PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

A classe Intent

Professor: Danilo Giacobo

OBJETIVOS DA AULA

- Alternar entre múltiplas telas de uma aplicação Android.
- Conhecer a classe *Intent*.
- Associar as telas do aplicativo a *Activities*.
- Realizar a chamada de *Activities* com e sem parâmetros.
- Recuperar dados após a execução de uma *Activity*.
- Apresentar técnicas para chamar as telas e os recursos nativos dos dispositivos Android.

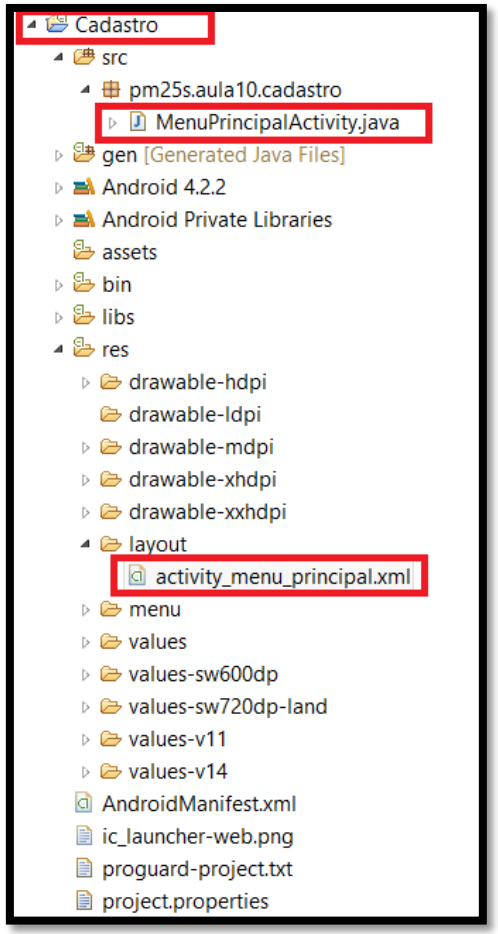
INTRODUÇÃO

- O desenvolvimento de software para dispositivos móveis difere em vários aspectos do tradicional desenvolvimento de aplicações para *desktop* e *Web*.
- Ao projetar um aplicativo para dispositivos móveis devemos levar em consideração a disposição das informações na tela para o usuário.
- Telas com muitos componentes visuais requerem a utilização de barras de rolagem, o que dificulta consideravelmente o uso da aplicação.
- Uma técnica interessante é apresentar telas com poucos componentes visuais, sendo estes agrupados por conteúdo.
- Preferencialmente, os recursos mais utilizados devem apresentados de forma mais simples, possuindo atalho nas telas iniciais. Os recursos menos utilizados podem ser estar dentro de menus ou submenus.

ESTUDO DE CASO

- Para exemplificarmos o uso de múltiplas telas, iremos desenvolver nesta aula um protótipo de um programa para um lançamento de venda, o qual será formado inicialmente por duas telas: Menu Principal e Tela de Lançamento.
- Na tela principal, o usuário poderá optar por abrir a tela de lançamento ou sair do aplicativo; na tela de lançamento, o usuário informará os dados da venda de um produto, como o código do mesmo, quantidade e valor de venda. Ao final, funcionalidades, como uma tela para pesquisa de produtos e tela de confirmação, serão apresentadas.
- Para testar as funcionalidades, criaremos um projeto Android com o nome de Cadastro.
- A *Activity* principal será chamada de MenuPrincipalActivity.java.
- O arquivo XML da interface gráfica será chamado de activity_menu_principal.xml.

ESTRUTURA DO PROJETO



INTERFACE GRÁFICA DO MENU

```
<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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MenuPrincipalActivity"
    android:orientation="vertical" >

    <Button
        android:id="@+id/btLancamento"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/lancamento" />

    <Button
        android:id="@+id/btSair"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/sair" />

</LinearLayout>
```

CODIFICAÇÃO DA CLASSE ACTIVITY DO MENU

```java
package pm25s.aula10.cadastro;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;

public class MenuPrincipalActivity extends Activity {

    private Button btLancamento;
    private Button btSair;

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

        btLancamento = (Button) findViewById(R.id.btLancamento);
        btSair = (Button) findViewById(R.id.btSair);

        btLancamento.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btLancamentoOnClick();
            }
        });
        btSair.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btSairOnClick();
            }
        });
    }

    public void btLancamentoOnClick() {

    }

    public void btSairOnClick() {
        finish();
    }
}
```

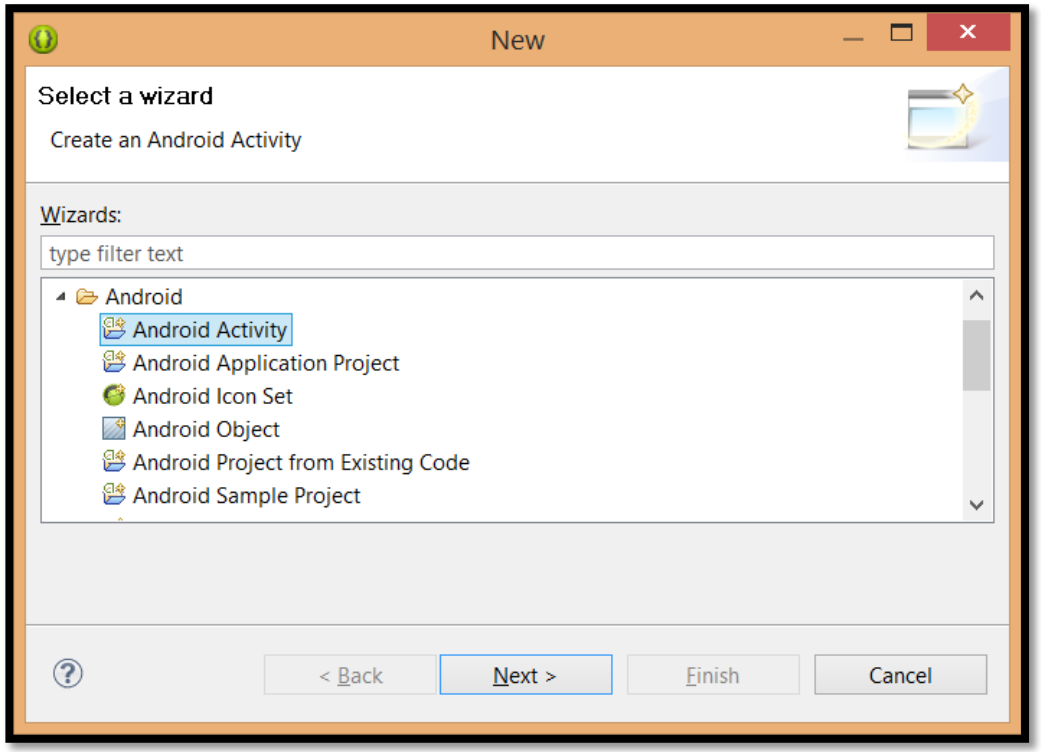
CLASSES *ACTIVITY*

- Activities são classes do Android que possuem a função de realizar alguma atividade ou tarefa dentro da aplicação.
- Na maioria das vezes (porém, não sempre) as Activities são associadas à tela.
- Na nossa aplicação, até o momento, temos uma situação na qual cada tela é tratada por uma *Activity*.
- Toda *Activity* herda funcionalidades da classe *Activity* ou de alguma de suas subclasses (ListActivity, MapActivity, etc.).
- Para ser iniciada, uma *Activity* precisa executar o método **onCreate()**.

CRIANDO UMA SEGUNDA TELA PARA A APLICAÇÃO

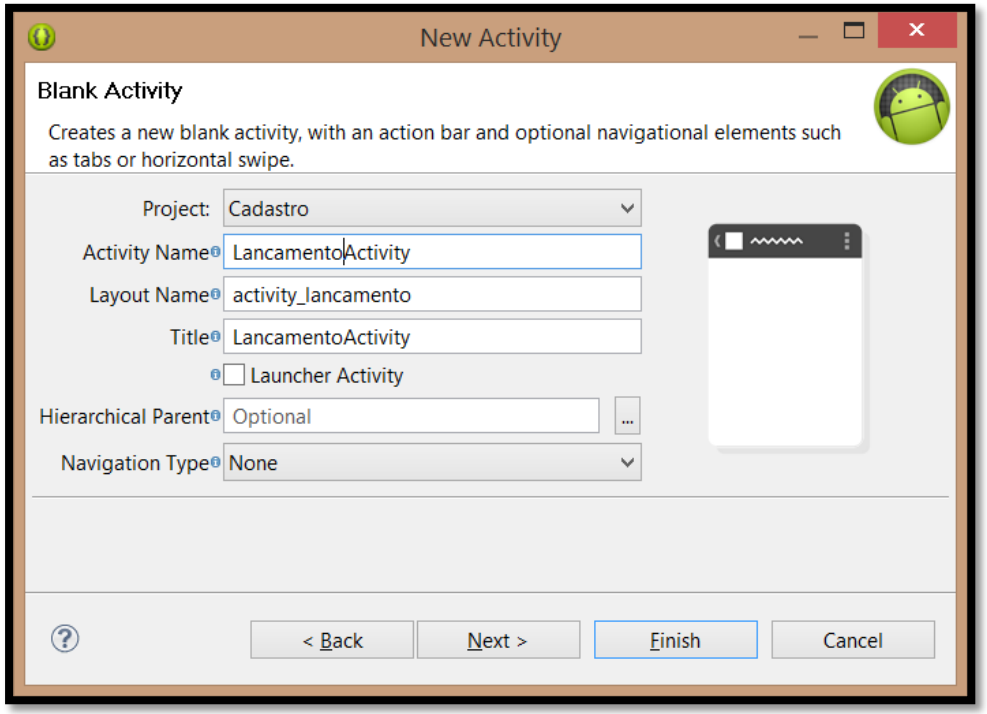
- Para a apresentação da tela de lançamento, o primeiro passo é criar uma *Activity* que tratará a tela de lançamento. Basicamente uma *Activity* é uma classe Android.

- Para criar uma *Activity* que processará a tela de lançamento, no IDE Eclipse, deve-se clicar com o botão direito no projeto, escolhendo a opção **New > Other**. Na janela apresentada, escolhemos a opção **Android Activity**, a qual se encontra dentro da categoria Android.



CRIANDO UMA SEGUNDA TELA PARA A APLICAÇÃO

- Na tela seguinte, deve-se selecionar um *template* para a criação da *Activity*.
- Para o nosso exemplo, iremos selecionar a opção **Blank Activity**.
- Na próxima tela, é informado o nome da *Activity* (.java) bem com o nome da tela (.xml).
- Para o exemplo da aula, utilizaremos **LancamentoActivity** e **activity_lancamento**.
- Para concluir, clicamos no botão **Finish**.



INTERFACE GRÁFICA DA TELA DE LANÇAMENTO

- O código de activity_lancamento.xml é apresentado abaixo:

```xml
<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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".LancamentoActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/codigo" />

    <EditText
        android:id="@+id/etCodigo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="number" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/quantidade" />

    <EditText
        android:id="@+id/etQuantidade"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/valor" />

    <EditText
        android:id="@+id/etValor"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="numberDecimal" />

    <Button
        android:id="@+id/btConfirmar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/confirmar" />

    <Button
        android:id="@+id/btListarProdutos"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/listar_produtos" />

</LinearLayout>
```

CÓDIGO JAVA DA TELA DE LANÇAMENTO

```java
package pm25s.aula10.cadastro;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

public class LancamentoActivity extends Activity {

    private EditText etCodigo;
    private EditText etQuantidade;
    private EditText etValor;
    private Button btConfirmar;
    private Button btListarProdutos;

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

        etCodigo = (EditText) findViewById(R.id.etCodigo);
        etQuantidade = (EditText) findViewById(R.id.etQuantidade);
        etValor = (EditText) findViewById(R.id.etValor);
        btConfirmar = (Button) findViewById(R.id.btConfirmar);
        btListarProdutos = (Button) findViewById(R.id.btListarProdutos);
```

```java
        btConfirmar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btConfirmarOnClick();
            }
        });

        btListarProdutos.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btListarProdutosOnClick();
            }
        });
    }

    public void btConfirmarOnClick() {

    }

    public void btListarProdutosOnClick() {

    }
}
```

CLASSE *INTENT*

- A tradução literal de *Intent* é *intenção*, e esta palavra traduz bem a função dessa classe. Por meio dela um aplicativo Android consegue "intencionar" os comandos, não que os mesmos sejam necessariamente executados, mas se o dispositivo permitir, essa intenção se transformará em ação.
- É possível, via classe *Intent*, iniciar o Bluetooth do aparelho (caso ele esteja desligado), ligar o GPS ou, ainda, abrir o *browser* e acessar uma página, e até mesmo abrir o programa de envio de SMS, já preenchendo o destinatário e a mensagem assim como a sua lista de contatos do celular.
- Uma das características mais utilizadas da classe *Intent* é a execução de novas *Activities*.

CHAMANDO UMA NOVA TELA

- Iremos acessar a classe *MenuPrincipalActivity.java* do nosso projeto e no método *btLancamentoOnClick*, incluiremos a lógica descrita abaixo:

```
import android.content.Intent;
```

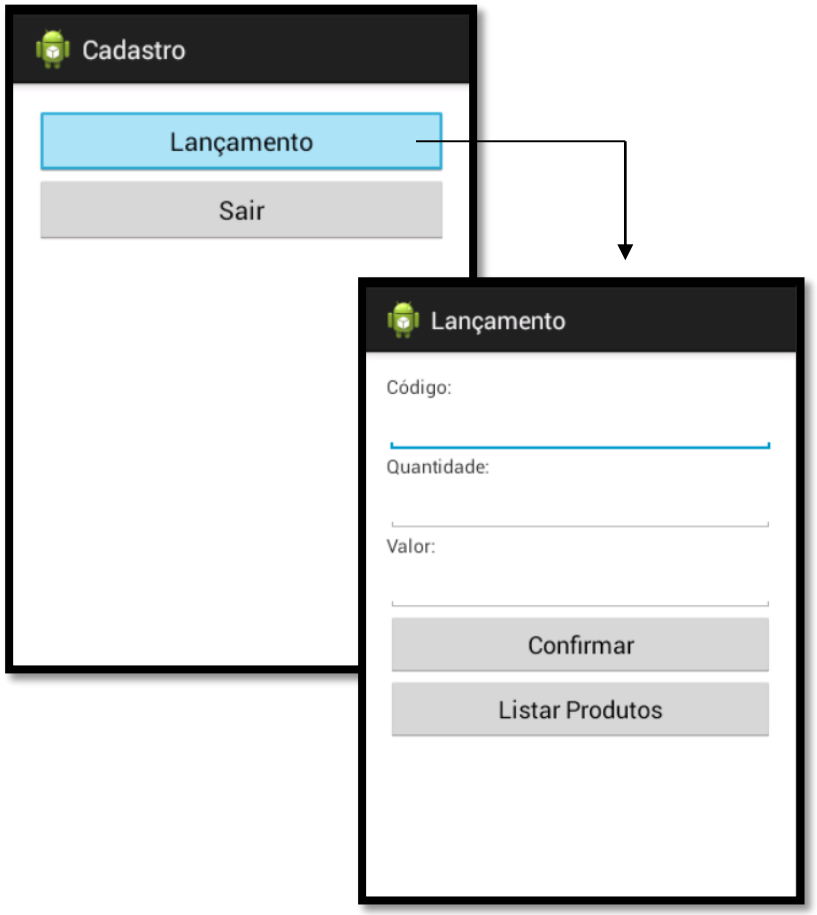
```
public void btLancamentoOnClick() {
    Intent i = new Intent(this, LancamentoActivity.class);
    startActivity(i);
}
```

- Para registrar a intenção de executar uma nova *Activity*, devemos instanciar um objeto do tipo *Intent*, passando por parâmetro a *Activity* que executará a ação e também a *Activity* que será executada.

- Por fim, executamos o comando *startActivity*, passando a intenção que será executada. Após executar o aplicativo, a tela de lançamento será apresentada (ver próximo slide).

CHAMANDO UMA NOVA TELA

- Uma vez apresentada a tela de lançamento, o usuário pode optar por retornar para a tela anterior e para isso, basta clicar no botão *Voltar* do emulador/celular.
- Esse botão fecha a *Activity* atual e retorna para a *Activity* anterior.
- Na plataforma Android não é necessário criar botões para voltar em todas as telas, já que o botão *Voltar* do *device* já faz esta função.



CHAMANDO UMA ACTIVITY PASSANDO PARÂMETRO

- Em algumas situações é necessário passar dados entre uma *Activity* e outra, como, por exemplo, fazer com que a *Activity* que chama envie dados para a *Activity* chamada, fazendo com que esses dados sejam apresentados na tela.
- Para exemplificar esta situação, criaremos uma nova *Activity* chamada **ConfirmarActivity.java**, a qual apresentará uma tela **activity_confirmar.xml** com os dados digitados na tela de lançamento.
- O procedimento para criar uma nova *Activity* foi apresentado com detalhes em slides anteriores e desta forma, nos próximos slides serão apresentados apenas os códigos da interface e da lógica de negócio.

CHAMANDO UMA ACTIVITY PASSANDO PARÂMETRO

```
<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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".ConfirmarActivity" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/codigo_produto" />

    <TextView
        android:id="@+id/tvCod"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/quantidade" />
```

```
    <TextView
        android:id="@+id/tvQtd"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/valor" />

    <TextView
        android:id="@+id/tvValor"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text=""
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <Button
        android:id="@+id/btConfirmarFinal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/confirmar" />
</LinearLayout>
```

CHAMANDO UMA ACTIVITY PASSANDO PARÂMETRO

```java
package pm25s.aula10.cadastro;

import android.app.Activity;
import android.app.AlertDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class ConfirmarActivity extends Activity {

    private TextView tvCod;
    private TextView tvQtd;
    private TextView tvValor;
    private Button btConfirmar;

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

        tvCod = (TextView) findViewById(R.id.tvCod);
        tvQtd = (TextView) findViewById(R.id.tvQtd);
        tvValor = (TextView) findViewById(R.id.tvValor);
        btConfirmar = (Button) findViewById(R.id.btConfirmarFinal);

        btConfirmar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                btConfirmarOnClick();
            }
        });
    }
```

```java
    public void btConfirmarOnClick() {
        AlertDialog.Builder alert = new AlertDialog.Builder(this);
        alert.setTitle("Atenção");
        alert.setMessage("Operação indisponível no momento.");
        alert.setNeutralButton("OK", null);
        alert.show();
    }
}
```

CHAMANDO UMA ACTIVITY PASSANDO PARÂMETRO

- Para chamar a *Activity* anterior, devemos codificar o botão **Confirmar** de **LancamentoActivity.java**, conforme detalhado abaixo:

```
public void btConfirmarOnClick() {
    int cod = Integer.parseInt(etCodigo.getText().toString());
    double qtd = Double.parseDouble(etQuantidade.getText().toString());
    double valor = Double.parseDouble(etValor.getText().toString());

    Intent i = new Intent(this, ConfirmarActivity.class);
    i.putExtra("codigo", cod);
    i.putExtra("quantidade", qtd);
    i.putExtra("valor", valor);

    startActivity(i);
}
```

- Será necessário incluir uma chamada para o método **receberDados()** antes do fechamento do método **onCreate()** na classe *ConfirmarActivity.java*.

CHAMANDO UMA ACTIVITY PASSANDO PARÂMETRO

- O código do método **receberDados()** pode ser colocado ao final da classe **ConfirmarActivity.java**.

```
public void receberDados() {
    Intent i = getIntent();

    int cod = i.getIntExtra("codigo", 0);
    double qtd = i.getDoubleExtra("quantidade", 0);
    double valor = i.getDoubleExtra("valor", 0);

    tvCod.setText(String.valueOf(cod));
    tvQtd.setText(String.valueOf(qtd));
    tvValor.setText(String.valueOf(valor));
}
```

- O resultado final pode ser visto no próximo slide.

CHAMANDO UMA ACTIVITY PASSANDO PARÂMETRO

RECUPERANDO PARÂMETROS DE OUTRA ACTIVITY

- Até o momento, vimos duas situações específicas de uso da classe *Intent*: a primeira, quando chamamos uma *Activity* de forma simples, somente para apresentar uma nova tela; a segunda quando chamamos uma *Activity* passando algumas informações, apresentadas na nova tela.
- Agora iremos trabalhar com a recuperação de dados de uma *Activity* chamada. Imagine, no exemplo desenvolvido, que o usuário não conheça todos os códigos dos produtos, portanto, ele teria que ter acesso a uma tela de busca ou listagem de produtos (para este exemplo, foi criado o botão **Listar Produtos**).
- Apresentando na tela a listagem com os produtos desejados, o usuário pode clicar no nome do produto e, automaticamente, já retornamos à tela anterior, com o código do produto mostrado no respectivo campo.
- Para codificar essa funcionalidade, criaremos uma nova *Activity* chamada **ListarProdutoActivity.java**.

RECUPERANDO PARÂMETROS DE OUTRA ACTIVITY

- Como se trata de uma lista, será utilizada uma *Activity* ligeiramente diferente, especializada para lista, chamada *ListActivity*. Ao utilizar este tipo de *Activity*, não é necessário a criação de um arquivo XML para a interface gráfica, assim, o arquivo XML criado automaticamente pode ser excluído.
- Em seguida devemos modificar o arquivo **ListarProdutoActivity.java** para ele herdar a funcionalidade de *ListActivity*, bem como apresentar uma lista estática de produtos na tela.
- O código desta classe é apresentado no slide seguinte.

RECUPERANDO PARÂMETROS DE OUTRA ACTIVITY

```java
package pm25s.aula10.cadastro;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class ListarProdutoActivity extends ListActivity {

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

        String produtos[] = {"Coca-Cola", "Guaraná", "Pepsi", "Fanta", "Aquarius", "H2OH", "Água Mineral"};

        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, produtos);

        setListAdapter(adapter);
    }

    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {

    }
}
```

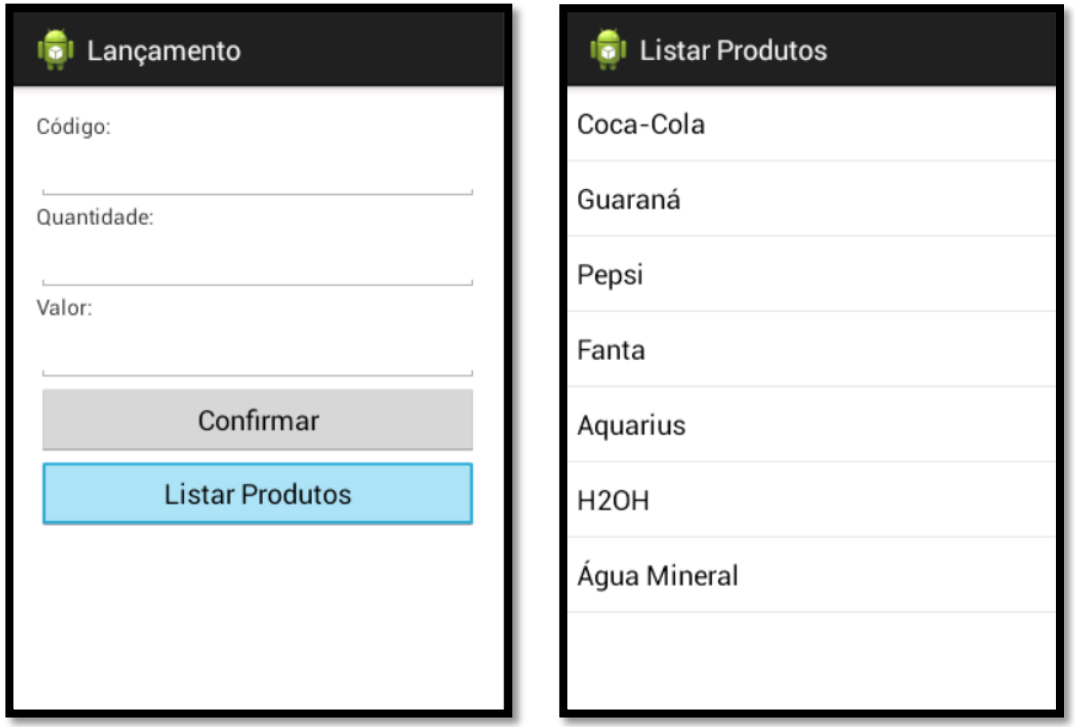
RECUPERANDO PARÂMETROS DE OUTRA ACTIVITY

- O passo seguinte é fazer a chamada dessa *Activity*, bem como codificar o método que receberá os dados enviados como retorno.
- Para isso, na classe **LancamentoActivity.java**, codificaremos o método **btListarProdutoOnClick**() e sobrescreveremos o método **onActivityResult()**, conforme apresentado abaixo:

```
public void btListarProdutosOnClick() {
    Intent i = new Intent(this, ListarProdutoActivity.class);
    startActivityForResult(i, 1);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {

}
```

RECUPERANDO PARÂMETROS DE OUTRA ACTIVITY

- O resultado visual da chamada da tela para **Listar Produtos** até o momento é apresentado na figura abaixo:



RECUPERANDO PARÂMETROS DE OUTRA ACTIVITY

- Na tela anterior o usuário tem a possibilidade de interagir clicando no nome do item da lista.
- Para o exemplo desenvolvido, para o usuário clicar em um dos itens da lista, o mesmo deve retornar para a tela de lançamento, formatando o campo **Código do Produto** com o código do item selecionado. Para um exemplo didático, o código do produto será sequencial e crescente, começando com um para Coca-Cola e terminando com 7 para Água Mineral.
- Desta forma, o código do método **OnListItemClick()** é mostrado em detalhes:

```
@Override
protected void onListItemClick(ListView l, View v, int position, long id) {
    int codProd = position + 1;

    Intent i = getIntent();
    i.putExtra("codProd", codProd);

    // aqui poderiam ser retornados outros campos, caso necessário

    setResult(1, i);
    finish();
}
```

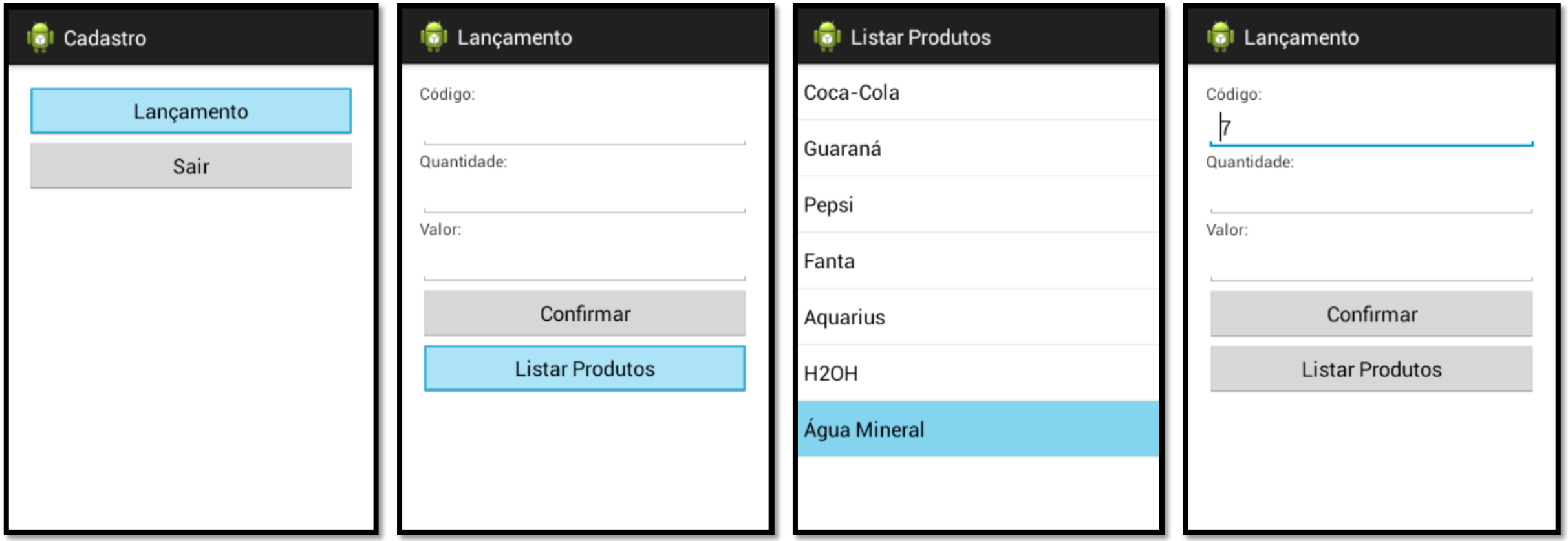
RECUPERANDO PARÂMETROS DE OUTRA ACTIVITY

- Na classe **LancamentoActivity.java**, o método **onActivityResult()** deve ser codificado da seguinte forma:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(data != null) {
        if(requestCode == 1) {
            int codProd = data.getIntExtra("codProd", 0);
            etCodigo.setText(String.valueOf(codProd));
        }
    }
}
```

- Ao executar o aplicativo, o mesmo se comporta conforme a sequencia de imagens contidas no slide seguinte.

RECUPERANDO PARÂMETROS DE OUTRA ACTIVITY



CLASSE *INTENT*: TELAS "NATIVAS" E RECURSOS ANDROID

- A classe *Intent* possui inúmeras funções no Android, podendo ser utilizada para abrir novas janelas (associadas a *Activities*), passar parâmetros, não passar parâmetros ou ainda receber os parâmetros de outras *Activities*.
- Outro recurso muito interessante desta classe é a chamada de telas e recursos nativos do dispositivo, como, por exemplo, chamar a tela responsável por fazer ligações, enviar mensagens ou até mesmo ligar/desligar recursos como, Bluetooth e GPS.
- Nos próximos slides serão mostrados alguns códigos interessantes utilizando a classe *Intent*.

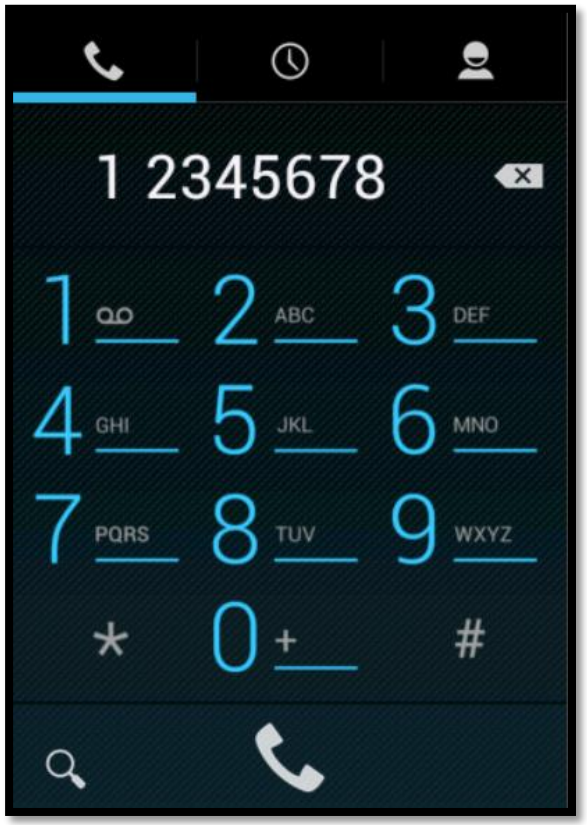
CLASSE *INTENT*: TELAS "NATIVAS" E RECURSOS ANDROID

- O primeiro exemplo é o código para fazer com que uma aplicação Android realize uma ligação.

```xml
<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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="vertical"
    tools:context=".MainActivity" >

    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Digite o Número de Telefone: " />

    <EditText
        android:id="@+id/etTelefone"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:inputType="phone" />

    <Button
        android:id="@+id/btDiscar"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Discar"
        android:onClick="realizarLigacao" />

</LinearLayout>
```

```java
package pm25s.aula10.realizarligacao;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class MainActivity extends Activity {

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

    public void realizarLigacao(View v)
    {
        EditText etTelefone = (EditText) findViewById(R.id.etTelefone);
        String telefone = etTelefone.getText().toString();
        Uri uri = Uri.parse("tel:"+telefone);
        Intent i = new Intent(Intent.ACTION_DIAL, uri);
        startActivity(i);
    }
}
```

CLASSE *INTENT*: TELAS "NATIVAS" E RECURSOS ANDROID



CLASSE *INTENT*: TELAS "NATIVAS" E RECURSOS ANDROID

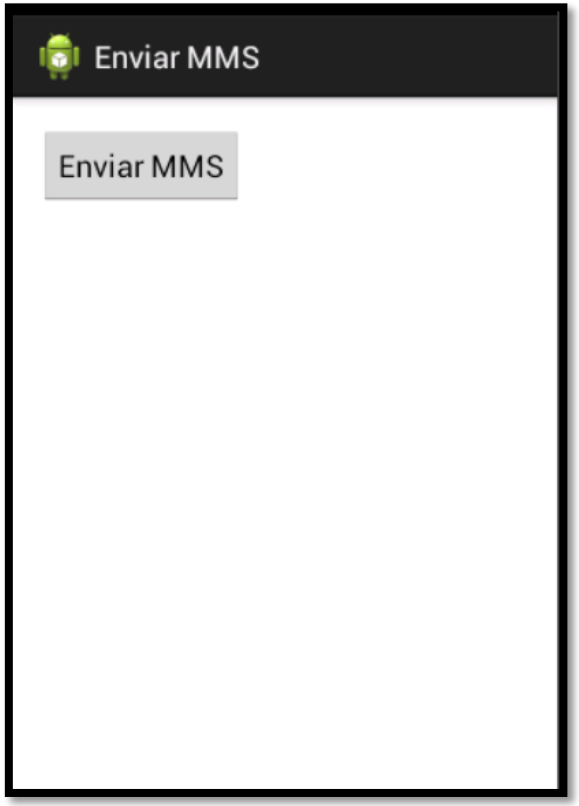
- O segundo exemplo é sobre abrir a tela de envio de mensagens com os campos já formatados. Para exemplificar, iremos formatar os campos para o envio de uma mensagem multimídia (MMS). A interface gráfica do exemplo é apresentada abaixo.

```
1  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
2      xmlns:tools="http://schemas.android.com/tools"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:paddingBottom="@dimen/activity_vertical_margin"
6      android:paddingLeft="@dimen/activity_horizontal_margin"
7      android:paddingRight="@dimen/activity_horizontal_margin"
8      android:paddingTop="@dimen/activity_vertical_margin"
9      tools:context=".MainActivity" >
10
11     <Button
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:text="@string/enviar"
15         android:onClick="enviarMMS" />
16
17 </RelativeLayout>
```

CLASSE *INTENT*: TELAS "NATIVAS" E RECURSOS ANDROID

- O código Java do segundo exemplo é apresentado abaixo:

```java
package pm25s.aula10.enviarmms;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.view.View;

public class MainActivity extends Activity {

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

    public void enviarMMS(View v) {
        Intent i = new Intent(Intent.ACTION_SEND);

        String caminhoCartao = Environment.getExternalStorageDirectory().toString();
        String path = "file://" + caminhoCartao + "/foto.jpg";

        i.putExtra("sms_body", "Clique na imagem acima");
        i.putExtra(Intent.EXTRA_STREAM, Uri.parse(path));
        i.setType("image/jpg");

        startActivity(i);
    }
}
```



CLASSE *INTENT*: TELAS "NATIVAS" E RECURSOS ANDROID

- O último exemplo apresenta uma forma de, via aplicativo Android, iniciar o **Bluetooth** de um dispositivo.

```
<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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:orientation="horizontal"
    tools:context=".MainActivity" >

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

    <ToggleButton
        android:id="@+id/tbBluetooth"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textOn="Ligado"
        android:textOff="Desligado"
        android:onClick="ligarBluetooth" />

</LinearLayout>
```

Dica: Talvez seja necessário incluir a permissão de acesso do aplicativo ao recurso de Bluetooth no arquivo **AndroidManifest.xml**.

CLASSE *INTENT*: TELAS "NATIVAS" E RECURSOS ANDROID

- O código Java do último exemplo é apresentado abaixo:

```
1  package pm25s.aula10.bluetooth;
2
3  import android.app.Activity;
4  import android.bluetooth.BluetoothAdapter;
5  import android.content.Intent;
6  import android.os.Bundle;
7  import android.view.View;
8  import android.widget.ToggleButton;
9
10 public class MainActivity extends Activity {
11
12     @Override
13     protected void onCreate(Bundle savedInstanceState) {
14         super.onCreate(savedInstanceState);
15         setContentView(R.layout.activity_main);
16     }
17
18     public void ligarBluetooth(View v) {
19         ToggleButton tbBluetooth = (ToggleButton) findViewById(R.id.tbBluetooth);
20
21         if(tbBluetooth.isChecked()) {
22             Intent i = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
23             startActivityForResult(i, 2);
24         }
25     }
26 }
```