

FUNDAÇÃO INSTITUTO DE EDUCAÇÃO DE BARUERI

Instituto Técnico de Barueri “Brasília Flores de Azevedo”

Tecnologias Emergentes

Tutorial Chamando outra Lista e Detalhe da Lista

versão 1.0

Prof. Adriano Domingues

2019

AULA 15 – Chamando outra Lista e Detalhe da Lista

Para realizar esta aula você deve terminar o Tutorial Aula 12 Lista com RecyclerView e 13 Login Banco Externo e possuir o script abaixo:

```
--SCRIPT DO BANCO DE DADOS AULA06
use master
go
drop DATABASE TEEM_ANDROID
go
CREATE DATABASE TEEM_ANDROID
GO
USE TEEM_ANDROID
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

CREATE TABLE produto(
codigo varchar(50) PRIMARY KEY,
descricao VARCHAR(100) NOT NULL unique,
qtde int not null,
valor_unit decimal(15,2) not null,
status int NOT NULL)  --0 INATIVO / 1 ATIVO
GO

INSERT INTO PRODUTO VALUES (1, 'Coca-cola', 10, 4.99, 1);
GO
INSERT INTO PRODUTO VALUES (2, 'Bolacha', 30, 2.99, 1);
GO
INSERT INTO PRODUTO VALUES (3, 'Chocolate', 100, 3.99, 1);
GO
INSERT INTO PRODUTO VALUES (4, 'Salgado', 20, 1.99, 1);
GO
INSERT INTO PRODUTO VALUES (5, 'Suco', 30, 4.99, 1);
GO

select * from produto;
GO
```

```

--drop table item_produto
Create table ITEM_PRODUTO (
codigo int primary key identity,
descricao varchar(200) not null,
qtde int not null,
valor_unit decimal(15,2) not null,
cod_prod varchar(50),
status int NOT NULL,
CONSTRAINT FK_ITEM_PROD foreign key (cod_prod) references PRODUTO (codigo));
GO

insert into ITEM_PRODUTO values ('carne', 100, 2.9, 4, 1);
go
insert into ITEM_PRODUTO values ('frango', 50, 1.9, 4, 1);
go
insert into ITEM_PRODUTO values ('laranja', 50, 4.9, 5, 1);
go
insert into ITEM_PRODUTO values ('limão', 40, 4.9, 5, 1);
go
insert into ITEM_PRODUTO values ('melancia', 30, 5.9, 5, 1);
go

select * from ITEM_PRODUTO;
go

```

Uma nova tabela de itens dos produtos foi criada para a primeira lista chamar a segunda, quando clicarmos em algum dos produtos.

Antes vamos dar vida ao objeto Switch, chamado status_produto, toda vez que ele for clicado a situação do produto passa de ativo para inativo, ou seja, disponível para indisponível, ou vice-versa.

Para isso acontecer vamos criar mais um método na classe Conexao, que será o responsável por alterar a situação do produto na tabela correspondente:

```

public static void alterarStatusProduto(Context context,
                                         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();
        Toast.makeText(context,
            "STATUS DO PRODUTO ALTERADO!!!",
            Snackbar.LENGTH_LONG).show();
    } catch (SQLException e) {
        Toast.makeText(context,
            e.getMessage().toString(),
            Snackbar.LENGTH_LONG).show();
    }
}

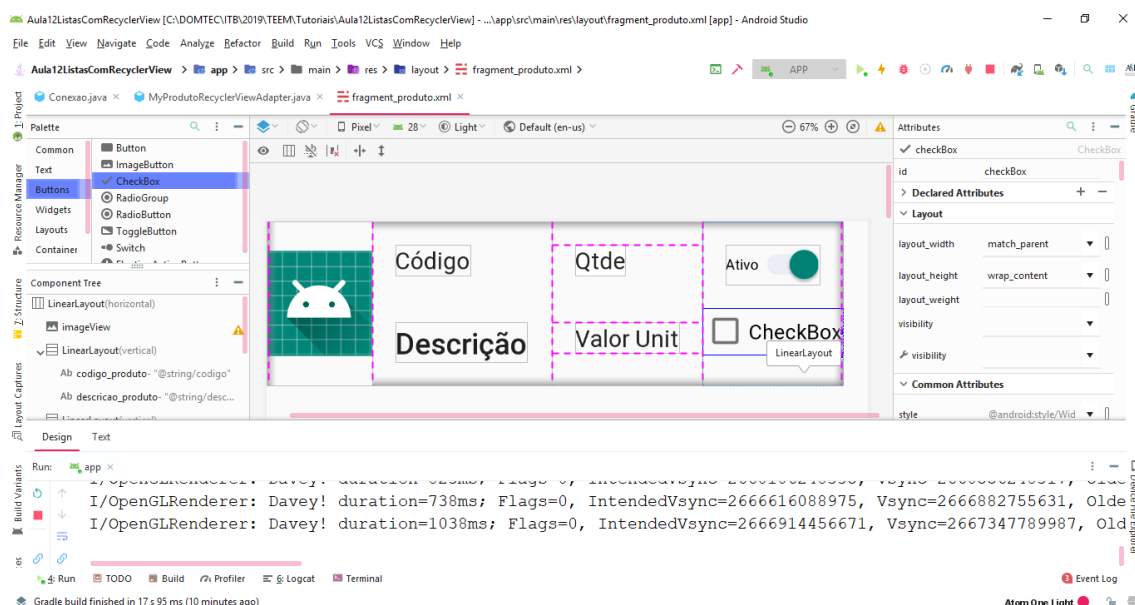
```

Vamos alterar o código da classe `MyProdutoRecyclerViewAdapter` para ficar na espera do clique no objeto `Switch` chamado `status_produto`, informando o status atual do produto para a troca de situação. Insira o código abaixo a partir da linha 68:

```
holder.mStatusView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int status = 99;
        if(!holder.mStatusView.isChecked()) {
            status = 0;
            holder.mStatusView.setTextOff("OFF");
            holder.mStatusView.setText("Indisponível");
        }
        else {
            status = 1;
            holder.mStatusView.setTextOn("ON");
            holder.mStatusView.setText("Disponível");
        }
        Conexao.alterarStatusProduto(v.getContext(),
            holder.mCodigoView.getText().toString(),
            status);
    }
});
```

Teste o aplicativo neste ponto, e verifique no banco de dados se o status foi alterado corretamente.

Vamos adicionar um objeto que muitos alunos querem para seu TCC, um `CheckBox`, altere o id dele para **chkStatusProduto**:



Ele vai fazer a mesma função do Switch, habilitar e desabilitar o produto:

```
holder.mStatusView2.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v) {
        int status = 99;
        if(!holder.mStatusView2.isChecked()) {
            status = 0;
            //CheckBox
            holder.mStatusView2.setText("Indisponível");
        }
        else {
            status = 1;
            //CheckBox
            holder.mStatusView2.setText("Disponível");
        }

        Conexao.alterarStatusProduto(v.getRootView(),
            holder.mCodigoView.getText().toString(),
            status);
    }
});
```

Realizaremos uma alteração nos dois métodos de clique, para os dois objetos serem alterados ao mesmo tempo, isto é, quando clicarmos no objeto CheckBox o objeto Switch também será alterado.

```
//Quando o Switch é clicado
holder.mStatusView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int status = 99;
        if(!holder.mStatusView.isChecked()) {
            status = 0;
            //Switch
            holder.mStatusView.setTextOff("OFF");
            holder.mStatusView.setText("Indisponível");
            //CheckBox
            holder.mStatusView2.setChecked(false);
            holder.mStatusView2.setText("Indisponível");
        }
        else {
            status = 1;
            //Switch
            holder.mStatusView.setTextOn("ON");
            holder.mStatusView.setText("Disponível");
            //CheckBox
            holder.mStatusView2.setChecked(true);
            holder.mStatusView2.setText("Disponível");
        }

        Conexao.alterarStatusProduto(v.getRootView(),
            holder.mCodigoView.getText().toString(),
            status);
    }
});
```

```

//Quando o CheckBox é clicado
holder.mStatusView2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        int status = 99;
        if(!holder.mStatusView2.isChecked()) {
            status = 0;
            //Switch
            holder.mStatusView.setChecked(false);
            holder.mStatusView.setTextOff("OFF");
            holder.mStatusView.setText("Indisponível");
            //CheckBox
            holder.mStatusView2.setText("Indisponível");
        }
        else {
            status = 1;
            //Switch
            holder.mStatusView.setChecked(true);
            holder.mStatusView.setTextOn("ON");
            holder.mStatusView.setText("Disponível");
            //CheckBox
            holder.mStatusView2.setText("Disponível");
        }

        Conexao.alterarStatusProduto(v.getRootView(),
            holder.mCodigoView.getText().toString(),
            status);
    }
});

```

Não podemos esquecer de adicionar ao método onBindViewHolder() as linhas abaixo, para configurar o status dos objetos Switch e CheckBox no carregamento da tela.

```

// O status capturado é apresentado, de acordo com o valor
// Se o valor for 1 o produto mostra que está disponível
// Senão o produto será apresentado como indisponível
if(mValues.get(position).status == 1){
    holder.mStatusView.setChecked(true);
    holder.mStatusView.setTextOn("ON");
    holder.mStatusView.setText("Disponível");
} else{
    holder.mStatusView.setChecked(false);
    holder.mStatusView.setTextOff("OFF");
    holder.mStatusView.setText("Indisponível");
}

```

```
// Implementação do CheckBox adicionado
if(mValues.get(position).status == 1){
    holder.mStatusView2.setChecked(true);
    holder.mStatusView2.setText("Disponível");
}else{
    holder.mStatusView2.setChecked(false);
    holder.mStatusView2.setText("Indisponível");
}
```

Realize os testes no emulador e verifique se o produto é disponível ou indisponível quando clicamos nos objetos Switch ou CheckBox programados.

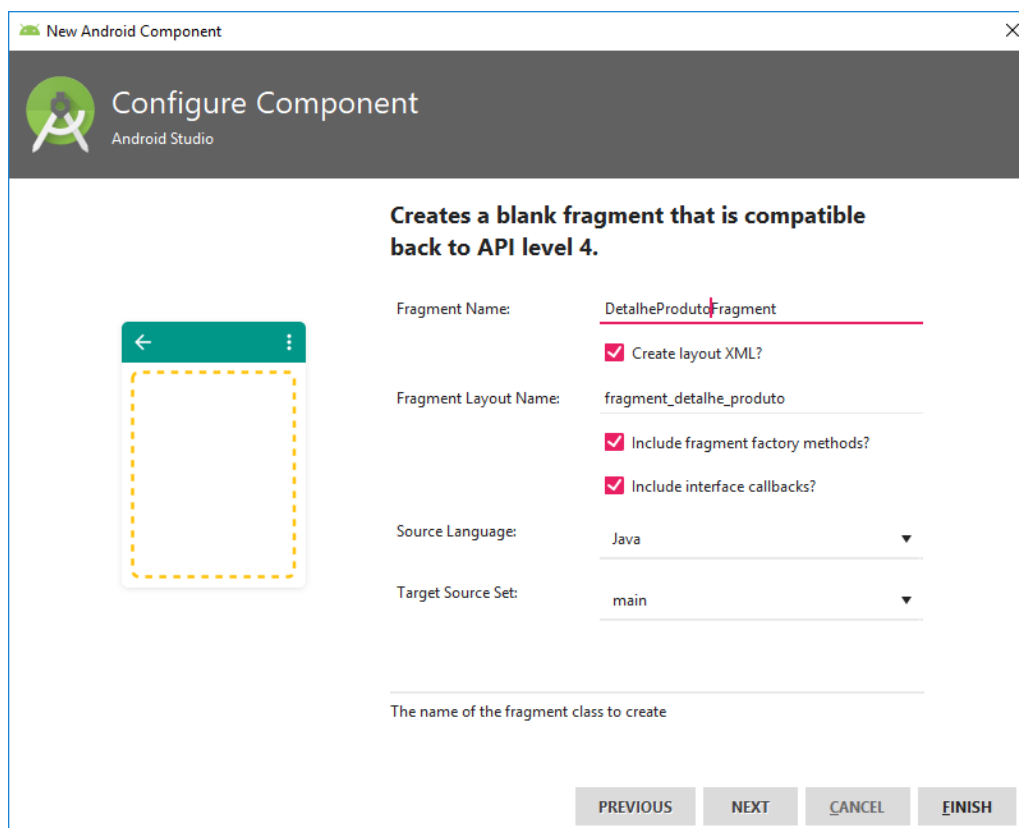
Agora, vamos programar a chamada do detalhamento de um produto clicado, isto é, quando clicarmos em algum produto da lista apresentada um novo fragmento com os dados daquele produto será mostrado.

Para isso acontecer, vamos novamente inserir um fragmento em branco.

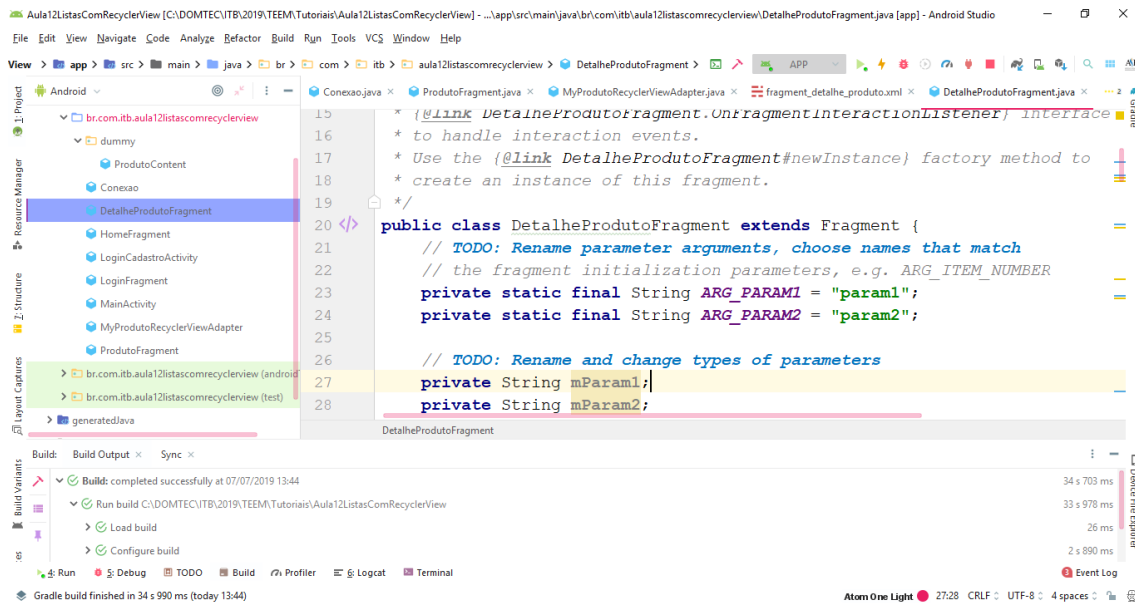
Clique com o botão direito em Android > app ... escolha o item New > Fragment > Fragment (Blank)

Trocamos o nome do fragmento para DetalheProdutoFragment e consequentemente o layout será chamado de fragment_detalhe_produto.xml

Clique no FINISH após as alterações:



Após o carregamento do fragmento a classe `DetalheProdutoFragment` será aberta:



Este é o código XML da tela montada na imagem acima:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".DetalheProdutoFragment">

    <!-- TODO: Update blank fragment layout -->
    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <TextView
            android:id="@+id/lblDetalheCodigo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_marginTop="25dp"
            android:text="@string/codigo"
            app:layout_constraintStart_toStartOf="parent"

app:layout_constraintTop_toBottomOf="@+id/ivDetalheImagem" />

        <ImageView
            android:id="@+id/ivDetalheImagem"
            android:layout_width="0dp"
            android:layout_height="300dp"
            android:layout_marginStart="8dp"
            android:layout_marginEnd="8dp"
            android:src="@mipmap/ic_launcher"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/txtDetalheCodigo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginBottom="10dp"
            android:text="TextView"

app:layout_constraintBottom_toTopOf="@+id/txtDetalheDescricao"

app:layout_constraintStart_toStartOf="@+id/txtDetalheDescricao" />
```

```
<TextView
    android:id="@+id/lblDetalheDescricao"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="8dp"
    android:text="@string/descricao"
    app:layout_constraintEnd_toStartOf="@+id/txtDetalheDescricao"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/lblDetalheCodigo" />

<TextView
    android:id="@+id/txtDetalheDescricao"
    android:layout_width="0dp"
    android:layout_height="50dp"
    android:layout_marginStart="8dp"
    android:layout_marginEnd="7dp"
    android:text="TextView"

    app:layout_constraintBaseline_toBaselineOf="@+id/lblDetalheDescricao"
    "
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.207"
    app:layout_constraintStart_toEndOf="@+id/lblDetalheDescricao" />

<TextView
    android:id="@+id/lblDetalheQtde"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:layout_marginBottom="10dp"
    android:text="@string/quantidade"
    app:layout_constraintBottom_toTopOf="@+id/lblDetalheValor"
    app:layout_constraintStart_toStartOf="@+id/txtDetalheDescricao"
    app:layout_constraintTop_toBottomOf="@+id/txtDetalheDescricao"
/>

<TextView
    android:id="@+id/txtDetalheQtde"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="12dp"
    android:layout_marginTop="10dp"
    android:text="TextView"
    app:layout_constraintStart_toEndOf="@+id/lblDetalheValor"
    app:layout_constraintTop_toBottomOf="@+id/txtDetalheDescricao"
/>

<TextView
    android:id="@+id/lblDetalheValor"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/valor_unitario"

    app:layout_constraintBaseline_toBaselineOf="@+id/txtDetalheValor"
    app:layout_constraintEnd_toEndOf="@+id/lblDetalheQtde"
    app:layout_constraintStart_toEndOf="@+id/lblDetalheQtde" />
```

```
<TextView
    android:id="@+id/txtDetalheValor"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dp"
    android:text="TextView"

    app:layout_constraintStart_toStartOf="@+id/txtDetalheQtde"

    app:layout_constraintTop_toBottomOf="@+id/txtDetalheQtde" />

    <TextView
        android:id="@+id/lblDetalheSituacao"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="8dp"
        android:text="@string/status"
        app:layout_constraintEnd_toEndOf="@+id/lblDetalheValor"

    app:layout_constraintTop_toBottomOf="@+id/lblDetalheValor" />

    <TextView
        android:id="@+id/txtDetalheSituacao"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:text="TextView"

    app:layout_constraintStart_toStartOf="@+id/txtDetalheValor"

    app:layout_constraintTop_toBottomOf="@+id/txtDetalheValor" />

    <Button
        android:id="@+id/btnDetalheComprar"
        android:layout_width="150dp"
        android:layout_height="100dp"
        android:layout_marginTop="10dp"
        android:layout_marginEnd="16dp"
        android:text="COMPRAR"
        app:layout_constraintEnd_toEndOf="parent"

    app:layout_constraintTop_toBottomOf="@+id/txtDetalheSituacao" />

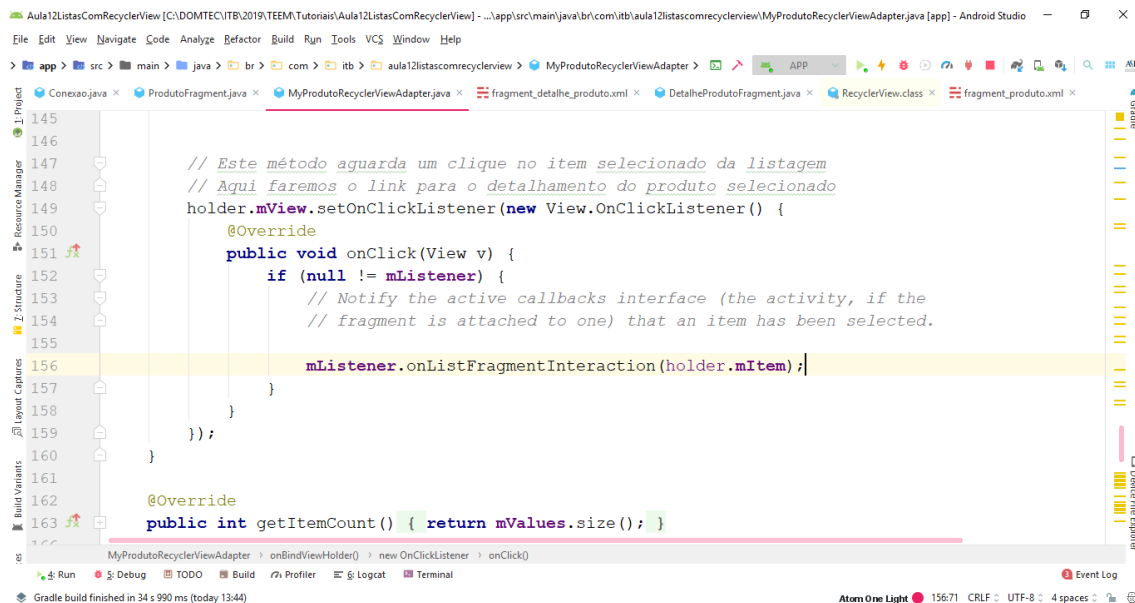
</android.support.constraint.ConstraintLayout>

</FrameLayout>
```

Montada a tela vamos programar o envio dos dados do produto selecionado na lista:

Vamos abrir a classe MyProdutoRecyclerViewAdapter.java para programar o clique no produto selecionado na lista:

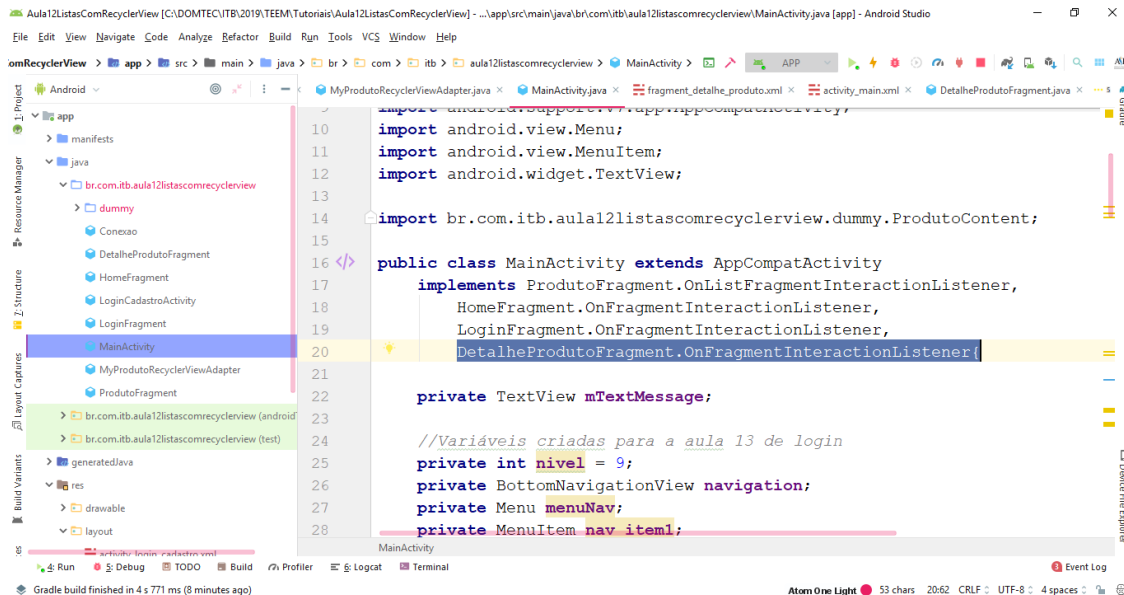
Mais especificamente dentro do método onBindViewHolder(), temos um método setOnClickListener() do objeto mView, onde é esperado o clique em um produto da lista, e retornado os dados deste item clicado para a atividade MainActivity.java:



Vamos abrir a classe MainActivity.java:

Insira o código abaixo na assinatura da classe:

```
public class MainActivity extends AppCompatActivity
    implements ProdutoFragment.OnListFragmentInteractionListener,
        HomeFragment.OnFragmentInteractionListener,
        LoginFragment.OnFragmentInteractionListener,
        DetalheProdutoFragment.OnFragmentInteractionListener{
```



O código abaixo definirá a chamada do fragmento DetalheProdutoFragment, passando os parâmetros do item selecionado para a tela de detalhamento, e deve ser incluído no método onListFragmentInteraction():

```
@Override
public void onListFragmentInteraction(ProdutoContent.ProdutoItem item)
{
    String codigo = item.codigo;
    String descricao = item.descricao;
    int qtde = item.qtde;
    double valor = item.valor_unit;
    int situacao = item.status;

    FragmentTransaction ft =
    getSupportFragmentManager().beginTransaction();
    ft.replace(R.id.fragment_container,
    DetalheProdutoFragment.newInstance(
        codigo, descricao, qtde, valor, situacao
    ))
    .addToBackStack(null)
    .commit();
}
```

Abra o fragmento DetalheProdutoFragment.java e substitua com o código abaixo:

```
package br.com.itb.aula12listascomrecyclerview;

import android.content.Context;
import android.net.Uri;
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.Button;
import android.widget.TextView;

/**
 * A simple {@link Fragment} subclass.
 * Activities that contain this fragment must implement the
 * {@link DetalheProdutoFragment.OnFragmentInteractionListener}
 * interface
 * to handle interaction events.
 * Use the {@link DetalheProdutoFragment#newInstance} factory method
 * to
 * create an instance of this fragment.
 */
public class DetalheProdutoFragment extends Fragment {
    // TODO: Rename parameter arguments, choose names that match
    // the fragment initialization parameters, e.g. ARG_ITEM_NUMBER
    private static final String ARG_PARAM1 = "param1";
    private static final String ARG_PARAM2 = "param2";
    private static final String ARG_PARAM3 = "param3";
    private static final String ARG_PARAM4 = "param4";
    private static final String ARG_PARAM5 = "param5";

    // TODO: Rename and change types of parameters
    private String mParam1;
    private String mParam2;
    private int mParam3;
    private double mParam4;
    private int mParam5;

    private TextView txtCodigo;
    private TextView txtDescricao;
    private TextView txtQtde;
    private TextView txtValor;
    private TextView txtSituacao;
    private Button btnComprar;

    private OnFragmentInteractionListener mListener;

    public DetalheProdutoFragment() {
        // Required empty public constructor
    }
```

```

/**
 * Use this factory method to create a new instance of
 * this fragment using the provided parameters.
 *
 * @param param1 Parameter 1.
 * @param param2 Parameter 2.
 * @return A new instance of fragment DetalheProdutoFragment.
 */
// TODO: Rename and change types and number of parameters
public static DetalheProdutoFragment newInstance(String param1, String
param2,
                                                    int param3, double
param4, int param5) {
    DetalheProdutoFragment fragment = new DetalheProdutoFragment();
    Bundle args = new Bundle();
    args.putString(ARG_PARAM1, param1);
    args.putString(ARG_PARAM2, param2);
    args.putInt(ARG_PARAM3, param3);
    args.putDouble(ARG_PARAM4, param4);
    args.putInt(ARG_PARAM5, param5);
    fragment.setArguments(args);
    return fragment;
}

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (getArguments() != null) {
        mParam1 = getArguments().getString(ARG_PARAM1);
        mParam2 = getArguments().getString(ARG_PARAM2);
        mParam3 = getArguments().getInt(ARG_PARAM3);
        mParam4 = getArguments().getDouble(ARG_PARAM4);
        mParam5 = getArguments().getInt(ARG_PARAM5);
    }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_detalhe_produto,
container, false);

    // Inicializando componentes da tela de detalhamento do produto
    txtCodigo = view.findViewById(R.id.txtDetalheCodigo);
    txtDescricao = view.findViewById(R.id.txtDetalheDescricao);
    txtQtde = view.findViewById(R.id.txtDetalheQtde);
    txtValor = view.findViewById(R.id.txtDetalheValor);
    txtSituacao = view.findViewById(R.id.txtDetalheSituacao);
    btnComprar = view.findViewById(R.id.btnDetalheComprar);

    // Captura dos valores recebidos na passagem de parâmetro do
    construtor
    txtCodigo.setText(mParam1);
    txtDescricao.setText(mParam2);
    txtQtde.setText(String.valueOf(mParam3));
    txtValor.setText(String.valueOf(mParam4));
    if (mParam5==1) {
        txtSituacao.setText("Disponível");
        btnComprar.setEnabled(true);
    }
    else {
        txtSituacao.setText("Indisponível");
        btnComprar.setEnabled(false);
    }
}

```

```

        return view;
    }

    // TODO: Rename method, update argument and hook method into UI
    event
    public void onPressed(Uri uri) {
        if (mListener != null) {
            mListener.onFragmentInteraction(uri);
        }
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        if (context instanceof OnFragmentInteractionListener) {
            mListener = (OnFragmentInteractionListener) context;
        } else {
            throw new RuntimeException(context.toString()
                + " must implement OnFragmentInteractionListener");
        }
    }

    @Override
    public void onDetach() {
        super.onDetach();
        mListener = null;
    }

    /**
     * This interface must be implemented by activities that contain
     * this
     * fragment to allow an interaction in this fragment to be
     * communicated
     * to the activity and potentially other fragments contained in
     * that
     * activity.
     * <p>
     * See the Android Training lesson <a href=
    "http://developer.android.com/training/basics/fragments/communicating.h
    tml"
     * >Communicating with Other Fragments</a> for more information.
     */
    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        void onFragmentInteraction(Uri uri);
    }
}

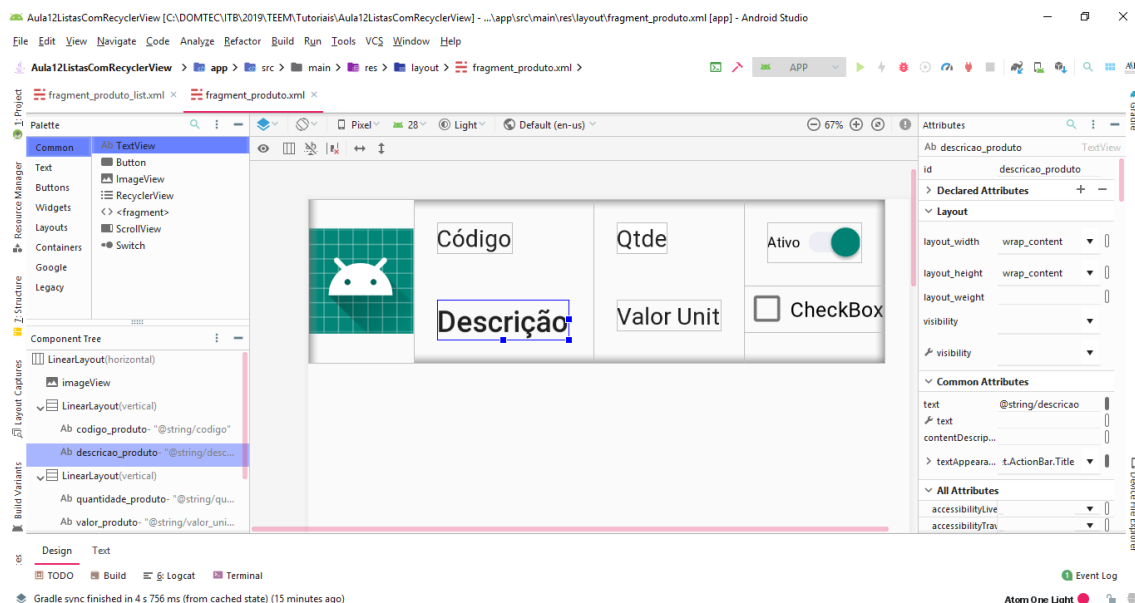
```


As dimensões do `FrameLayout` chamado `fragment_container` contido no layout `activity_main.xml` foram alteradas, segue o código XML da alteração:

```
<FrameLayout
    android:id="@+id/fragment_container"
    android:layout_width="389dp"
    android:layout_height="606dp"
    android:layout_marginStart="8dp"
    android:layout_marginTop="43dp"
    android:layout_marginEnd="8dp"
    android:layout_marginBottom="8dp"
    android:isScrollContainer="true"
    android:scrollbarAlwaysDrawVerticalTrack="true"
    android:scrollbarStyle="insideInset"
    android:scrollbars="vertical"
    app:layout_constraintBottom_toTopOf="@+id/navigation"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
```

Teste o aplicativo, verifique que os produtos indisponíveis não podem ser comprados, o botão de compra é desabilitado.

Vamos adicionar mais uma funcionalidade no layout do produto da lista, tornando o objeto `descricao_produto` clicável:



Vamos abrir a classe MyProdutoRecyclerViewAdapter para adicionar um `setOnClickListener` ao objeto que apresenta o nome do produto:



```

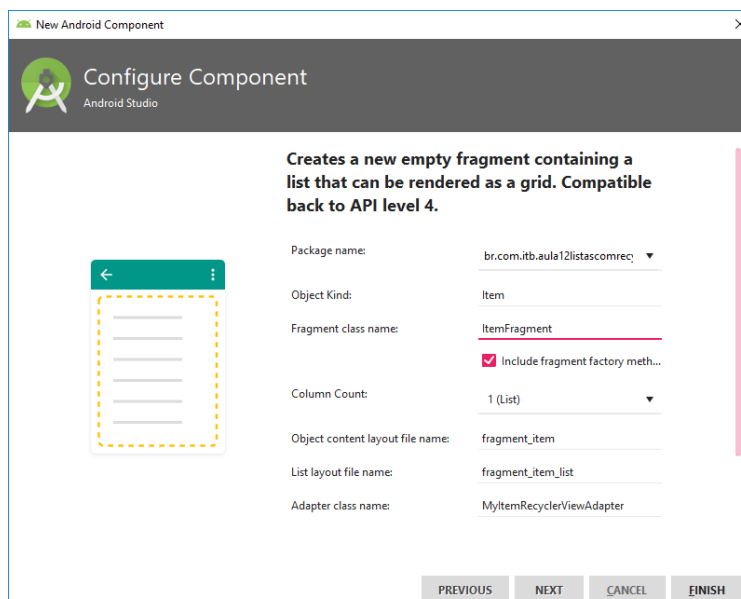
//Habilitar o clique no objeto descricao_produto
holder.mDescricaoView.setOnClickListener(new
View.OnClickListener() {
    @Override
    public void onClick(View v) {
        //Chamar nova lista com itens do produto chamado

    }
});

```

Para continuar a implementação do método acima vamos criar um novo fragmento de listagem, clique com o botão direito em Android > app, em seguida escolha New > Fragment > Fragment (List)

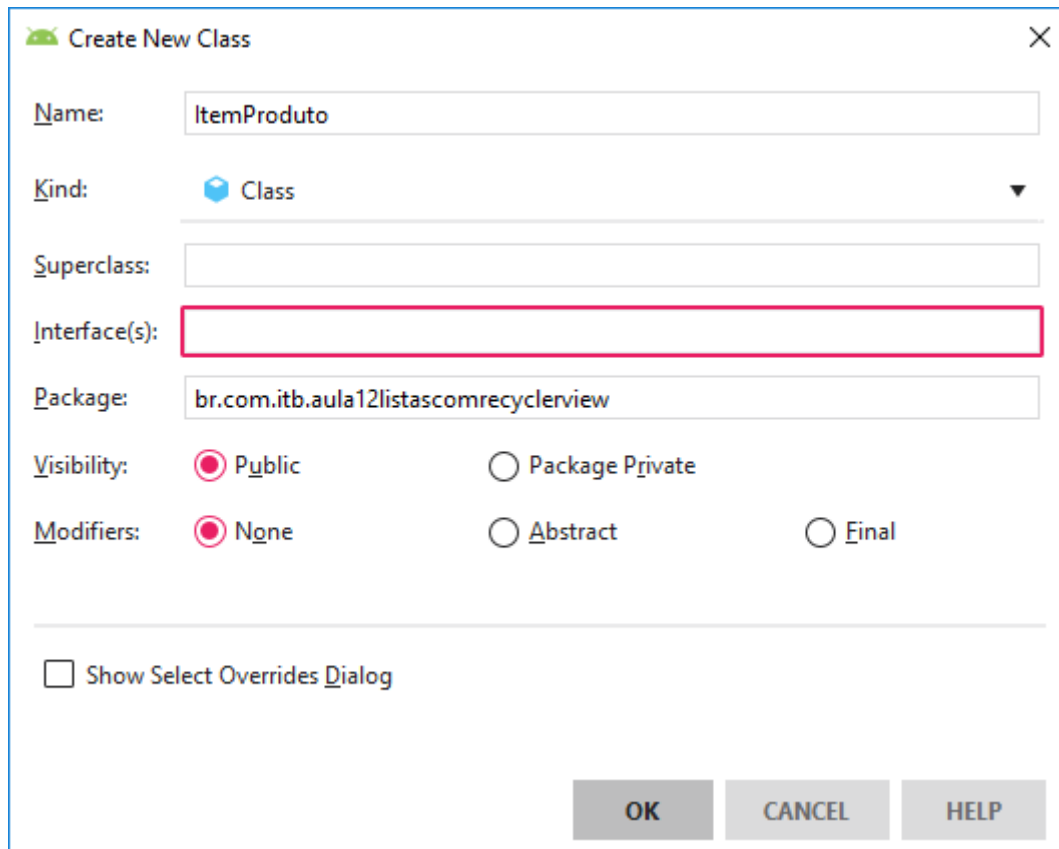
Não altere nenhuma opção e clique em FINISH, observe na imagem abaixo:



Vamos criar uma classe chamada Item:

Clique com o botão direito em Android > app > java > br.com.itb.aula12listascomrecyclerview

Escolha New > Java Class



Create New Class

Name: ItemProduto

Kind: Class

Superclass:

Interface(s):

Package: br.com.itb.aula12listascomrecyclerview

Visibility: ☒ Public ☐ Package Private

Modifiers: ☒ None ☐ Abstract ☐ Final

☐ Show Select Overrides Dialog

OK CANCEL HELP

```
package br.com.itb.aula12listascomrecyclerview;

public class ItemProduto {
    private String codigo;
    private String descricao;
    private int qtde;
    private double valor;
    private int status;

    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }
}
```

```
public String getDescricao() {  
    return descricao;  
}  
  
public void setDescricao(String descricao) {  
    this.descricao = descricao;  
}  
  
public int getQtde() {  
    return qtde;  
}  
  
public void setQtde(int qtde) {  
    this.qtde = qtde;  
}  
  
public double getValor() {  
    return valor;  
}  
  
public void setValor(double valor) {  
    this.valor = valor;  
}  
  
public int getStatus() {  
    return status;  
}  
  
public void setStatus(int status) {  
    this.status = status;  
}  
  
    public ItemProduto itemProduto(){  
        return this;  
    }  
  
public ItemProduto(String codigo, String descricao,  
                    int qtde, double valor, int status){  
    this.codigo = codigo;  
    this.descricao = descricao;  
    this.qtde = qtde;  
    this.valor = valor;  
    this.status = status;  
}  
  
}
```

Vamos definir o layout do arquivo fragment_item.xml

```
<?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="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">

    <ImageView
        android:id="@+id/imageViewItem"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="1"
        app:srcCompat="@mipmap/ic_launcher" />

    <LinearLayout
        android:layout_width="123dp"
        android:layout_height="wrap_content"
        android:orientation="vertical">

        <TextView
            android:id="@+id/codigo_item_produto"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="@dimen/text_margin"
            android:text="@string/codigo"

            android:textAppearance="@android:style/TextAppearance.DeviceDefault
            .SearchResult.Subtitle" />

        <TextView
            android:id="@+id/descricao_item_produto"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="@dimen/text_margin"
            android:text="@string/descricao"

            android:textAppearance="@android:style/TextAppearance.DeviceDefault
            .Widget.ActionBar.Title" />

    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <TextView
            android:id="@+id/quantidade_item_produto"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_margin="@dimen/text_margin"
            android:background="@android:color/background_light"
            android:text="@string/quantidade"

            android:textAppearance="@android:style/TextAppearance.DeviceDefault
            .SearchResult.Subtitle" />

    </LinearLayout>
</LinearLayout>
```

```

<TextView
    android:id="@+id/valor_item_produto"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_margin="@dimen/text_margin"
    android:text="@string/valor_unitario"

    android:textAppearance="@android:style/TextAppearance.DeviceDefault
.SearchResult.Subtitle" />

</LinearLayout>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Switch
        android:id="@+id/status_item_produto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/text_margin"
        android:checked="true"

        android:switchTextAppearance="@android:style/TextAppearance.Holo.Se
archResult.Subtitle"
        android:text="@string/ativo"
        android:textSize="10sp"
        android:thumbTint="@color/colorPrimary" />

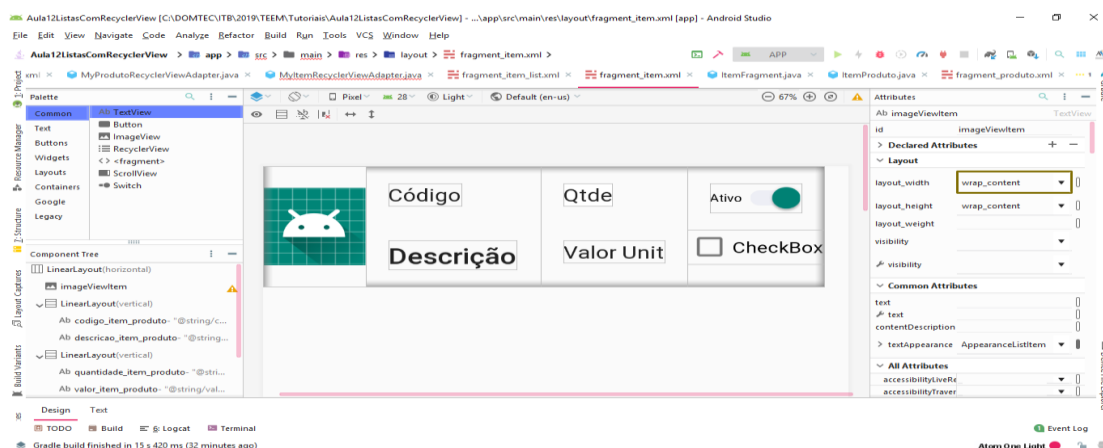
    <CheckBox
        android:id="@+id/chkStatusItemProduto"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="CheckBox" />

</LinearLayout>

</LinearLayout>

```

O layout do item terá a seguinte aparência:



Adicionaremos uma listagem do item do produto na classe Conexao:

```
//Método de pesquisa de itens na base SQL Server
public static ArrayList<ItemProduto> pesquisarItensFiltro(String
texto){
    ArrayList<ItemProduto> lista = new ArrayList<ItemProduto>();
    try{
        //Pesquisa no banco de dados
        PreparedStatement pst = Conexao.conexaoBD()
            .prepareStatement("select * from item_produto " +
                "where cod_prod = '" + 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()){
            ItemProduto item = new ItemProduto(
                res.getString(1),
                res.getString(2),
                res.getInt(3),
                res.getDouble(4),
                res.getInt(6)
            );
            lista.add(item);
        }
    } catch (SQLException e) {
        e.getMessage().toString();
    }
    return lista;
}
```

Vamos criar uma classe chamada ItemProdutoController:

```
package br.com.itb.aula12listascomrecyclerview;

import java.util.ArrayList;

public class ItemProdutoController {
    public ArrayList<ItemProduto> listagemItemProduto(String codigo){
        ArrayList<ItemProduto> listagem;
        listagem = Conexao.pesquisarItensFiltro(codigo);
        return listagem;
    }
}
```

Faremos alguns ajustes na classe ItemFragment.java

```

public class ItemFragment extends Fragment {

    // TODO: Customize parameter argument names
    private static final String ARG_COLUMN_COD_PROD = "codigo-
produto";
    // TODO: Customize parameters
    private String mColumnCodigo = "1";
    private OnListFragmentInteractionListenerItem mListenerItem;
    private ItemProdutoController itemProdutoController;

    /**
     * Mandatory empty constructor for the fragment manager to instantiate
     the
     * fragment (e.g. upon screen orientation changes).
     */
    public ItemFragment() {
    }

    // TODO: Customize parameter initialization
    @SuppressWarnings("unused")
    public static ItemFragment newInstance(String codigoProduto) {
        ItemFragment fragment = new ItemFragment();
        Bundle args = new Bundle();
        args.putString(ARG_COLUMN_COD_PROD, codigoProduto);
        fragment.setArguments(args);
        return fragment;
    }

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

        if (getArguments() != null) {
            mColumnCodigo = getArguments().getString(ARG_COLUMN_COD_PROD);
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_item_list,
            container, false);

        // Criação do objeto Controller
        itemProdutoController = new ItemProdutoController();

        // Set the adapter
        if (view instanceof RecyclerView) {
            Context context = view.getContext();
            RecyclerView recyclerView = (RecyclerView) view;
            recyclerView.setLayoutManager(new
                LinearLayoutManager(context));
            recyclerView.setAdapter(new MyItemRecyclerViewAdapter(
                itemProdutoController.listagemItemProduto(mColumnCodigo),
                mListenerItem));
        }
        return view;
    }
}

```



```

@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof
OnListFragmentInteractionListenerItem) {
        mListenerItem = (OnListFragmentInteractionListenerItem)
context;
    } else {
        throw new RuntimeException(context.toString()
            + " must implement
OnListFragmentInteractionListenerItem");
    }
}

@Override
public void onDetach() {
    super.onDetach();
    mListenerItem = null;
}

/**
 * This interface must be implemented by activities that contain
this
 * fragment to allow an interaction in this fragment to be
communicated
 * to the activity and potentially other fragments contained in
that
 * activity.
 * <p/>
 * See the Android Training lesson <a href=
"http://developer.android.com/training/basics/fragments/communicatin
g.html"
 * >Communicating with Other Fragments</a> for more information.
 */
public interface OnListFragmentInteractionListenerItem {
    // TODO: Update argument type and name
    void onListFragmentInteractionItem(ItemProduto item);
}
}

```

Observe que o nome da interface `OnListFragmentInteractionListener` foi alterada para `OnListFragmentInteractionListenerItem`, para diferenciar do método que foi implementado na classe `ProdutoFragment`, também chamado `OnListFragmentInteractionListener`.

E para que o clique no nome do produto, que chamei de `descricao_produto` funcione adequadamente, um novo Listener chamado `Listener2` foi criado na classe `ProdutoFragment`, e uma nova interface de interação será chamada de `OnListFragmentInteractionListener2` para a implementação da chamada da segunda lista.

A lógica é, pesquisamos um produto, se este produto possuir itens, uma nova lista será chamada, quando clicarmos no nome do produto, se clicarmos em qualquer parte da apresentação do produto na lista, uma página de detalhamento do produto será chamada.

Segue a implementação da classe ProdutoFragment.java, com um novo Listener, chamado Listener2:

```
package br.com.itb.aula12listascomrecyclerview;

import android.content.Context;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v7.widget.GridLayoutManager;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.SearchView;
import java.util.ArrayList;
import br.com.itb.aula12listascomrecyclerview.dummy.ProdutoContent;
import br.com.itb.aula12listascomrecyclerview.dummy.ProdutoContent.ProdutoItem;

/**
 * A fragment representing a list of Items.
 * 

<p/>
 * Activities containing this fragment MUST implement the {@link OnListFragmentInteractionListener}
 * interface.
 */
public class ProdutoFragment extends Fragment {

    // TODO: Customize parameter argument names
    private static final String ARG_COLUMN_COUNT = "column-count";
    // TODO: Customize parameters
    private int mColumnCount = 1;
    private OnListFragmentInteractionListener mListener;
    private OnListFragmentInteractionListener2 mListener2;
    private ArrayList<ProdutoItem> lista = new
ArrayList<ProdutoItem>();
    private int pesquisou = 0;
    SearchView texto;
    private View view;

    /**
     * Mandatory empty constructor for the fragment manager to
     instantiate the
     * fragment (e.g. upon screen orientation changes).
     */
    public ProdutoFragment() {
    }

    // TODO: Customize parameter initialization
    @SuppressWarnings("unused")
    public static ProdutoFragment newInstance(int columnCount) {
        ProdutoFragment fragment = new ProdutoFragment();
        Bundle args = new Bundle();
        args.putInt(ARG_COLUMN_COUNT, columnCount);
        fragment.setArguments(args);
        return fragment;
    }
}


```

```

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

    if (getArguments() != null) {
        mColumnCount = getArguments().getInt(ARG_COLUMN_COUNT);
    }
}

@Override
public View onCreateView(LayoutInflater inflater, ViewGroup
container,
                        Bundle savedInstanceState) {
    view = inflater.inflate(R.layout.fragment_produto_list,
container, false);

    // Realizar a pesquisa personalizada no banco de dados
    // Através da SearchView svPesquisar
    // Busca o texto digitado no objeto svPesquisar (SearchView)
    texto = view.findViewById(R.id.svPesquisar);

    // O texto comentado abaixo é referente à ação na perda do foco
do SearchView
    /*texto.setOnQueryTextFocusChangeListener(new
View.OnFocusChangeListener() {
        @Override
        public void onFocusChange(View v, boolean hasFocus) {

        }
    });*/

    // perform set on query text listener event
    texto.setOnQueryTextListener(new SearchView.OnQueryTextListener()
{
        @Override
        public boolean onQueryTextSubmit(String query) {
            // do something on text submit
            lista = Conexao.pesquisarProdutosFiltro(query);
            pesquisou = 1;

            Context context = view.getContext();
            //Altere o código neste ponto para carregar o
RecyclerView
            RecyclerView recyclerView = (RecyclerView)
view.findViewById(R.id.list);
            if (mColumnCount <= 1) {
                recyclerView.setLayoutManager(new
LinearLayoutManager(context));
            } else {
                recyclerView.setLayoutManager(new
GridLayoutManager(context, mColumnCount));
            }
            // Se realizou pesquisa por filtro, envia a lista acima
criada
            if(pesquisou == 0)
                recyclerView.setAdapter(new
MyProdutoRecyclerViewAdapter(ProdutoContent.ITEMS, mListener,
mListener2));
            else
                recyclerView.setAdapter(new
MyProdutoRecyclerViewAdapter(lista, mListener, mListener2));
            return false;
        }
    }
}

```

```

        @Override
        public boolean onQueryTextChanged(String newText) {
            // do something when text changes
            return false;
        }
    });

    // Código original comentado, para carregamento do RecyclerView
    sem filtro
    /**/ Set the adapter
    if (view instanceof RecyclerView) {
        Context context = view.getContext();
        RecyclerView recyclerView = (RecyclerView) view;
        if (mColumnCount <= 1) {
            recyclerView.setLayoutManager(new
LinearLayoutManager(context));
        } else {
            recyclerView.setLayoutManager(new
GridLayoutManager(context, mColumnCount));
        }
        // Se realizou pesquisa por filtro, envia a lista acima
        criada
        if (pesquisou == 0)
            recyclerView.setAdapter(new
MyProdutoRecyclerViewAdapter(ProdutoContent.ITEMS, mListener));
        else
            recyclerView.setAdapter(new
MyProdutoRecyclerViewAdapter(lista, mListener));
    }
    /**/
    return view;
}

@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnListFragmentInteractionListener) {
        mListener = (OnListFragmentInteractionListener) context;
        mListener2 = (OnListFragmentInteractionListener2) context;
    }
    else {
        throw new RuntimeException(context.toString()
            + " must implement
OnListFragmentInteractionListener");
    }
}

@Override
public void onDetach() {
    super.onDetach();
    mListener = null;
    mListener2 = null;
}

```

```

/**
 * This interface must be implemented by activities that contain
 * this
 * fragment to allow an interaction in this fragment to be
 * communicated
 * to the activity and potentially other fragments contained in
 * that
 * activity.
 * <p/>
 * See the Android Training lesson <a href=
"http://developer.android.com/training/basics/fragments/communicatin
g.html"
 * >Communicating with Other Fragments</a> for more information.
 */
public interface OnListFragmentInteractionListener {
    // TODO: Update argument type and name
    void onListFragmentInteraction(ProdutoItem item);
}

public interface OnListFragmentInteractionListener2 {
    // TODO: Update argument type and name
    void onListFragmentInteraction2(ProdutoItem item);
}
}

```

Se atentem para o final desta classe, onde foi criada uma nova interface, para um novo Listener, que espera o clique em algum objeto da tela, no caso, no nome do produto.

Voltando para a classe MyProdutoRecyclerViewAdapter.java finalizamos a implementação da chamada do clique do botão:

Abaixo, este novo import deve ser inserido:

```

import
br.com.itb.aula12listascomrecyclerview.ProdutoFragment.OnListFragmentInteractionListener2;

```

Um novo objeto Listener foi criado e o construtor alterado:

```

private final OnListFragmentInteractionListener2 mListener2;

public MyProdutoRecyclerViewAdapter(ArrayList<ProdutoItem> items,
                                   OnListFragmentInteractionListener
listener,
                                   OnListFragmentInteractionListener2
listener2) {
    mValues = items;
    mListener = listener;
    mListener2 = listener2;
    //Adicionei este método para informar alterações no objeto de pesquisa
    notifyDataSetChanged();
}

```

Esta é a nova implementação do método que irá esperar o clique no nome do produto (descrição):

```
//Habilitar o clique no objeto descricao_produto
holder.mDescricaoView.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v) {
        // Chamar nova lista com itens do produto chamado
        if (null != mListener2) {
            mListener2.onListFragmentInteraction2(holder.mItem);
        }
    }
});
```

A classe MyItemRecyclerViewAdapter.java também deve ser alterada para atender os requisitos da classe ItemProduto.java criada acima:

```
package br.com.itb.aula12listascomrecyclerview;

import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.CheckBox;
import android.widget.Switch;
import android.widget.TextView;

import br.com.itb.aula12listascomrecyclerview.ItemFragment.OnListFragmentInteractionListenerItem;
import br.com.itb.aula12listascomrecyclerview.dummy.DummyContent.DummyItem;
import br.com.itb.aula12listascomrecyclerview.ItemProduto;

import java.util.List;

/**
 * {@link RecyclerView.Adapter} that can display a {@link ItemProduto} and makes a call to the
 * specified {@link OnListFragmentInteractionListenerItem}.
 * TODO: Replace the implementation with code for your data type.
 */
public class MyItemRecyclerViewAdapter extends RecyclerView.Adapter<MyItemRecyclerViewAdapter.ViewHolder> {

    private final List<ItemProduto> mValues;
    private final ItemFragment.OnListFragmentInteractionListenerItem mListenerItem;

    public MyItemRecyclerViewAdapter(List<ItemProduto> items,
        ItemFragment.OnListFragmentInteractionListenerItem listenerItem) {
        mValues = items;
        mListenerItem = listenerItem;
    }
}
```

```

@Override
public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
{
    View view = LayoutInflater.from(parent.getContext())
        .inflate(R.layout.fragment_item, parent, false);
    return new ViewHolder(view);
}

@Override
public void onBindViewHolder(final ViewHolder holder, int position)
{
    holder.mItem = mValues.get(position);
    holder.mCodigoView.setText(mValues.get(position).getCodigo());

    holder.mDescricaoView.setText(mValues.get(position).getDescricao());

    holder.mQtdeView.setText(String.valueOf(mValues.get(position).getQtde()));

    holder.mValorView.setText(String.valueOf(mValues.get(position).getValor()));

    // O status capturado é apresentado, de acordo com o valor
    // Se o valor for 1 o produto mostra que está disponível
    // Senão o produto será apresentado como indisponível
    if(mValues.get(position).getStatus() == 1){
        holder.mStatusView.setChecked(true);
        holder.mStatusView.setTextOn("ON");
        holder.mStatusView.setText("Disponível");
    }else{
        holder.mStatusView.setChecked(false);
        holder.mStatusView.setTextOff("OFF");
        holder.mStatusView.setText("Indisponível");
    }

    // Implementação do CheckBox adicionado
    if(mValues.get(position).getStatus() == 1){
        holder.mStatusView2.setChecked(true);
        holder.mStatusView2.setText("Disponível");
    }else{
        holder.mStatusView2.setChecked(false);
        holder.mStatusView2.setText("Indisponível");
    }

    //Quando o Switch é clicado
    holder.mStatusView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            int status = 99;
            if(!holder.mStatusView.isChecked()) {
                status = 0;
                //Switch
                holder.mStatusView.setTextOff("OFF");
                holder.mStatusView.setText("Indisponível");
                //CheckBox
                holder.mStatusView2.setChecked(false);
                holder.mStatusView2.setText("Indisponível");
            }
            else {
                status = 1;
                //Switch
                holder.mStatusView.setTextOn("ON");
                holder.mStatusView.setText("Disponível");
                //CheckBox
                holder.mStatusView2.setChecked(true);
                holder.mStatusView2.setText("Disponível");
            }
        }
    });
}

```

```

        Conexao.alterarStatusProduto(v.getRootView(),
            holder.mCodigoView.getText().toString(),
            status);
    }
});

//Quando o CheckBox é clicado
holder.mStatusView2.setOnClickListener(new View.OnClickListener()
{
    @Override
    public void onClick(View v) {
        int status = 99;
        if(!holder.mStatusView2.isChecked()) {
            status = 0;
            //Switch
            holder.mStatusView.setChecked(false);
            holder.mStatusView.setTextOff("OFF");
            holder.mStatusView.setText("Indisponível");
            //CheckBox
            holder.mStatusView2.setText("Indisponível");
        }
        else {
            status = 1;
            //Switch
            holder.mStatusView.setChecked(true);
            holder.mStatusView.setTextOn("ON");
            holder.mStatusView.setText("Disponível");
            //CheckBox
            holder.mStatusView2.setText("Disponível");
        }

        Conexao.alterarStatusProduto(v.getRootView(),
            holder.mCodigoView.getText().toString(),
            status);
    }
});

holder.mView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (null != mListenerItem) {
            // Notify the active callbacks interface (the
activity, if the
            // fragment is attached to one) that an item has been
selected.

mListenerItem.onListFragmentInteractionItem(holder.mItem);
        }
    }
});

@Override
public int getItemCount() {
    return mValues.size();
}
}

```



```

public class ViewHolder extends RecyclerView.ViewHolder {
    public final View mView;
    public final TextView mCodigoView;
    public final TextView mDescricaoView;
    public final TextView mQtdeView;
    public final TextView mValorView;
    public final Switch mStatusView;
    public CheckBox mStatusView2;
    public ItemProduto mItem;

    public ViewHolder(View view) {
        super(view);
        mView = view;
        mCodigoView = (TextView)
view.findViewById(R.id.codigo_item_produto);
        mDescricaoView = (TextView)
view.findViewById(R.id.descricao_item_produto);
        mQtdeView = (TextView)
view.findViewById(R.id.quantidade_item_produto);
        mValorView = (TextView)
view.findViewById(R.id.valor_item_produto);
        mStatusView = (Switch)
view.findViewById(R.id.status_item_produto);
        mStatusView2 = (CheckBox)
view.findViewById(R.id.chkStatusItemProduto);
    }

    @Override
    public String toString() {
        return super.toString() + " '" + mDescricaoView.getText()
+ "'";
    }
}

```

Por fim, na classe MainActivity, vamos inserir as novas interfaces na assinatura da classe:

```

public class MainActivity extends AppCompatActivity
    implements ProdutoFragment.OnListFragmentInteractionListener,
        ProdutoFragment.OnListFragmentInteractionListener2,
        HomeFragment.OnFragmentInteractionListener,
        LoginFragment.OnFragmentInteractionListener,
        DetalheProdutoFragment.OnFragmentInteractionListener,
        ItemFragment.OnListFragmentInteractionListenerItem{

```

Ainda na classe MainActivity.java, vamos inserir a implementação dos métodos de ação:

```
@Override
public void onListFragmentInteractionItem(ItemProduto item) {
    String codigo = item.getCodigo();
    String descricao = item.getDescricao();
    int qtde = item.getQtde();
    double valor = item.getValor();
    int situacao = item.getStatus();

    FragmentTransaction ft =
getSupportFragmentManager().beginTransaction();
    ft.replace(R.id.fragment_container,
DetalheProdutoFragment.newInstance(
        codigo, descricao, qtde, valor, situacao
    ))
        .addToBackStack(null)
        .commit();
}

@Override
public void onListFragmentInteraction2(ProdutoContent.ProdutoItem
item) {
    String codigo = item.codigo;

    FragmentTransaction ft =
getSupportFragmentManager().beginTransaction();
    ft.replace(R.id.fragment_container, ItemFragment.newInstance(
        codigo
    ))
        .addToBackStack(null)
        .commit();
}
```

O método onListFragmentInteractionItem faz com que o detalhamento do item selecionado na lista de itens do produto será executado e mostrado na tela.

O método onListFragmentInteraction2 faz com que a segunda lista, a lista dos itens apareça, de acordo com o código do produto selecionado.

Execute o aplicativo, faça o login, realize a pesquisa do termo “salgado”, quando clicar no nome do produto uma nova lista vai aparecer, se clicar em qualquer outra parte do produto selecionado um detalhamento vai aparecer.

Também programei o detalhamento do item selecionado na segunda lista.

Referência Bibliográfica

[1] <http://developer.android.com>. Acessado em 28/04/2019.