

FUNDAÇÃO INSTITUTO DE EDUCAÇÃO DE BARUERI

Instituto Técnico de Barueri “Brasília Flores de Azevedo”

Tecnologias Emergentes

Tutorial Venda com Master/Detail Activity

PARTE 1

versão 1.0

Prof. Adriano Domingues

2019

## AULA 21 – Venda com Master/Detail Activity

Iniciaremos a aula com a execução do script de banco de dados abaixo:

```
--SCRIPT DO BANCO DE DADOS AULA 21 VENDA
use master
go
drop DATABASE TEEM_ANDROID_VENDA
go
CREATE DATABASE TEEM_ANDROID_VENDA
GO
USE TEEM_ANDROID_VENDA
GO
CREATE TABLE login(
id INT PRIMARY KEY IDENTITY,
usuario VARCHAR(100) NOT NULL unique,
senha VARCHAR(20) NOT NULL,
email VARCHAR(100) NOT NULL unique,
nivel_acesso VARCHAR(100) NOT NULL)
GO
INSERT INTO login VALUES ('admin', '123456', 'admin@admin.com', 0);
GO
select * from login;
GO
--drop table produto
CREATE TABLE produto(
codigo varchar(50) PRIMARY KEY,
descricao VARCHAR(100) NULL unique,
qtde int null,
valor_unit decimal(15,2) null,
nomeImagem varchar(200) null,
imagem varbinary(MAX) null,
status int NOT NULL) --0 INATIVO / 1 ATIVO
GO
INSERT INTO PRODUTO(codigo, descricao, qtde, valor_unit, status, nomeImagem,
imagem)
SELECT 1, 'Coca-cola', 10, 4.99, 1, 'coca_cola.jpg', *
FROM OPENROWSET(BULK N'C:\DOMTEC\ITB\2019\TEEM\imagens\coca_cola.jpg',
SINGLE_BLOB) Load
GO
INSERT INTO PRODUTO(codigo, descricao, qtde, valor_unit, status, nomeImagem,
imagem)
SELECT 2, 'Bolacha', 30, 2.99, 1, 'bolacha.jpg', *
FROM OPENROWSET(BULK N'C:\DOMTEC\ITB\2019\TEEM\imagens\bolacha.jpg', SINGLE_BLOB)
Load
GO
INSERT INTO PRODUTO(codigo, descricao, qtde, valor_unit, status, nomeImagem,
imagem)
SELECT 3, 'Chocolate', 100, 3.99, 1, 'chocolate.jpg', *
FROM OPENROWSET(BULK N'C:\DOMTEC\ITB\2019\TEEM\imagens\chocolate.jpg',
SINGLE_BLOB) Load
GO
INSERT INTO PRODUTO(codigo, descricao, qtde, valor_unit, status, nomeImagem,
imagem)
SELECT 4, 'Salgado', 20, 1.99, 1, 'salgado.jpg', *
FROM OPENROWSET(BULK N'C:\DOMTEC\ITB\2019\TEEM\imagens\salgado.jpg', SINGLE_BLOB)
Load
GO
INSERT INTO PRODUTO(codigo, descricao, qtde, valor_unit, status, nomeImagem,
imagem)
SELECT 5, 'Suco', 30, 4.99, 1, 'suco.jpg', *
FROM OPENROWSET(BULK N'C:\DOMTEC\ITB\2019\TEEM\imagens\suco.jpg', SINGLE_BLOB)
Load
GO
select * from produto;
GO
```

```
--drop table venda -- 0-CANCELADA 1-VENDIDO 2-PAGO 3-ENTREGUE
Create Table Venda(
codigo int primary key identity,
data_venda date null,
valor_total money null,
desconto numeric(15,2) null,
status int null)
GO
INSERT INTO VENDA (data_venda, valor_total, desconto, status) values ('2019-10-11', 50.0, 0.05, 1);
go
select * from venda;
go
--drop table item_venda
Create table ITEM_VENDA (
codigo int primary key identity,
cod_venda int CONSTRAINT fk_item_venda foreign key (cod_venda) references venda (codigo),
cod_prod varchar(50) CONSTRAINT fk_item_produto foreign key (cod_prod) references produto (codigo),
qtde int null,
valor_unit decimal(15,2) null,
status int NULL)
GO

insert into ITEM_VENDA (cod_venda, cod_prod, qtde, valor_unit, status) values (1, 2, 1, 2.99, 1);
go
insert into ITEM_VENDA (cod_venda, cod_prod, qtde, valor_unit, status) values (1, 1, 2, 4.99, 1);
go

select * from ITEM_VENDA;
go
```

Vale salientar neste ponto que cada grupo do TCC já possui um método de venda, com seu próprio Modelo Relacional, alguns com tabelas de Estoque, outros apenas debitando a quantidade da tabela Produto.

Há a necessidade de adequação do que estamos realizando aqui neste tutorial para a sua realidade, pode-se utilizar a desculpa que o estoque das vendas realizadas pelo Mobile é separado ou diferente da venda realizada no projeto Desktop (presencial), essa é uma boa justificativa, mas não garante a integração do projeto como objetivo final.

Ou talvez garanta se o débito na quantidade de produtos em estoque for o mesmo do projeto Desktop, isto é, se tenho na loja 10 produtos para vender, e vendi 3 no Mobile, então na loja teremos 7 disponíveis. Tem que controlar isso para garantir a integração do projeto.

## Criação do Projeto no Android Studio

Neste ponto vamos criar o projeto no Android Studio, como temos feito durante todo o ano:

Application Name: Aula21\_Venda

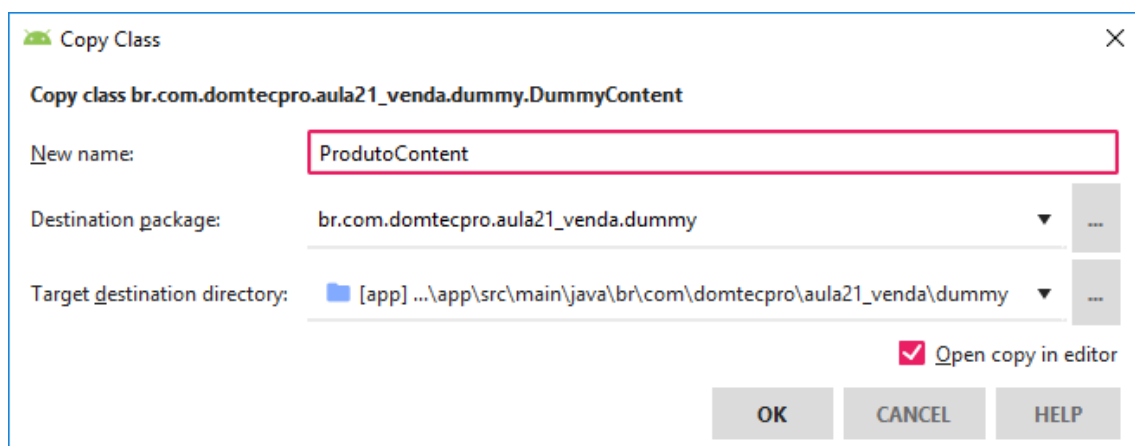
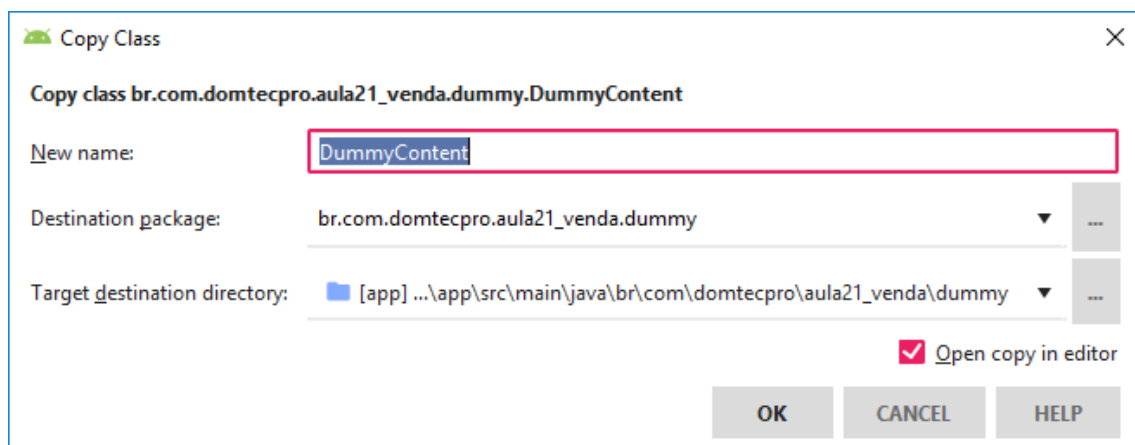
Package Name (Domain Name): br.com.domtecpro.aula21\_venda

Save Location: <NA SUA Z:> SE TIVER ESPAÇO

API 22

Escolha o exemplo **Master/Detail Flow**

Vamos realizar uma cópia do arquivo DummyContent e chamar esta cópia de ProdutoContent: é o famoso Ctrl+C e Ctrl+V no arquivo DummyContent e quando a janela abaixo aparecer nomeie o arquivo como ProdutoContent:



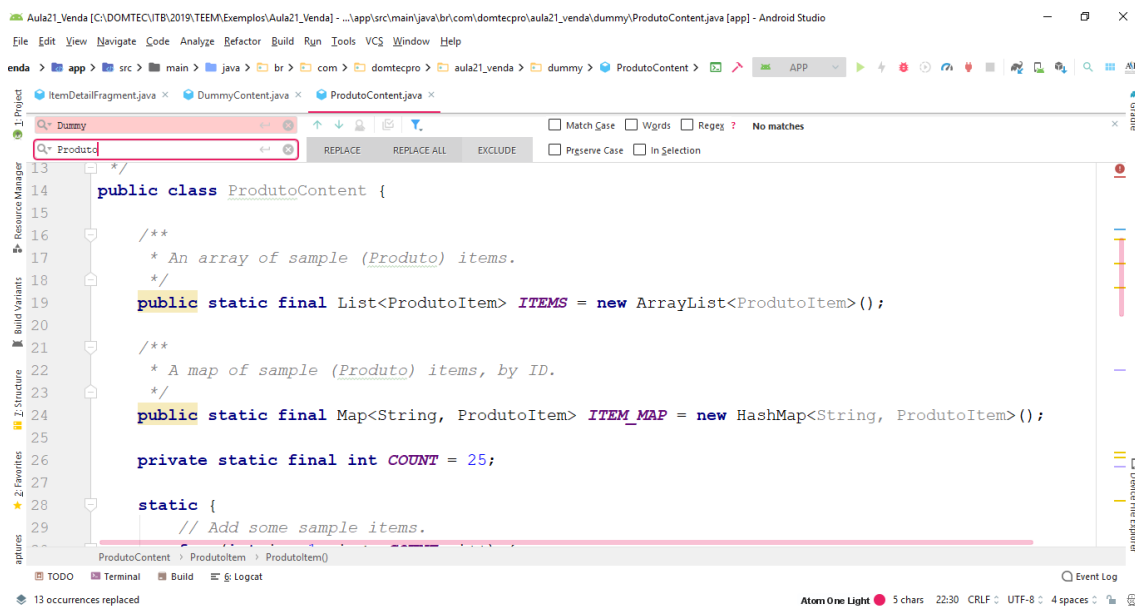
E clique no **OK**.

Elimine todas as referências ao DummyContent no arquivo de ProdutoContent, assim como DummyItem será ProdutoItem:



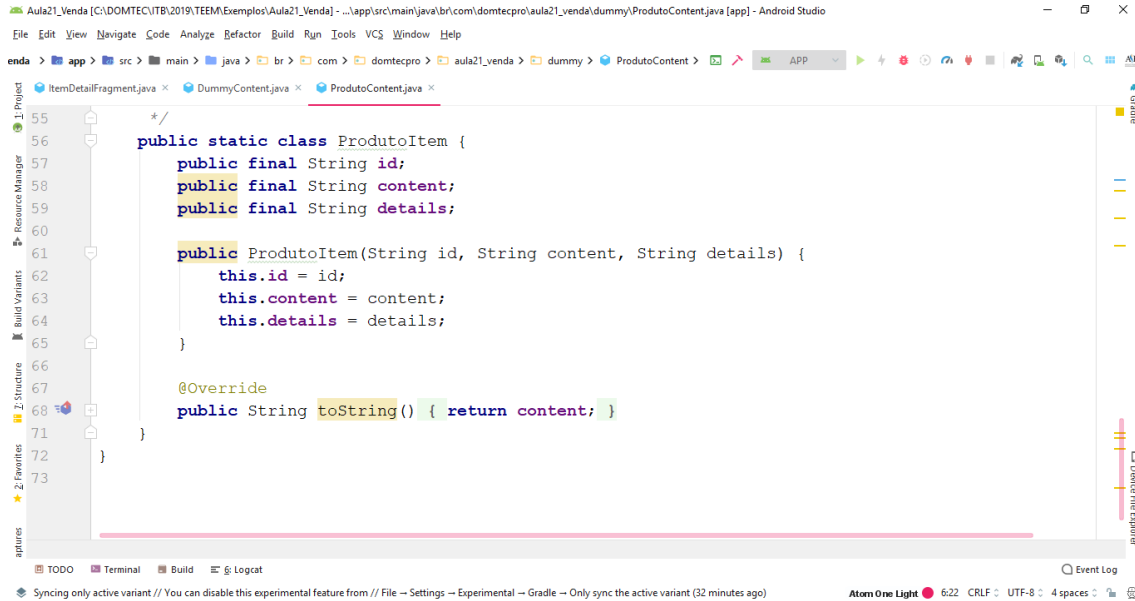
```
10  * Android template wizards.
11  *
12  * <p>
13  * TODO: Replace all uses of this class before publishing your app.
14  */
15  public class ProdutoContent {
16
17      /**
18       * An array of sample (dummy) items.
19       */
20      public static final List<DummyItem> ITEMS = new ArrayList<>();
21
22      /**
23       * A map of sample (dummy) items, by ID.
24       */
25      public static final Map<String, DummyItem> ITEM_MAP = new HashMap<>();
26
27      private static final int COUNT = 25;
28
29      static {
```

Utilize o Ctrl+R se for o caso, não altere o pacote, ele ainda será chamado de **package** br.com.domtepro.aula21\_venda.dummy;



```
13  */
14  public class ProdutoContent {
15
16      /**
17       * An array of sample (Produto) items.
18       */
19      public static final List<ProdutoItem> ITEMS = new ArrayList<ProdutoItem>();
20
21      /**
22       * A map of sample (Produto) items, by ID.
23       */
24      public static final Map<String, ProdutoItem> ITEM_MAP = new HashMap<String, ProdutoItem>();
25
26      private static final int COUNT = 25;
27
28      static {
29          // Add some sample items.
```

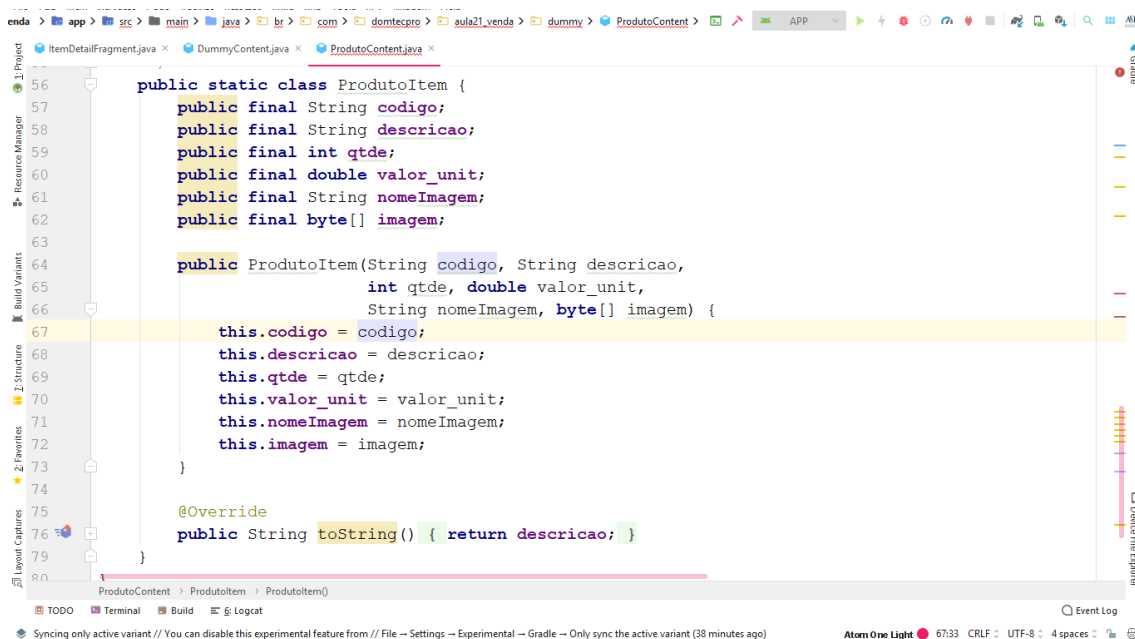
Vamos para a linha 56 para alterarmos os atributos e o método construtor do ProdutoItem, que é uma classe de negócio embutida no ProdutoContent, e deve corresponder à tabela produto do nosso banco de dados:



The screenshot shows the Android Studio interface with the file explorer on the left displaying the project structure: app > src > main > java > br > com > domtepro > aula21\_venda > dummy > ProdutoContent > ProdutoContent.java. The main editor shows the code for the ProdutoItem class, with line 56 highlighted. The code defines three final String attributes (id, content, details) and a constructor that initializes them. The toString() method is overridden to return the content attribute.

```
55  */
56  public static class ProdutoItem {
57      public final String id;
58      public final String content;
59      public final String details;
60
61      public ProdutoItem(String id, String content, String details) {
62          this.id = id;
63          this.content = content;
64          this.details = details;
65      }
66
67      @Override
68      public String toString() { return content; }
69  }
70
71
72
73
```

Depois da alteração ficará assim:



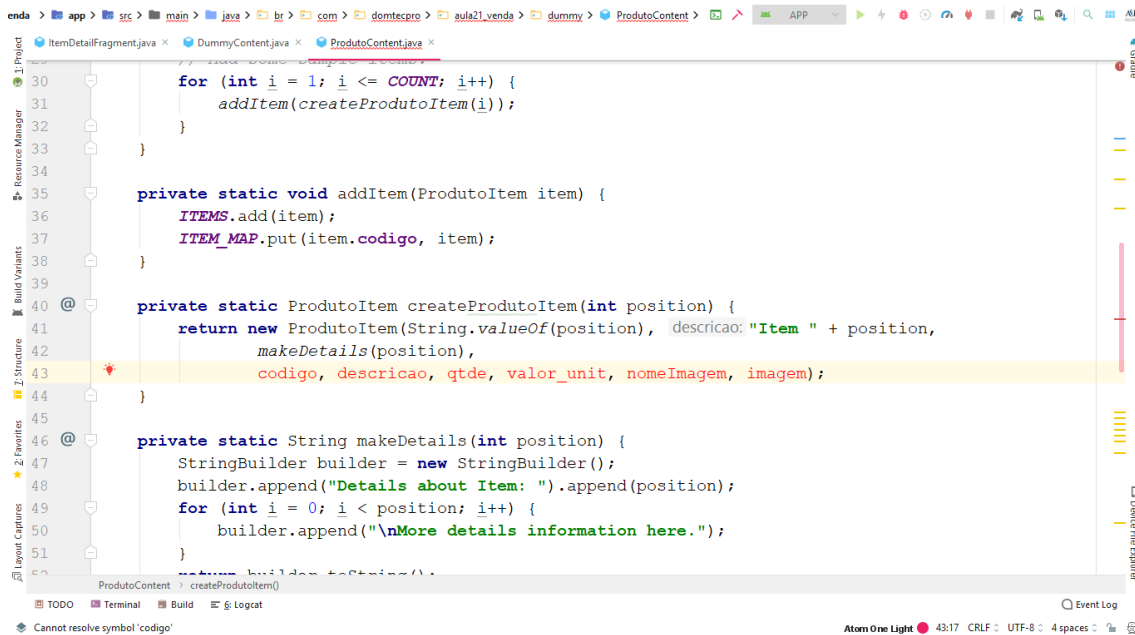
The screenshot shows the same Android Studio interface, but the code for the ProdutoItem class has been updated. Line 56 is still highlighted. The attributes are now codigo (String), descricao (String), qtde (int), valor\_unit (double), nomeImagem (String), and imagem (byte[]). The constructor takes these six parameters and initializes the instance variables. The toString() method is overridden to return the descricao attribute.

```
56  public static class ProdutoItem {
57      public final String codigo;
58      public final String descricao;
59      public final int qtde;
60      public final double valor_unit;
61      public final String nomeImagem;
62      public final byte[] imagem;
63
64      public ProdutoItem(String codigo, String descricao,
65                          int qtde, double valor_unit,
66                          String nomeImagem, byte[] imagem) {
67          this.codigo = codigo;
68          this.descricao = descricao;
69          this.qtde = qtde;
70          this.valor_unit = valor_unit;
71          this.nomeImagem = nomeImagem;
72          this.imagem = imagem;
73      }
74
75      @Override
76      public String toString() { return descricao; }
77  }
78
79
80
```

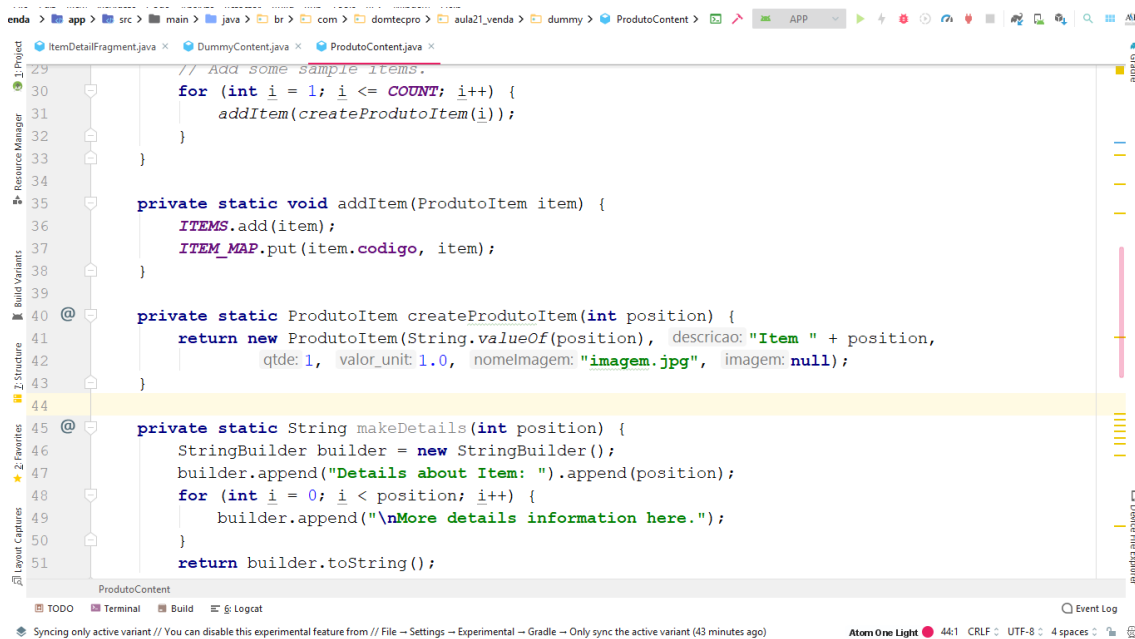
Na linha 37 corrija o erro provocado pela troca do id por código:

```
private static void addItem(ProdutoItem item) {
    ITEMS.add(item);
    ITEM_MAP.put(item.codigo, item);
}
```

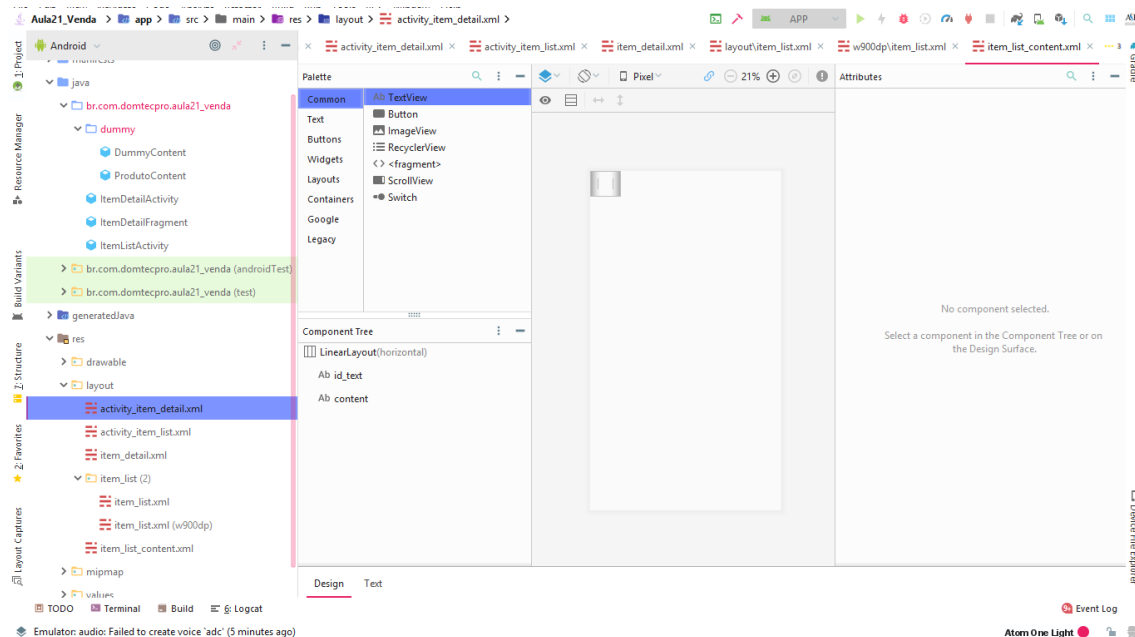
Na linha 41, adicione os valores de carregamento padrão da listagem estática:



Ficará assim:

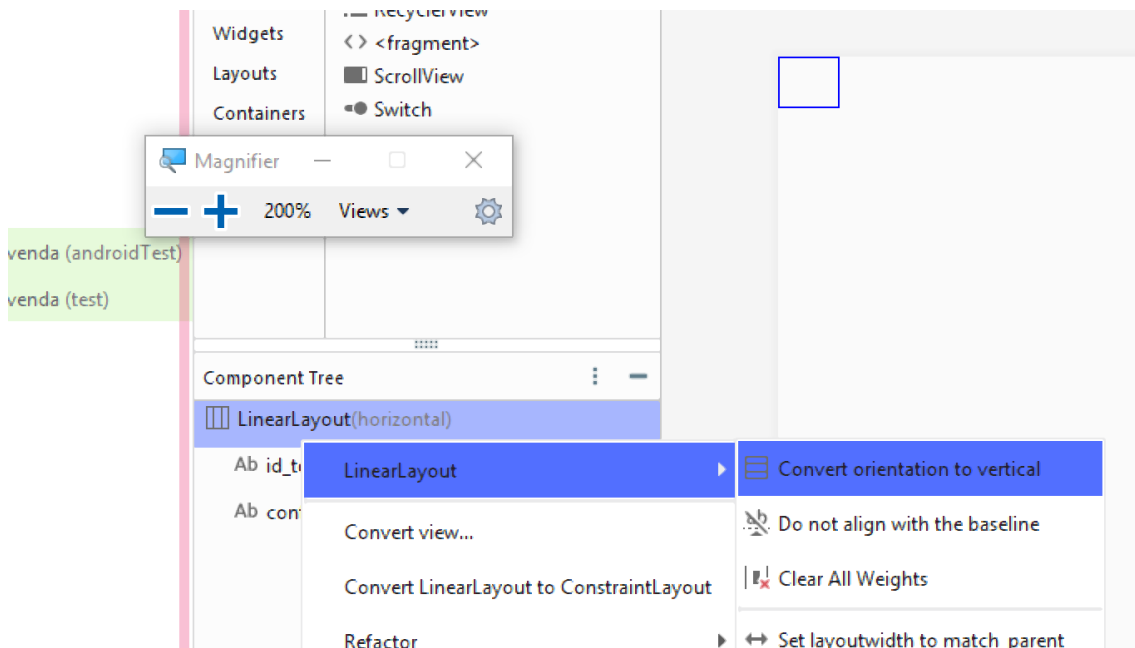


Observe abaixo que este exemplo Master/Detail Flow cria vários arquivos de layout, pois também possui telas para rodar em Tablet.



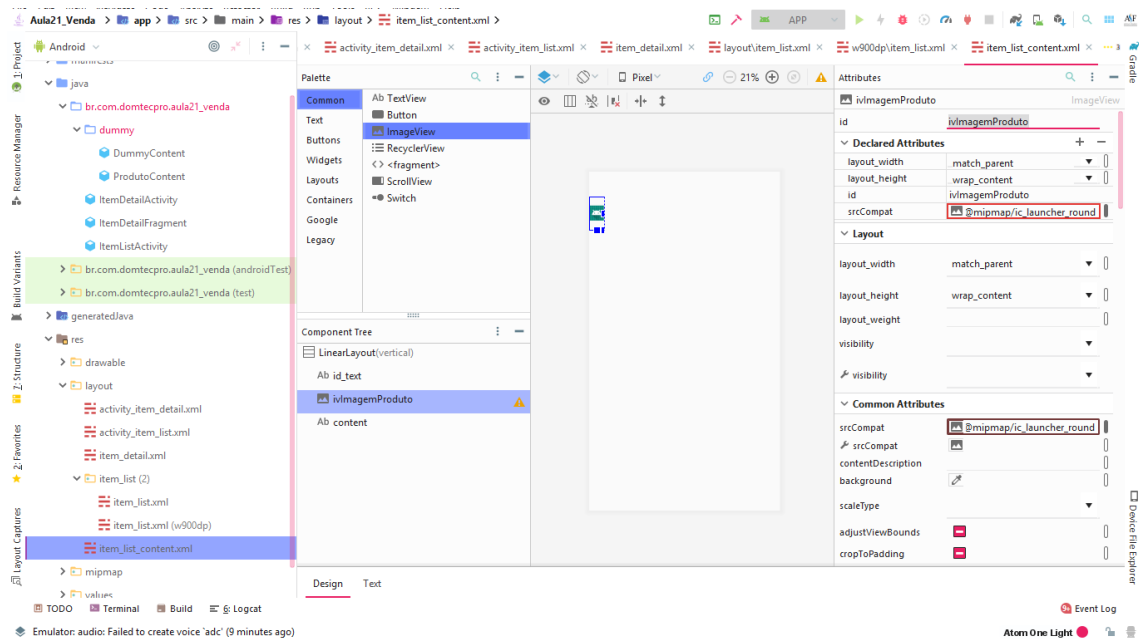
Vamos alterar o layout item\_list\_content.xml para carregar o nome do produto e a imagem dele.

Altere a orientação do LinearLayout de horizontal para vertical:

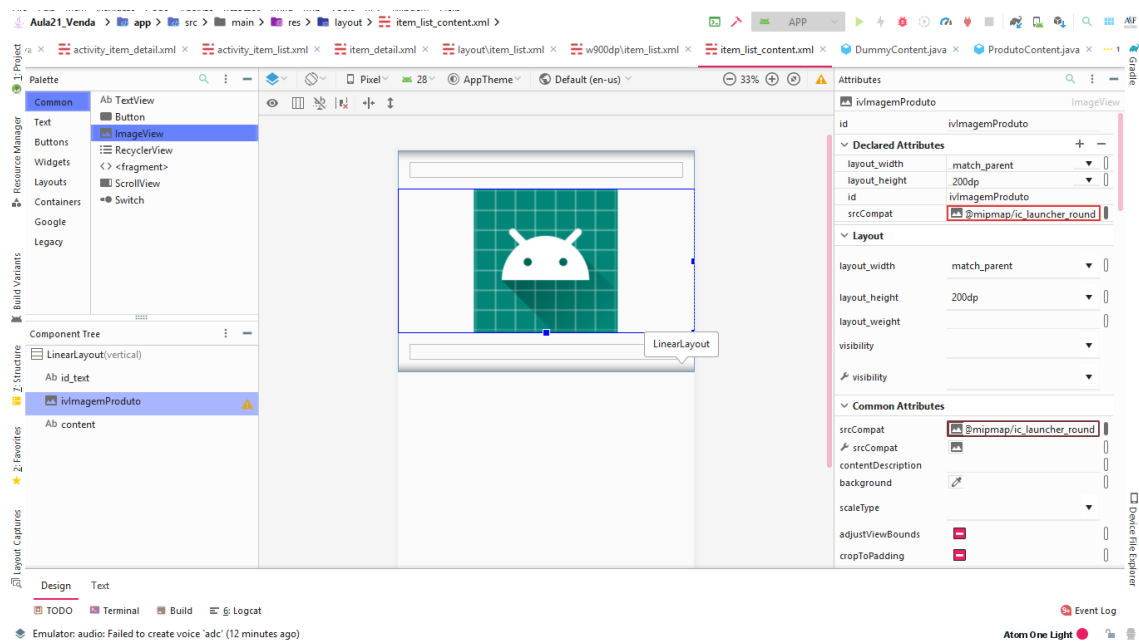


Adicione um objeto ImageView entre o código e a descrição, e o nomeie como **ivImagemProduto**:





Vou efetuar os ajustes de margem e padding para melhorar a aparência o item do produto na lista:



Segue o código XML para comparação ou adequação, como preferir:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:id="@+id/id_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/text_margin"
        android:textAppearance="?attr/textAppearanceListItem" />

    <ImageView
        android:id="@+id/ivImagemProduto"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        app:srcCompat="@mipmap/ic_launcher_round" />

    <TextView
        android:id="@+id/content"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/text_margin"
        android:textAppearance="?attr/textAppearanceListItem" />
</LinearLayout>
```

Observe que delimitei o tamanho da altura da imagem para 200dp, pois nossas imagens tem tamanhos diversos.

Vamos agora alterar as classes Java, começando pela ItemDetailFragment:

Na linha 12 altere o import:

```
import br.com.domtecproua21_venda.dummy.ProdutoContent;
```

Na linha 30 altere a declaração do mItem:

```
private ProdutoContent.ProdutoItem mItem;
```

Na linha 47 altere a construção do mItem:

```
mItem =
    ProdutoContent.ITEM_MAP.get(getArguments().getString(ARG_ITEM_ID));
```

Na linha 52 altere o código, mItem.content para mItem.descricao:

```
appBarLayout.setTitle(mItem.descricao);
```

E na linha 64 de mItem.details também para mItem.descricao:

```
((TextView)
rootView.findViewById(R.id.item_detail)).setText(mItem.descricao);
```

Vamos agora alterar o código da classe ItemListActivity:

Altere o import da linha 17:

```
import br.com.domtecproua21_venda.dummy.ProdutoContent;
```

Altere a linha 69 de DummyContent para ProdutoContent:

```
recyclerView.setAdapter(new SimpleItemRecyclerViewAdapter(this,
ProdutoContent.ITEMS, mTwoPane));
```

Na linha 76 altere DummyContent.DummyItem para ProdutoContent.ProdutoItem:

```
private final List<ProdutoContent.ProdutoItem> mValues;
```

Na linha 81 novamente:

```
ProdutoContent.ProdutoItem item = (ProdutoContent.ProdutoItem)
view.getTag();
```

Na linha 84 de id para código:

```
arguments.putString(ItemDetailFragment.ARG_ITEM_ID, item.codigo);
```

Na linha 93 de id para código:

```
intent.putExtra(ItemDetailFragment.ARG_ITEM_ID, item.codigo);
```

Na linha 100 altere novamente de Dummy para Produto:

```
SimpleItemRecyclerViewAdapter(ItemListActivity parent,
                                List<ProdutoContent.ProdutoItem> items,
                                boolean twoPane) {
```

Na linha 117 de id para código:

```
holder.mIdView.setText(mValues.get(position).codigo);
```

Na linha 118 de content para descrição:

```
holder.mContentView.setText(mValues.get(position).descricao);
```

Teste o aplicativo neste ponto para verificar o carregamento da listagem estática dos produtos, agora vamos adicionar a classe de Conexão:

```
package br.com.domtecpro.aula21_venda;

import android.os.StrictMode;
import android.support.design.widget.Snackbar;
import android.view.View;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import br.com.domtecpro.aula21_venda.dummy.ProdutoContent;

public class Conexao {
    public static Connection conexaoBD() {
        Connection conexao = null;
        try {
            StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy
                .Builder().permitAll().build();
            StrictMode.setThreadPolicy(policy);

            Class.forName("net.sourceforge.jtds.jdbc.Driver").newInstance();
            //NÃO ESQUECER DE DESCOBRIR O IP DA MÁQUINA ONDE ESTÁ O SQL
            SERVER
            //
            String IP = "172.19.0.73";
            String IP = "192.168.0.20";
            conexao =
            DriverManager.getConnection("jdbc:jtds:sqlserver://" + IP + ";" +
            "databaseName=TEEM_ANDROID_VENDA;user=sa;password=123456;");
        } catch (Exception e) {
            e.getMessage().toString();
        }
        return conexao;
    }
}
```

```
//Método de pesquisa de produtos na base SQL Server
public static ArrayList<ProdutoContent.ProdutoItem>
pesquisarProdutos() {
    ArrayList<ProdutoContent.ProdutoItem> lista = new
ArrayList<ProdutoContent.ProdutoItem>();
    try{
        //Pesquisa no banco de dados
        PreparedStatement pst =
Conexao.conexaoBD().prepareStatement("select * from produto");
        //Resultado da pesquisa realizada
        ResultSet res = pst.executeQuery();
        //Se existir linhas no objeto res, a lista será carregada com
produtos
        while(res.next()) {
            ProdutoContent.ProdutoItem produto = new
ProdutoContent.ProdutoItem(
                res.getString(1),
                res.getString(2),
                res.getInt(3),
                res.getDouble(4),
                res.getString(5),
                res.getBytes(6)
            );
            lista.add(produto);
        }
    } catch (SQLException e) {
        e.getMessage().toString();
    }
    return lista;
}

//Método de pesquisa de produtos na base SQL Server
public static ArrayList<ProdutoContent.ProdutoItem>
pesquisarProdutosFiltro(String texto) {
    ArrayList<ProdutoContent.ProdutoItem> lista = new
ArrayList<ProdutoContent.ProdutoItem>();
    try{
        //Pesquisa no banco de dados
        PreparedStatement pst = Conexao.conexaoBD().prepareStatement(
            "select * from produto " +
            "where codigo like '%" + texto.toString() + "%' " +
            "or descricao like '%" + texto.toString() + "%'");
        //Resultado da pesquisa realizada
        ResultSet res = pst.executeQuery();
        //Se existir linhas no objeto res, a lista será carregada com
produtos
        while(res.next()) {
            ProdutoContent.ProdutoItem produto = new
ProdutoContent.ProdutoItem(
                res.getString(1),
                res.getString(2),
                res.getInt(3),
                res.getDouble(4),
                res.getString(5),
                res.getBytes(6)
            );
            lista.add(produto);
        }
    } catch (SQLException e) {
        e.getMessage().toString();
    }
    return lista;
}
```

```
public static int pesquisarUsuario(String usuario, String senha,
                                   View view){
    try{
        PreparedStatement pst =
            Conexao.conexaoBD().prepareStatement(
                "select * from login WHERE usuario = ?
and senha = ?");
        pst.setString(1, usuario);
        pst.setString(2, senha);
        ResultSet res = pst.executeQuery();
        if(res.next()){
            Snackbar.make(view,
                "LOGIN ENCONTRADO!!!",
Snackbar.LENGTH_LONG).show();
            return res.getInt(5);
        }else{
            Snackbar.make(view,
                "LOGIN NÃO ENCONTRADO!!!",
Snackbar.LENGTH_LONG).show();
        }
    } catch (SQLException e) {
        Snackbar.make(view,
            e.getMessage().toString(),
Snackbar.LENGTH_LONG).show();
    }
    return 9;
}

public static void inserirUsuario(View view, String usuario,
String
    senha, String email){
    try{
        PreparedStatement pst =
            Conexao.conexaoBD().prepareStatement("insert
into login values (?, ?, ?, ?)");
        pst.setString(1, usuario);
        pst.setString(2, senha);
        pst.setString(3, email);
        int nivel = 1;
        pst.setInt(4, nivel);
        pst.executeUpdate();
        Snackbar.make(view,
            "LOGIN INSERIDO!!!",
Snackbar.LENGTH_LONG).show();
    } catch (SQLException e) {
        Snackbar.make(view,
            e.getMessage().toString(),
Snackbar.LENGTH_LONG).show();
    }
}
```

```
public static void alterarStatusProduto(View view,
                                         String codigo,
                                         int situacaoAtual){

    try{
        PreparedStatement pst =
            Conexao.conexaoBD().prepareStatement("'" +
            "update produto " +
            "set status = ? where codigo = ?");

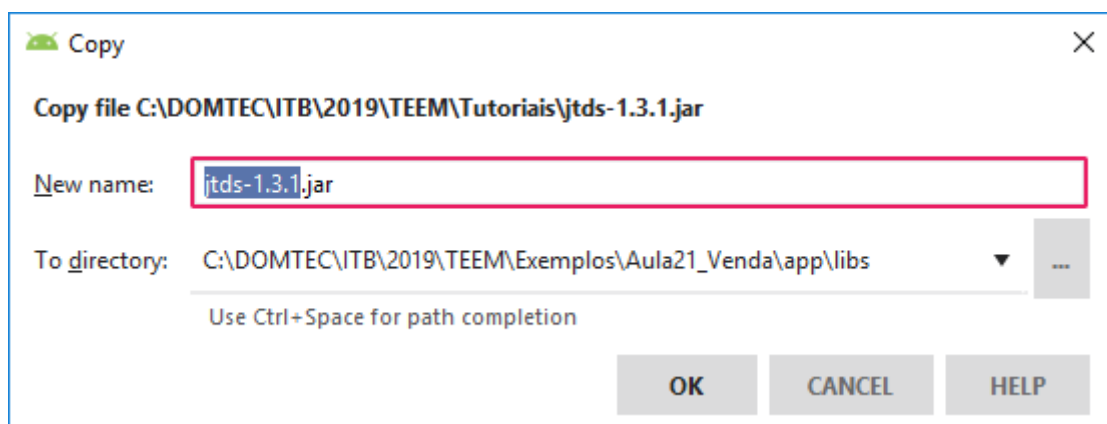
        if(situacaoAtual == 1) {
            pst.setInt(1, 1);
        }else{
            pst.setInt(1, 0);
        }
        pst.setString(2, codigo);

        pst.executeUpdate();
        Snackbar.make(view,
            "STATUS DO PRODUTO ALTERADO!!!",
            Snackbar.LENGTH_LONG).show();
    } catch (SQLException e) {
        Snackbar.make(view,
            e.getMessage().toString(),
            Snackbar.LENGTH_LONG).show();
    }
}
```

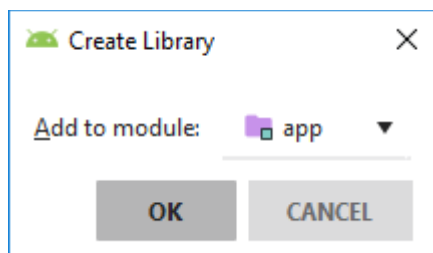
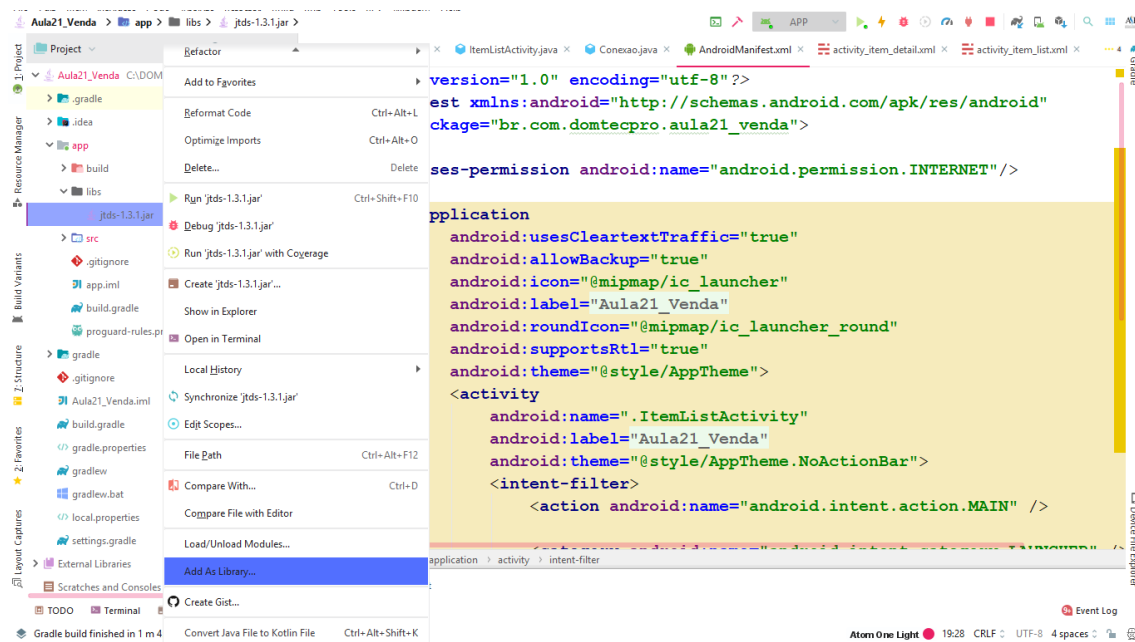
Agora abra o AndroidManifest.xml e insira o pedido de permissão de INTERNET e o parâmetro de liberação do buffer de transmissão de dados:

```
<uses-permission android:name="android.permission.INTERNET"/>
<application
    android:usesCleartextTraffic="true"
```

Não esqueça de copiar o arquivo do driver JTDS na pasta libs:



E adicionar na biblioteca:



E agora com todos os requisitos para conexão completos, vamos alterar o ProdutoContent para solicitar a listagem de produtos do banco de dados:

Voltaremos nossa atenção para linha 28 da classe ProdutoContent:

A estrutura *static* irá sofrer as alterações abaixo:

```
static {
    ArrayList<ProdutoContent.ProdutoItem> lista;
    lista = Conexao.pesquisarProdutos();

    if(lista==null){
        // Add some sample items.
        for (int i = 1; i <= COUNT; i++) {
            addItem(createProdutoItem(i));
        }
    }else{
        for(ProdutoContent.ProdutoItem produto: lista){
            addItem(produto);
        }
    }
}
```



Teste o aplicativo neste ponto, observe que a imagem de cada produto ainda não foi carregada adequadamente, vamos alterar a classe `ItemListActivity` para carregar as imagens:

Primeiro vamos alterar a classe `ViewHolder` presente na classe `ItemListActivity` na linha 129:

```
class ViewHolder extends RecyclerView.ViewHolder {
    final TextView mIdView;
    final TextView mContentView;
    final ImageView mImageView;

    ViewHolder(View view) {
        super(view);
        mIdView = (TextView) view.findViewById(R.id.id_text);
        mContentView = (TextView) view.findViewById(R.id.content);
        mImageView = (ImageView)
view.findViewById(R.id.ivImagemProduto);
    }
}
```

E na linha 124 adicione o método de carregamento da imagem:

```
public Bitmap carregaImagem(byte[] bytes) {
    try { // Tenta converter o blob em uma imagem
        BitmapFactory.Options options = new BitmapFactory.Options();
        options.inPreferredConfig = Bitmap.Config.ARGB_8888;
        // 'bytes' é o Blob que vem do banco, está vindo preenchido
        corretamente
        Bitmap bmp = BitmapFactory.decodeByteArray(bytes, 0,
bytes.length, options);
        return bmp;
    } catch (Exception e) {
        return null;
    } // Fim do controle try/catch
}
```

Então com os devidos imports realizados na linha 122 será inserido o código de carregamento da imagem que vem do banco de dados para a listagem:

```
holder.mImageView.setImageBitmap(carregaImagem(mValues.get(position).i
magem));
```

Teste o aplicativo neste ponto novamente e verifique o carregamento das imagens do banco de dados:



Ao clicar em um item qualquer uma tela de detalhe é aberta, e é nesta tela que vamos agora trabalhar, para verificar os dados totais do produto, como quantidade e preço, e acionar o botão de compra deste produto, que irá enviar este item para o “carrinho” ou “sacola”.

## Referência Bibliográfica

- [1] <http://developer.android.com>. Acessado em 28/04/2019.
- [2] <https://pedroalvaiojunior.wordpress.com/2012/07/20/dica-armazenando-arquivos-de-imagem-no-sql-server-2008-e-r2-atraves-do-comando-openrowset-em-conjunto-com-a-opcao-bulk/>. Acessado em 11/10/2019.
- [3] <https://www.guj.com.br/t/de-blob-para-bitmap-e-vice-versa/246586/8>. Acessado em 11/10/2019.