro será mínimo, tendo como base os componentes-filhos do layout atual.
 - ▶ **match_parent:** Mantém o tamanho herdado pelo componente-pai. Caso não haja um componente-pai, o tamanho será máximo (ou seja, o tamanho da tela).

LinearLayout

► Mais propriedades:

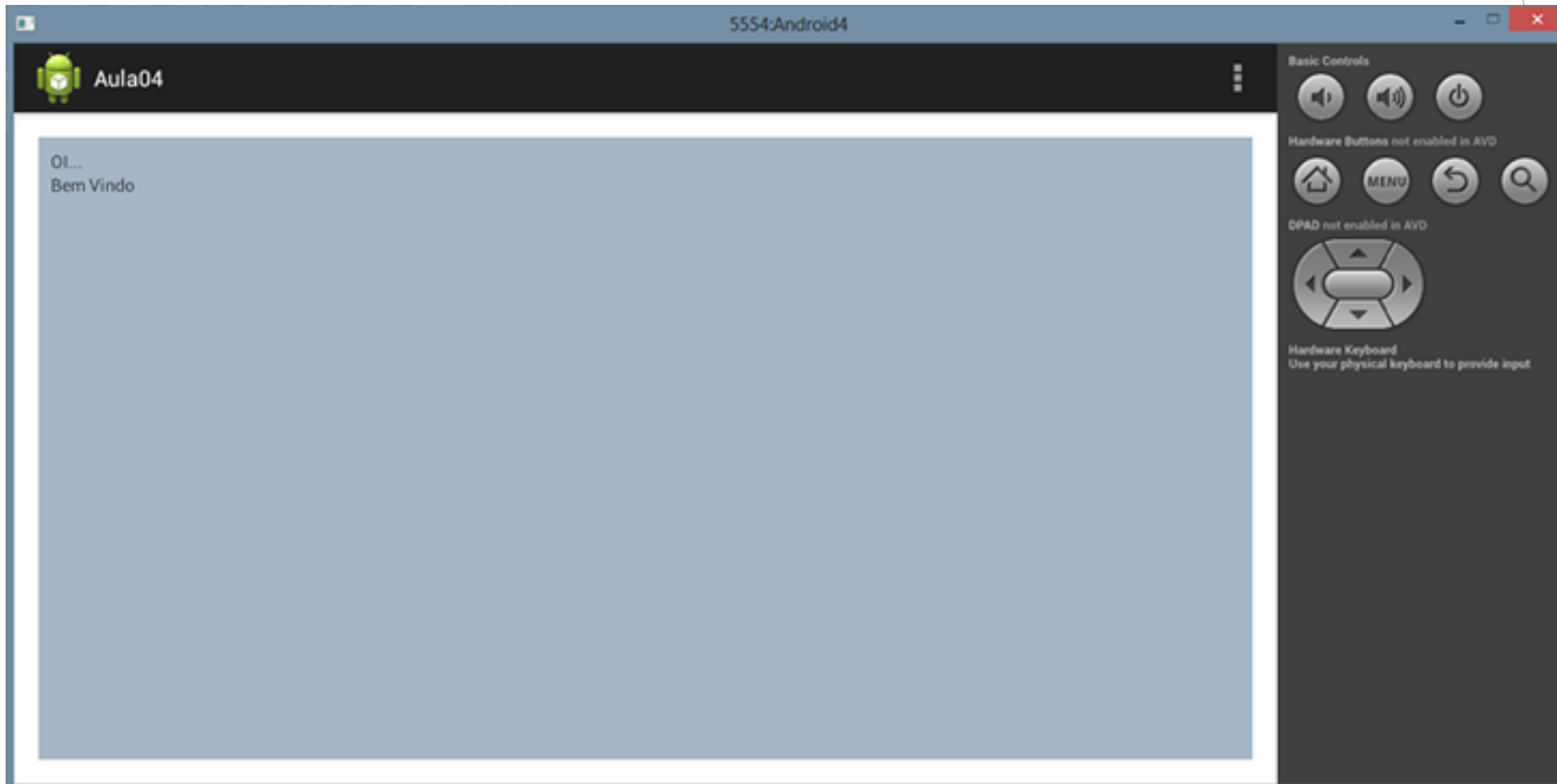
- **android:layout_height="match_parent"**: Este parâmetro define a altura da tela que estamos criando.
- **android:gravity="center_horizontal"**: Este parâmetro define o alinhamento que os componentes deste layout terão. Neste caso, o alinhamento será central e horizontal.
- **android:text="Exemplo"**: Este parâmetro define o texto do componente atual.
- **android:id="@+id/linearDados"**: Este parâmetro define um identificador ao elemento gráfico. Caso seja necessário manipular este elemento via código Java, usamos este identificador para acessá-lo
- **android:background="#A5B6C7"**: Este parâmetro define um background ao Layout que pode ser uma cor em formato hexadecimal ou uma imagem

LinearLayout

► Mais propriedades:

- **android:orientation="vertical"** : Este parâmetro define a orientação dos elementos na tela, que pode ser na vertical ou horizontal.
- **android:padding="10dp"**: Define o espaçamento dos componentes gráficos (TextView, EditText, etc) em relação ao layout
- **android:margin="10dp"**: Define a margem do elemento gráfico em relação ao elemento que o contém.
- OBS: as propriedades padding e margin podem ser configuradas separadamente usando as direções: Top, Bottom, Left, Right. Quando estas propriedades não estão configuradas separadamente, o valor definido é aplicado em todas as direções.

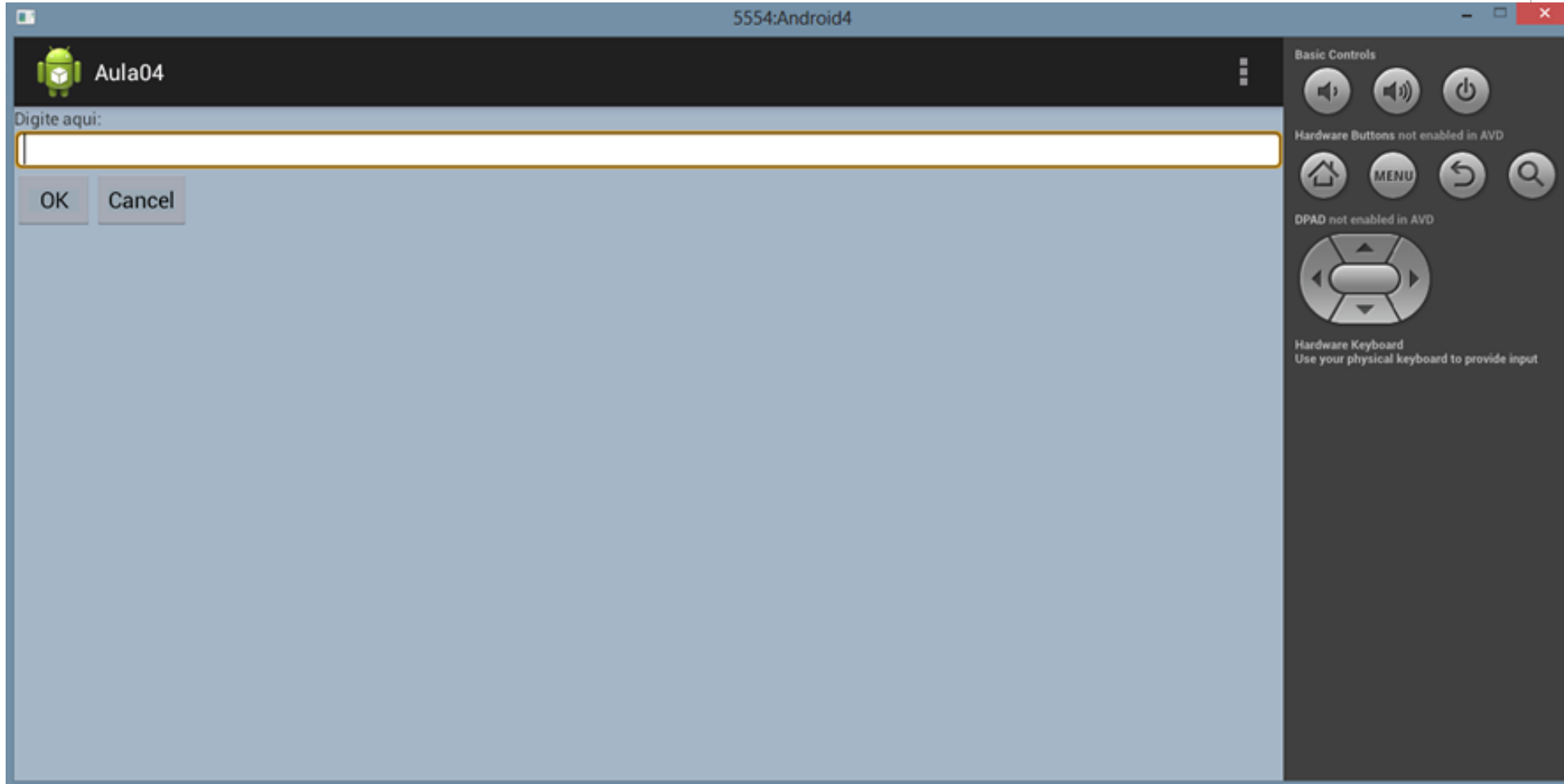
Exemplo 01 - Boas vindas com LinearLayout



RelativeLayout

- ▶ Relative Layout é um layout do tipo relativo, ou seja, ao contrário do Linear Layout, que especifica sempre uma direção horizontal ou vertical, no relative layout posicionamos os elementos por referência à outros elementos.
- ▶ Por exemplo, dizemos se o botão estará abaixo de um campo de texto, do lado direito ou até mesmo em cima dele.
- ▶ Para definir o posicionamento dos elementos na tela em um RelativeLayout devemos configurar as propriedades:
 - ▶ `android:layout_below="@id/label"` : define que o elemento gráfico está abaixo de um outro elemento definido pelo parâmetro @id
 - ▶ `android:layout_above="@id/label"`: define que o elemento gráfico está acima de um outro elemento definido pelo parâmetro @id
 - ▶ `android:layout_toRightOf="@id/ok"`: define que o elemento gráfico está a direita de um outro elemento definido pelo parâmetro @id
 - ▶ `android:layout_toLeftOf="@id/ok"`: define que o elemento gráfico está a esquerda de um outro elemento definido pelo parâmetro @id

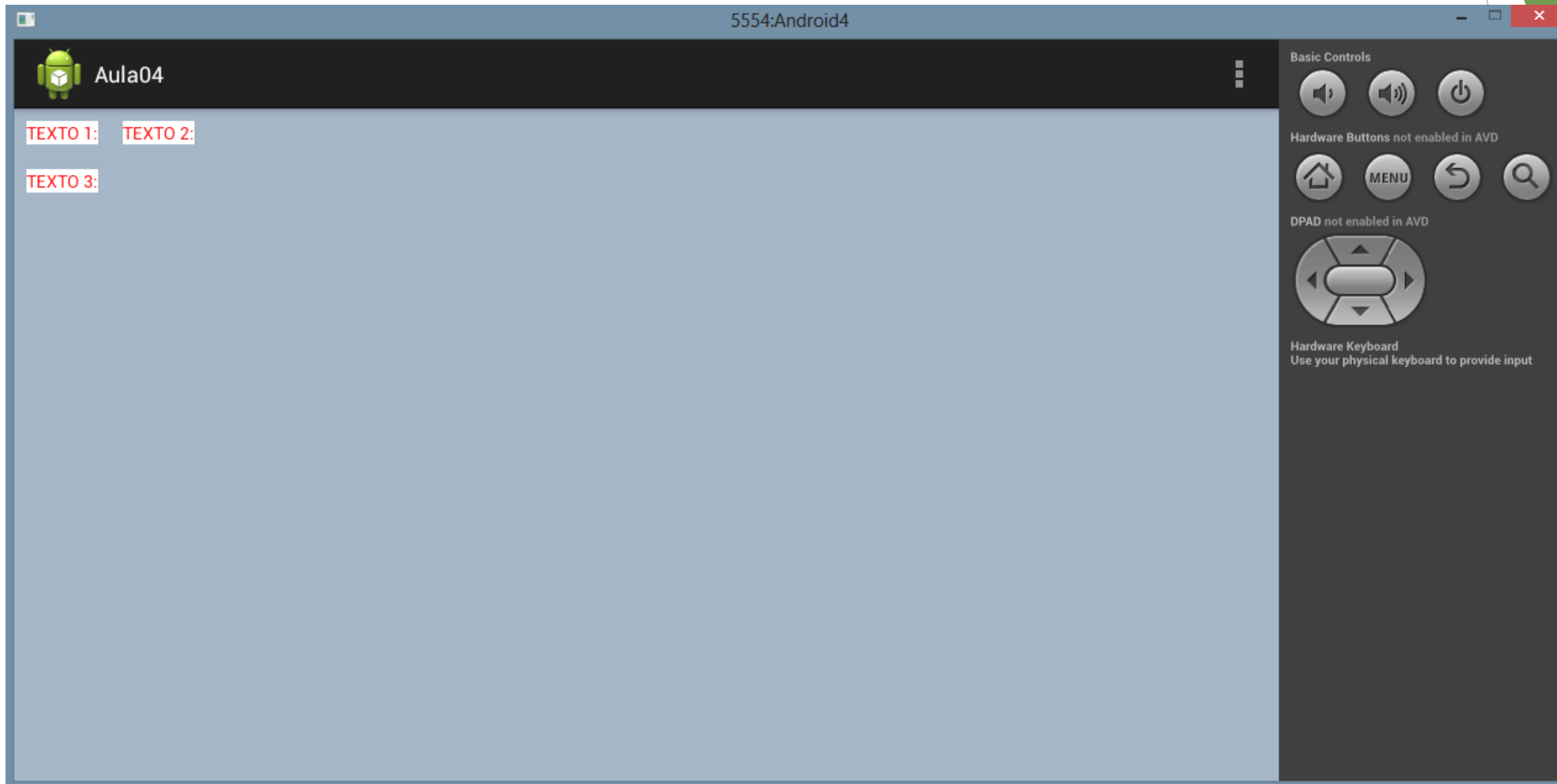
Exemplo 02 - Digite aqui com RelativeLayout



TableLayout

- ▶ Este layout comumente é usado quando precisamos listar vários componentes em uma mesma linha, ao longo de uma mesma tela, no formato de uma tabela. Por exemplo, criar um layout com 18 *TextView's* divididas 3 a 3, ao longo de 6 linhas.
- ▶ Para criar as linhas em um *TableLayout* usamos o componente *TableRow*.
- ▶ O número de colunas no componente *TableLayout* é definido pelo objeto *TableRow* que contém a maioria dos componentes.
- ▶ A altura de cada linha é determinada pelo componente mais alto dessa linha. Da mesma forma, a largura de uma coluna é definida pelo elemento mais largo nessa coluna - a não ser que configuramos para que as colunas da tabela se alonguem para preencher a largura da tela.
- ▶ Por padrão, os componentes são adicionados da esquerda para direita em uma linha, mas podemos especificar o local exato de um componente.
- ▶ OBS: por padrão as linhas e colunas são numeradas a partir de 0.
- ▶ Uma propriedade importante é `android:stretchColumns="0,1"` que indica que as colunas devem ser alongadas horizontalmente,

Exemplo 03 - Layout de tabelas



TableLayout

- ▶ Propriedades importantes:
 - ▶ `android:layout_column="1"`: Por padrão, ao se adicionar elementos em um componente TableRow, o primeiro deles é colocado na primeira coluna, o segundo é colocado na segunda coluna e assim por diante.
 - ▶ Para começar em uma coluna diferente, precisamos especificar o número da coluna. Neste exemplo, estamos especificando que o elemento está na coluna nº 1 (ou seja segunda coluna)



TableLayout

- ▶ Propriedades importantes:
 - ▶ `android_layout_span="2"`. Por padrão cada elemento gráfico é inserido em uma coluna. Caso seja necessário que este elemento ocupe mais de uma coluna, devemos especificar quantas colunas este elemento ocupa (limitado ao número de colunas existente na tabela). Neste exemplo, o elemento ocuparia 2 colunas.

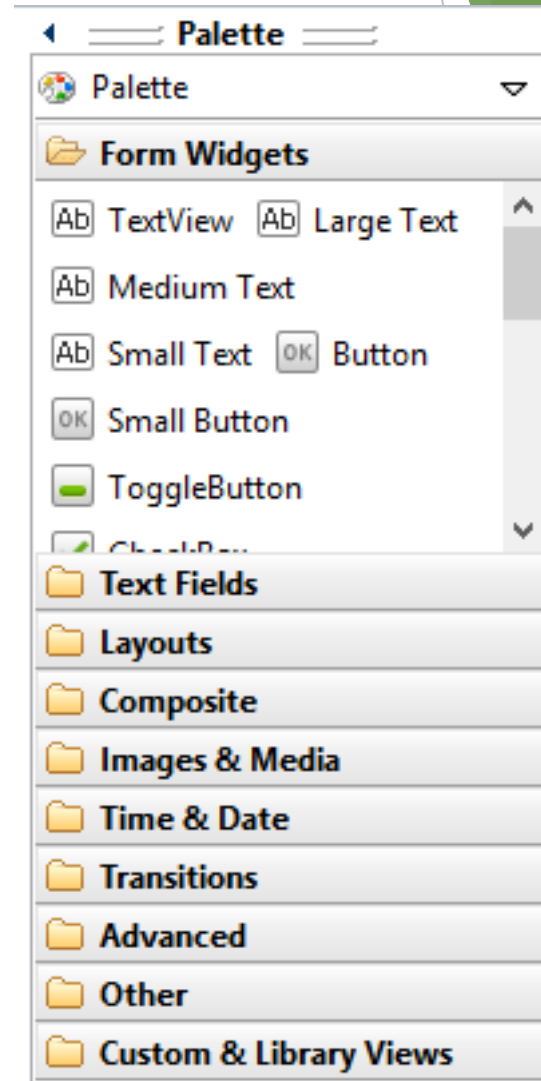


Parte 02: Widgets

TextView

- ▶ O Widget TextView é utilizado para apresentar um texto não editável na tela.
- ▶ Qualquer componente gráfico pode ser adicionado arrastando da paleta até a tela gráfica ou criando o código XML deste elemento.
- ▶ O código XML que representa um TextView pode ser representado:

```
<TextView  
    android:id="@+id/label"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="TEXTO 1:"  
/>
```



TextView

- ▶ Propriedades:
 - ▶ **android:text="Texto 1"**: Este parâmetro define o texto que é exibido na TextView
 - ▶ **android:id="@+id/tvNome "**: Este parâmetro define um identificado textView. Caso seja necessário manipular este elemento via código Java, usamos este identificador para acessá-lo
- ▶ OBS: De acordo com a documentação do android é considerado uma boa prática “exteriorizar” strings, arrays de string, imagens, cores, e outros recursos que você ou outra pessoa possa gerenciá-los separadamente do código de seu aplicativo.
- ▶ Para isto, adicione uma nova String: **res -> values -> strings**
`<string name="nome">Nome</string>`
- ▶ Em seguida configure o parâmentro android:text conforme abaixo
`android:text="@string/nome"`

TextView

- ▶ Propriedades:
 - ▶ **android:textColor="#A5B6C7"**: Este parâmetro define uma cor ao texto exibido. A cor definida deve estar em formato hexadecimal.
 - ▶ **android:textSize="20dp"**: *Este parâmetro define o tamanho do texto.*
 - ▶ **android:textStyle="bold"**: *Define o estilo do texto(negrito, itálico ou normal)*
 - ▶ **android:textAllCaps="true"**: *Define se o texto exibido aparecerá em caixa alta (true) ou em caixa baixa(false)*
 - ▶ **android:layout_gravity="center_horizontal"**: *Define o alinhamento do texto*
 - ▶ **android:typeface="serif"**: *Define os padrões de fonte, ou famílias, que no caso do Android são 3 famílias: Droid Sans, Droid Sans Mono e Droid Serif*

TextView

- ▶ Adicionando sombras:
 - ▶ **android:shadowColor:** cor da sombra
 - ▶ **android:shadowRadius:** o raio da sombra
 - ▶ **android:shadowDx:** o distanciamento horizontal da sombra em relação ao texto
 - ▶ **android:shadowDy:** o distanciamento vertical da sombra em relação ao texto

ImageView

- ▶ O Widget ImageView é usado para adicionar uma imagem em uma activity(tela)
- ▶ Os parâmetros id, gravity, e outras propriedades comuns a todos os widgets são configurados da mesma forma aos já apresentados
- ▶ Definindo a imagem (propriedade src):
`android:src="@drawable/ic_launcher"`
- ▶ OBS: Antes de utilizar uma imagem, é necessário coloca-la na pasta de imagem(@drawable). Para isto copie e cole a imagem na pasta específica.

```
<ImageView  
    android:layout_width="wrap_content"  
    android:layout_height="86dp"  
    android:src="@drawable/ferrari" />
```

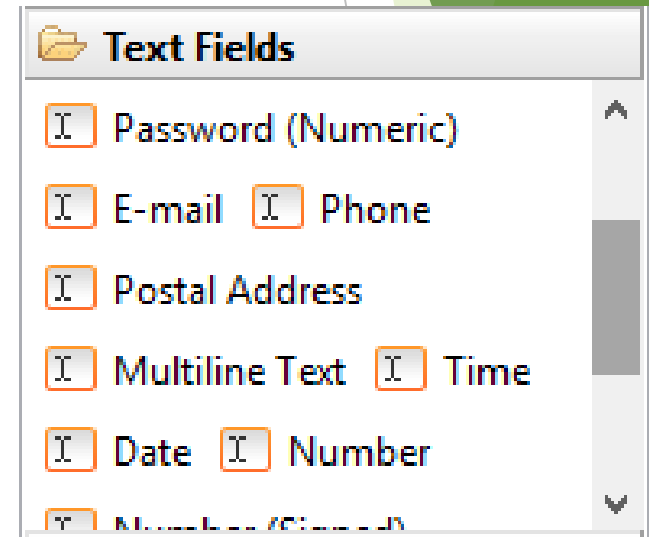
Exemplo 04 - Inserindo imagem



EditText

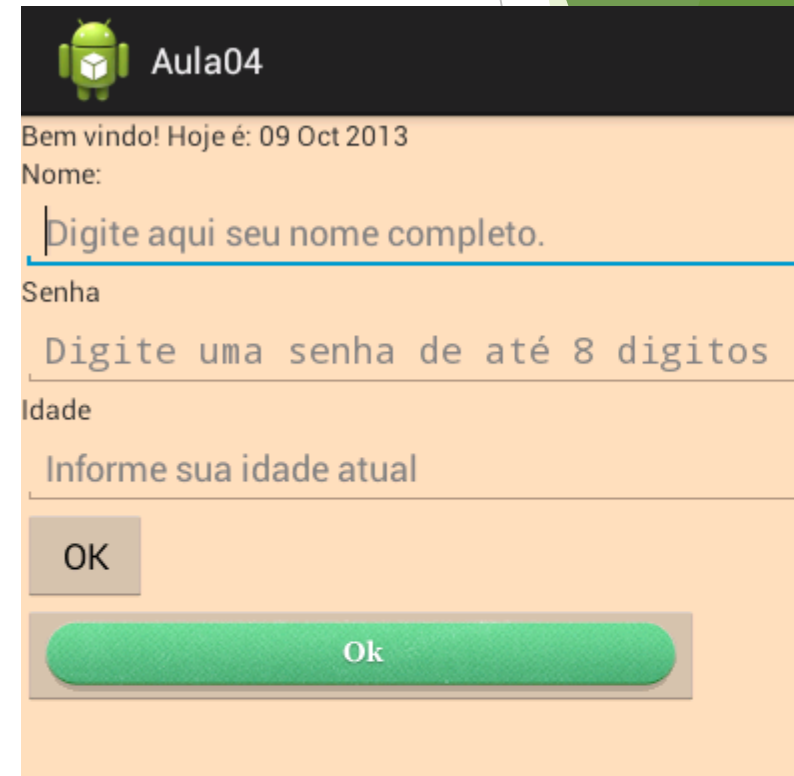
- ▶ Um EditText é um componente gráfico que permite ao usuário interagir com o aplicativo através da inserção de textos.
- ▶ Quando o usuário tocar em um **EditText** , automaticamente será exibido o teclado virtual para que uma informação seja passada.
- ▶ Na paleta de Widgets é possível incluir EditText com entradas pré-configuradas para permitir apenas números, campos no formato senha(password), etc.
- ▶ Uma EditText também poderá ser adicionada via código XML, conforme abaixo:

```
<EditText
    android:id="@+id/editText1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
</EditText>
```



EditText

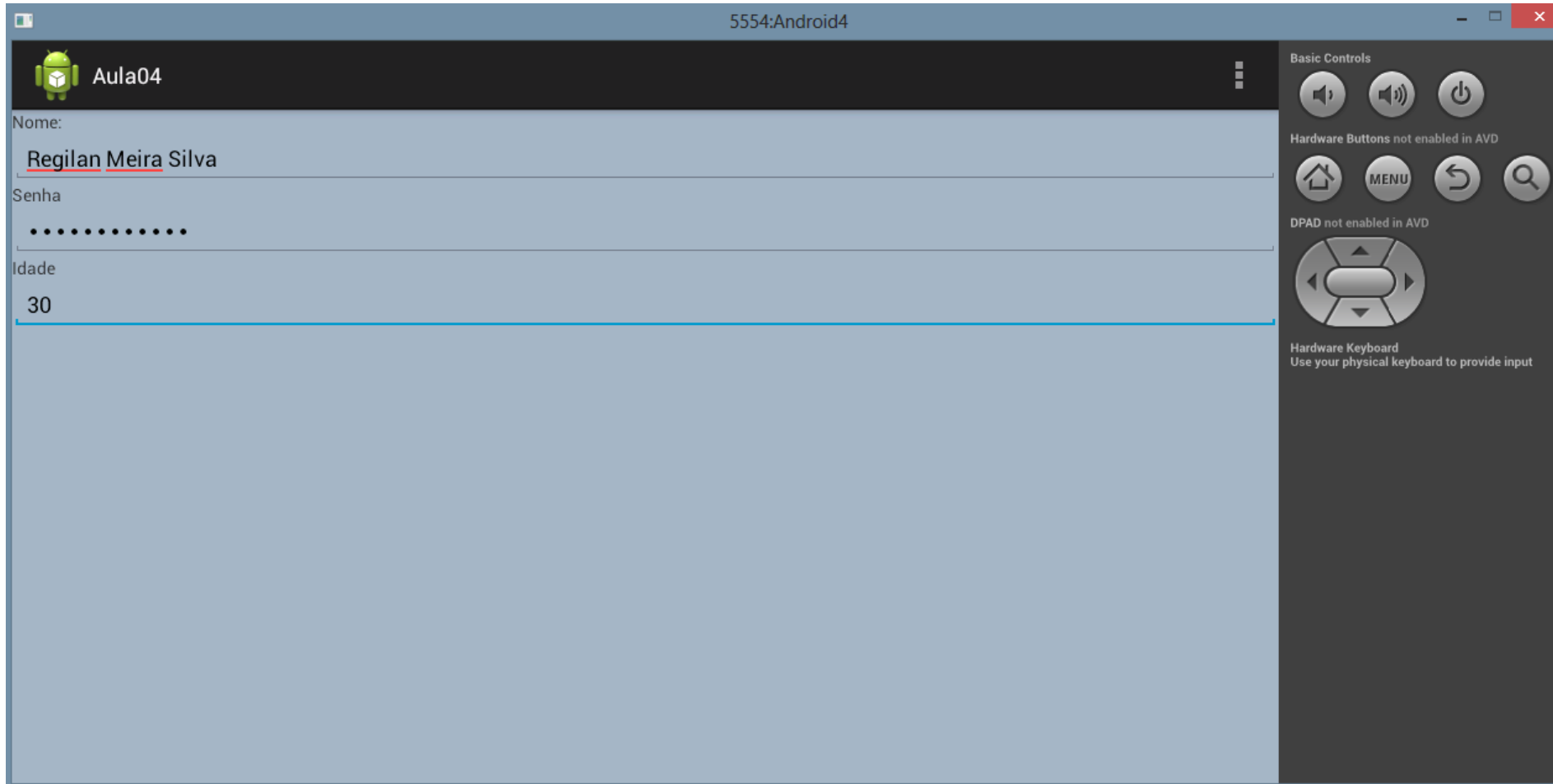
- ▶ Exibindo ajuda:
 - ▶ **android:hint:** este atributo exibe uma dica dentro de um componente EditText, a qual ajuda o usuário a entender o objetivo do componente. Quando o usuário iniciar a digitação neste componente, a dica de ajuda desaparece.



The screenshot shows an Android application window titled "Aula04" with an Android icon. The window contains a login form with the following elements:

- A greeting: "Bem vindo! Hoje é: 09 Oct 2013".
- A label "Nome:" followed by an EditText field with the hint text "Digite aqui seu nome completo.".
- A label "Senha" followed by an EditText field with the hint text "Digite uma senha de até 8 digitos".
- A label "Idade" followed by an EditText field with the hint text "Informe sua idade atual".
- A small "OK" button.
- A large green "Ok" button.

Exemplo 05 - EditText

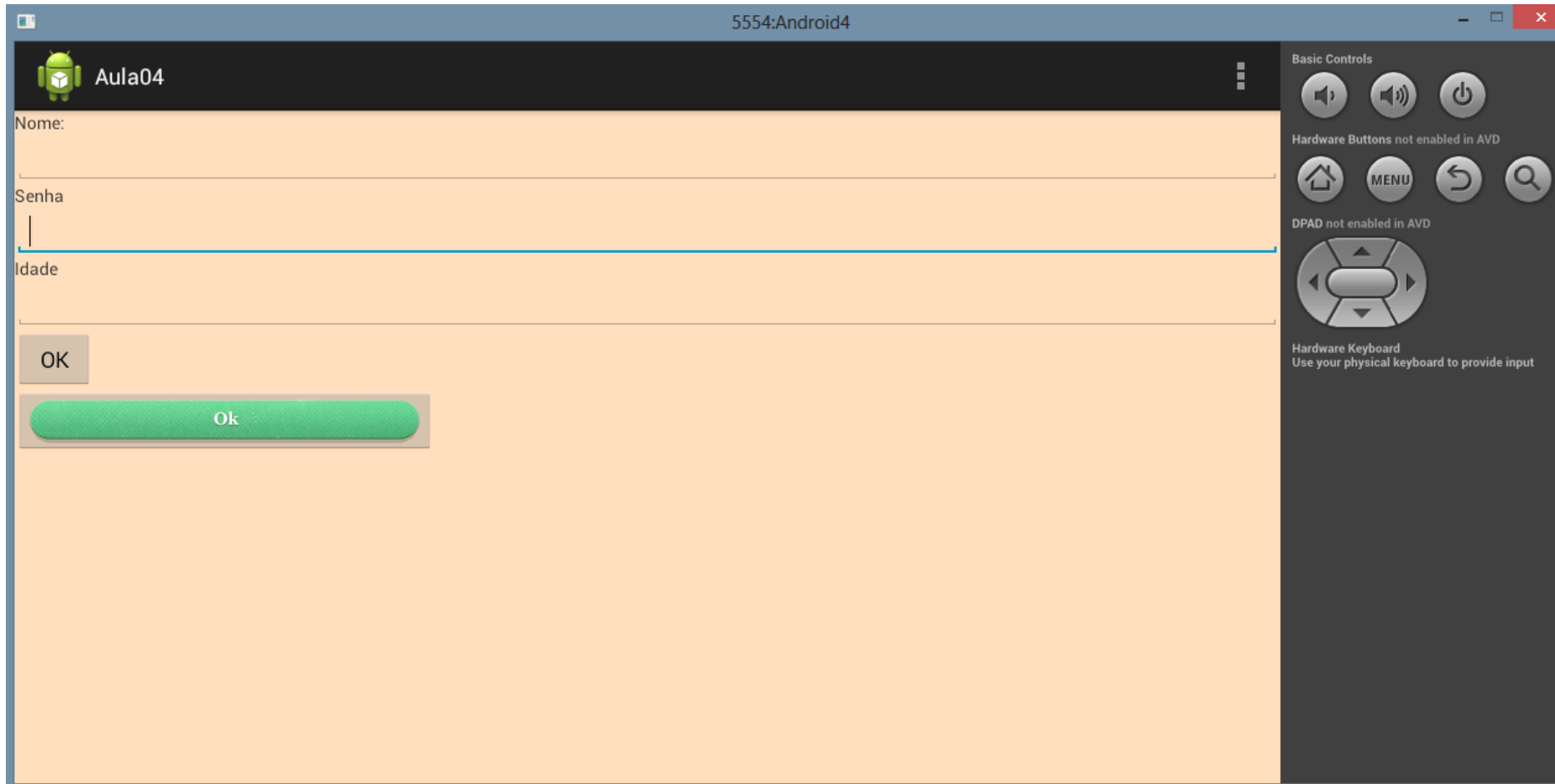


Button

- ▶ Um Button é um componente gráfico que permite ao usuário interagir com o aplicativo através de cliques(toques) no botão.
- ▶ Em geral os botões acompanham código JAVA que é acionado para realizar uma determinada função assim que o usuário do aplicativo toca-lo. Usamos para isto a propriedade onClick para chamar uma função no código JAVA a qual o formulário está relacionado.
- ▶ As propriedades id, text, background, margin e outras propriedades comuns a todos os widgets são configuradas da mesma forma que os controles já apresentados

```
<Button  
    android:id="@+id/button1"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="OK"  
    android:onClick="cadastrar"  
/>
```

Exemplo 06 - Button e ImageButton



Na próxima aula...

- ▶ string.xml e internacionalização
- ▶ Layouts em diferentes orientações

