

PROGRAMAÇÃO PARA DISPOSITIVOS MÓVEIS

Prof. Luiz Carlos Querino Filho
luiz.querino@fatec.sp.gov.br

Fatec Garça – 2018

pdm-04



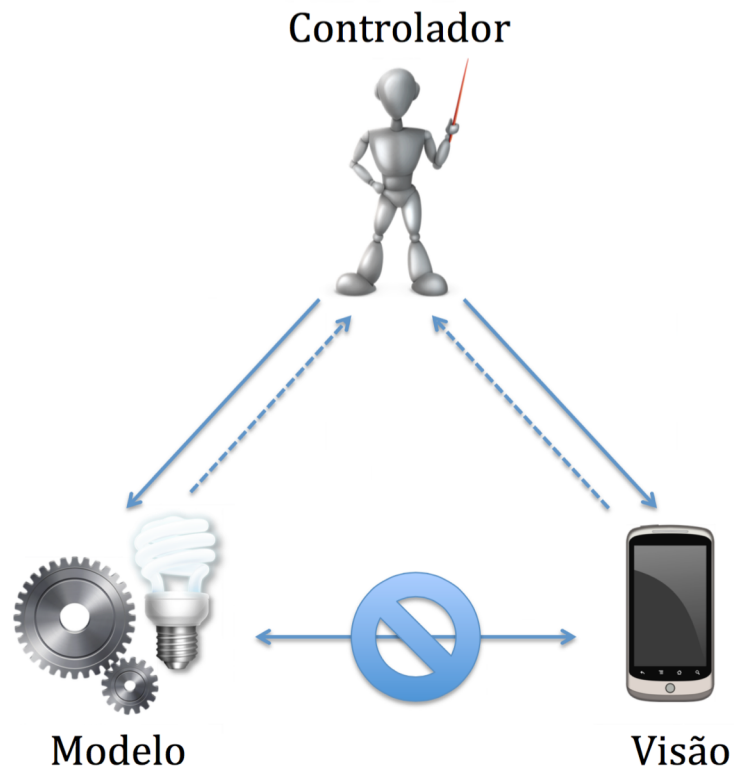
MVC: MODEL / VIEW / CONTROLLER

MVC

- O padrão de projeto MVC estabelece uma divisão de um aplicativo em três “campos”:
 - **Model** (Modelo): os dados e as regras de negócio do aplicativo, ou seja, sua lógica fundamental
 - **View** (Visão): a interface do aplicativo com os usuários
 - **Controller** (Controlador): o elemento que faz a ligação entre o Modelo e a Visão.
- Dividir um aplicativo em camadas facilita sua manutenção e reutilização de seus componentes em outros projetos.
- No Android, podemos dizer que os elementos de um projeto se encaixam desse modo no MVC:
 - **Modelo**: Classes Java básicas para os dados e lógica
 - **Visão**: Arquivos XML
 - **Controlador**: Activity

MVC

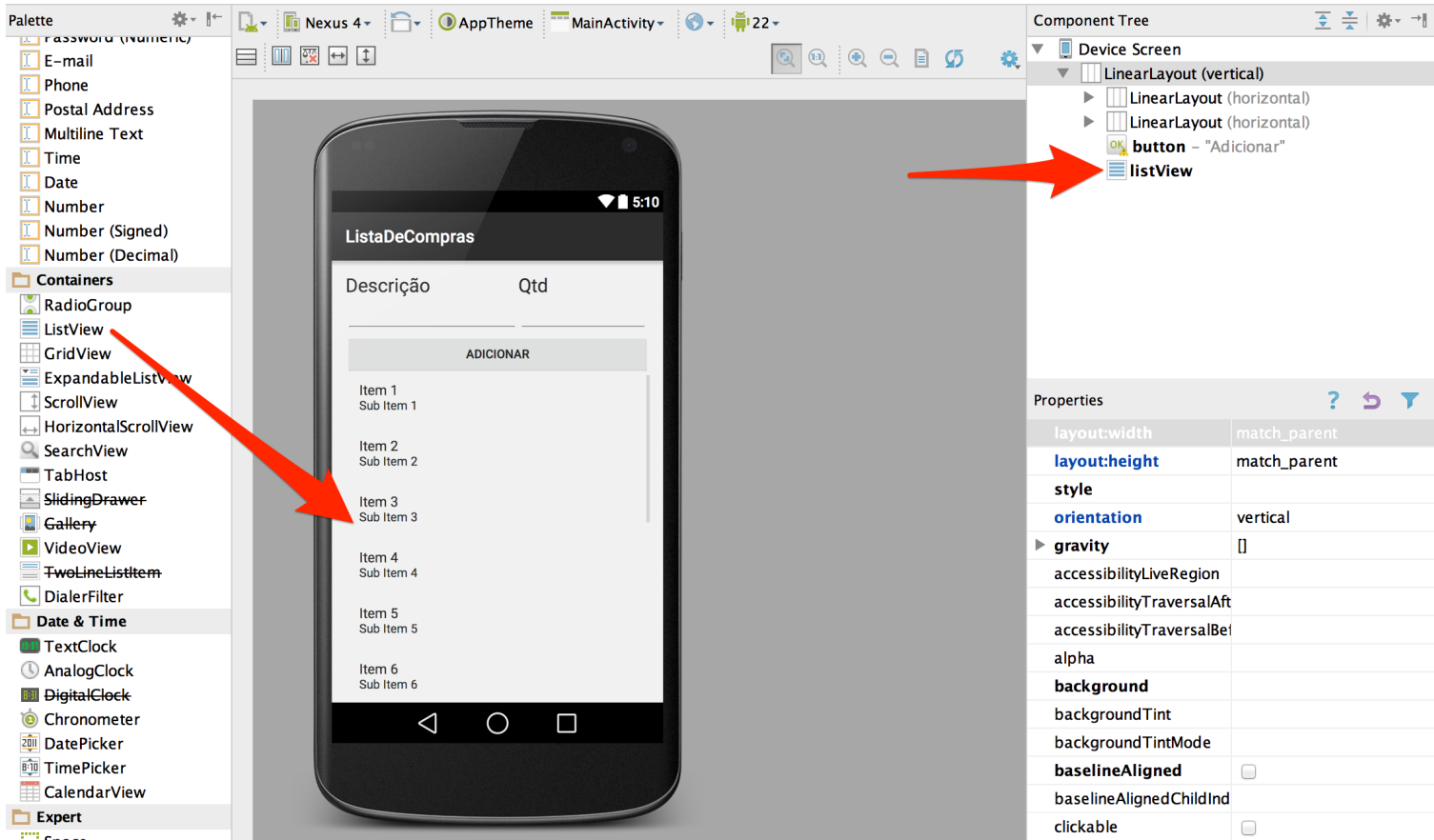
- Para maximizar a reutilização, o ideal é que não haja ligação direta entre a visão e o modelo.



LISTVIEW

ListView

- Com o widget ListView, é possível exibir um conjunto de dados em lista.

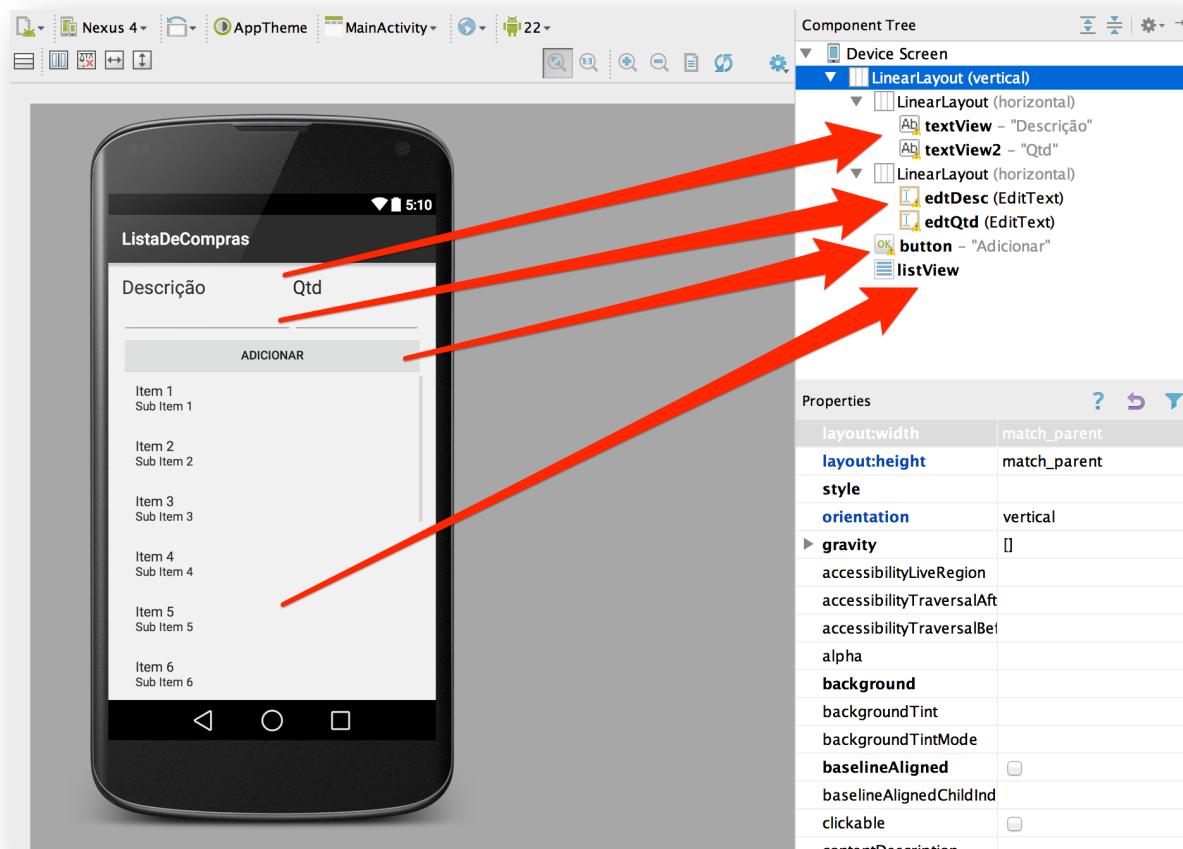


Adapter

- Os dados que serão exibidos em um **ListView** devem ser provenientes de um objeto da classe **Adapter**.
- Se usarmos dados existentes em um vetor ou ArrayList, devemos usar um **ArrayAdapter**.
- Caso os dados venham de um banco de dados, usamos o **SimpleCursorAdapter**.
- Ao instanciar o Adapter, definimos a origem dos dados (vetor, arraylist ou cursor de um banco de dados)
- Também devemos definir um layout básico (em XML) a ser usado nas linhas do ListView.
- Pode ser usado um layout “default”, existente no Android SDK, ou um layout criado dentro do projeto.

Exemplo Prático

- O projeto ListaDeCompras possibilita ao usuário adicionar itens a serem comprados em um ListView. Monte um layout seguindo o modelo abaixo:



- Atribua ao evento `onClick` do botão o método `adicionar`

Classe Modelo: Item

- Crie uma classe Java básica para um item na lista, contendo sua descrição, quantidade, indicador se o mesmo já foi comprado e preço:

```
package com.example.alunos.listadecompras;

import java.text.DecimalFormat;

public class Item implements Comparable {

    private String descricao;
    private int quantidade;
    private boolean comprado;
    private double preco;

    public Item() {
        comprado = false;
    }

    public double getPreco() {
        return preco;
    }

    public void setPreco(double preco) {
        this.preco = preco;
    }

    public boolean isComprado() {
        return comprado;
    }

    public void setComprado(boolean comprado) {
        this.comprado = comprado;
    }
}
```

Classe Modelo: Item

```
public String getDescricao() {  
    return descricao;  
}  
  
public void setDescricao(String descricao) {  
    this.descricao = descricao;  
}  
  
public int getQuantidade() {  
    return quantidade;  
}  
  
public void setQuantidade(int quantidade) {  
    this.quantidade = quantidade;  
}  
  
@Override  
public String toString() {  
    String retorno = descricao + '\n' + "Qtd: " + quantidade;  
    if (comprado)  
        retorno += " " + DecimalFormat.getCurrencyInstance().format(preco);  
    return retorno;  
}  
  
public int compareTo(Object objeto) {  
    Item outroItem = (Item)objeto;  
  
    return this.getQuantidade() - outroItem.getQuantidade();  
}
```

Classe Modelo: **Item**

- Repare que a classe possui os getters e setters para os seus campos privados.
- O método **toString()** retorna uma descrição textual do item, que será mostrada nas linhas do ListView.
- O método **compareTo()** possibilita que façamos a ordenação de itens, usando como critério o valor do campo **descricao**.

Código da Activity (1ª Parte)

```
package com.example.alunos.listadecompras;
```

```
import ...
```

```
public class MainActivity extends ActionBarActivity {
```

```
    ArrayList<Item> lista;  
    ArrayAdapter<Item> adapter;  
    EditText edtDesc, edtQtd;  
    Button button;  
    int posicao = -1;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);
```

```
    lista = new ArrayList<Item>();  
    adapter = new ArrayAdapter<Item>(this, android.R.layout.simple_list_item_checked, lista);
```

```
    edtDesc = (EditText)findViewById(R.id.edtDesc);  
    edtQtd = (EditText)findViewById(R.id.edtQtd);  
    button = (Button)findViewById(R.id.button);
```

```
    ListView listView = (ListView)findViewById(R.id.listView);  
    listView.setAdapter(adapter);  
    listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
```

```
    listView.setOnItemClickListener((parent, view, position, id) → {  
        if (lista.get(position).isComprado()) {  
            lista.remove(position);  
            adapter.notifyDataSetChanged();  
        } else {
```

COMENTÁRIOS – 1ª Parte

- As variáveis globais da classe serão acessíveis em mais de um método.
- A variável **posicao** vai guardar a posição de um elemento tocado na lista para sua posterior alteração.
- No **onCreate()**, criamos um **ArrayList** para guardar os objetos da classe **Item** instanciados, assim como um **ArrayAdapter** quer fará a ligação do **ArrayList** com o **ListView**.
- No **ArrayAdapter** também definimos o layout que será usado no **ListView** (`android.R.layout.simple_list_item_checked`)
- Criamos um **onItemClickListener** para o **ListView**, que removerá o item tocado já comprado.

Código da Activity (2ª Parte)

```
    } else {
        Toast.makeText(MainActivity.this,
            "Você comprou " + lista.get(position).getDescricao(),
            Toast.LENGTH_SHORT)
            .show();
        lista.get(position).setComprado(true);
        AlertDialog.Builder construtor = new AlertDialog.Builder(MainActivity.this);
        final View dialogo = getLayoutInflater().inflate(R.layout.dialogo, null);
        construtor.setView(dialogo);
        construtor.setPositiveButton("OK", (dialog, which) → {
            final EditText edtPreco = (EditText) dialogo.findViewById(R.id.edtPreco);
            double preco = Double.parseDouble(edtPreco.getText().toString());
            lista.get(position).setPreco(preco);
            adapter.notifyDataSetChanged();
        });
        construtor.setNegativeButton("Cancelar", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }
        });
        construtor.create().show();
    }
});
listView.setOnItemLongClickListener((parent, view, position, id) → {
    edtDesc.setText(lista.get(position).getDescricao());
    edtQtd.setText(String.valueOf(lista.get(position).getQuantidade()));
    button.setText("Alterar");
    posicao = position;
    return true;
});
}
```

COMENTÁRIOS – 2ª Parte

- Caso o item não tenha sido comprado, o toque nele realiza a abertura de um **AlertDialog** onde o usuário poderá definir um **preço** para o **Item**.
- Além de definir o preço, o Item é marcado como **comprado**, com essa variável definida para **true**.
- Também definimos um `onItemLongClickListener()` que será usado para iniciar uma alteração.
- Quando ele acontecer, salvamos a variável **position** que indica a posição tocada na variável global **posicao**.

Código da Activity (3ª Parte)

```
public void adicionar(View view){
    Item item = new Item();
    item.setDescricao(edtDesc.getText().toString());
    item.setQuantidade(Integer.parseInt(edtQtd.getText().toString()));

    if (posicao < 0)
        lista.add(item);
    else {
        lista.set(posicao, item);
        posicao = -1;
        button.setText("Adicionar");
    }

    Collections.sort(lista);

    adapter.notifyDataSetChanged();
    edtDesc.setText("");
    edtQtd.setText("");
    edtDesc.requestFocus();
}
```


COMENTÁRIOS – 3ª Parte

- O método adicionar será executado quando o usuário clicar no botão.
- Neste método obtemos os valores de descrição e preço informados nos **EditText**, instanciamos um objeto **Item** e colocamos estes valores neste objeto.
- Em seguida, adicionamos este objeto ao **ArrayList** e atualizamos o adapter, para que o novo elemento apareça no **ListView**.
- O método **Collections.sort()** ordena o **ArrayList**, usando para isso o método **compareTo()** desta classe.

Código da Activity (4ª Parte - Final)

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();

    String mensagem = "";
    int qtdtotal = 0;
    double valortotal = 0;

    switch (id) {
        case R.id.total:
            mensagem = "Itens na lista: " + lista.size();
            break;
        case R.id.totalqtd:
            for (int i = 0; i < lista.size(); i++) {
                qtdtotal = qtdtotal + lista.get(i).getQuantidade();
            }
            mensagem = "Quantidade total: " + qtdtotal;
            break;
        case R.id.totalcompra:
            for (Item itemAtual : lista) {
                if (itemAtual.isComprado())
                    valortotal += itemAtual.getQuantidade() * itemAtual.getPreco();
            }
            mensagem = "Valor total: " + DecimalFormat.getCurrencyInstance().format(valortotal);
    }
    Toast.makeText(this, mensagem, Toast.LENGTH_LONG).show();
    return true;
}
```

COMENTÁRIOS – 4ª Parte

- O método **onOptionsItemSelected()** implementa uma ação para cada um dos itens de menu, definidos no arquivo `res\menu\menu_main.xml`
- Para obter os totais indicados nos menus, percorremos o **ArrayList**, acumulando os valores em variáveis que serão posteriormente exibidas ao usuário.

\res\menu\menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto"
      xmlns:tools="http://schemas.android.com/tools" tools:context=".MainActivity">

    <item android:id="@+id/total" android:title="Total de itens"
          android:orderInCategory="100" app:showAsAction="never" />

    <item android:id="@+id/totalqtd" android:title="Total em quantidade"
          android:orderInCategory="101" app:showAsAction="never" />

    <item android:id="@+id/totalcompra" android:title="Valor da Compra"
          android:orderInCategory="102" app:showAsAction="never" />

</menu>
```

BIBLIOGRAFIA

- QUERINO FILHO, L. C. Desenvolvendo seu Primeiro Aplicativo Android. Novatec Editora. 2013
- DEITEL, H. et al. Android for Programmers: An App-Driven Approach. Pearson Education. 2012.