

Curso Android para *Iniciantes*



Lana Mesquita
Cristiane Mayara
Brunno Melo

Contatos



- Lana Mesquita
lanabeatriz.mesquita@gmail.com
- Cristiane Mayara
cristiane.mayara@gmail.com
- Brunno Melo
brunnomelo@gmail.com
- Grupo Facebook:
<https://www.facebook.com/groups/CursoAndroidCAET2014>

Ementa de hoje

1. LogCat
2. Toast
3. Utilizando o adb
4. Activity
4. Ciclo de vida de uma Activity
5. Gerenciadores de Layout
6. Componentes de tela
 - View
 - TextView
 - ImageView
 - Button
7. Prática: Criar projeto da Calculadora

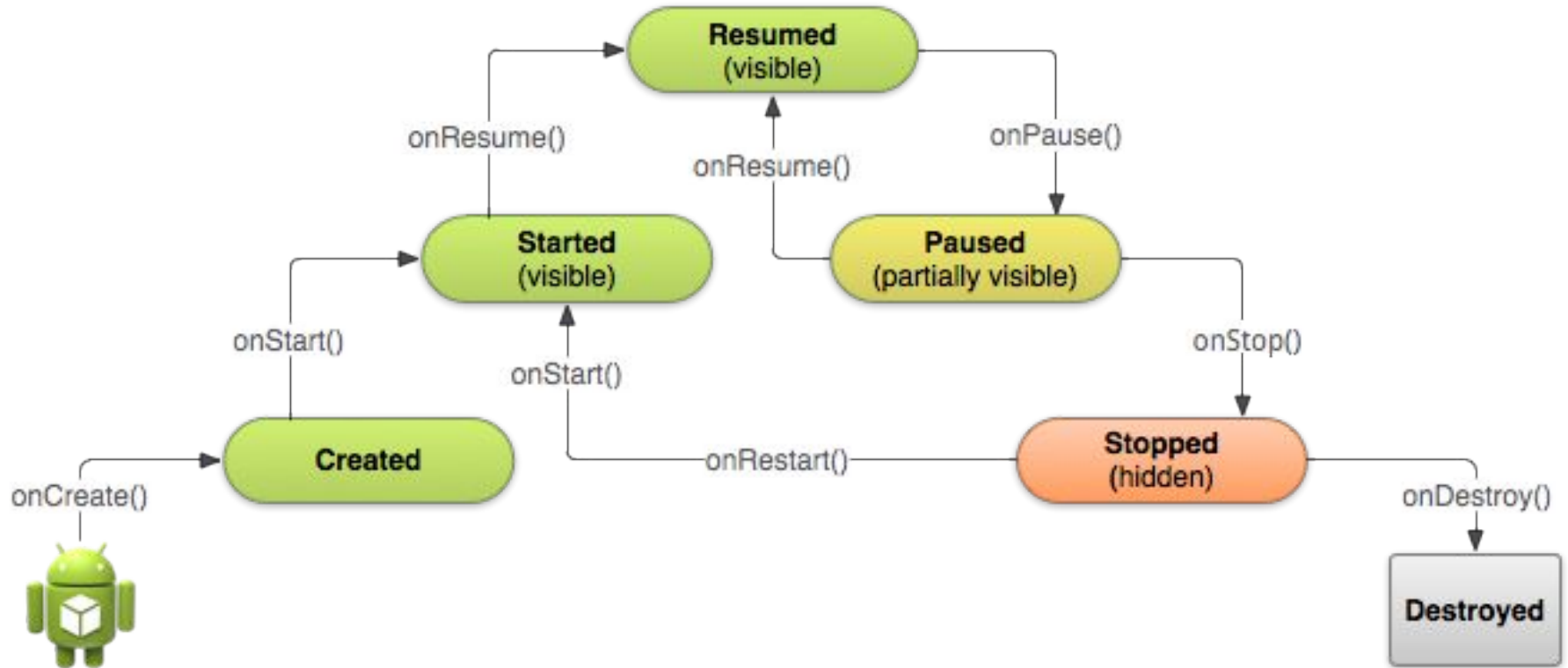


Activity

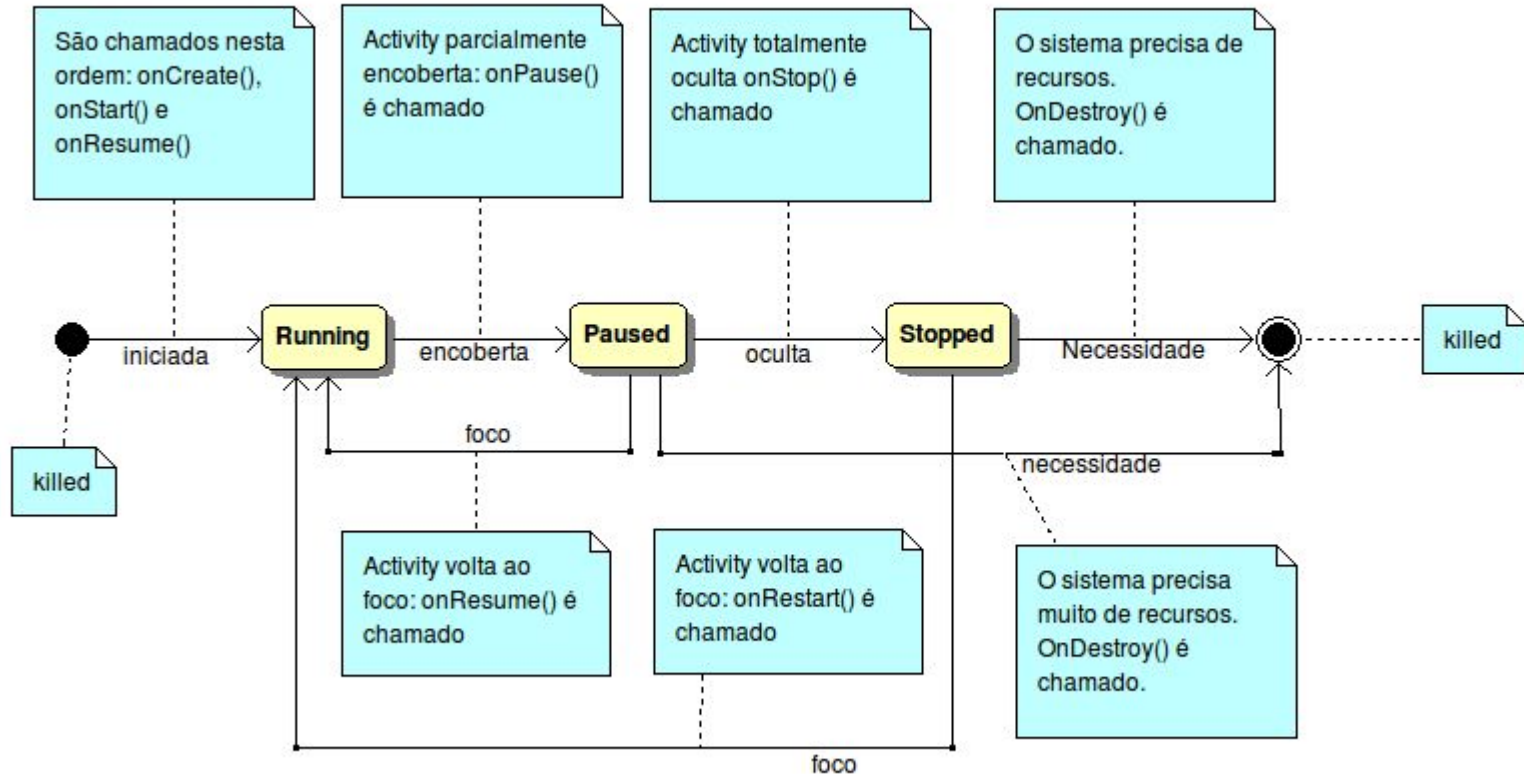
- é uma classe que deve herdar da classe `android.app.Activity`
- representa uma tela da aplicação
- é responsável por tratar eventos gerados nessa tela
- possui diferentes estados (lifecycle) que são chamados em métodos na Activity.
- é possível definir o comportamento de cada um destes métodos.



Ciclo de vida de uma Activity



Ciclo de vida de uma Activity



Activity



- dependendo da complexidade da aplicação, não precisa implementar todos os métodos do ciclo.
- é nestes métodos que são tratados eventos como:
 - receber uma ligação, usuário trocar de aplicativo,
 - não perder conteúdo se o usuário girar a tela,
 - não utilizar processamento de recursos quando a activity não estiver ativa.

Activity

- Uma Activity deve sempre implementar o método **onCreate()**, que deve chamar o método **setContentView(view)** para definir o layout
- Três passos importantes para criar uma Activity:
 1. Criar uma Activity (ex: TesteActivity.java)
 2. Declarar no Manifest:

```
<activity android:name=".MainActivity" android:label="@string/app_name">  
  <intent-filter>  
    <action android:name="android.intent.action.MAIN" />  
    <category android:name="android.intent.category.LAUNCHER" />  
  </intent-filter>  
</activity>
```

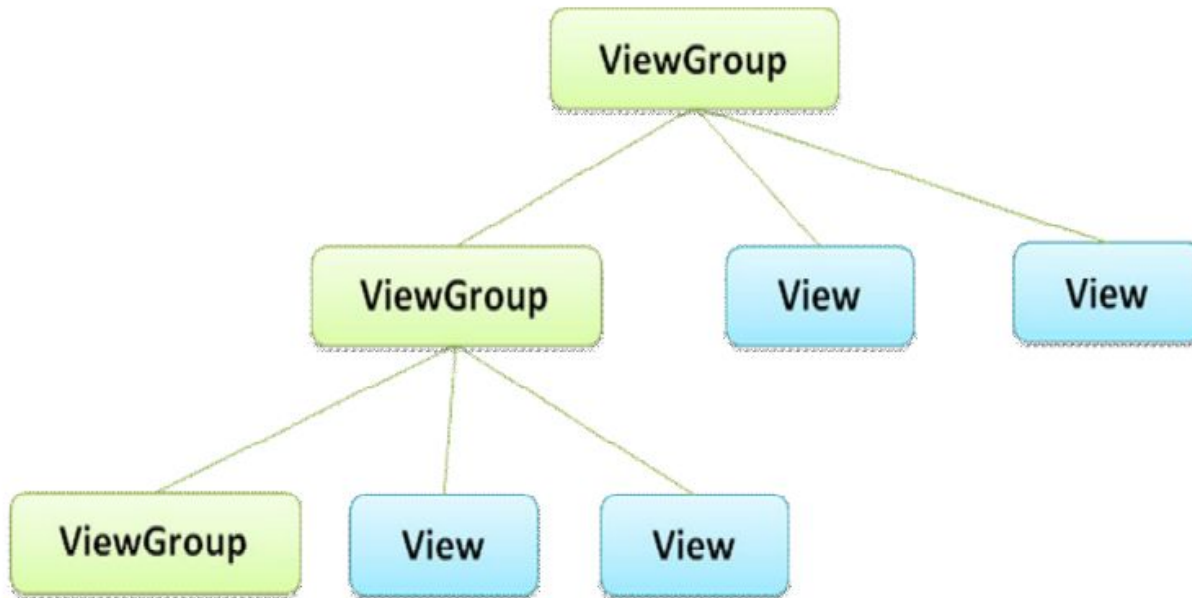
3. Criar um layout para a activity (ex: teste_layout)

Layout

- Em Android, todos os componentes de interface gráfica são representados por subclasses de **android.view.View** que representam os componentes gráficos (os chamados **widgets**) como TextView, Button , TextEdit, RadioButton, Checkbox, etc.
- A classe **android.view.ViewGroup**, que representa um contêiner de Views e também ViewGroups. Ela é a classe base para componentes de **layouts**, como LinearLayout, FrameLayout, AbsoluteLayout, RelativeLayout, TableLayout, etc.

Layout

- um ViewGroup ser composto por um ou mais ViewGroups é o fator que permite que layouts complexos (layouts aninhados)



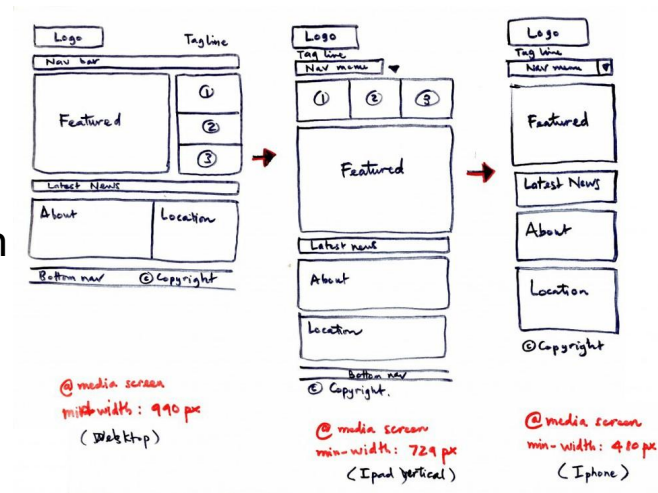
Layout

- Um layout define a estrutura visual para a interface do usuário.

- Duas maneiras de trabalhar com layout:

- **Declarar elementos UI em XML.** Android fornece um vocabulário XML simples que corresponde à visão classes e subclasses, tais como os de widgets e layouts.

- Instanciar elementos de layout em tempo de execução e manipular suas propriedades **em código**



Layout

Parâmetros

Para que os componentes possam ser acomodados de acordo com o layout de seu pai, os mesmos devem manter uma relação de obediência.

Parâmetros essenciais:

`android:layout_width` - largura da view

`android:layout_height` - altura da view

wrap_content : se ajusta ao conteúdo

match_parent: ocupa todo o layout pai (*fill_parent* até a API 7)

Layout

Parâmetros

Outros parâmetros:

`android:id` - utilizado para identificar o componente e chamá-lo no código

`android:padding` - cria uma espaço interno (moldura) ao redor da view

`android:margin` - cria uma espaço externo (moldura) entre views

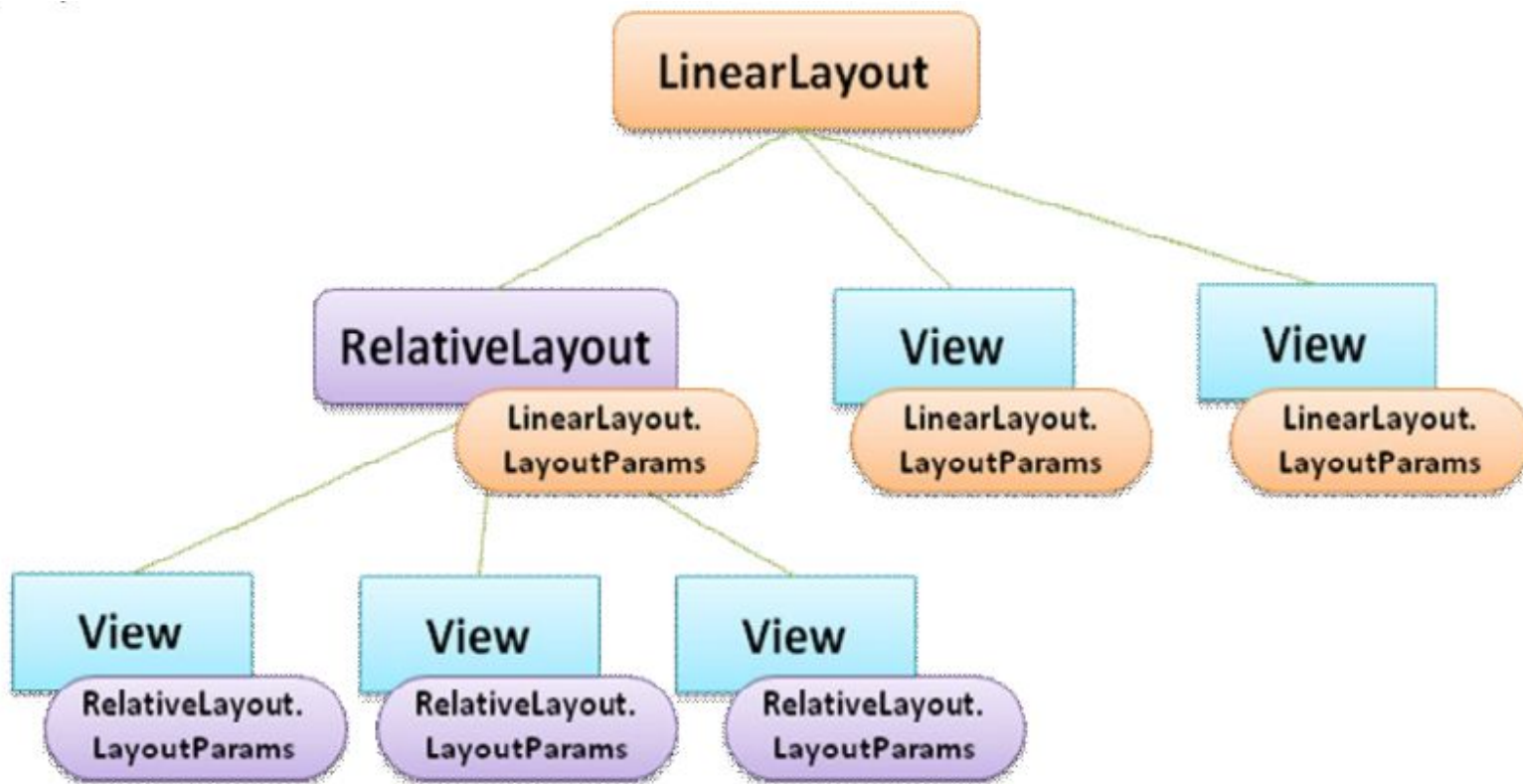
`android:background` - seta cor ou imagem de fundo

`android:gravity` - seta a localização da view em relação ao layout pai

`android:background` - seta cor ou imagem de fundo

Layout

Parâmetros



Layout

Parâmetros

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <Button
        android:id="@+id/btnButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button ABC"/>

</RelativeLayout>
```



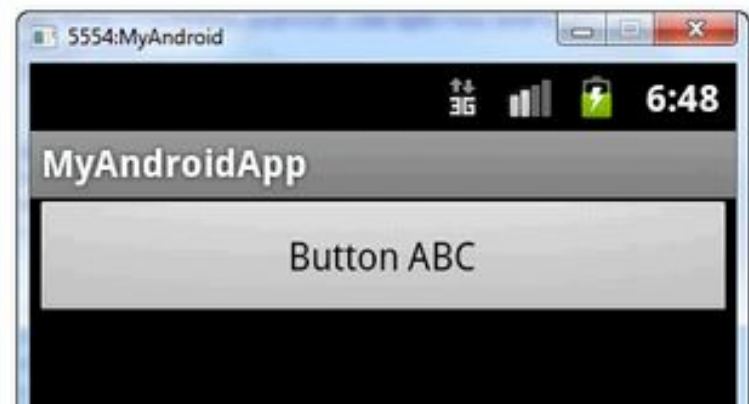
Layout

Parâmetros

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <Button
        android:id="@+id/btnButton1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Button ABC"/>

</RelativeLayout>
```



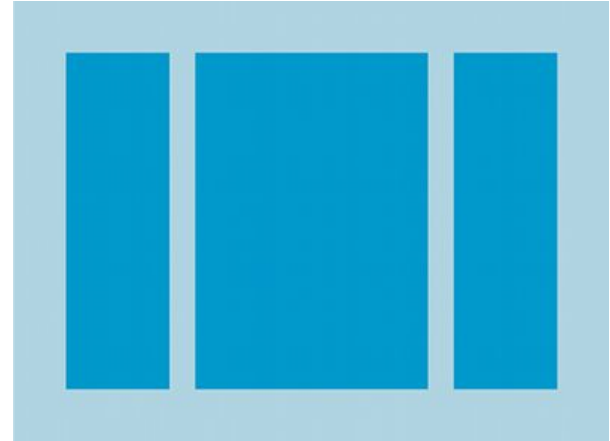
Gerenciadores de Layout

LinearLayout

LinearLayout é um gerenciador que alinha todas as seus componentes em uma única direção, verticalmente ou horizontalmente.

Deve conter o parâmetro:

`android:orientation`



Exemplo:

```
<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="@string/hello_world" />

</LinearLayout>
```

Gerenciadores de Layout

LinearLayout

O parâmetro *weight* cria pesos para as *views* dentro do LinearLayout.

`android:weightSum` - usa no LinearLayout

`android:layout_weight` - usa nas views dentro do LinearLayout

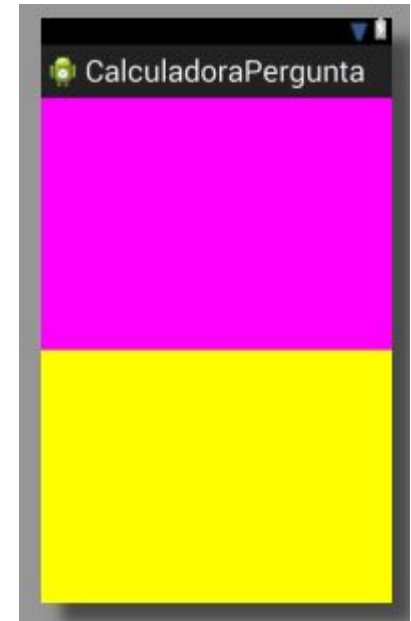
Exemplo:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical"
  android:weightSum="10" >

  <View
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="5"
    android:background="#FF00FF" />

  <View
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="5"
    android:background="#FFFF00" />

</LinearLayout>
```



Gerenciadores de Layout

RelativeLayout

RelativeLayout é um gerenciador que alinha seus componentes em posições relativas, podendo ser especificada em relação à outro componente.

Alguns parâmetros utilizados nos componentes dentro de um RelativeLayout:

`android:layout_alignParentTop` - alinha a view à borda superior do layout.

`android:layout_centerVertical` - alinha a view ao centro do layout verticalmente.

`android:layout_below` - alinha a margem desta view logo abaixo de outra view

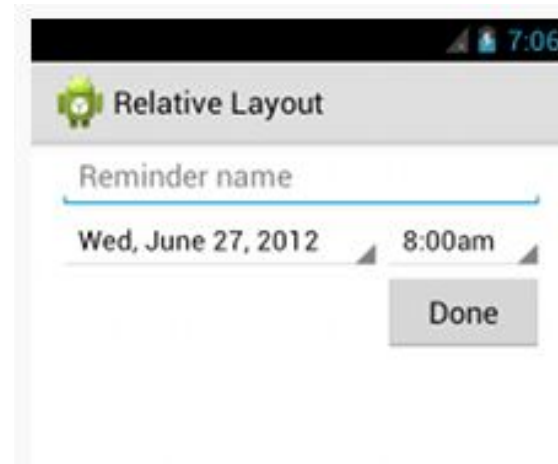
`android:layout_toRightOf` - alinha esta view a direita de outra



Gerenciadores de Layout

RelativeLayout

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp" >
    <EditText
        android:id="@+id/name"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/reminder" />
    <Spinner
        android:id="@+id/dates"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentLeft="true"
        android:layout_toLeftOf="@+id/times" />
    <Spinner
        android:id="@+id/times"
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/name"
        android:layout_alignParentRight="true" />
    <Button
        android:layout_width="96dp"
        android:layout_height="wrap_content"
        android:layout_below="@id/times"
        android:layout_alignParentRight="true"
        android:text="@string/done" />
</RelativeLayout>
```



Gerenciadores de Layout

FrameLayout

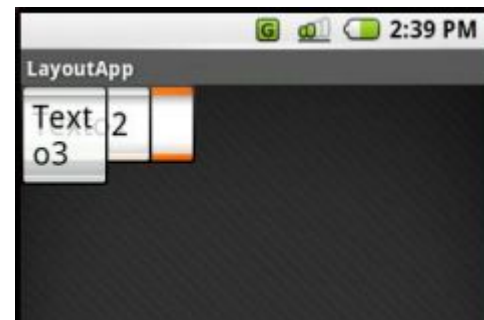
O FrameLayout arranja seus filhos de acordo com uma pilha de componentes que são adicionados, sendo que o topo da pilha contém o objeto que foi adicionado por último.

O tamanho total de um FrameLayout é definido pelo seu maior filho mais o espaçamento (padding) e **todos os componentes são agrupados no canto superior esquerdo do layout.**



Gerenciadores de Layout

FrameLayout

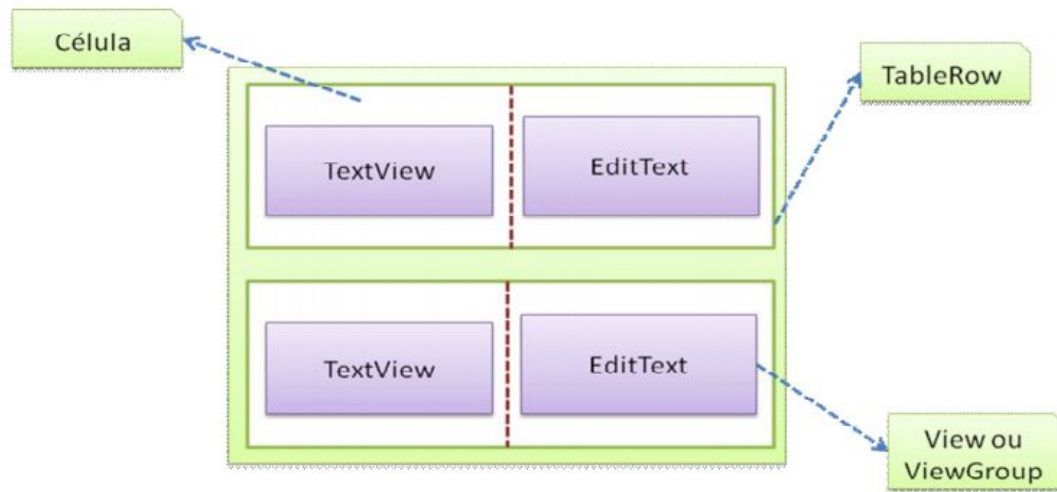


```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <EditText android:layout_width="120px"
        android:layout_height="wrap_content" android:text="Texto1"
        android:layout_weight="1" />
    <EditText android:layout_width="90px"
        android:layout_height="wrap_content" android:text="Texto2"
        android:layout_weight="1" />
    <EditText android:layout_width="60px"
        android:layout_height="wrap_content" android:text="Texto3" />
</FrameLayout>
```

Gerenciadores de Layout

TableLayout

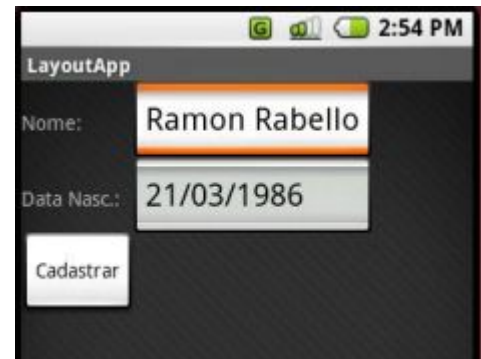
O TableLayout comporta seus filhos em linhas e colunas. Cada filho é representado pelo componente TableRow (um tipo de LinearLayout horizontal)



Gerenciadores de Layout

TableLayout

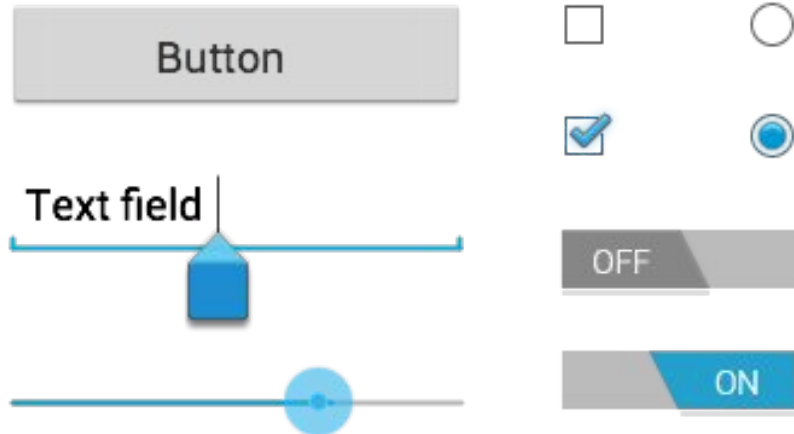
```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="fill_parent">
    <TableRow>
        <TextView android:text="Nome:" />
        <EditText android:text="Ramon Rabello" />
    </TableRow>
    <TableRow>
        <TextView android:text="Data Nasc.:" />
        <EditText android:text="21/03/1986" />
    </TableRow>
    <TableRow>
        <Button android:text="Cadastrar" />
    </TableRow>
</TableLayout>
```



Widgets

A partial list of available widgets includes

- Button
- TextView
- EditText
- ListView
- CheckBox
- RadioButton
- Gallery
- Spinner



Widgets

Control Type	Description	Related Classes
Button	A push-button that can be pressed, or clicked, by the user to perform an action.	Button
Text field	An editable text field. You can use the <code>AutoCompleteTextView</code> widget to create a text entry widget that provides auto-complete suggestions	EditText, AutoCompleteTextView
Checkbox	An on/off switch that can be toggled by the user. You should use checkboxes when presenting users with a group of selectable options that are not mutually exclusive.	CheckBox
Radio button	Similar to checkboxes, except that only one option can be selected in the group.	RadioGroup RadioButton
Toggle button	An on/off button with a light indicator.	ToggleButton
Spinner	A drop-down list that allows users to select one value from a set.	Spinner
Pickers	A dialog for users to select a single value for a set by using up/down buttons or via a swipe gesture. Use a <code>DatePicker</code> widget to enter the values for the date (month, day, year) or a <code>TimePicker</code> widget to enter the values for a time (hour, minute, AM/PM), which will be formatted automatically for the user's locale.	DatePicker, TimePicker

Widgets

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="horizontal">
    <EditText android:id="@+id/edit_message"
        android:layout_weight="1"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:hint="@string/edit_message" />
    <Button android:id="@+id/button_send"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/button_send"
        android:onClick="sendMessage" />
</LinearLayout>
```

Widgets

EditText

`android:inputType :`

`"text"`

Teclado normal

`"textEmailAddress"`

Teclado especial para e-mail

`"textUri"`

Teclado com o caractere /

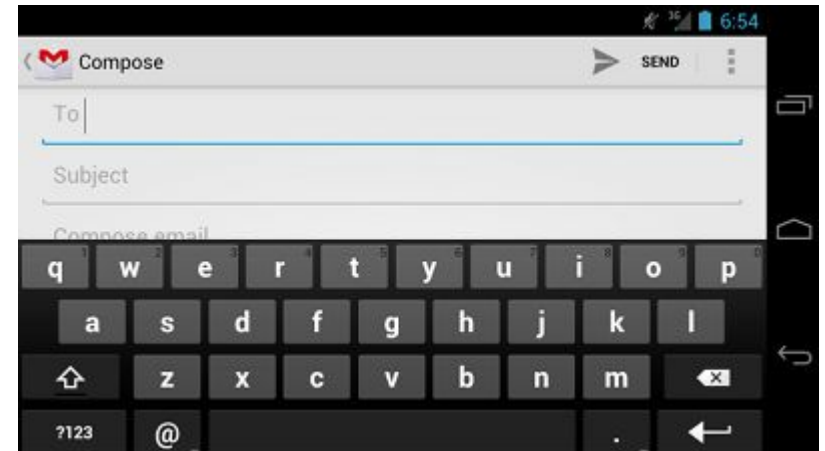
`"number"`

Teclado com números

`"phone"`

Teclado de ligação

```
<EditText
    android:id="@+id/email_address"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/email_hint"
    android:inputType="textEmailAddress" />
```



Widgets

EditText

O mesmo parâmetro também pode se referir ao tipo de entrada e é possível colocar mais de um tipo neste parâmetro:

`"textCapSentences"`

Normal text keyboard that capitalizes the first letter for each new sentence.

`"textCapWords"`

Normal text keyboard that capitalizes every word. Good for titles or person names.

`"textAutoCorrect"`

Normal text keyboard that corrects commonly misspelled words.

`"textPassword"`

Normal text keyboard, but the characters entered turn into dots.

`"textMultiLine"`

Normal text keyboard that allow users to input long strings of text that include line breaks (carriage returns).

Widgets

EditText

```
EditText editText = (EditText) findViewById(R.id.search);
editText.setOnEditorActionListener(new OnEditorActionListener() {
    @Override
    public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {
        boolean handled = false;
        if (actionId == EditorInfo.IME_ACTION_SEND) {
            sendMessage();
            handled = true;
        }
        return handled;
    }
});
```

LogCat

Existe 5 tipos de logs são eles:

V — Verbose

D — Debug

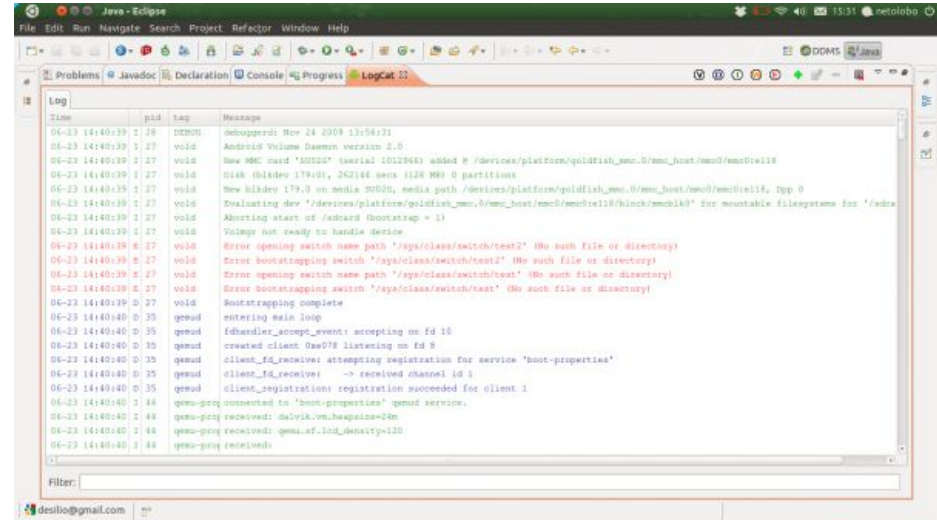
I — Info

W — Warning

E — Error

Para abrir a view do LogCat:

menu Window -> Show View -> Other -> Android -> LogCat



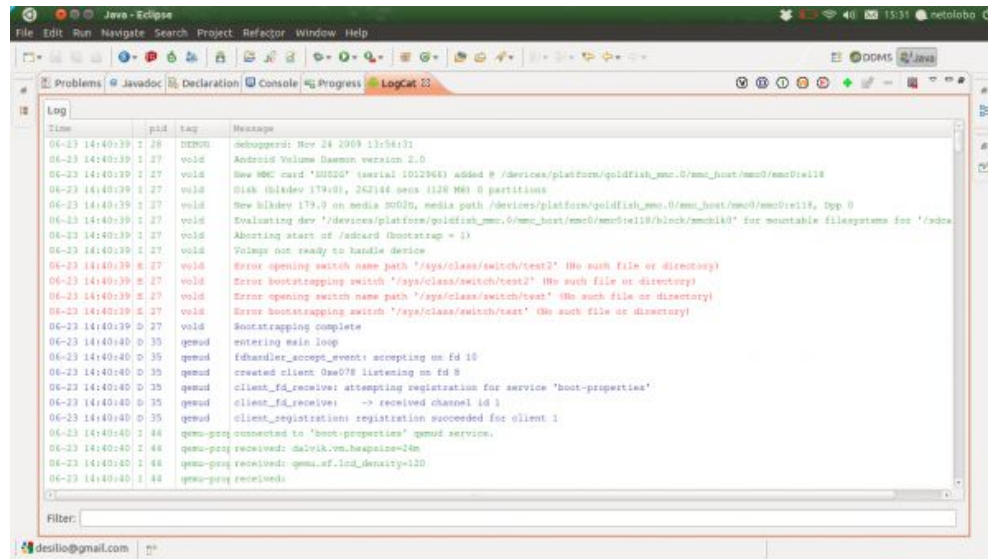
LogCat

A view contém todos os logs do Android, incluindo:

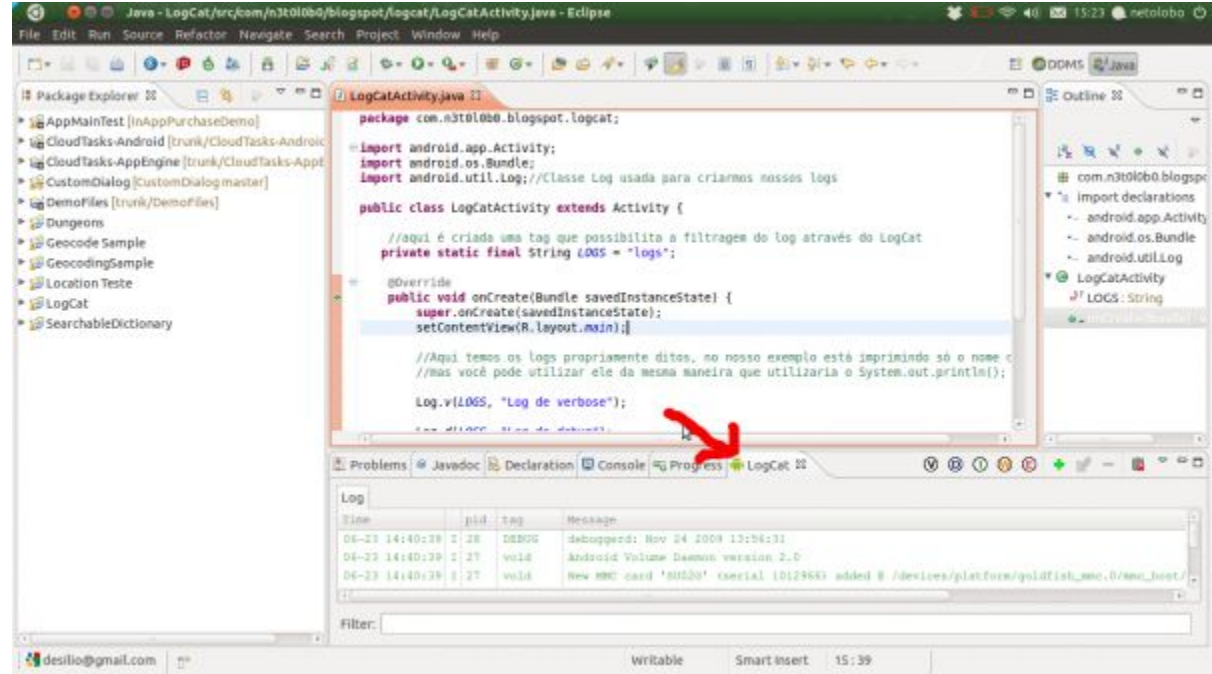
- os logs criados em código
- os logs do sistema

É possível filtrar quais logs deseja ver, por tipo, tag, projeto etc.

Métodos chamados em código para criar um log: `Log.v()` `Log.d()` `Log.i()` `Log.w()` and `Log.e()`



LogCat



- Prática: Insira o método abaixo no método onCreate de uma Activity.
Log.i("teste_info", "Teste do LogCat!");

LogCat

Boas práticas:

Log.e : Indicam erro. Pode ser usado, por exemplo, dentro de um tratamento de um *catch*.

Log.w : Utilizado para indicar um comportamento inesperado, porém não indica com certeza um erro.

Log.i : Utilizado para postar informações úteis para o log. Por exemplo : se você conectou com sucesso a um servidor.

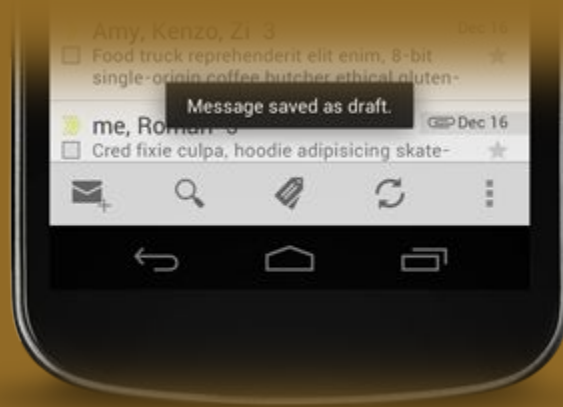
Log.d : Utilizado para fins de depuração. Se você quiser imprimir um monte de mensagens para que você possa registrar o fluxo exato de seu programa, use este. Se você quiser manter um registro de valores de variáveis , use este .

Log.v : Utilizado para registrar detalhes de uma parte específica do seu aplicativo. Como eventos que ocorrem.

Log.wtf : Utilizado para reportar uma falha que nunca deveria acontecer.

Toast

- Toast é uma mensagem de notificação que é exibida um determinado período de tempo, e automaticamente desaparece.
- Pode ser usado para fins de depuração.



Toast

Exemplo:

```
Context context = getApplicationContext();  
CharSequence text = "Hello toast!";  
int duration = Toast.LENGTH_SHORT;  
  
Toast toast = Toast.makeText(context, text, duration);  
toast.show();
```

Para setar sua localização na tela:

```
toast.setGravity(Gravity.TOP|Gravity.LEFT, 0, 0);
```

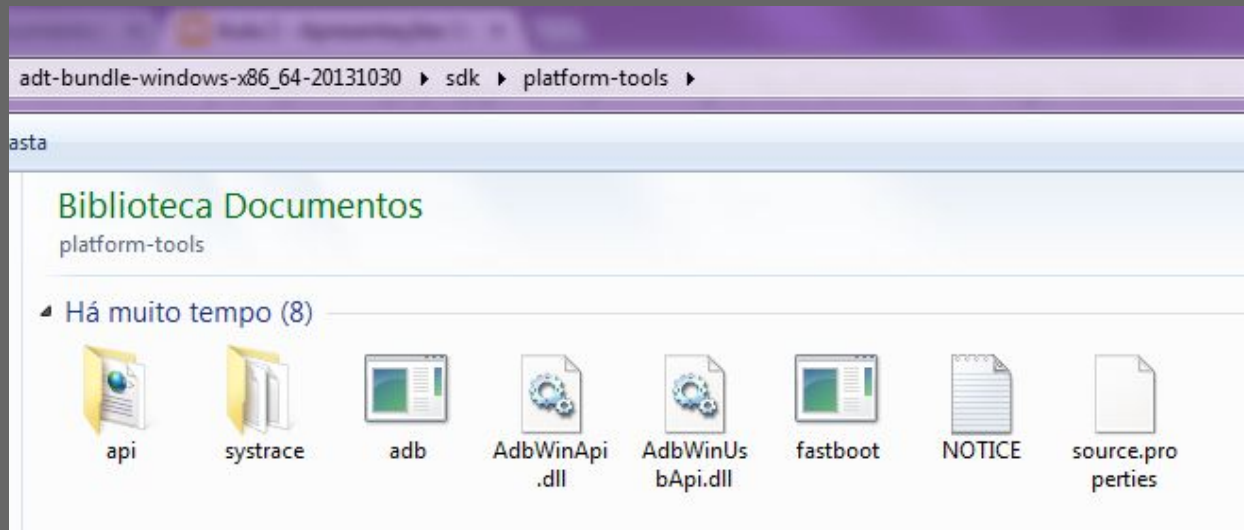
adb

- O adb (**Android Debug Bridge**) consiste em uma aplicação cliente-servidor que permite gerenciar o estado de dispositivos reais ou emulados.
- Além da **comunicação entre computador e dispositivo Android**, o ADB permite ainda que se instalem aplicações, que se copie informação entre o PC e o equipamento Android e também que se corram alguns comandos na shell do Android.



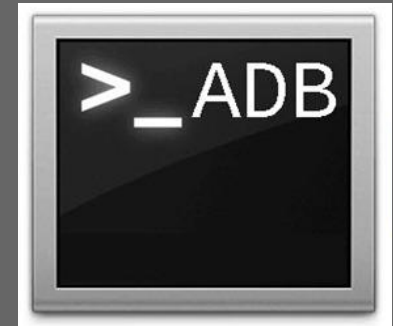
adb

- Endereço: sdk -> platform-tools -> adb



adb

- local: `.../adt-bundle-linux-x86_64-20140321/sdk/platform-tools`
- o acesso deve ser via Terminal (Ubuntu) e via Prompt de Comando (Windows)
- Dicas:
<http://pplware.sapo.pt/smartphones-tablets/android/como-usar-o-android-debug-bridge-adb/>



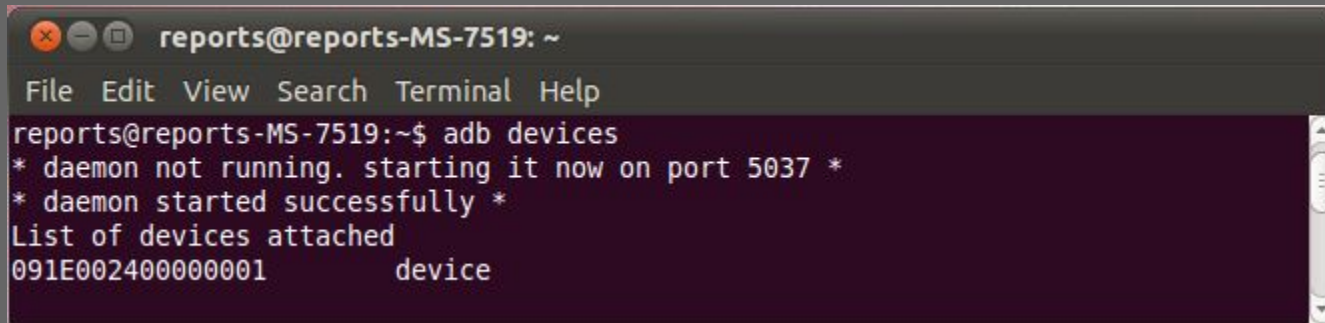
adb

- Alguns comandos em terminal:

`adb devices` : para verificar os dispositivos disponíveis

`adb shell` : entrar no shell do Android

`adb push` : enviar arquivos ao dispositivo

A screenshot of a terminal window titled 'reports@reports-MS-7519: ~'. The window has a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal output shows the command 'reports@reports-MS-7519:~\$ adb devices' being executed. The response is '* daemon not running. starting it now on port 5037 *' followed by '* daemon started successfully *'. Below this, it says 'List of devices attached' and then shows '091E002400000001' followed by 'device' on the next line. The terminal has a dark background and a scrollbar on the right side.

```
reports@reports-MS-7519: ~
File Edit View Search Terminal Help
reports@reports-MS-7519:~$ adb devices
* daemon not running. starting it now on port 5037 *
* daemon started successfully *
List of devices attached
091E002400000001      device
```


adb

- Dica: se o dispositivo não aparecer na lista de dispositivos mate o processo e reinicie o adb utilizando os dois comandos abaixo.

```
adb kill-server
```

```
adb start-server
```

- Dica (Ubuntu) : criar um atalho para adb

```
sudo ln -s
```

```
~/opt/adt-bundle-linux-x86_64-20140321/sdk/platform-tools/a  
db
```



Dicas

1. Atalhos Eclipse:

Ctrl+space : mostra possíveis opções para o preenchimento de um campo

Ctrl+shift+F : organiza o código e identa



Prática

