Módulo 01

android

Prof. Josias Paes

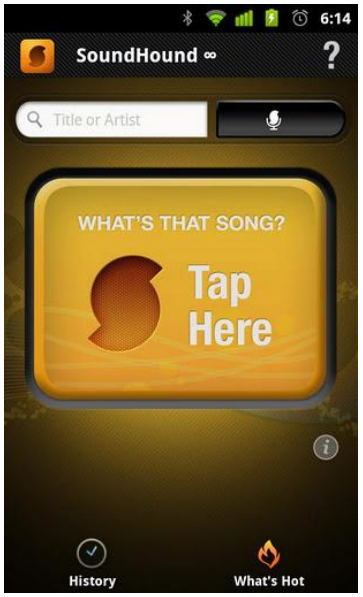
Agenda

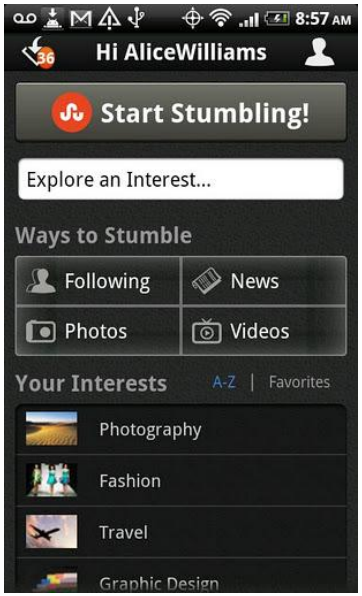
- Activity
  - Users Interface
    - Activitys
    - Fragments
- Intent

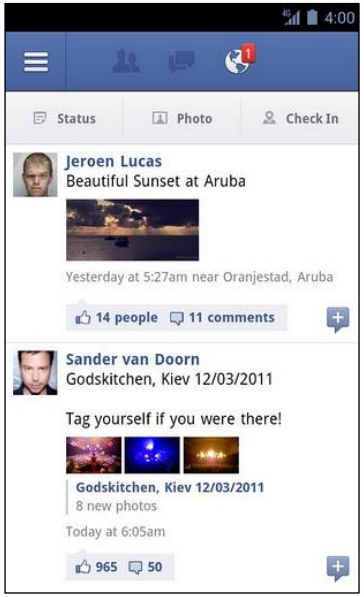
Activity


Google
Apps
Inbox
honeycombdemo@gmail.com
1
JR Raphael
Feb 27
Meeting today at 1:30 — Everyone's on-board. Shouldn't take more than an hour. Oh, and don't forget to bring ...
JR Raphael 2
Feb 27
Note to self — Remember to set up a Gmail account for the Honeycomb demo
eSarcasm .com
Feb 27
Hey, how's it going? — What do you think of the Xoom? I can't decide whether to get it or wait to see what other ...
PM
12
25
Tue, May 17
San Francisco
Cloudy
H:61°
L:50°
52°
GNews Tech
Facebook
Android Power
eSarcasm
2D Glasses
TECH
CNN
Different dimensions for glasses
12:25

Activity
Users Interface







GUI em Android são codificadas utilizando a tecnologia XML!

- Cada elemento é representado como uma TAG (seja uma *view* ou um *layout*)
  - View – Componente de interface: rótulos, botões, listas, caixa de textos
  - Layout – Local onde é possível inserir diversas Views. É um repositório ou container. É possível inserir layouts dentro de outros layouts

```
<?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">
</RelativeLayout>
```

- Nessa TAG podemos trabalhar com diferentes atributos de configuração do componente

- **As Tags mais conhecidas são:**
- TextView
- EditText
- Button
- Spinner
- ImageView
- ProgressBar
- CheckBox
- RadioButton
- ...

```
<TextView
    android:id="@+id/tvNomeusuario"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Digite seu nome" />
```

- Os atributos de TAGS mais comuns são:
  - **android:id** – cria o identificador do componente gráfico
  - **android:text** – indica o texto inicial a ser exibido no componente
  - **android:layout_width** – define a largura do componente
  - **android:layout_height** – define a altura do componente

```
<TextView
    android:id="@+id/tvNomeusuario"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Digite seu nome" />
```

- Fill Model
  - Todas as views dentro de um layout deve definir seu ...
    - **android:layout_width** e o
    - **android:layout_height**

  ... para que os mesmos possam se adequar como desejado na tela do dispositivo. Os valores possíveis são:
    - **wrap_content** – significa que o View deve preencher o seu espaço natural para que não fique “grande”;
    - **match_parent / fill_parent** – significa que o View deve ocupar todo o espaço permitido na tela.

- Gravity
  - Com essa propriedade é possível alterar o alinhamento dos elementos na tela:
    - **android:gravity="left|center"**
    - **android:gravity="right"**
    - **android:gravity="center"**
    - **android:gravity="bottom"**

- Padding
  - Espaço definido entre os Views
    - **android:padding="5px"**
    - **android:paddingLeft="5px"**
    - **android:paddingRight="5px"**
    - **android:paddingTop="10px"**
    - **android:paddingBottom="10px"**

- Em alguns atributos (por exemplo, **android:id** ou **android:text)** é muito comum instruções que precedem o **@**
  - **@+id/**algumacoisa
  - **@id/**algumacoisa
  - **@string/**algumacoisa
  - **@drawable/**algumacoisa

- O **@** apresenta uma sinalização para que possa existir uma interligação do código XML com o código Java. É o sinal para que informações sejam registradas/utilizadas a partir da classe R.
  - Todo componente que possui o **@+id/** pode ser utilizando pelo código Java
  - Um atributo que faça uso do recurso **@string/** está utilizando constantes definidas no arquivo strings.xml
    - Também pode ser utilizado diretamente no código Java
  - Um atributo que faça uso do recurso **@drawable** está utilizando uma imagem existente no projeto Android9

Activity
Users Interface

- Como acontece a interação do arquivo XML com o código Java?
  - Acontece dentro do método onCreate() através do método:
    - **setContentView(R.layout.main);**
      - **main** é o nome do arquivo XML
  - Para acessar os componentes de interface do arquivo XML é necessário utilizar o método:
    - **findViewById(R.id.idDoElemento);**
    - Para que funcione é preciso definir o **id** do elemento previamente:
      - **android:id="@+id/elemento"**

Activity
Users Interface

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

    <TextView
        android:id="@+id/tvNomeusuario"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Digite seu nome" />

</LinearLayout>
```

```java
public class MainActivity extends Activity{

    private TextView tvNomeUsuario;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        tvNomeUsuario = (TextView)findViewById(R.id.tvNomeUsuario);
    }
}
```

Classe View

- A classe *View* é a classe mãe de todos os componentes visuais do Android.

- Uma *View* geralmente é chamada de componente.

- Existe dois tipos de componentes:
  - **Widgets**: que herdam diretamente da classe *View (Button,* ImageView e TextView).
  - **Layouts**: que herdam diretamente da subclasse *ViewGroup* e são popularmente chamados de layouts.

- A função de um layout é organizar a disposição dos componentes na tela.

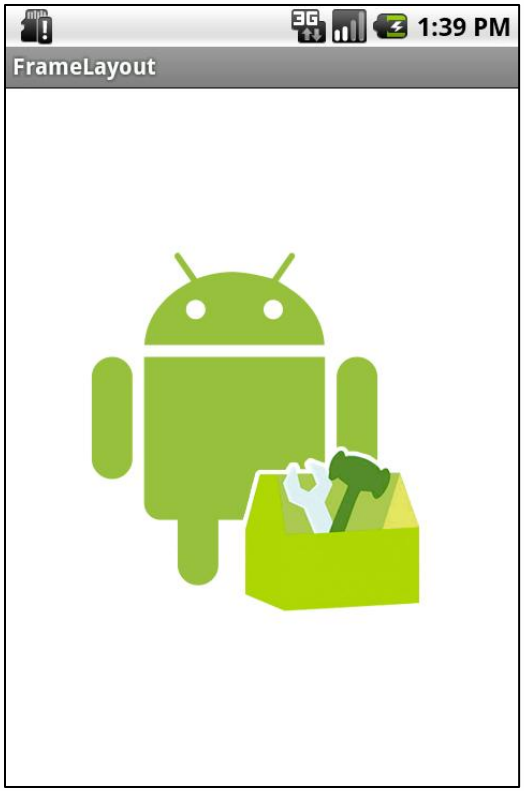
- Os principais gerenciadores de layout são:
  - **FrameLayout:** O tipo mais comum e simples de layout, utilizado quando um componente precisa ocupar a tela inteira.
  - **LinearLayout:** Utilizado para organizar os componentes na horizontal ou na vertical.
  - **RelativeLayout:** Permite posicionar um componente relativo à outro, por exemplo, abaixo, acima, ou ao lado de um componente já existente.
  - **SlidingPaneLayout:** Permite criar um menu dinâmico na lateral da aplicação. Ex.: Facebook

FrameLayout

- A classe *FrameLayout*, é a mais simples de todos os gerenciadores de layout do Android.
- Utilize quando a tela possuir apenas um componente que pode preencher a tela inteira.
- Um componente inserido no *FrameLayout* ***SEMPRE*** será posicionado no canto superior esquerdo da tela.
- Pode ocupar a tela inteira ou não.
- É possível inserir mais de um componente no *FrameLayout,* mas sempre os últimos componentes inseridos ficarão na frente dos demais, seguindo o conceito de pilha.

FrameLayout

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

  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icon"/>

</FrameLayout>
```

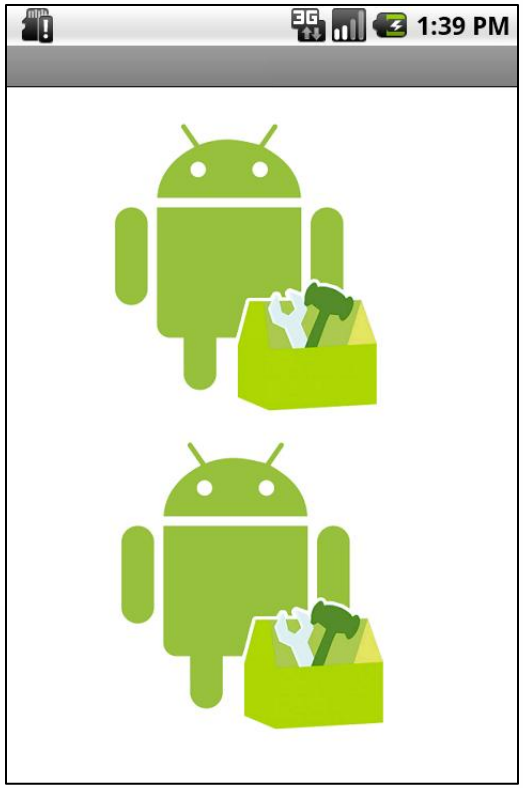


LinearLayout

- A classe *LinearLayout* é um dos gerenciadores de layout mais utilizados.
- É possível organizar os componentes na horizontal, ou na vertical.
  - Através do atributo *android:orientation.*
- Se o parâmetro *android:orientation* não for definido, por padrão o LinearLayout organiza os componentes na horizontal.
- Caso os componentes extrapolem a área útil da tela, os componentes que estiverem fora desta área não serão exibidos.

LinearLayout

```
<?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="vertical">

  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icon"/>

  <ImageView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icon"/>

</LinearLayout>
```



RelativeLayout

- A classe *RelativeLayout* pode posicionar os componentes ao lado, abaixo, ou acima de um outro componente já existente.

- Para isso é necessário definir um id para cada componente.

- Como o posicionamento de um componente depende de um outro componente, o componente a ser referenciado **DEVE** aparecer antes no layout do arquivo XML.

- Os seguintes atributos podem ser usados para informar a posição de um componente:

  - **android:layout_toLeftOf=*"@id/idDoElemento"***
  - **android:layout_toRightOf=*"@id/idDoElemento"***
  - **android:layout_alignTop=*"@id/idDoElemento"***
  - **android:layout_alignLeft=*"@id/idDoElemento"***
  - **android:layout_alignRight=*"@id/idDoElemento"***
  - **android:layout_alignBelow=*"@id/idDoElemento "***

RelativeLayout

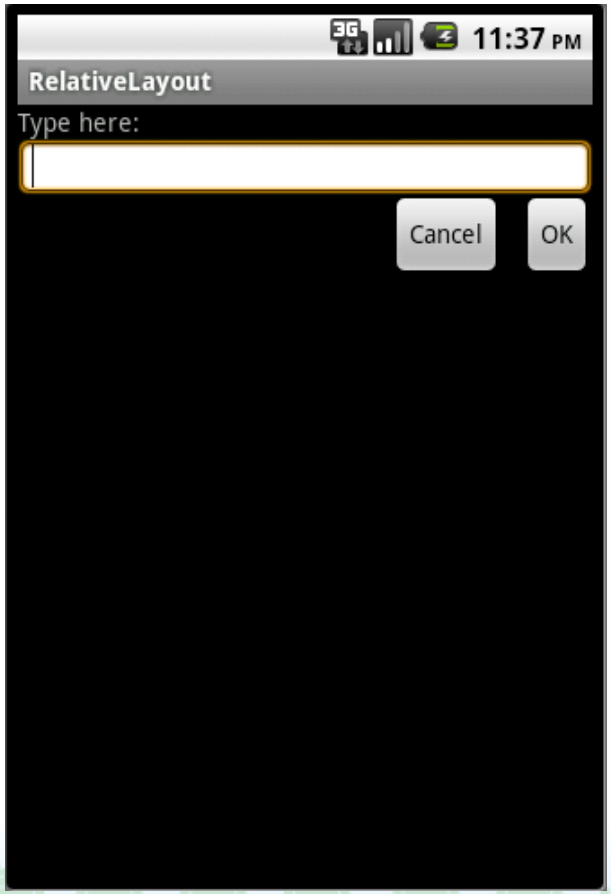
- Posição Relativas ao Container

- É usado para relacionar a posição de uma View com o container definido no arquivo XML:
  - **android:layout_alignParentTop** – alinha o View no topo do container
  - **android:layout_alignParentBottom** – alinha o fundo do View no fundo do container
  - **android:layout_alignParentLeft** – alinha o lado esquerdo do View com o lado esquerdo do container
  - **android:layout_alignParentRight** – alinha o lado direito do View com o lado direito do container

PS: Todos estes são atribuídos através de uma valor booleano

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">

  <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/label"/>

  <EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_below="@+id/label"/>

</RelativeLayout>
```



SlidingPaneLayout

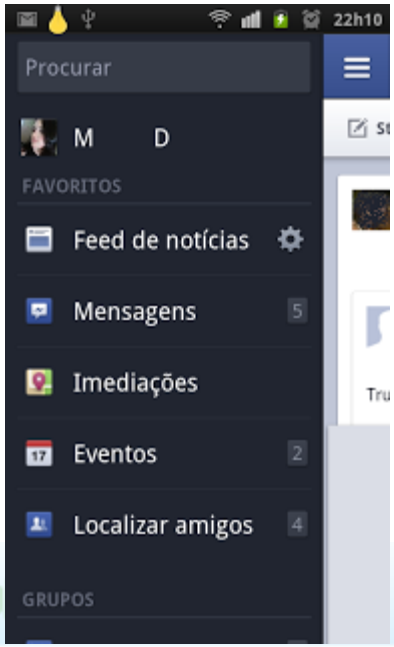
- A classe *SlidingPaneLayout* cria dinamicamente um menu lateral de fácil acesso em sua interface gráfica.

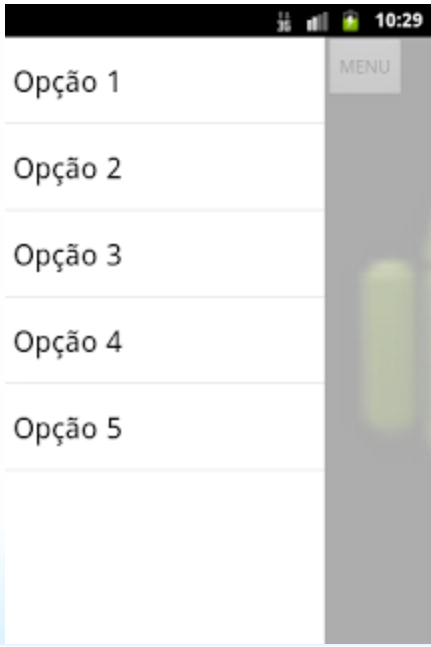
- Para isso é necessário criar dois (e somente dois) layouts no interior do XML.

- A classe irá identificar automaticamente que o primeiro layout adicionado é o menu e o segundo layout o corpo da activity.

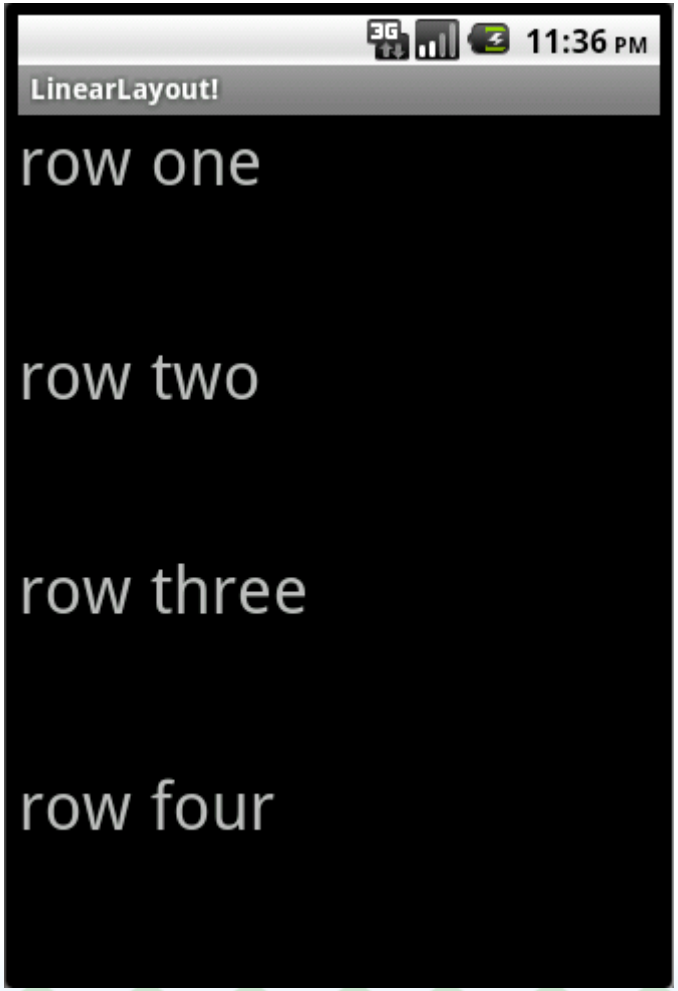
SlidingPaneLayout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.SlidingPaneLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/sliding_pane_layout"
  android:layout_width="match_parent"
  android:layout_height="match_parent" >

  <!-- Menu Lateral -->
  <ListView
    android:id="@+id/left_pane"
    android:layout_width="280dp"
    android:layout_height="match_parent"
    android:layout_gravity="left" />

  <!-- Conteúdo da tela -->
  <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#ff333333" >

  </RelativeLayout>

</android.support.v4.widget.SlidingPaneLayout>
```



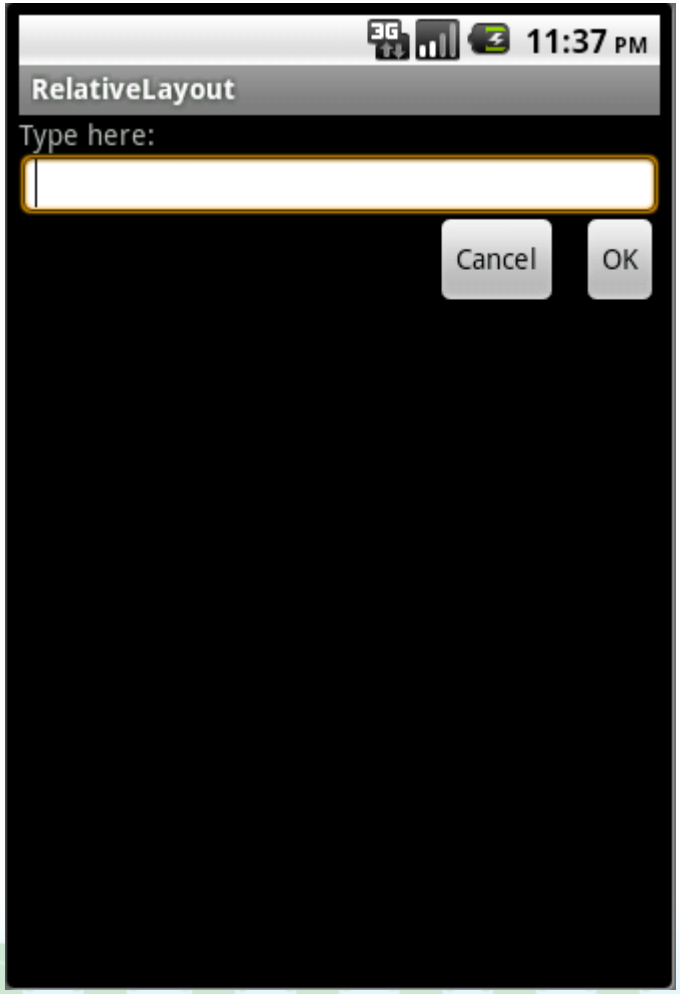


Views - TextView




```
<TextView
    android:id="@+id/tvNome"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/nome"
    android:textColor="#E7E7E7"
    android:textStyle="bold|italic"
    android:textSize="14sp" />
```

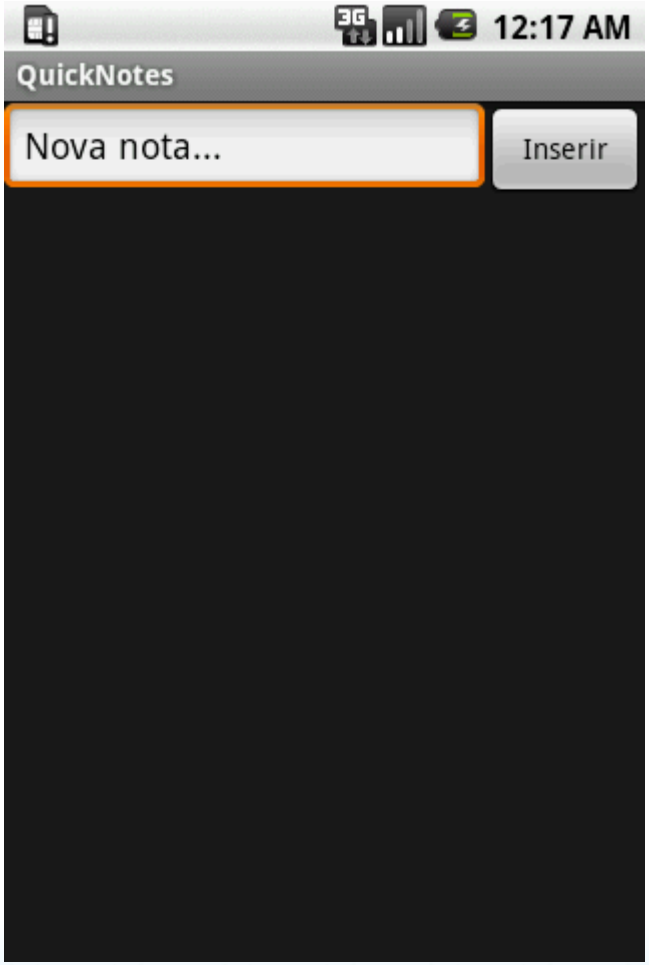
Views - Button




```
<Button
    android:id="@+id/btOk"
    android:layout_width="fill_parent"
    android:layout_height="48dp"
    android:layout_marginRight="3dp"
    android:background="@drawable/bt_post_clear" />
```

```
public class MainActivity extends Activity implements OnClickListener{

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button botao = (Button) findViewById(R.id.botao);
        botao.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
      // Implementar aqui o que fazer
      // quando acontecer um evento.
    }

}
```

Views – EditText




```
<EditText
    android:id="@+id/etSenha"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/edit_text_holo_light"
    android:hint="@string/msg05"
    android:inputType="textPassword"
    android:textColor="#E7E7E7"
    android:textSize="16sp" />
```

Métodos relevantes:

```
EditText et = (EditText) findViewById(R.id.edittext);
et.setText();            //Seta um texto no EditText
et.getText().toString();
//Captura o texto de um EditText
```

Views – CheckBox


```
<CheckBox
    android:id="@+id/cbSalvar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/edit_text_holo_light"
    android:text="Salvar"
    android:textColor="#E7E7E7"
    android:textSize="16sp" />
```

Métodos relevantes:

CheckBox cb = (CheckBox) findViewById(R.id.*checkbox*);

//Verifica se o elementos está selecionado
cb.isChecked();
//Seleciona o check em questão
cb.setChecked();

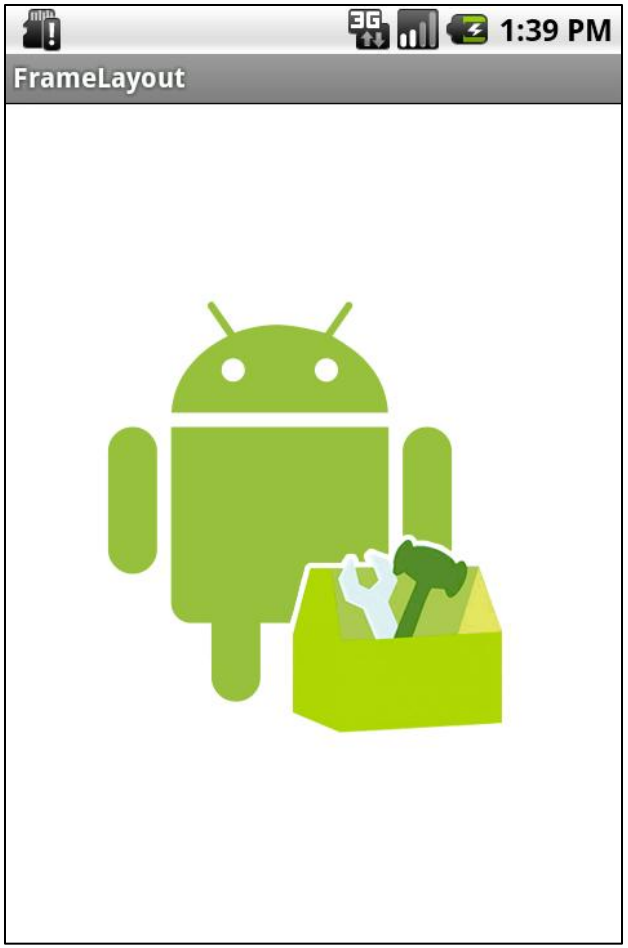
Views - RadioButton


```
<RadioGroup
    android:id="@+id/rgroup"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical" >

    <RadioButton
        android:id="@+id/rb01"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Radio Button ..." />

    <RadioButton
        android:id="@+id/rb02"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Radio Button ..." />
</RadioGroup>
```

- Métodos relevantes:

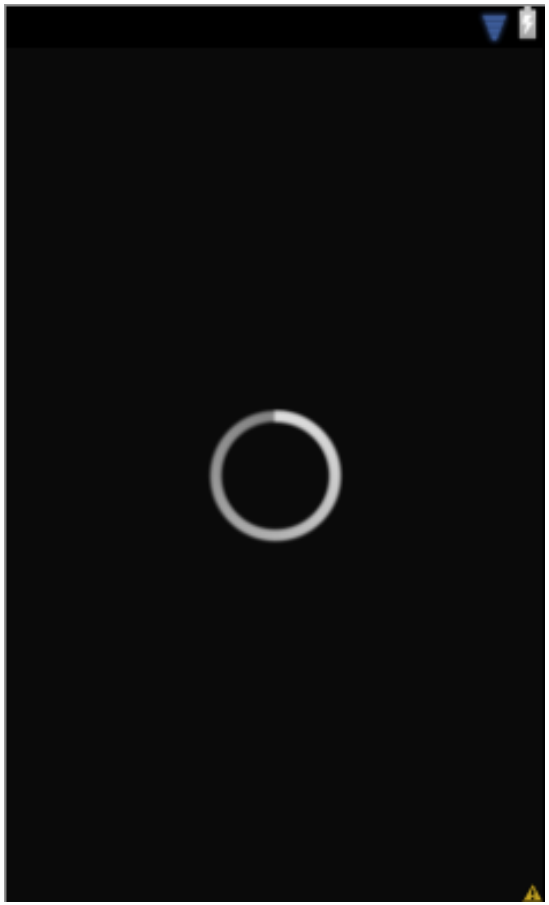
```
RadioButton rb = (RadioButton)findViewById(R.id.radio_red);
rb. getCheckedRadioButtonId()                    //Verifica se o elementos está selecionado

RadioGroup rg = (RadioGroup)findViewById(R.id.rgroup);
rg.check(R.id.radio_blue);          //Marca um RadionButton especifico
rg.clearCheck();                    //Desmarca os RadioButton selecionado
rg.getCheckedRadioButtonId();       //Captura o id do RadioButton selecionado
```

Views - ImageView




```
<ImageView
    android:id="@+id/ivFotoBigPost"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_gravity="center"
    android:layout_marginBottom="5dp"
    android:layout_marginLeft="5dp"
    android:layout_marginRight="5dp"
    android:layout_weight="1"
    android:adjustViewBounds="true"
    android:scaleType="fitXY" />
```

Views – ImageView



```
<ProgressBar
    android:id="@+id/progressBar"
    android:layout_width="100dip"
    android:layout_height="100dip"
    android:layout_gravity="center" />
```

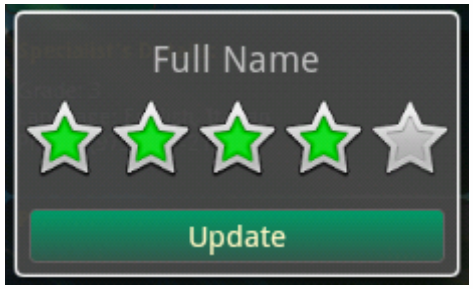
Métodos relevantes:

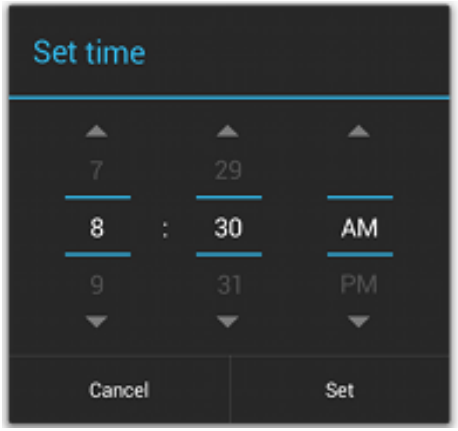
```
ProgressBar pb = (ProgressBar) findViewById(R.id.progressBar);

//Exibe
pb.setVisibility(View.VISIBLE);

//Esconde
pb.setVisibility(View.GONE);
```

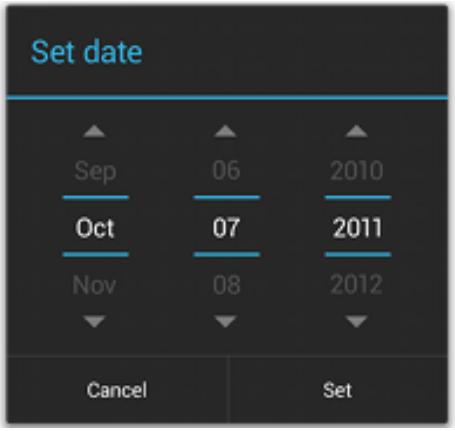
Activity
Users Interface

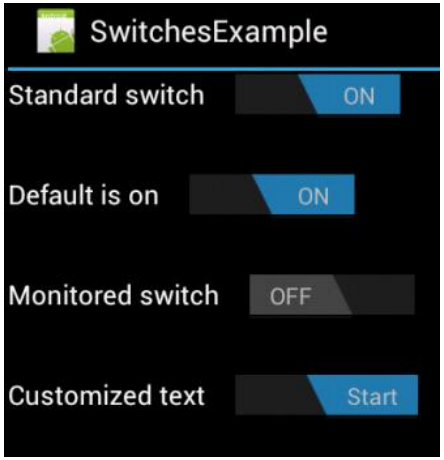
Views

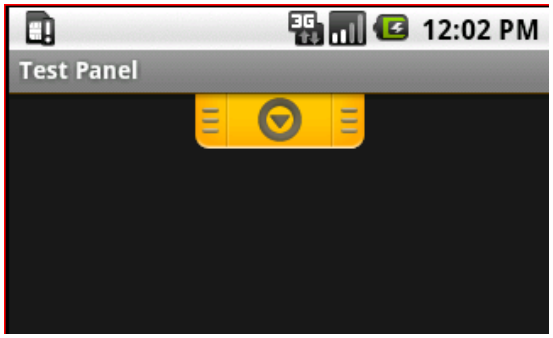
- Rating Bar
- Spinners
- Gallery
- DatePicker
- TimePicker
- Switch
- Sliding Drawer
- TabHost
- WebView
- Dentre outros

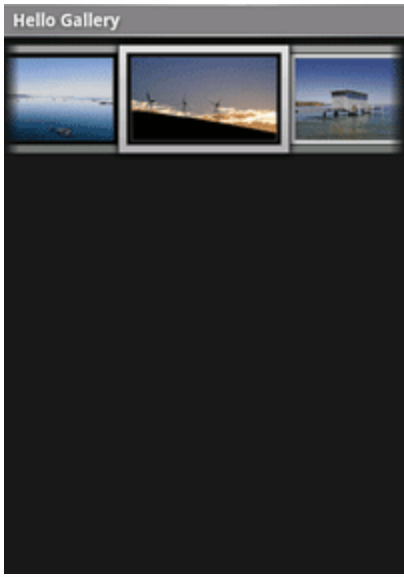


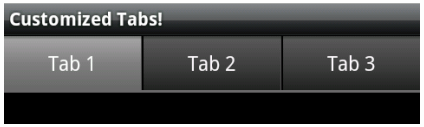


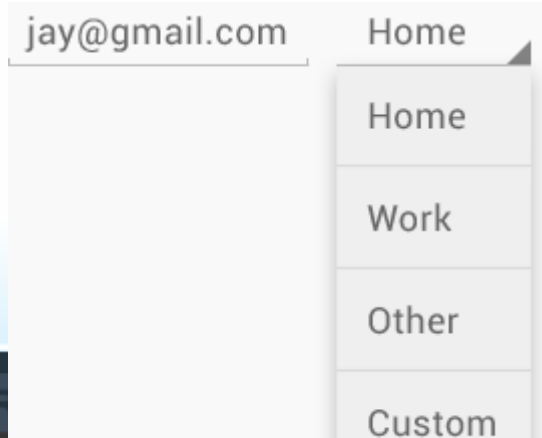








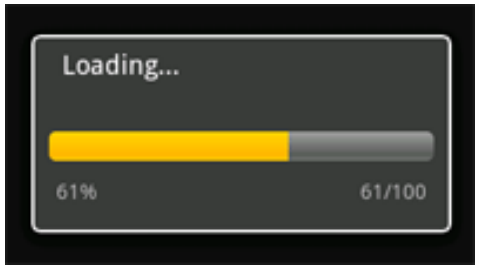


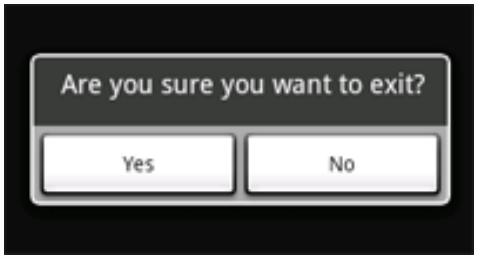


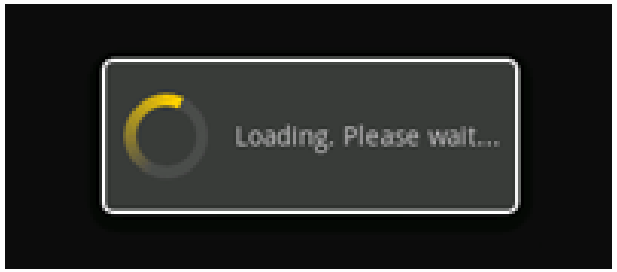
Dialogs
Users Interface

- Dialogs são componentes bastantes importantes em qualquer UI de qualquer software.

- São componentes utilizados para informar ao usuário a situação atual do sistema, mensagens de erro, avisos ou qualquer coisa que esteja relacionada com a prestação de informação.







Dialogs
Users Interface

- AlertDialog

```
public class TesteDialogActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        AlertDialog.Builder dialog = new AlertDialog.Builder(TesteDialogActivity.this);
        dialog.setTitle("Titulo Dialog");
        dialog.setMessage("Olá esse é o nosso primeiro dialog");
        dialog.show();

    }
}
```

Dialogs
Users Interface

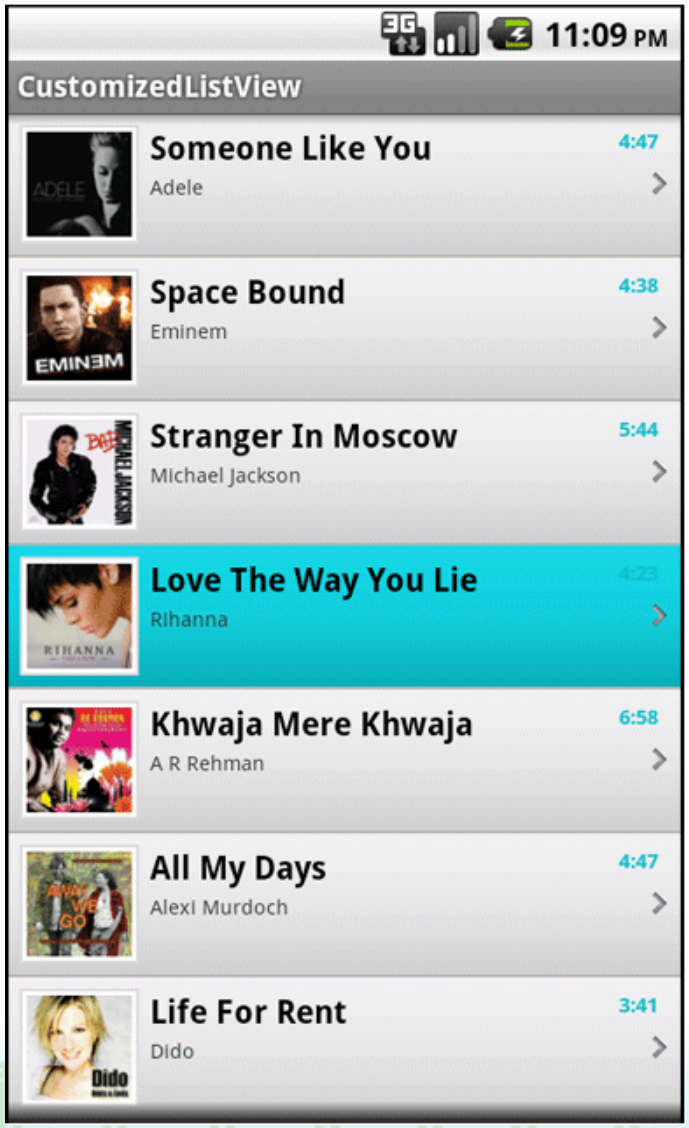
- AlertDialog

```
public class TesteDialogActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        AlertDialog.Builder dialog = new AlertDialog.Builder(TesteDialogActivity.this);
        dialog.setTitle("Titulo Dialog");
        dialog.setMessage("Olá esse é o nosso primeiro dialog");
        dialog.show();

    }
}
```

Dialogs
Users Interface

```
public class TesteDialogActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        AlertDialog.Builder dialog = new AlertDialog.Builder(TesteDialogActivity.this);
        dialog.setTitle("Titulo Dialog");
        dialog.setMessage("Olá esse é o nosso primeiro dialog");
        dialog.setPositiveButton("Ok", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        });
        dialog.setNegativeButton("Cancel", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        });
        dialog.setNeutralButton("Neutro", new OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

            }
        });
        dialog.show();

    }
}
```

ListView



- Componente mais Complexo do Android
- Passos:

1. Criar ListView na UI
2. Criar UI dos Itens
3. Criar Adapter
4. Interligar Adapter e ListView
5. Criar o Holder Content

ListView
Adapter

```
public class CommentsPostAdapter extends ArrayAdapter<Comentario> {
    private ArrayList<Comentario> items;
    private LayoutInflater inflater;
    public CommentsPostAdapter(Context context, int textViewResourceId, ArrayList<Comentario> items) {
        super(context, textViewResourceId, items);
        this.items = items;
        inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        Comentario comment = items.get(position);

        TextView tvComment;

        if (convertView == null) {
            convertView = inflater.inflate(R.layout.itemlist_comment, null);

            tvComment = (TextView)convertView.findViewById(R.id.tvComment);

            ProcessViewHolder holder = new ProcessViewHolder();
            holder.tvComment = tvComment;

            convertView.setTag(holder);
        }else{
            ProcessViewHolder holder = (ProcessViewHolder) convertView.getTag();
            tvComment = holder.tvComment;
        }
        tvComment.setText(comment.getComentario());
        return convertView;
    }
    public class ProcessViewHolder {
        public TextView tvComment;
    }
}
```

Módulo 01

android

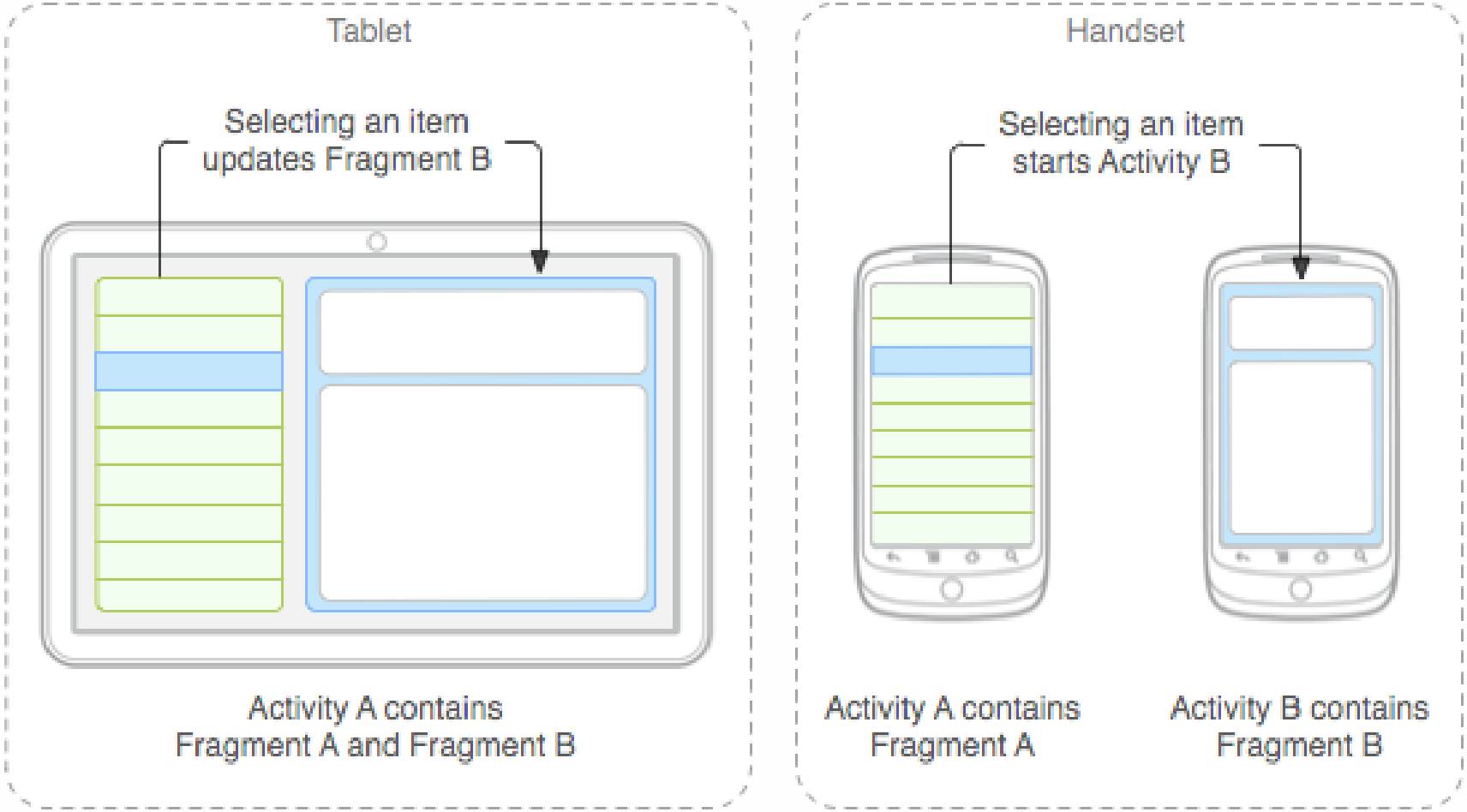
Prof. Josias Paes

Fragments
Users Interface

Fragments
Users Interface

- Android introduziu fragmentos no Android 3.0 (API nível 11)
- Veio para apoiar projetos de interface do usuário mais dinâmicas e flexíveis em telas grandes (comprimidas)
- Fragmentos permite construir telas sem a necessidade de gerenciar mudanças complexas em suas hierarquia
- Ao dividir o layout de uma atividade em fragmentos, você se torna capaz de modificar a aparência da atividade em tempo de execução

Fragments
Users Interface



Fragments
Users Interface

- Para criar uma UI baseada em Fragments deve-se seguir os seguintes passos:

1. Criar o layout do fragmento (layout tradicional)
2. Criar a classe que herda **Fragment**
   - Perceba que utilizando **Fragments** nós deixamos de utilizar a classe **Activity**
3. Criar o layout **gerenciador**
   - Layout responsável por organizar os fragmentos
4. Criar a classe que herda **FragmentActivity**
   - Classe que irá gerenciar os fragmentos

Fragments
Users Interface

1. Criar o layout do fragmento (layout tradicional)

asd.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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:background="@drawable/bg"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:id="@+id/tvText"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Digite seu nome"/>
    <EditText
        android:id="@+id/etNome"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />

</LinearLayout>
```

2. Criar a classe que herda **Fragment**

```
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;

public class ExampleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View layout = inflater.inflate(R.layout.asd, container, false);

        //Controlar componentes utilizando o View
        EditText et = (EditText)layout.findViewById(R.id.etNome);

        return layout;
    }
}
```

3. Criar o layout **gerenciador**

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <fragment android:name="com.example.news.ExampleFragment"
            android:id="@+id/list"
            android:layout_weight="1"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
    <fragment android:name="com.example.news.ExampleFragment2"
            android:id="@+id/viewer"
            android:layout_weight="2"
            android:layout_width="0dp"
            android:layout_height="match_parent" />
</LinearLayout>
```

Como um componente qualquer

4. Criar a classe que herda **FragmentActivity**

```
import android.os.Bundle;
import android.support.v4.app.FragmentActivity;

public class MainActivity extends FragmentActivity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

}
```