



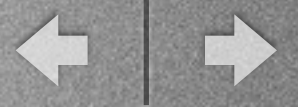
*Universidade Federal de Ouro Preto*  
*Departamento de Computação e Sistemas - DECSI*

## Computação Móvel

### Gerenciadores de *Layout* (Ref. Cap. 6)

*Vicente Amorim*  
*[vicente.amorim.ufop@gmail.com](mailto:vicente.amorim.ufop@gmail.com)*  
*[www.decom.ufop.br/vicente](http://www.decom.ufop.br/vicente)*





# Sumário

---

- \* Gerenciadores de *Layout*
  - *View / ViewGroup*
  - *FrameLayout*
  - *LinearLayout*
  - *TableLayout*
  - *RelativeLayout*
  - *AbsoluteLayout*
- \* Criação de *layouts* por API





## *View / ViewGroup*

---





## View / ViewGroup

---

### - Introdução

- ✓ O *layout* basicamente diz respeito a forma como os elementos são organizados na tela.
- ✓ Alguns *layouts* podem organizar elementos na horizontal ou vertical.
- ✓ *Layouts* possibilitam a apresentação de elementos da maneira desejada.
- ✓ *Layouts* funcionam basicamente como slots organizadores para a exibição de elementos.





## View / ViewGroup

---

### - Introdução

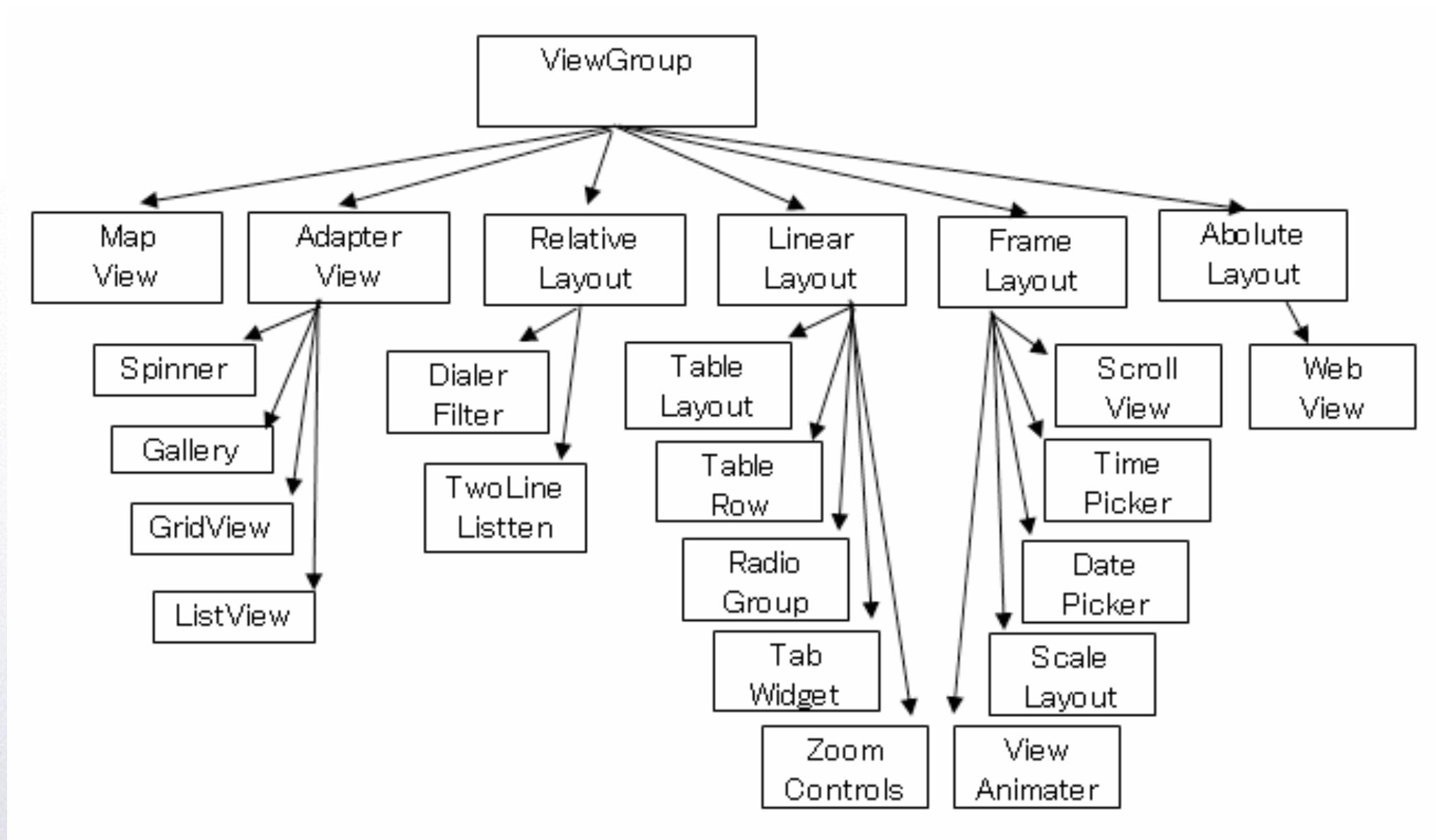
- ✓ Classe `android.view.View` é a classe-mãe de todos os componentes visuais do **Android**.
- ✓ Cada subclasse de `View` precisa implementar o método `onDraw(Canvas)`, responsável por desenhar o componente na tela.
- ✓ Separação entre *widgets* e gerenciadores de *layout*:
  - ◆ *Widget* : Elemento simples que herda da classe `View`: `Button`, `ImageView`, `TextView`, etc.
  - ◆ *Gerenciadores de layout* : Constituem subclasses de `android.view.ViewGroup` e são popularmente chamadas de *layouts*.





# View / ViewGroup

## - Introdução







## View / ViewGroup

---

### - Introdução

- ✓ *AbsoluteLayout*: Posiciona elementos na tela através de coordenadas x e y.
- ✓ *FrameLayout*: Utilizado por componentes que precisam ocupar a tela inteira.
- ✓ *LinearLayout*: Organiza os elementos na horizontal ou vertical.
- ✓ *TableLayout*: Organiza os elementos em formato de tabela.
- ✓ *RelativeLayout*: Posiciona elementos na tela utilizado como referência outros componentes.





# View / ViewGroup

## - Introdução

### ✓ Exemplo: AppNum18

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

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Escolha Sim ou Não" />
```

```
<Button android:id="@+id/btSim"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text=" Sim " />
```

```
<Button android:id="@+id/btNao"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text=" Não " />
```

```
</LinearLayout>
```





## View / ViewGroup

---

### - Introdução

- ✓ Por que os elementos ficaram dispostos um abaixo do outro?
- ✓ Gerenciador de *layout* `<LinearLayout>`: Organiza seus elementos filhos em uma lista vertical ou horizontal.
  - ◆ `android:orientation="vertical"` : Organiza os elementos em uma lista vertical; e
  - ◆ `android:orientation="horizontal"` : Organiza os elementos em uma lista horizontal.





## View / ViewGroup

---

### - Introdução

✓ Definição da altura e largura que o *layout* deve ocupar é feita através de dois outros atributos:

◆ **android:layout\_width** : Define a **largura** que o *layout* deve ocupar na tela; e

◆ **android:layout\_height** : Define a **altura** que o *layout* deve ocupar na tela.

✓ No momento da criação do *layout* devem ser definidos os elementos de tamanho o orientação.





## View / ViewGroup

### - Introdução

✓ Atributos de largura e altura podem receber os seguintes valores:

| Valor        | Descrição   |
|--------------|---|
| número       | Número inteiro especificando o tamanho do componente. Ex.: 50px.  |
| fill_parent  | Elemento irá ocupar todo o espaço definido pelo seu pai ( <i>layout</i> ). Deve ser utilizado sempre que algum <i>layout</i> ou componente precisar ocupar a tela inteira ou todo o espaço de <i>layout</i> definido pelo <i>layout</i> -pai. |
| match_parent | Idem ao <i>fill_parent</i> . A notação “ <i>fill_parent</i> ” foi descontinuada à partir do Android 2.2.  |
| wrap_content | Componente ocupará apenas o espaço necessário na tela. Ex.: Para TextView e ImageView o tamanho será suficiente para exibir o texto ou imagem requeridos.   |





## View / ViewGroup

---

### - Introdução

- ✓ Os parâmetros `android:layout_width` e `android:layout_height` são definidos pela classe `ViewGroup.LayoutParams`.
- ✓ Se a tela for construída diretamente via API será necessário conhecer a classe acima.
- ✓ Cada classe de `layout` define uma subclasse de parâmetros (`LayoutParams`).
- ✓ Se uma nova classe de `layout` é criada, então uma nova subclasse de parâmetros também deve ser definida.





## View / ViewGroup

---

- Entendendo *wrap\_content* e *match\_parent*

✓ Exemplo: AppNum19 (*wrap\_content*)

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#8B8B83" tools:context=".MainActivity">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:src="@drawable/android_green"/>
</FrameLayout>
```





## View / ViewGroup

---

- Entendendo *wrap\_content* e *match\_parent*

✓ Exemplo: AppNum20 (*match\_parent*)

```
<FrameLayout 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:background="#8B8B83" tools:context=".MainActivity">

    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:src="@drawable/android_green"/>
</FrameLayout>
```





## View / ViewGroup

---

- Entendendo *wrap\_content* e *match\_parent*

✓ Exemplo: AppNum21 (*match\_parent*)

```
<FrameLayout 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:background="#8B8B83" tools:context=".MainActivity">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent" android:src="@drawable/android_green"/>
</FrameLayout>
```





## View / ViewGroup

---

- Entendendo *wrap\_content* e *match\_parent*

✓ Exemplo: AppNum22

```
<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:padding="10dp" android:orientation="vertical" tools:context=".MainActivity">

    <TextView android:text="@string/nome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:inputType="text" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right" android:text="@string/ok" />
</LinearLayout>
```





## View / ViewGroup

---

- Entendendo *wrap\_content* e *match\_parent*

✓ Exemplo: AppNum23

```
<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:padding="10dp" android:orientation="vertical" tools:context=".MainActivity">

    <TextView android:text="@string/nome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text" />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right" android:text="@string/ok" />
</LinearLayout>
```





# *FrameLayout*

---





# FrameLayout

---

## - Introdução

- ✓ Definido pela classe [\*android.widget.FrameLayout\*](#).
- ✓ Mais simples de todos os gerenciadores de *layout*.
- ✓ Vários componentes podem ser adicionados, sendo que os últimos ficarão sobre os primeiros.
- ✓ Componente sempre é posicionado à partir do canto superior esquerdo.
- ✓ Dependendo do tamanho pode ocupar a tela inteira.





# FrameLayout

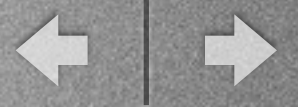
## - Introdução

### ✓ Exemplo: AppNum24 - Testando o *FrameLayout*

```
<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:padding="10dp" android:orientation="vertical" tools:context=".MainActivity">

    <TextView android:text="@string/nome"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text" />
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:layout_gravity="right"
            android:text="@string/ok" />
        <ProgressBar
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:layout_gravity="center" />
    </FrameLayout>
</LinearLayout>
```





# *LinearLayout*

---





# LinearLayout

---

## - Introdução

- ✓ Definido pela classe `android.widget.LinearLayout`.
- ✓ Gerenciador de *layout* mais utilizado.
- ✓ Também contém os atributos “`android:layout_width`” e “`android:layout_height`”.
- ✓ Possível controlar atributos que gerenciam o modo de exibição dos componentes: `vertical` ou `horizontal` - `android:orientation`.





# LinearLayout

---

## - Introdução

✓ Exemplo: AppNum25

✓ Exibindo elementos alinhados horizontalmente.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:orientation="horizontal"
    android:padding="10dp">

    <ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:src="@drawable/android_blue"
        android:contentDescription="@string/content_description"/>

    <ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:src="@drawable/android_green"
        android:contentDescription="@string/content_description"/>
</LinearLayout>
```

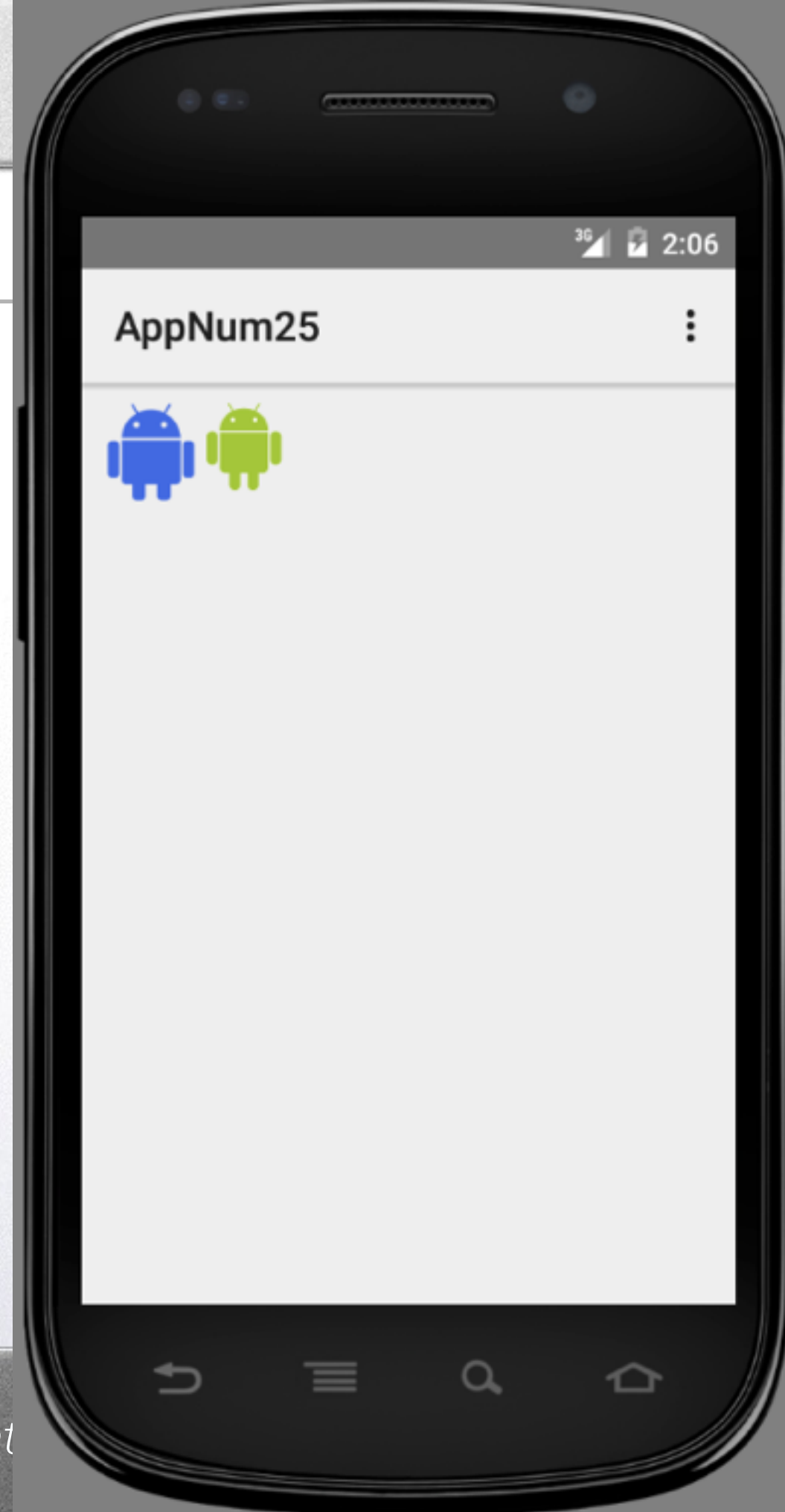




# LinearLayout

## - Introdução

✓ Exemplo: AppNum25







# LinearLayout

---

- Controle de alinhamento

- ✓ Componentes dentro de um *LinearLayout* podem ser alinhados de diversas formas na tela.

- ✓ Atributo `android:layout_gravity` deve ser utilizado para alinhamento dos componentes.

- ✓ Possíveis valores:

- ◆ `top`, `bottom`, `left`, `right`, `center_vertical`, `fill_vertical`, `center_horizontal`, `fill_horizontal`, `center` e `fill`.





# LinearLayout

---

## - Introdução

### ✓ Exemplo: AppNum26

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

    <ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:src="@drawable/android_green"
        android:layout_gravity="left"/>

    <ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:src="@drawable/android_blue"
        android:layout_gravity="center"/>

    <ImageView android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:src="@drawable/android_green"
        android:layout_gravity="right"/>
</LinearLayout>
```

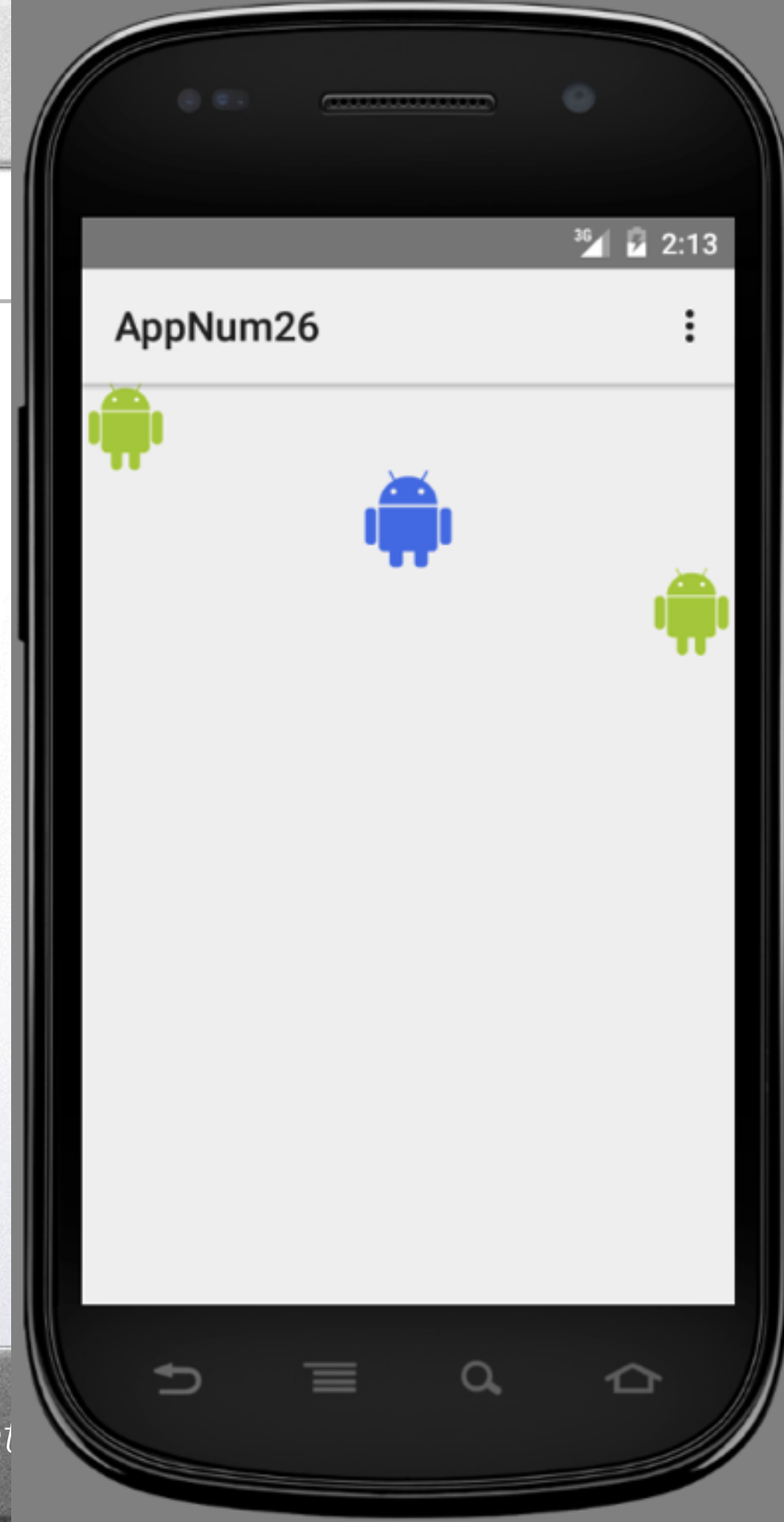




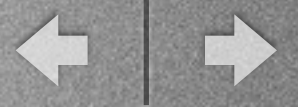
# LinearLayout

## - Introdução

✓ Exemplo: AppNum26







## *LinearLayout*

---

- Controle de peso e relevância

- ✓ Elementos podem receber um peso cada um.

- ✓ Os que receberem maior peso, ocuparão um espaço maior na tela.

- ✓ Peso dado através do atributo “*android:layout\_weight*”.

- ✓ Inteiros como possíveis valores.





# LinearLayout

## - Controle de peso e relevância

### ✓ Exemplo: AppNum27

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

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Nome: "/>

    <EditText android:layout_width="match_parent"
        android:layout_height="wrap_content" android:textColor="#ff0000"/>

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="E-mail: "/>

    <EditText android:layout_width="match_parent"
        android:layout_height="wrap_content"/>
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Observações: "/>

    <EditText android:layout_width="match_parent"
        android:layout_height="0dp" android:layout_weight="1" />

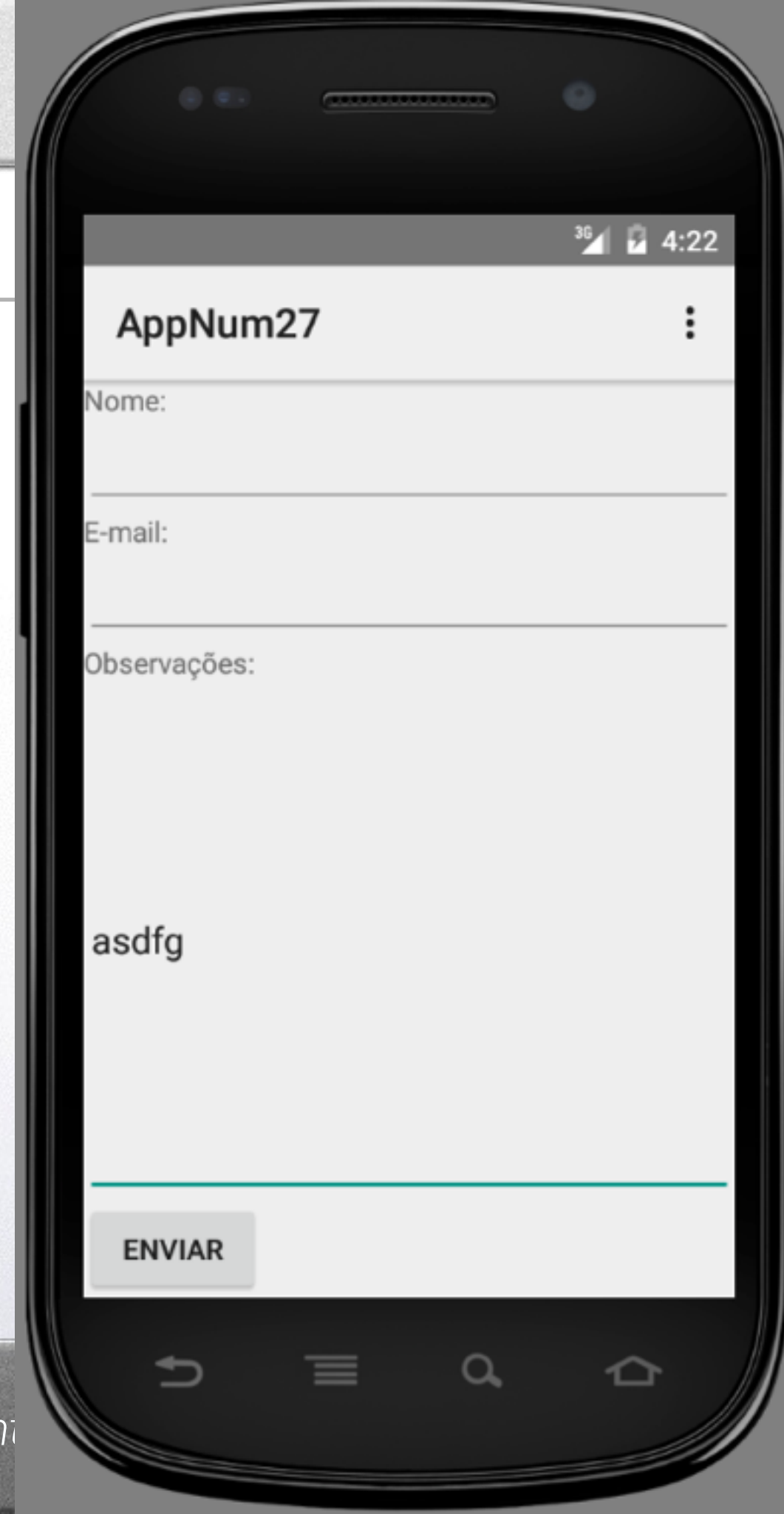
    <Button android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="Enviar"/>
</LinearLayout>
```





## LinearLayout

- Controle de peso e relevância
  - ✓ Exemplo: AppNum23







## LinearLayout

---

- Controle de peso e relevância

- ✓ Na AppNum27, o campo observações ocupou o espaço restante pois era o único a possuir “peso”.

- ✓ **Importante:** Sempre que utilizar o atributo de peso “*android:layout\_weight*”, deve-se definir a altura ou largura da *view* como 0dp.

- ◆ Notação indica que a largura/altura da *view* deve respeitar o peso atribuído.

- ✓ Sempre que um elemento visual precisar utilizar o espaço restante, basta adicionar o valor “1” (um) ao *android:layout\_weight*.





# LinearLayout

---

## - Controle de peso e relevância

✓ Exemplo: AppNum28

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

    <EditText android:layout_width="match_parent" android:layout_height="0dip"
        android:layout_weight="1" android:text="Texto (weight = 1)"/>

    <EditText android:layout_width="match_parent" android:layout_height="0dip"
        android:layout_weight="2" android:text="Texto (weight = 2)"/>

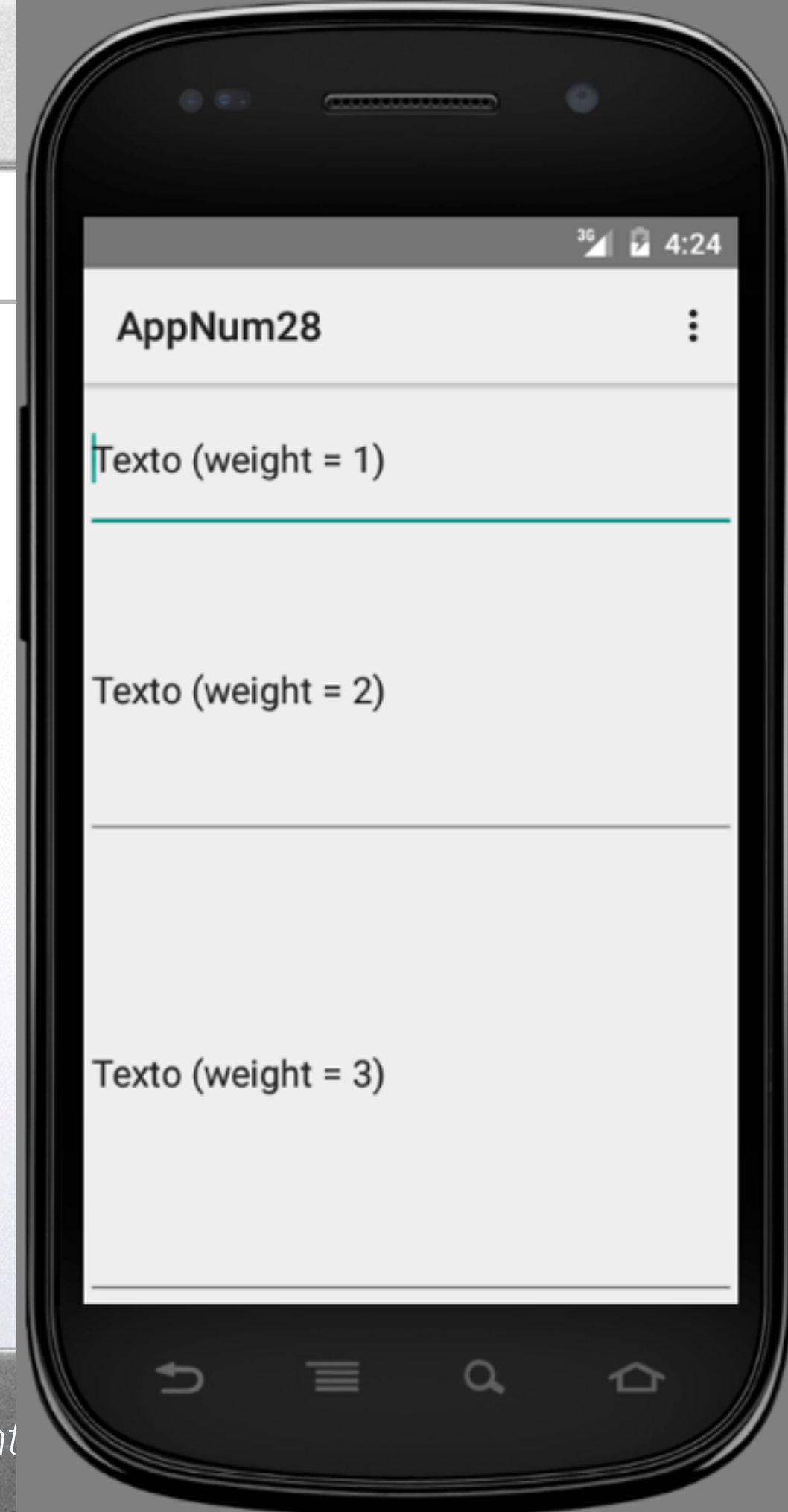
    <EditText android:layout_width="match_parent" android:layout_height="0dip"
        android:layout_weight="3" android:text="Texto (weight = 3)"/>
</LinearLayout>
```





## LinearLayout

- Controle de peso e relevância
  - ✓ Exemplo: AppNum28







# LinearLayout

---

## - Exercício:

- I I) Criar uma aplicação Android que contenha uma atividade que faça a entrada dos dados para o cadastro de um aluno.
- a) Os campos nome, endereço e data de nascimento devem estar localizados verticalmente dentro de um *LinearLayout*.
  - b) O valor das notas de quatro disciplinas devem ser pedidas. Tais campos devem ser localizados horizontalmente dentro de outro *LinearLayout*.
  - c) Um botão deve ser inserido no rodapé com o nome de “**Calcula**”. O mesmo, quando pressionado efetuará o cálculo da média das notas fornecidas.





# *TableLayout*

---





# TableLayout

---

## - Introdução

- ✓ Definido pela classe `android.widget.TableLayout`.
- ✓ Útil para a construção de formulários.
- ✓ Cada linha da tabela é constituída por um `android.widget.TableRow`.
- ✓ Possui atributos como `android:stretchColumns` e `android:shrinkColumns`:  
Recebem os índices das colunas que devem ser alteradas.
- ✓ **Importante:** Índices das colunas iniciam pelo “0” (zero).





## TableLayout

---

### - Ajuste das colunas

✓ Ex.: Para alterar a segunda coluna, o parâmetro `l` deve ser passado para os atributos *android:stretchColumns* / *android:shrinkColumns*.

◆ *android:stretchColumns*: Faz com que as colunas especificadas ocupem o espaço disponível na tela, expandindo-as.

◆ *android:shrinkColumns*: Faz com que as colunas especificadas sejam sempre exibidas na tela. Caso o texto seja maior que o espaço, duas ou mais linhas serão exibidas.





# TableLayout

## - Ajuste das colunas

### ✓ Exemplo: AppNum29 - Contração de colunas

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:shrinkColumns="2">

    <TableRow>
        <TextView android:text="Coluna 1" />
        <TextView android:text="Coluna 2" />
    </TableRow>

    <TableRow>
        <TextView android:text="Coluna 1"/>
        <TextView android:text="Coluna 2"/>
        <TextView android:text="Texto grande que vai sair da tela mas o
próprio atributo fará a tarefa de cortá-lo"/>
    </TableRow>

    <TableRow>
        <TextView android:text="Coluna 1"/>
        <TextView android:text="Coluna 2"/>
        <TextView android:text="Coluna 3"/>
    </TableRow>
</TableLayout>
```





## TableLayout

- Ajuste das colunas

✓ **Exemplo:** AppNum29 - Contração de col







# TableLayout

## - Ajuste das colunas

### ✓ Exemplo: AppNum30 - Expansão de colunas

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:stretchColumns="1">

    <TextView android:text="Lista de Produtos" />
    <View android:layout_height="20px" android:background="#FF909090"/>
    <TableRow>
        <TextView android:text="Produto A" />
        <TextView android:text="R$ 100,00" android:layout_gravity="right" />
    </TableRow>

    <TableRow>
        <TextView android:text="Produto B" />
        <TextView android:text="R$ 200,00" android:layout_gravity="right" />
    </TableRow>
    <TableRow>
        <TextView android:text="Produto C" />
        <TextView android:text="R$ 300,00" android:layout_gravity="right" />
    </TableRow>

    <View android:layout_height="20px" android:background="#FF909090"/>
    <LinearLayout android:layout_width="wrap_content" android:layout_height="match_parent"
        android:gravity="bottom">
        <Button android:layout_width="wrap_content" android:layout_height="80px" android:text="Cancelar"/>
        <Button android:layout_width="wrap_content" android:layout_height="80px" android:text="Comprar"/>
    </LinearLayout>
</TableLayout>
```





## TableLayout

- Ajuste das colunas

✓ **Exemplo:** AppNum30 - Expansão de colu







# TableLayout

---

## - Introdução

- ✓ Formulário com o *TableLayout* é similar a um formulário HTML.
- ✓ Pode posicionar elementos abaixo, acima ou ao lado de um já existente.
- ✓ Como o próprio nome diz, a posição de um elemento é sempre relativo a outro.
- ✓ Elemento referenciado precisa aparecer antes no *layout*.





# *GridLayout*

---





# GridLayout

---

## - Introdução

✓ Organiza as *views* em linhas e colunas utilizando os atributos *layout\_row* e *layout\_column*.

✓ Outros atributos:

◆ *android:layout\_columnSpan*: Fazer o merge entre várias colunas do *layout*;

◆ *android:layout\_rowSpan*: Fazer o merge entre várias linhas do *layout*;

◆ *android:layout\_gravity*: Especificar organização interna da *view*: *fill\_vertical*, *fill\_horizontal* e *fill*.

◆ GridLayout só está disponível à partir da **API Level 14** mas pode ser utilizado desde a **API Level 7** através da biblioteca de compatibilidade.





# GridLayout

## - Introdução

### ✓ Exemplo: AppNum3 I

```
<GridLayout 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:padding="@dimen/activity_horizontal_margin"
    tools:context=".MainActivity"
    android:columnCount="2" android:rowCount="2">

    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Botao 1" android:layout_column="0" android:layout_row="0"/>

    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Botao 2" android:layout_column="1" android:layout_row="0"/>

    <Button android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:text="Botao 3" android:layout_column="0" android:layout_row="1"/>
</GridLayout>
```





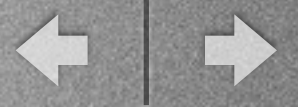
# GridLayout

## - Introdução

✓ **Exemplo:** AppNum31







# *RelativeLayout*

---





# RelativeLayout

---

## - Introdução

- ✓ Definido pela classe [\*android.widget.RelativeLayout\*](#).
- ✓ Pode posicionar elementos abaixo, acima ou ao lado de um já existente.
- ✓ Como o próprio nome diz, a posição de um elemento é sempre relativo a outro.
- ✓ Elemento referenciado precisa aparecer antes no *layout*.





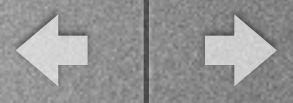
# RelativeLayout

## - Introdução

✓ Seguintes atributos podem ser utilizados para posicionamento de um elemento: (principais atributos)

| Atributo                                      | Descrição   |
|---|---|
| <code>android:layout_below</code>             | Posiciona abaixo do componente indicado.                        |
| <code>android:layout_above</code>             | Posiciona acima do componente indicado.                         |
| <code>android:layout_toRightOf</code>         | Posiciona à direita do componente indicado.                     |
| <code>android:layout_toLeftOf</code>          | Posiciona à esquerda do componente indicado.                    |
| <code>android:layout_alignParentTop</code>    | Alinha no topo do componente indicado.                          |
| <code>android:layout_alignParentBottom</code> | Alinha abaixo do componente indicado.                           |
| <code>android:layout_marginTop</code>         | Utilizado para definir espaço na margem superior do componente. |
| <code>android:layout_marginRight</code>       | Utilizado para definir um espaço à direita do componente.       |
| <code>android:layout_marginLeft</code>        | Utilizado para definir um espaço à esquerda do componente.      |





# RelativeLayout

## - Introdução

### ✓ Exemplo: AppNum32

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16dp"
    tools:context=".MainActivity"
    android:background="#8B8B83">
```

```
<TextView android:id="@+id/labelUsuario"
    android:layout_width="120px"
    android:layout_height="wrap_content"
    android:text="Usuário: " />
```

```
<EditText
    android:id="@+id/campoUsuario"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:drawable/editbox_background"
    android:layout_toRightOf="@id/labelUsuario" />
```

```
<TextView android:id="@+id/labelSenha"
    android:layout_width="120px"
    android:layout_height="wrap_content"
    android:layout_below="@id/campoUsuario"
    android:gravity="left"
    android:text="Senha: " />
```





# RelativeLayout

---

## - Introdução

### ✓ Exemplo: AppNum32 (cont.)

```
<EditText android:id="@+id/campoSenha"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@android:drawable/editbox_background"
    android:layout_toRightOf="@id/labelSenha"
    android:layout_alignTop="@id/labelSenha"
    android:password="true"/>
```

```
<Button android:id="@+id/btLogin"
    android:layout_width="wrap_content"
    android:layout_height="80px"
    android:layout_below="@id/campoSenha"
    android:layout_marginTop="20px"
    android:layout_alignParentRight="true"
    android:text="Login" />
```

```
</RelativeLayout>
```

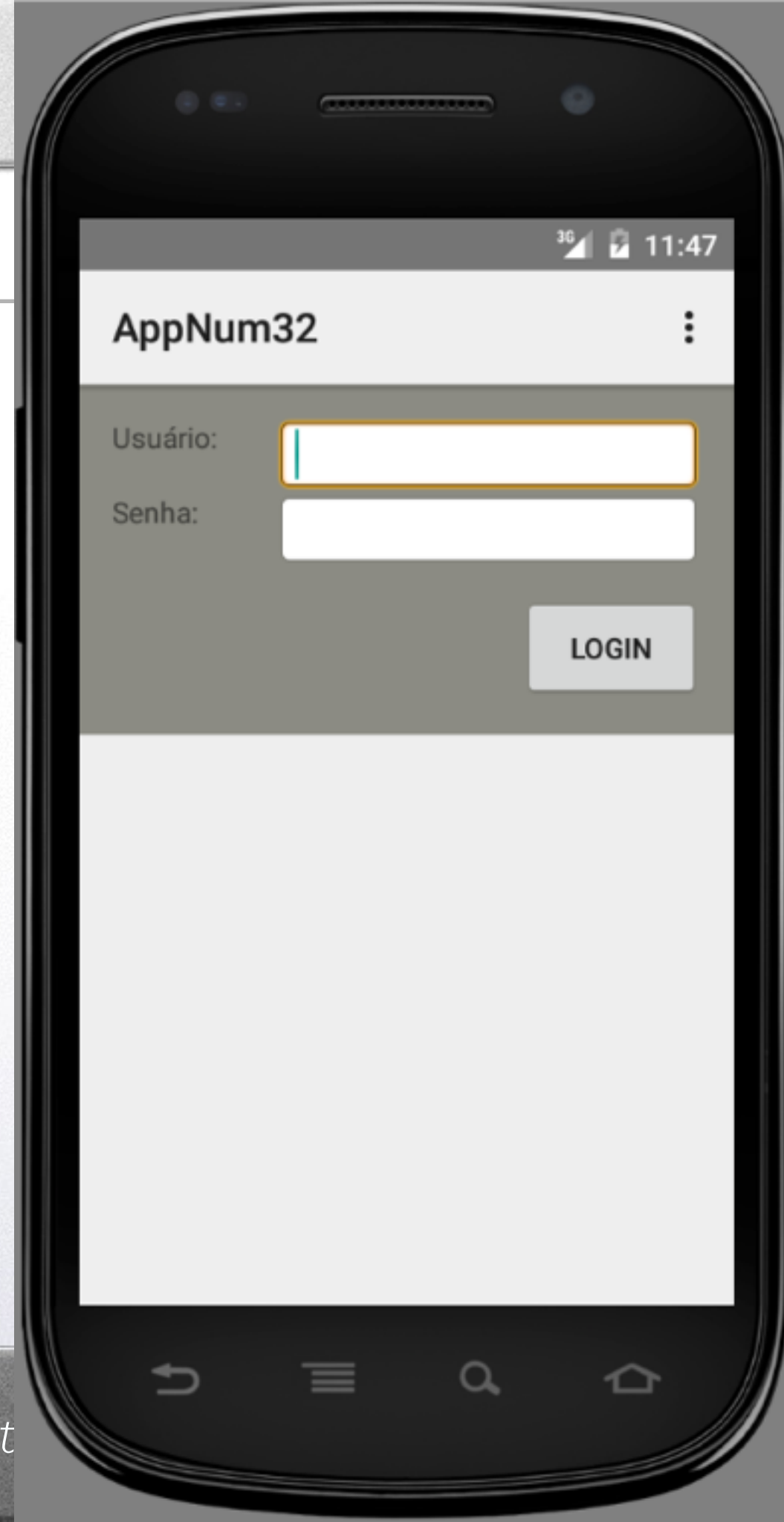




# RelativeLayout

## - Introdução

✓ **Exemplo:** AppNum32 (cont.)







# *AbsoluteLayout* (deprecated)

---





# AbsoluteLayout

---

## - Introdução

- ✓ Definido pela classe `android.widget.AbsoluteLayout`.
- ✓ Permite controlar a posição exata dos elementos na tela através do uso das coordenadas x e y.
- ✓ Uso dos atributos `android:layout_x` e `android:layout_y`.
- ✓ **Importante:** *Layout* pode diferir em outros dispositivos (uma vez que os valores são absolutos).





# AbsoluteLayout

---

## - Introdução

✓ Exemplo: AppNum33

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:background="#005793">
```

```
<TextView android:layout_x="5px" android:layout_y="10px"
    android:layout_width="wrap_content" android:layout_height="wrap_content"
    android:text="Usuário: "/>
```

```
<EditText android:layout_x="60px" android:layout_y="10px"
    android:layout_width="175px" android:layout_height="wrap_content"
    android:background="@android:drawable/editbox_background"/>
```





# AbsoluteLayout

## - Introdução

### ✓ Exemplo: AppNum33 (cont.)

```
<TextView android:layout_x="5px" android:layout_y="55px"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="Senha: "/>
```

```
<EditText android:layout_x="60px" android:layout_y="55px"  
    android:layout_width="175px" android:layout_height="wrap_content"  
    android:background="@android:drawable/editbox_background"  
    android:password="true"/>
```

```
<Button android:layout_x="180px" android:layout_y="100px"  
    android:layout_width="wrap_content" android:layout_height="60px"  
    android:text="Login"/>
```

```
</AbsoluteLayout>
```





# Criação de *layouts* por API

---





# Criação de *layouts* por API

---

## - Introdução

- ✓ Inicialmente, arquivos de *layout* podem ser definidos através do padrão XML.
- ✓ Entretanto, é possível se criar *layouts* também através da própria API **Android**.
- ✓ Também por esse método se faz necessário a definição de alguns atributos relacionados aos *layouts*.
- ✓ Os métodos `setContentView()` e `addView()` continuam a ser utilizados.





# Criação de *layouts* por API

✓ **Exemplo:** AppNum34 (criando campos para autenticação)

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
  
        //Cria layout  
        LinearLayout layout = new LinearLayout(this);  
        layout.setOrientation(LinearLayout.VERTICAL);  
        layout.setLayoutParams(new LinearLayout.LayoutParams(LayoutParams.MATCH_PARENT,  
LayoutParams.MATCH_PARENT));  
        layout.setPadding(10, 10, 10, 10);  
  
        //Cria TextView e EditText  
        TextView nome = new TextView(this);  
        nome.setText("Nome:");  
        nome.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));  
        layout.addView(nome);  
    }  
}
```





## Criação de *layouts* por API

---

✓ **Exemplo:** AppNum34 (criando campos para autenticação)(cont.)

```
EditText tnome = new EditText(this);
tnome.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
layout.addView(tnome);

//Focus
tnome.requestFocus();

//Campo senha
TextView senha = new TextView(this);
senha.setText("Senha:");
senha.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
layout.addView(senha);
```





## Criação de *layouts* por API

---

✓ **Exemplo:** AppNum34 (criando campos para autenticação)(cont.)

```
EditText tsenha = new EditText(this);
tsenha.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.WRAP_CONTENT));
layout.addView(tsenha);

//Botao alinhado a direita
Button ok = new Button(this);
ok.setLayoutParams(new LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
ok.setGravity(Gravity.RIGHT);
ok.setText("OK");
layout.addView(ok);

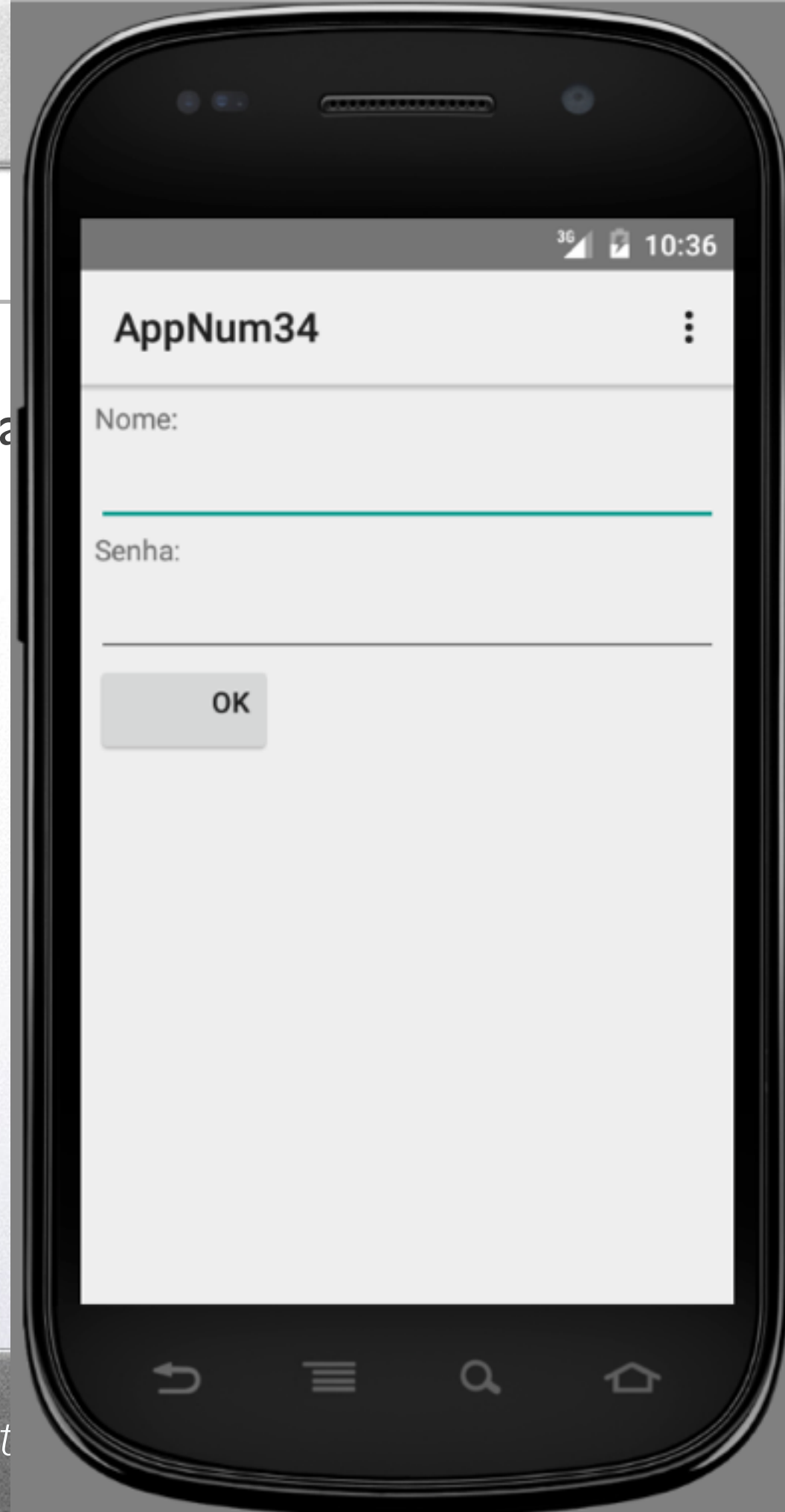
setContentView(layout);
```





## Criação de *layouts* por API

✓ **Exemplo:** AppNum34 (criando campos pa







# Criação de *layouts* por API

---

## - Introdução

- ✓ É possível se criar todas as telas de *layout* somente através da API.
- ✓ **Facilidade:** Os nomes dos métodos são muito parecidos com as palavras reservadas do XML.
- ✓ *Layouts* via XML é uma forma mais “elegante” de se criar uma separação entre lógica e interface gráfica.





# Criação de *layouts* por API

✓ **Exemplo:** AppNum35 (criação de *TableLayout* via API)

```
public class MainActivity extends Activity {

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

        //Cria layout
        TableLayout tabela = new TableLayout(this);
        tabela.setLayoutParams(new LayoutParams(LayoutParams.MATCH_PARENT, LayoutParams.MATCH_PARENT));

        //Expande coluna 1
        tabela.setColumnStretchable(1, true);

        //Linha 1
        TableRow linha1 = new TableRow(this);
        TextView nome = new TextView(this);
        nome.setText("Nome:");
        linha1.addView(nome);
        EditText tnome = new EditText(this);
        tnome.requestFocus();
        linha1.addView(tnome);
    }
}
```





# Criação de *layouts* por API

✓ **Exemplo:** AppNum35 (criação de *TableLayout* via API)

```
//Linha 2
TableRow linha2 = new TableRow(this);
TextView senha = new TextView(this);
senha.setText("Senha:");
linha2.addView(senha);
EditText tsenha = new EditText(this);
tsenha.setTransformationMethod(new PasswordTransformationMethod());
linha2.addView(tsenha);

//Linha3
TableRow linha3 = new TableRow(this);
linha3.setGravity(Gravity.RIGHT);

//Botão a direita
Button ok = new Button(this);
ok.setText("Login");
linha3.addView(ok);

//Adiciona as linhas ao layout
tabela.addView(linha1);
tabela.addView(linha2);
tabela.addView(linha3);

//Informa o layout
setContentView(tabela);
```





## Criação de *layouts* por API

---

### - *ScrollView*

- ✓ Especificado através da classe [\*android.widget.ScrollView\*](#).
- ✓ Permite que seja utilizado em telas que contêm muitos elementos e nas quais seja necessário fazer a rolagem da tela.
- ✓ É uma subclasse de [\*FrameLayout\*](#) - com a diferença de possuir um *scroll* nativo.
- ✓ Normalmente recebe outro *layout* internamente.





# Criação de *layouts* por API

## ✓ Exemplo: AppNum36 (ScrollView)

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

    <LinearLayout android:id="@+id/layout1"
        android:layout_width="match_parent" android:layout_height="wrap_content"
        android:orientation="vertical">
        <!-- TextViews serao adicionados aqui -->
    </LinearLayout>
</ScrollView>
```





# Criação de *layouts* por API

## ✓ Exemplo: AppNum36 (ScrollView) (cont.)

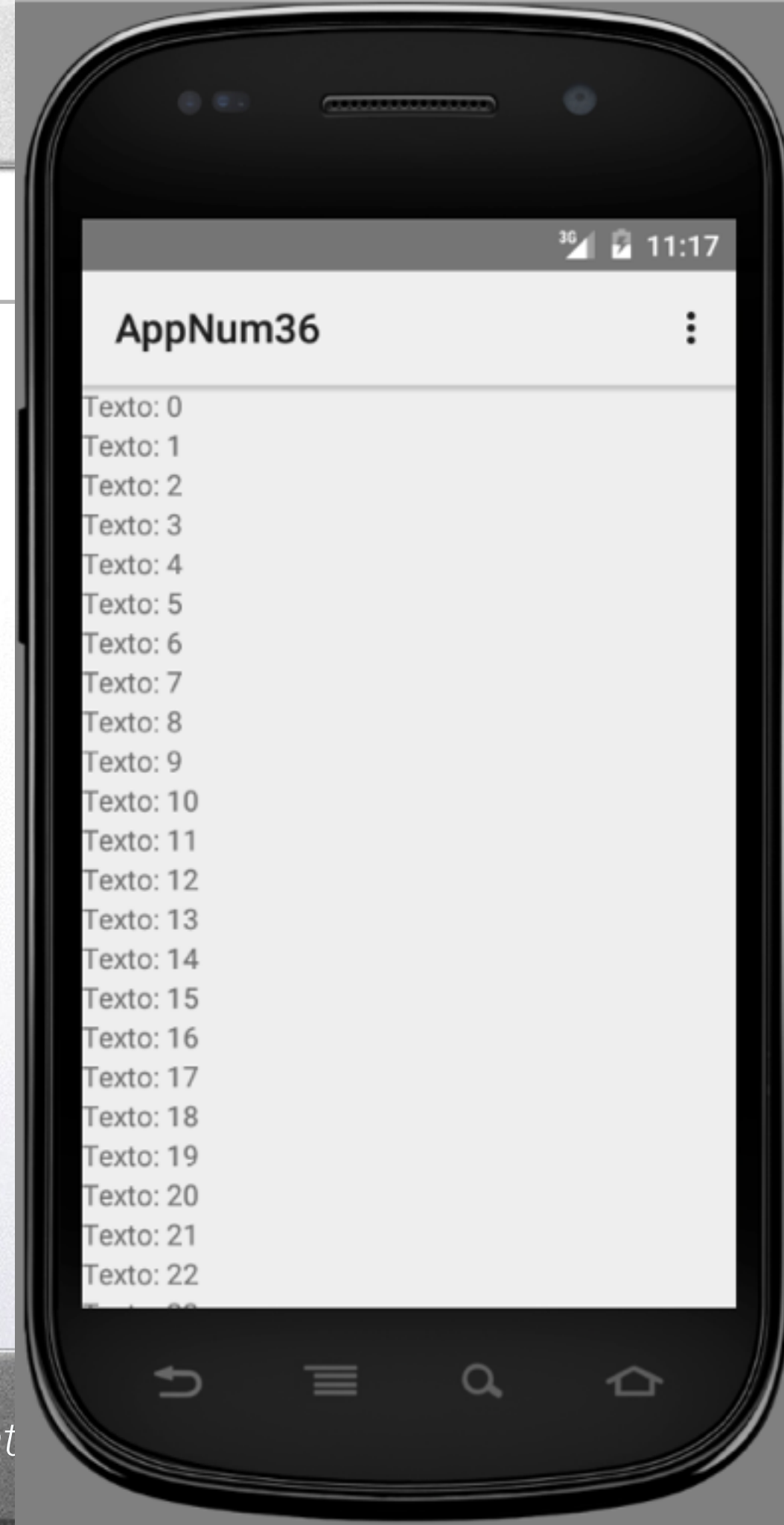
```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        LinearLayout layout = (LinearLayout) findViewById(R.id.layout1);  
  
        for (int i = 0; i < 100; i++) {  
            TextView text = new TextView(this);  
            text.setLayoutParams(new  
LayoutParams(LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));  
            text.setText("Texto: "+i);  
            layout.addView(text);  
        }  
    }  
}
```





## Criação de *layouts* por API

✓ **Exemplo:** AppNum36 (ScrollView) (cont.)







## Criação de *layouts* por API

---

### - *LayoutInflater*

- ✓ Exemplo anterior descreveu como adicionar *views* dinamicamente no *ScrollView*.
  - ◆ Foi necessário instanciar no código vários objetos do tipo *TextView*.
- ✓ É possível se criar um XML *layout* padrão para os casos onde os componentes são muito similares.
- ✓ *Layout* irá conter somente a parte da *view* que precisa ser adicionada.
- ✓ Utilização do componente *LayoutInflater*.





# Criação de *layouts* por API

---

## - *LayoutInflater*

✓ **Exemplo:** AppNum37 - *inflate\_textview.xml*

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/textView"  
    android:layout_width="wrap_content" android:layout_height="wrap_content"/>
```





# Criação de *layouts* por API

---

## - *LayoutInflater*

✓ **Exemplo:** AppNum37 - *activity\_main.xml*

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <LinearLayout android:id="@+id/layout1"
        android:layout_width="match_parent" android:layout_height="wrap_content"
        android:orientation="vertical">
        <!-- TextViews serao adicionados aqui -->
    </LinearLayout>
</ScrollView>
```





# Criação de *layouts* por API

## - *LayoutInflater*

✓ **Exemplo:** AppNum37 - *MainActivity.java*

```
public class MainActivity extends Activity {

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

        getActionBar().setDisplayHomeAsUpEnabled(true);
        setContentView(R.layout.activity_main);
        LinearLayout layout = (LinearLayout) findViewById(R.id.layout1);
        for (int i = 0; i < 100; i++) {
            //Cria o textview inflando o arquivo de layout
            LayoutInflater inflater = LayoutInflater.from(this);
            TextView text = (TextView) inflater.inflate(R.layout.inflate_textview, layout, false);
            //Utiliza-se normalmente
            text.setText("Texto: " + i);
            layout.addView(text);
        }
    }
    ...
}
```

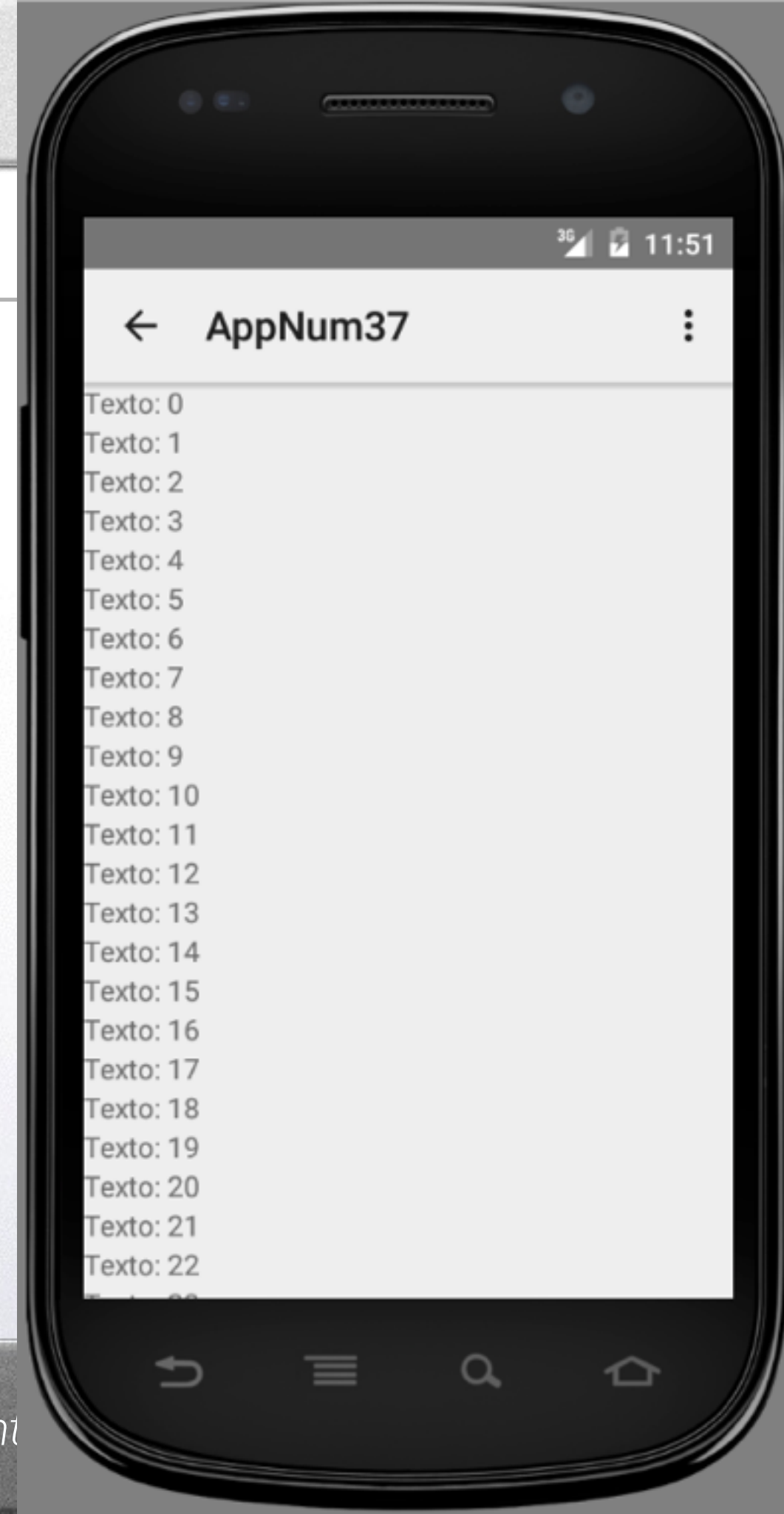




# Criação de *layouts* por API

- *LayoutInflater*

✓ **Exemplo:** AppNum37







# Criação de *layouts* por API

## - *LayoutInflater*

✓ **Importante:** Não é recomendável criar códigos de *layout* via API.

✓ Sempre é possível utilizar o *LayoutInflater* ao invés de codificar manualmente o *layout*.

◆ Nem sempre a *view* inflada será tão simples quanto a que foi utilizada no exemplo anterior. Parâmetros do *inflate*:

| Valor        | Descrição   |
|--------------|---|
| resource     | Arquivo XML com o <i>layout</i> que precisa ser inflado. O retorno será um objeto <i>View</i> desse <i>layout</i> .   |
| root         | <i>Layout</i> “container” no qual a <i>View</i> será adicionada. Deve ser informado para que a <i>View</i> conheça o tamanho e o posicionamento do <i>layout-pai</i> .          |
| attachToRoot | Indica se a <i>View</i> deve ser automaticamente adicionada no <i>layout</i> . É recomendado passar sempre <i>false</i> e posteriormente chamar o método <i>addView(view)</i> . |





## Criação de *layouts* por API

---

### - Exercício:

12) Alterar o programa anterior (média das notas dos alunos) para uma aplicação com *layout* construído dinamicamente via API. Sua aplicação deve considerar as seguintes funcionalidades:

- a) Opção de inserção das notas e 4 diferentes disciplinas para uma turma de no máximo 30 alunos;
- b) Exibição da média das notas de cada aluno;
- c) Exibição da média das notas de toda a turma.