



PROGRAMAÇÃO PARA  
DISPOSITIVOS MÓVEIS

# FRAGMENTOS

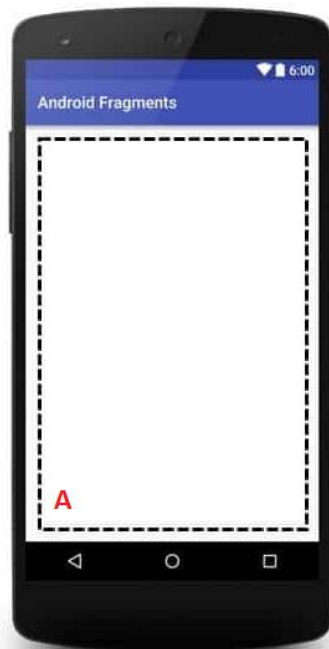
# O que são fragmentos?

- Um fragmento representa uma porção de interface de usuário (com ou sem comportamento) de uma atividade;
- É possível combinar fragmentos em uma única atividade ou distribuir e reusar fragmentos em múltiplas atividades;
- São “trechos modulares” de atividade.

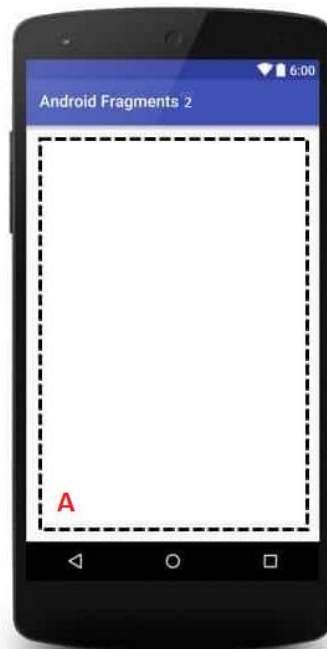
# O que são fragmentos?



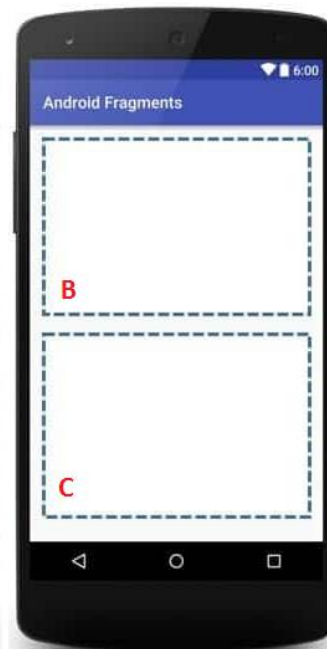
Atividade sem nenhum fragmento



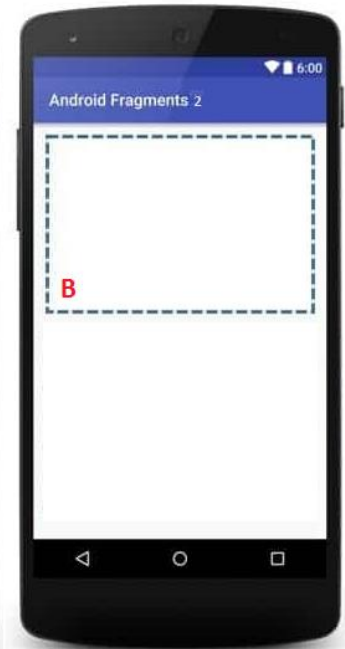
Atividade com um fragmento (A)



Outra atividade utilizando o fragmento (A)



Atividade com dois fragmentos (B) e (C)

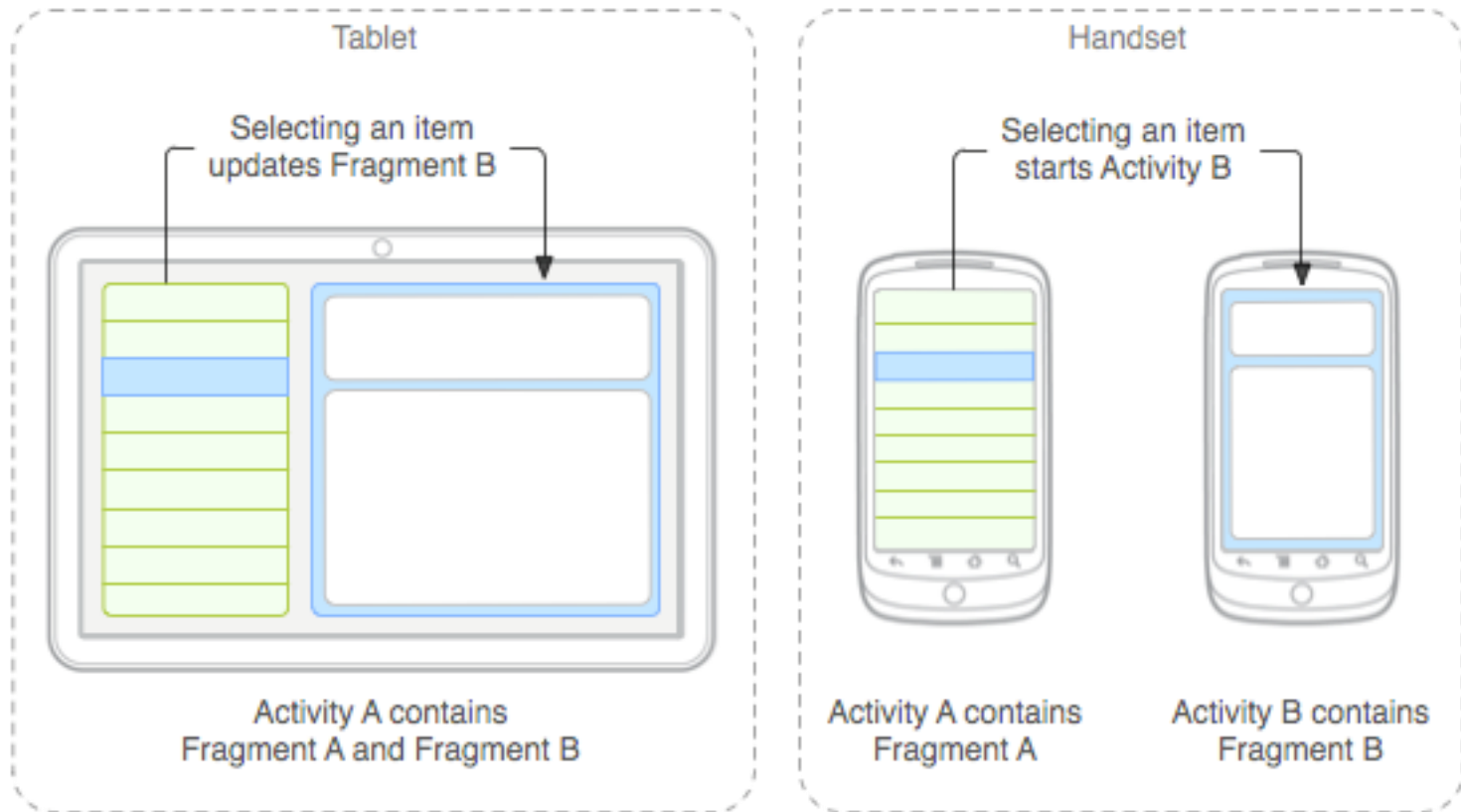


Atividade reutilizando o fragmento (B)

# Por que utilizar fragmentos?

- Modularização das views
- Separação da atividade e sua interface
- Acomodar diferentes tamanhos de tela e diferentes formatos

# Por que utilizar fragmentos?



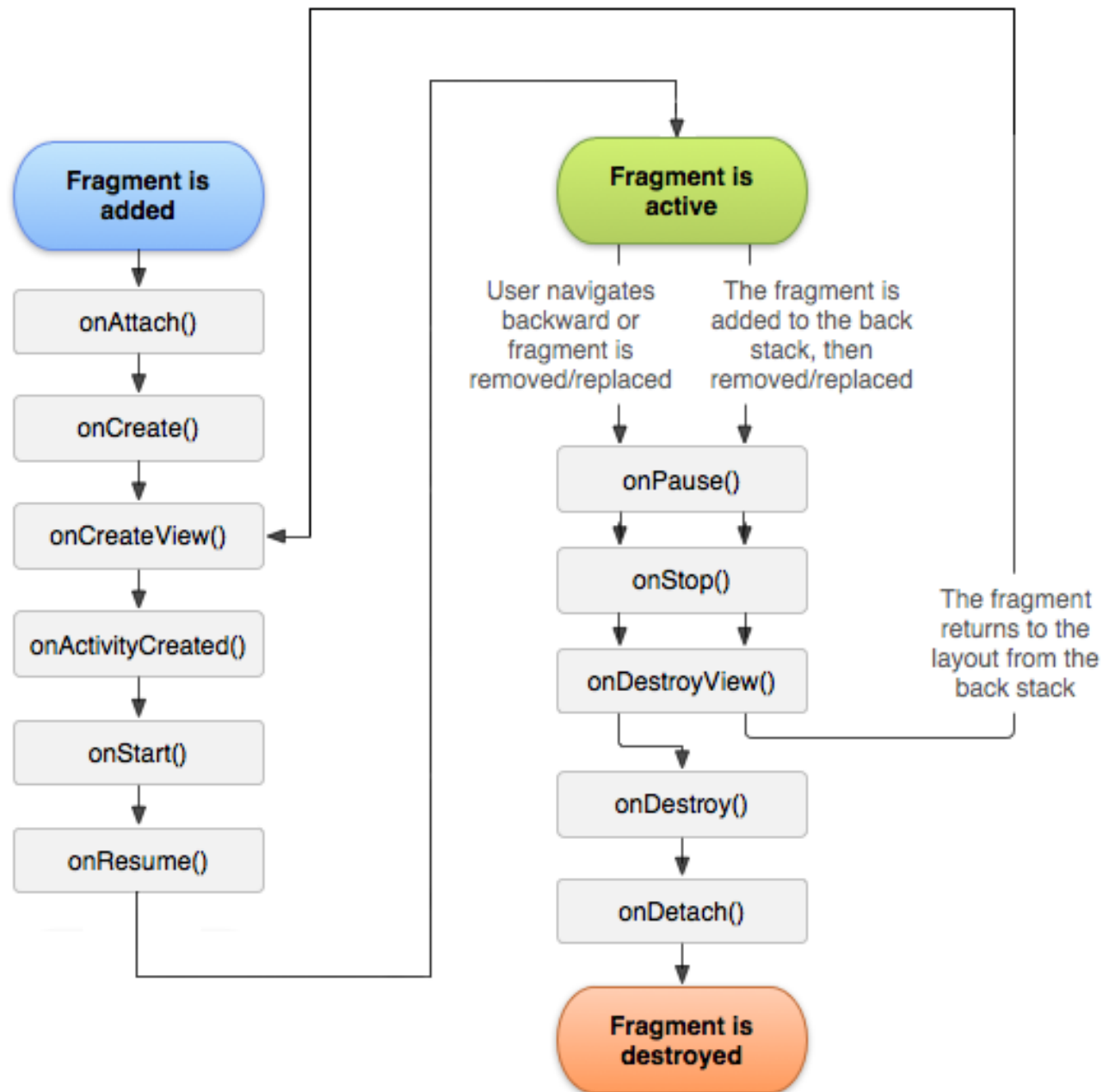
# Estrutura de fragmentos

- Fragmentos não existem por si só. Devem fazer parte de uma atividade;
- Consistem em um arquivo XML e um arquivo Java (como as atividades);

# Ciclo de vida de fragmentos

- Assim como atividades, os fragmentos também possuem um ciclo de vida;
- O ciclo de vida de fragmentos normalmente depende do ciclo de vida da atividade. Se uma atividade é pausada, o fragmento também será. O contrário não é verdadeiro.





# Arquivo XML

- O XML dos fragmentos é igual ao das atividades
- É um recurso de layout
  - Diretório “res/layout”
- Pode ter:
  - Parte de um layout
  - Layout completo

# Arquivo XML

- Quando um fragmento tem só parte de um layout?
  - Quando o fragmento está dentro do layout de uma atividade
- E quando o fragmento tem um layout completo?
  - Quando a atividade não tem layout (mas a atividade ainda precisa de uma classe Java)

# Arquivo Java

- Para criar um fragmento, deve-se criar sua classe estendendo-se de um dos seguintes elementos:
  - `android.app.Fragment`
  - **`android.support.v4.app.Fragment`** (para compatibilidade com versões anteriores do Android)

# Arquivo Java

- Principais tarefas do fragmento:
  - Inflar seu layout XML
  - Inicializar as views como objetos Java (binding)
  - Lidar com eventos da interface (listeners)

# Arquivo Java

- O método de callback “onCreateView()” é o principal método do fragmento
  - Diferentemente do “onCreate()” da atividade, é necessário inflar o layout em uma “View” e retorná-la no método
  - Para inflar o layout, utilizar o método “inflate()”, do objeto “LayoutInflater” recebido como parâmetro

# Exemplo de Arquivo Java

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

public class MyFragment extends Fragment {

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

# Fragmentos x Atividades

- Os conceitos gerais permanecem praticamente sem alteração:
  - A principal diferença é que os métodos que antes eram chamados diretamente da atividade (como “findViewById()”) agora são chamados utilizando a “View” criada e inflada



# Arquivo Java

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

    EditText editTextName = (EditText)
    view.findViewById(R.id.textName);

    return view;
}
```

# Adicionando à Atividade

- Duas maneiras de adicionar o fragmento à atividade
  - Estaticamente
  - Dinamicamente

# Adicionando à Atividade

- Estaticamente:
  - Adicionar o fragmento no próprio XML da atividade
- Dinamicamente:
  - O fragmento é adicionado via Java, na classe da atividade

# Exemplo Estático

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<FrameLayout
```

```
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent" >
```

```
    <fragment
```

```
        android:id="@+id/fragment_status"
```

```
        android:name="tsi.senac.MyFragment"
```

```
        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent" />
```

```
</FrameLayout>
```

# Exemplo Dinâmico

- O XML da atividade contém um `FrameLayout` (com id “frag\_container”) no qual será incluído o fragmento:

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/frag_container"></FrameLayout>
</LinearLayout>
```

# Exemplo Dinâmico

- No Java da atividade, o fragmento é inserido no FrameLayout:

```
public class MyActivity extends Activity {  
  
    private EditText editField;  
    private Button myButton;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_my);  
  
        if (savedInstanceState == null) {  
            MyFragment fragment = new MyFragment();  
            getSupportFragmentManager().beginTransaction()  
                .replace(R.id.frag_container,  
fragment).commit();  
        }  
    }  
}
```


O fragmento a ser adicionado

Onde o fragmento ficará

# Exemplo Dinâmico – Atividade sem XML

- Exemplo dinâmico, caso não haja um XML para a atividade:

```
public class MyActivity extends Activity {  
  
    private EditText editField;  
    private Button myButton;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        if (savedInstanceState == null) {  
            MyFragment fragment = new MyFragment();  
            getSupportFragmentManager().beginTransaction()  
                .add(android.R.id.content,  
fragment).commit();  
        }  
    }  
}
```



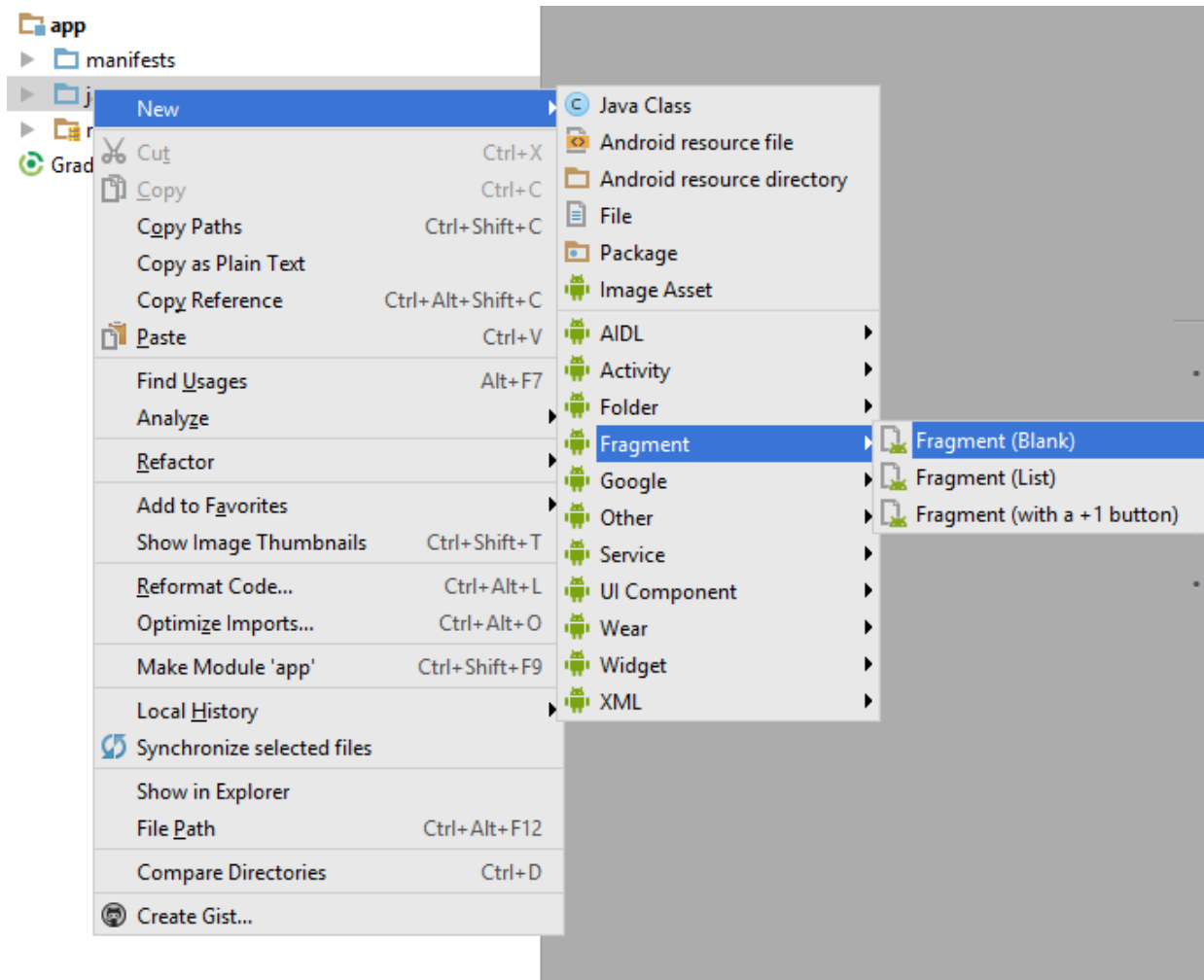
Argumento padrão, indicando que o fragmento ocupa todo o espaço da tela

# Android Studio e Fragmentos

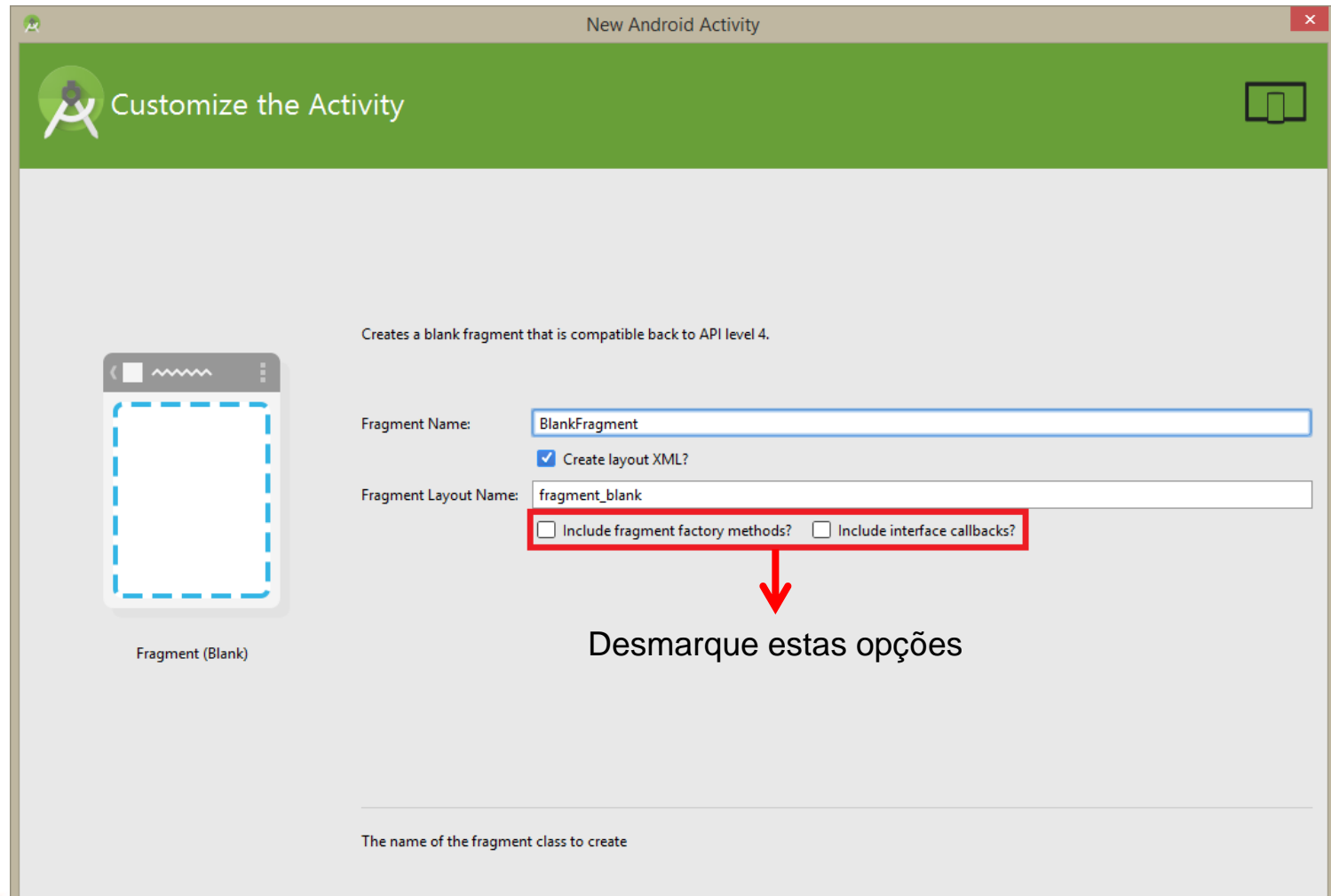
- O Android Studio permite criar uma atividade com um fragmento, um exemplo completo
  - Nesse caso, o fragmento é estático
- O Android Studio também permite criar somente um fragmento (XML + Java)
  - Nesse caso, ele precisará ser adicionado à atividade posteriormente



# Criando Fragmentos no Android Studio



# Criando Fragmentos no Android Studio



# Usos de Fragmentos

- Fragmentos são muito utilizados em:
  - Atividades com abas (Tabbed Activity)
  - Atividades com menu de navegação lateral (Navigation Drawer Activity)

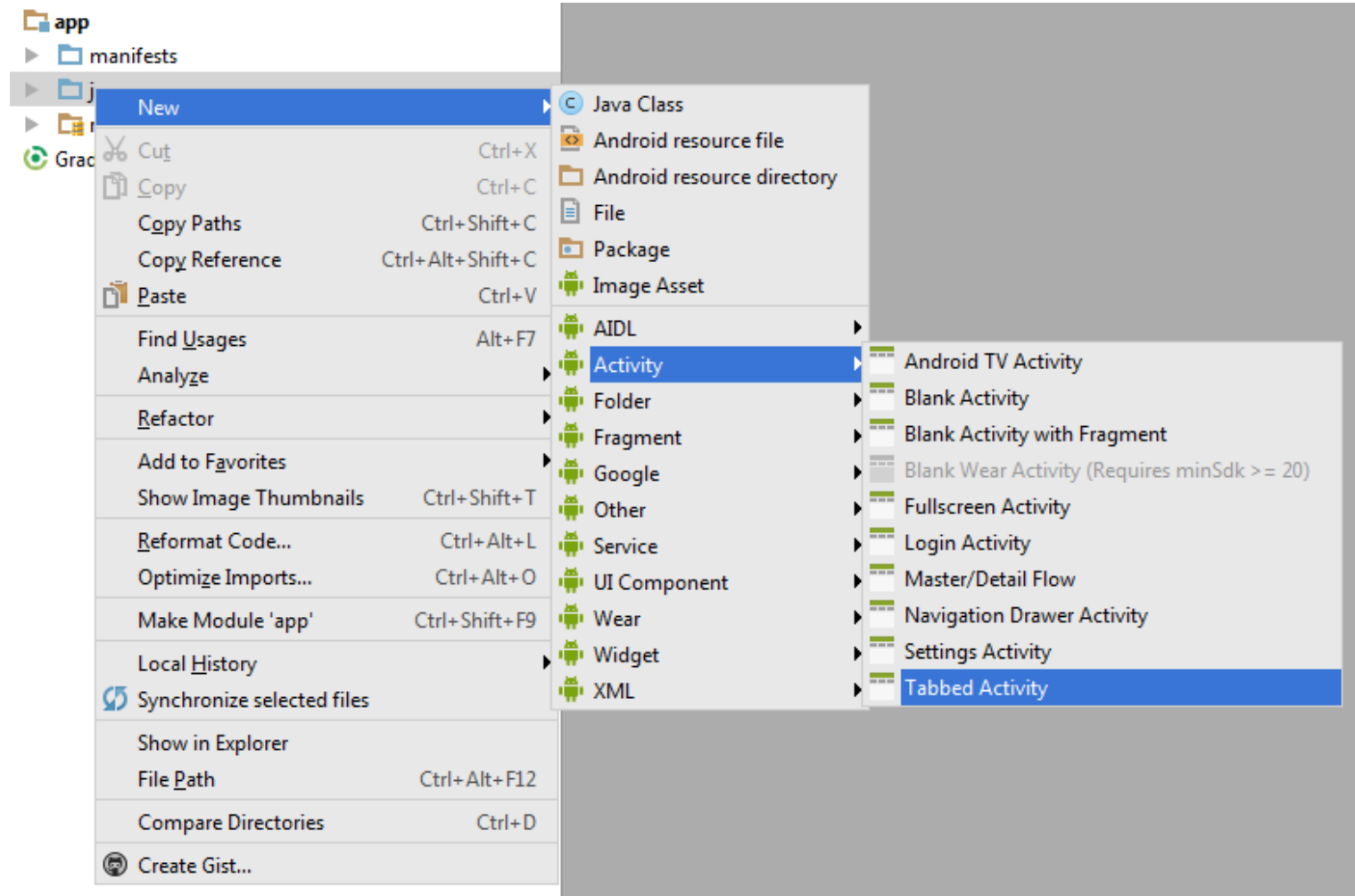
# Usos de Fragmentos

- O conteúdo principal fica separado em fragmentos
  - A atividade em si contém apenas as abas ou o menu lateral
  - Ao selecionar uma aba ou um item do menu, o fragmento correspondente é carregado

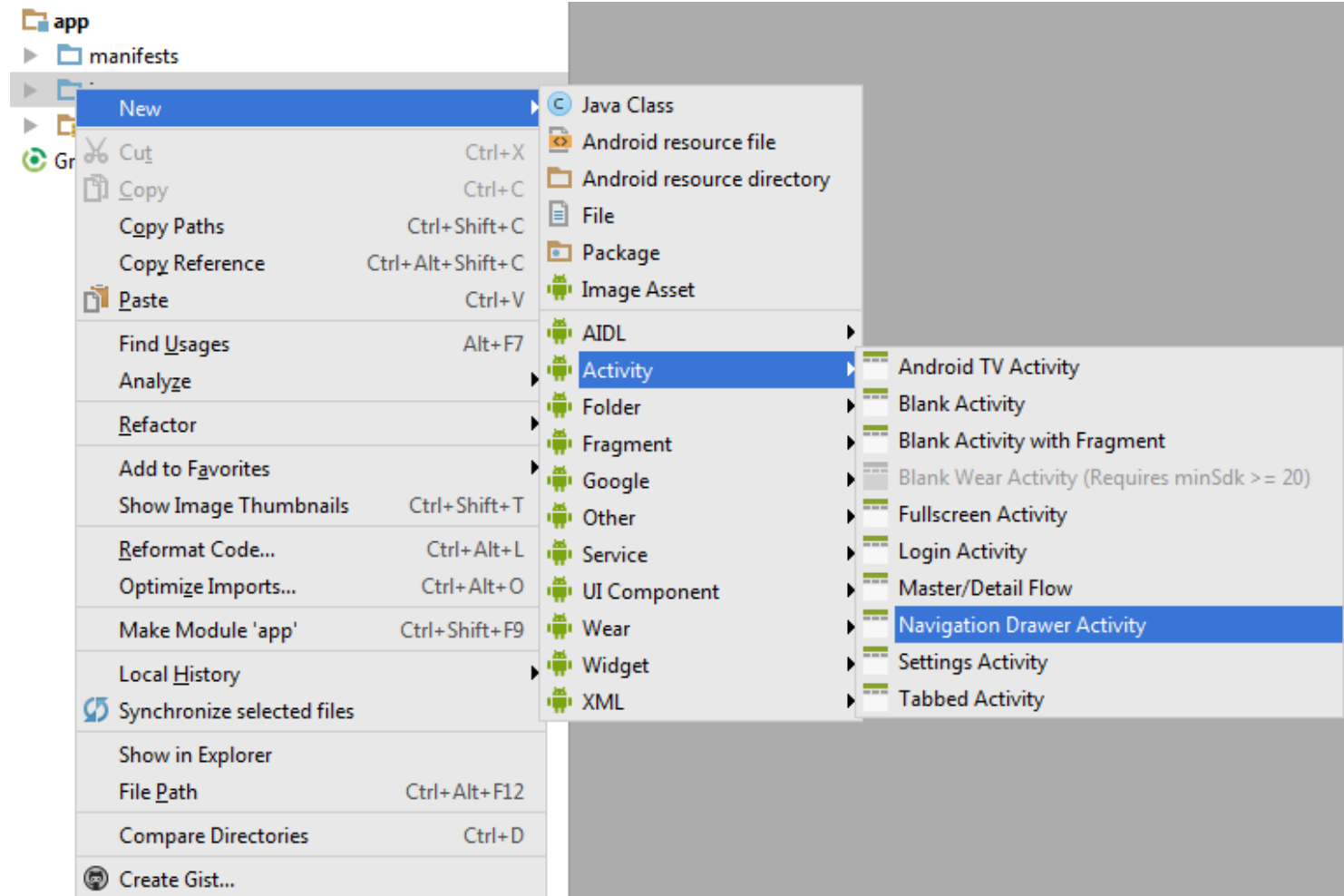
# Fragmentos no Android Studio

- O Android Studio permite criar facilmente os dois tipos de atividade, com classes e elementos de exemplo prontos

# Fragmentos no Android Studio



# Fragmentos no Android Studio



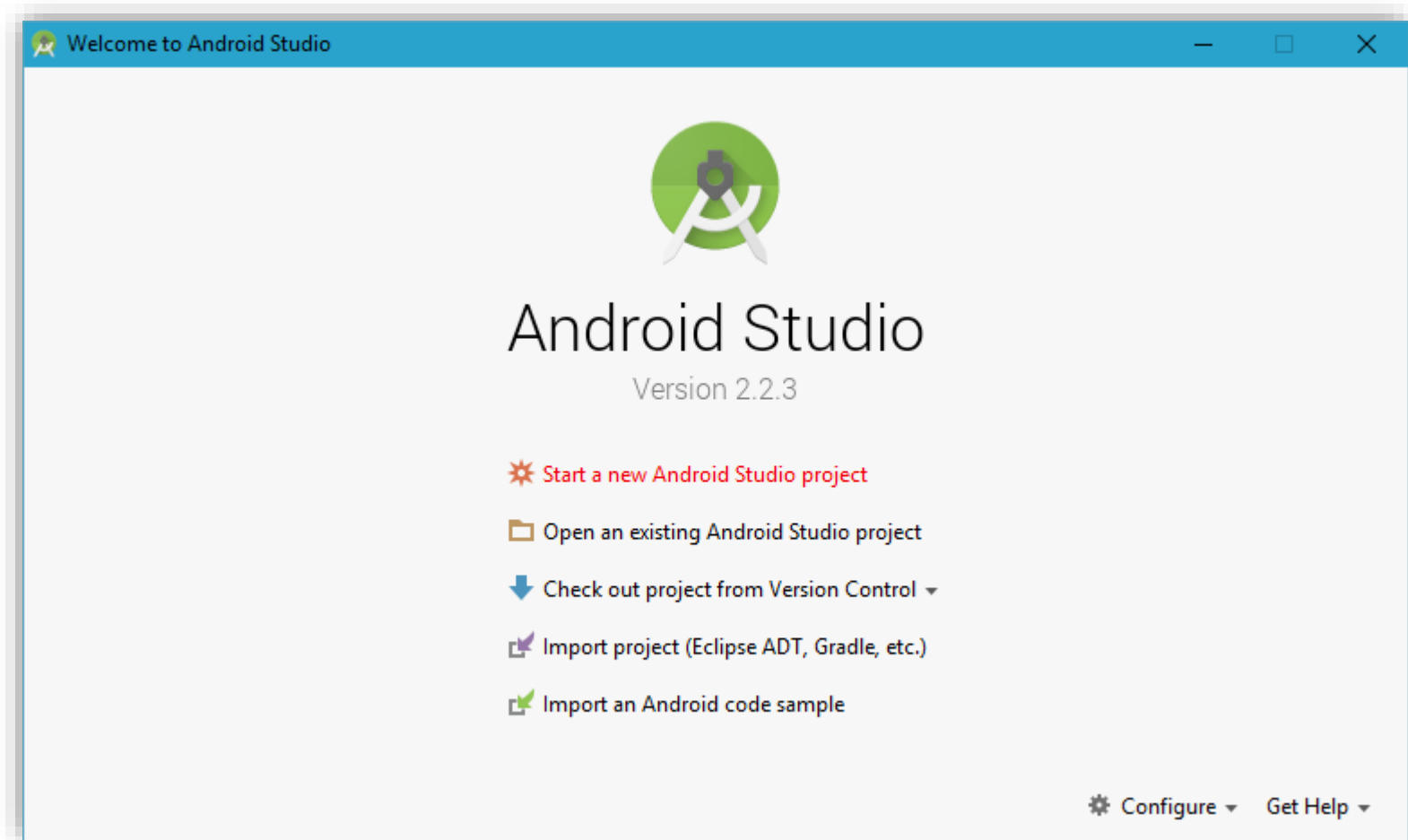
# CRIANDO FRAGMENTOS



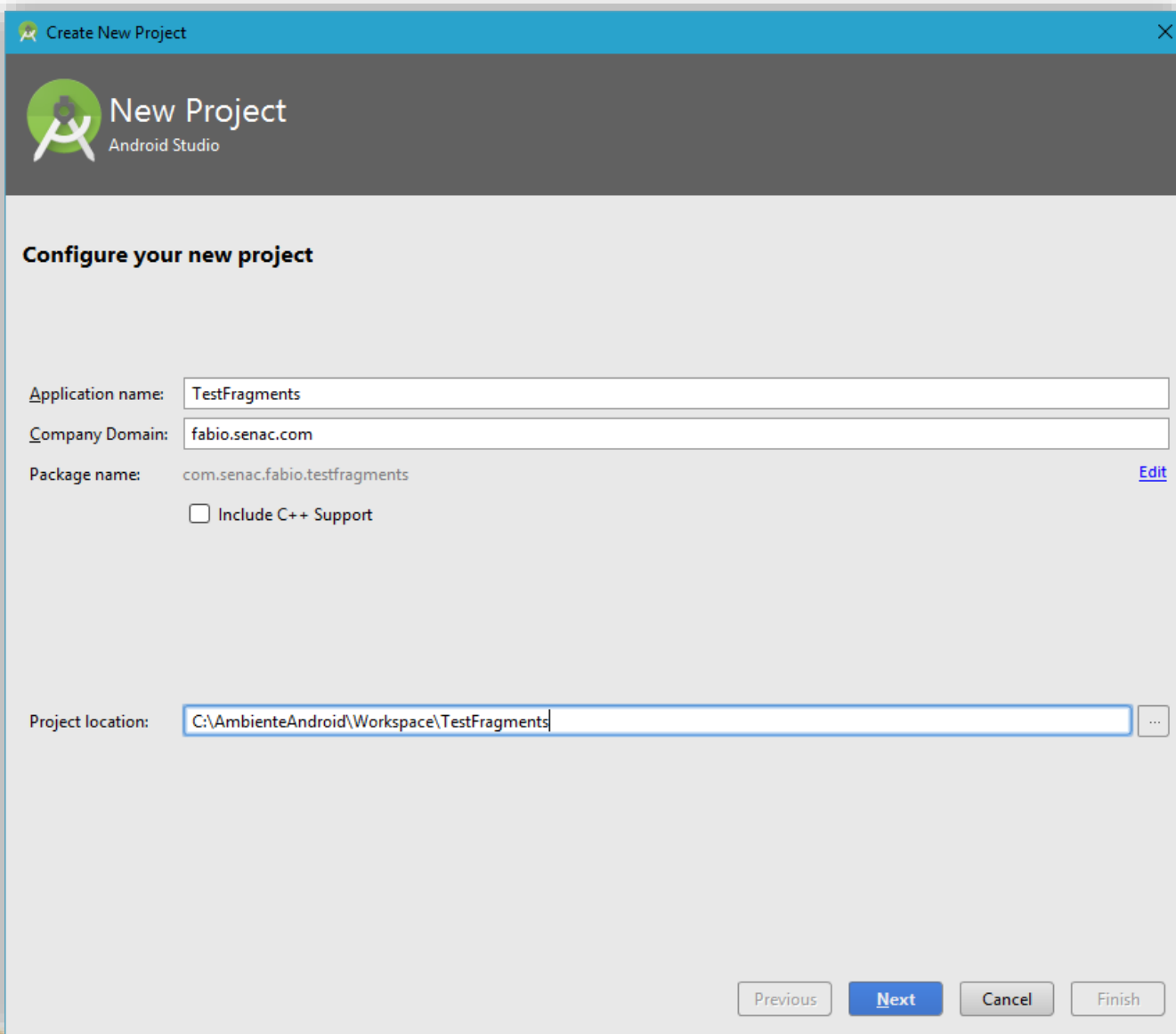
# Criando um Projeto para Fragmentos

- Vamos criar um novo projeto para lidar com fragmentos;
- No Android Studio, crie um novo projeto com o nome “TestFragments”. O projeto não deve conter nenhuma atividade inicialmente;
- Utilize as configurações padrão que usamos em aulas anteriores para sua criação.


# Criando um Projeto



# Criando um Projeto



Create New Project

 **New Project**  
Android Studio

**Configure your new project**

Application name:


Company Domain:


Package name:  [Edit](#)

☐ Include C++ Support

Project location:  ...

# Criando um Projeto

 Create New Project ✕

 Target Android Devices

**Select the form factors your app will run on**  
Different platforms may require separate SDKs

☒ Phone and Tablet

Minimum SDK API 23: Android 6.0 (Marshmallow) ▼  
Lower API levels target more devices, but have fewer features available.  
By targeting API 23 and later, your app will run on approximately **15,3%** of the devices that are active on the Google Play Store.  
[Help me choose](#) Stats load failed. Value may be out of date.

☐ Wear

Minimum SDK API 21: Android 5.0 (Lollipop) ▼

☐ TV

Minimum SDK API 21: Android 5.0 (Lollipop) ▼

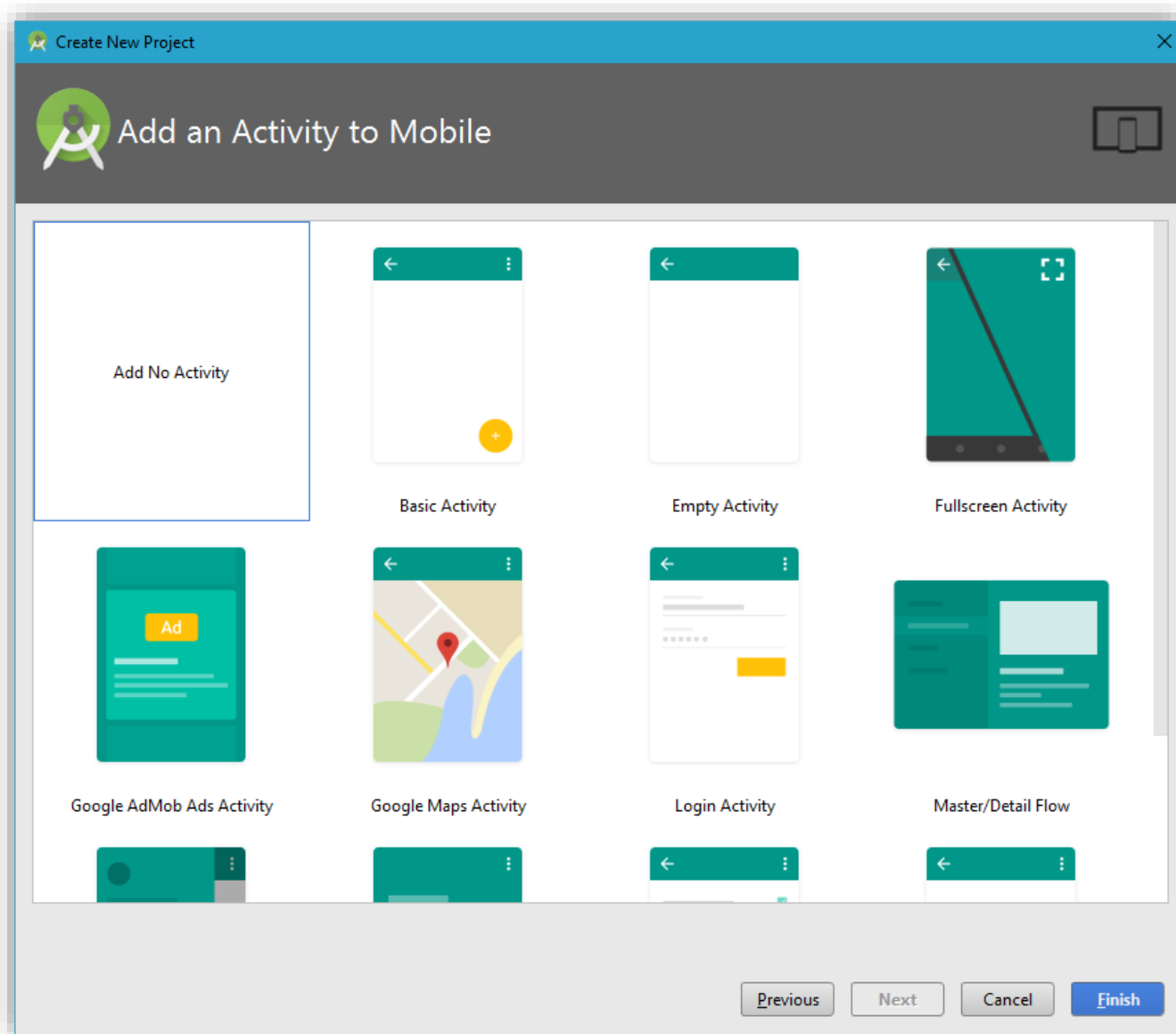
☐ Android Auto

☐ Glass (Not Available)

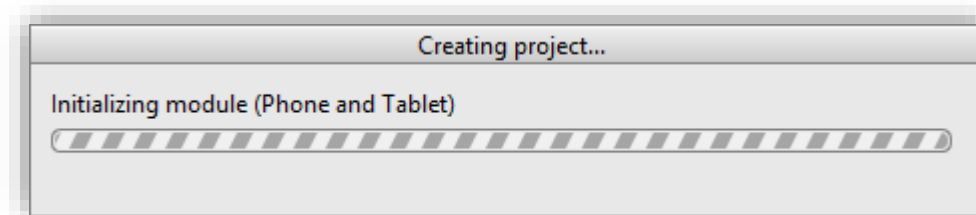
Minimum SDK  ▼

Previous Next Cancel Finish

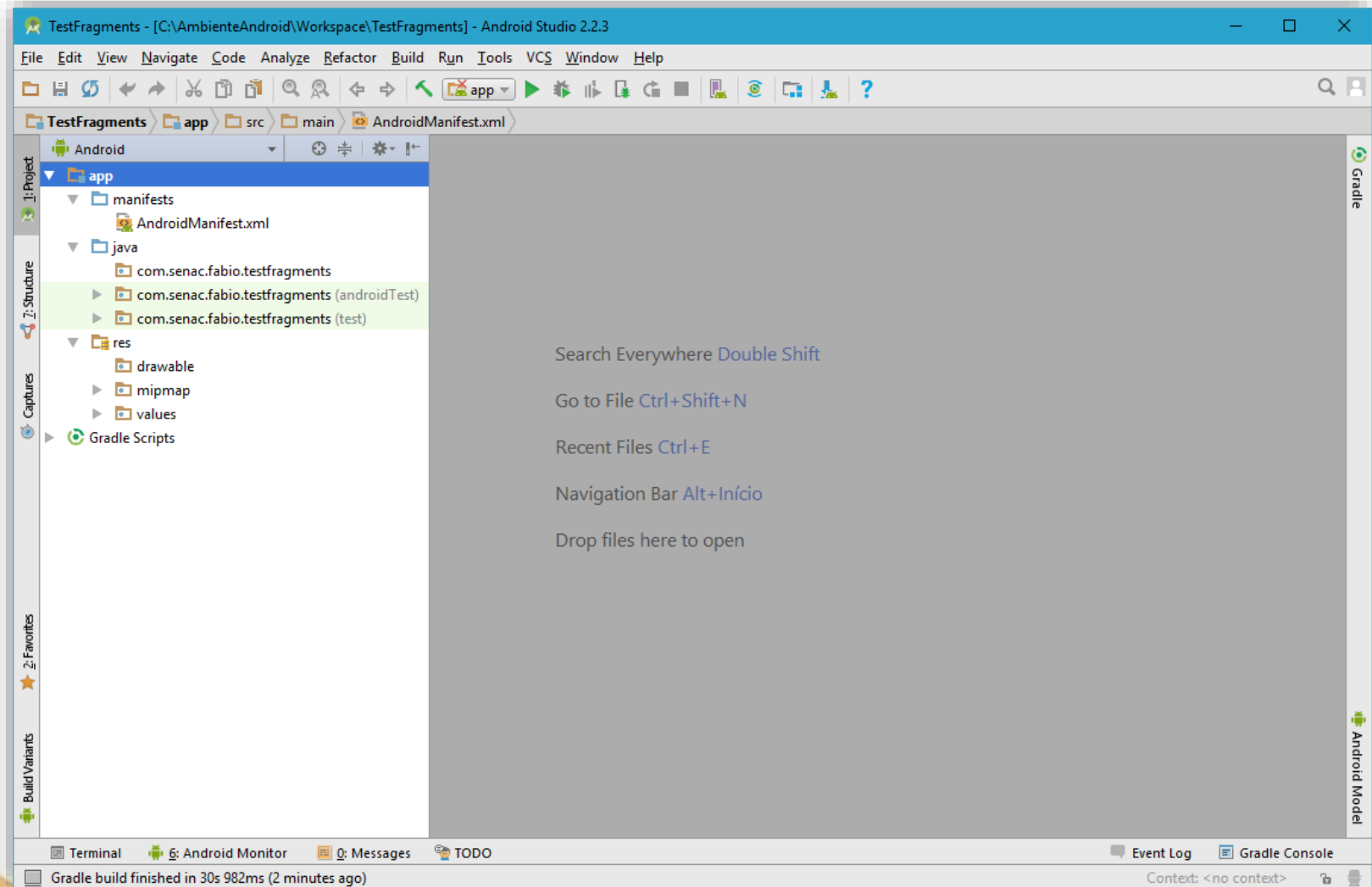
# Criando um Projeto



# Criando um Projeto



# Criando um Projeto

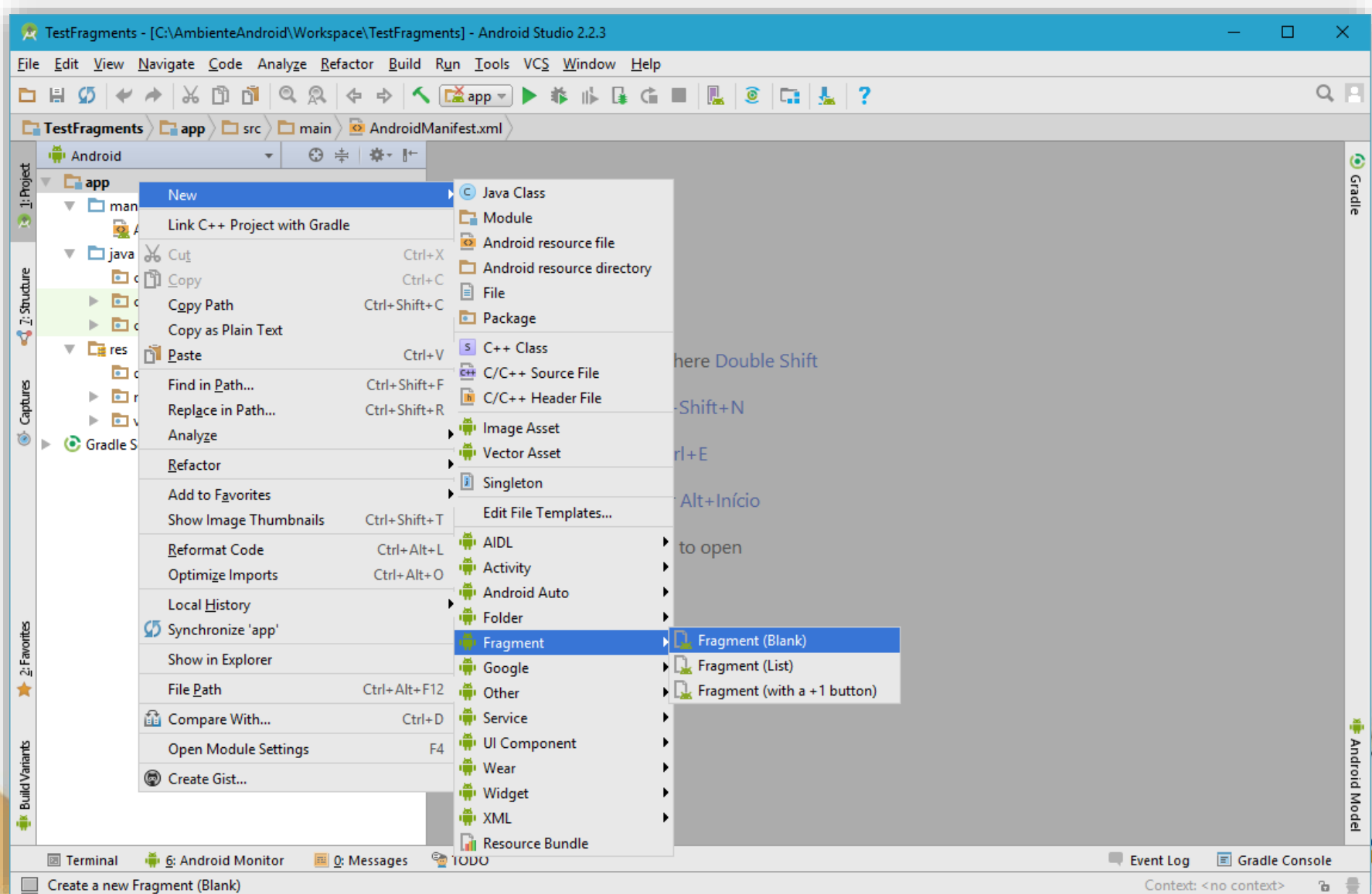


# Criando um Fragmento

- Com o projeto, podemos criar um fragmento;
- Clique com o botão direito em “app” e selecione as opções “New” > “Fragment” > “Fragment (Blank)” para que um novo fragmento vazio seja criado.



# Criando um Fragmento




# Criando um Fragmento


- No diálogo de criação de fragmentos, é possível configurar o novo fragmento;
- Dê ao fragmento o nome de **“FragmentoTeste01”**;
- Deixe marcada a opção para criação do layout do fragmento (**“Create layout XML ?”**);
- Desmarque as opções **“Include fragment factory methods?”** e **“Include interface callbacks?”** e clique em **“Finish”**.

# Criando um Fragmento

New Android Component

 **Configure Component**  
Android Studio

Creates a blank fragment that is compatible back to API level 4.



Fragment Name:

☒ Create layout XML?

Fragment Layout Name:

☐ Include fragment factory methods?

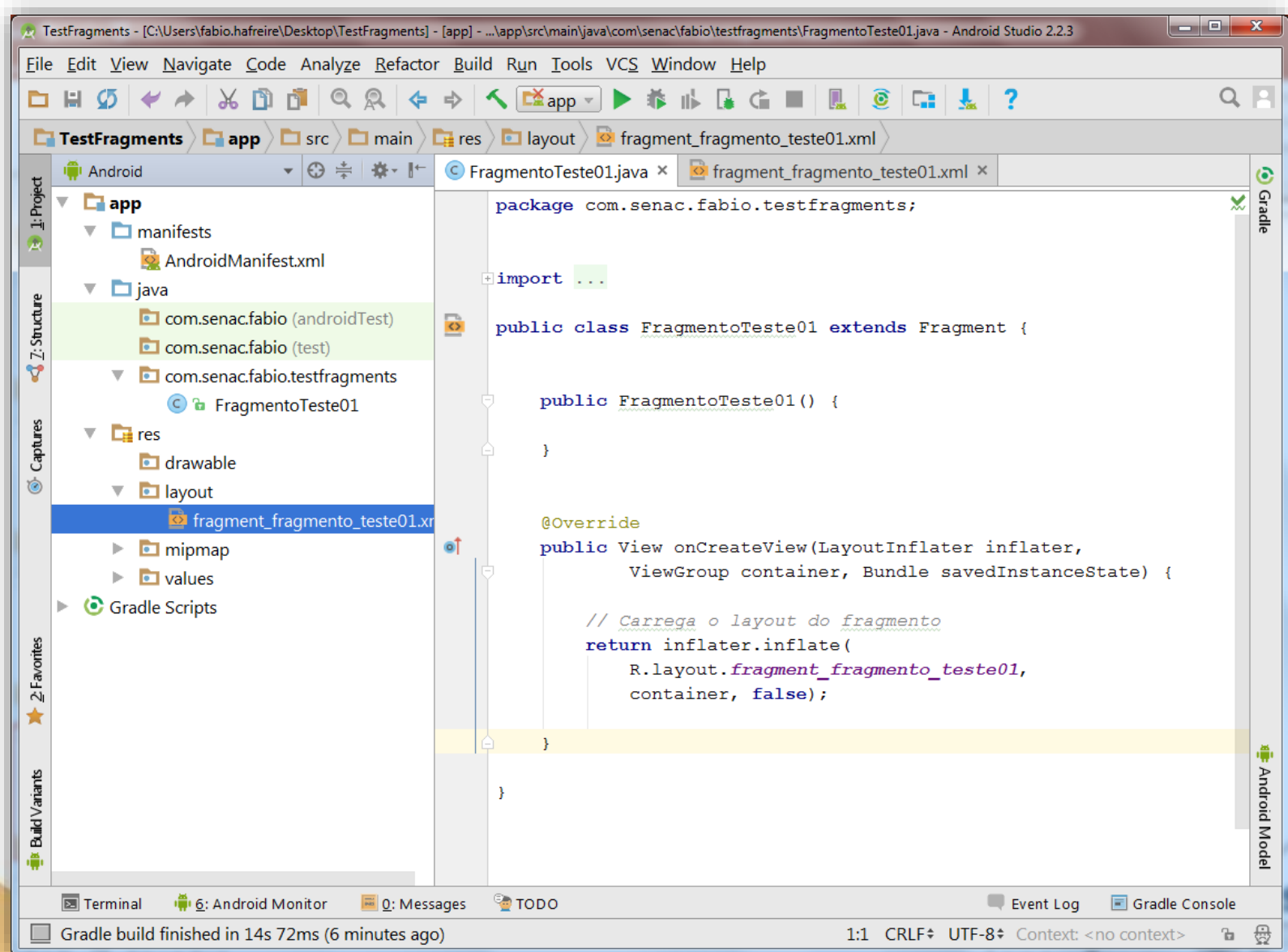
☐ Include interface callbacks?

Target Source Set:

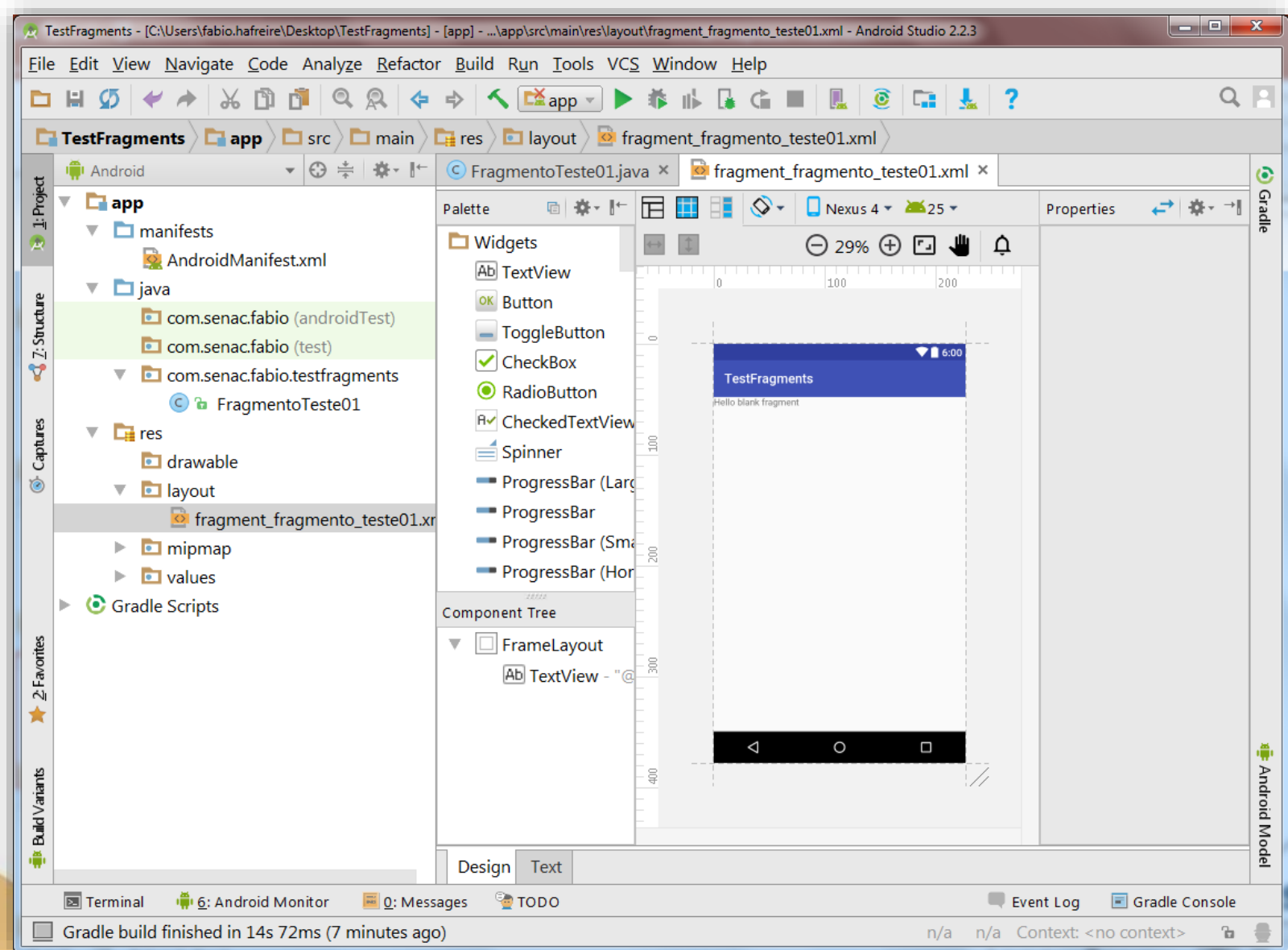
# Arquivos de Fragmentos

- Após a finalização do assistente de criação de fragmentos, serão criados dois novos arquivos:
  - A classe Java (“FragmentoTest01.java”) na pasta “java”;
  - O arquivo de layout do fragmento (“fragment\_fragmento\_teste01.xml”), na pasta “res/layout”

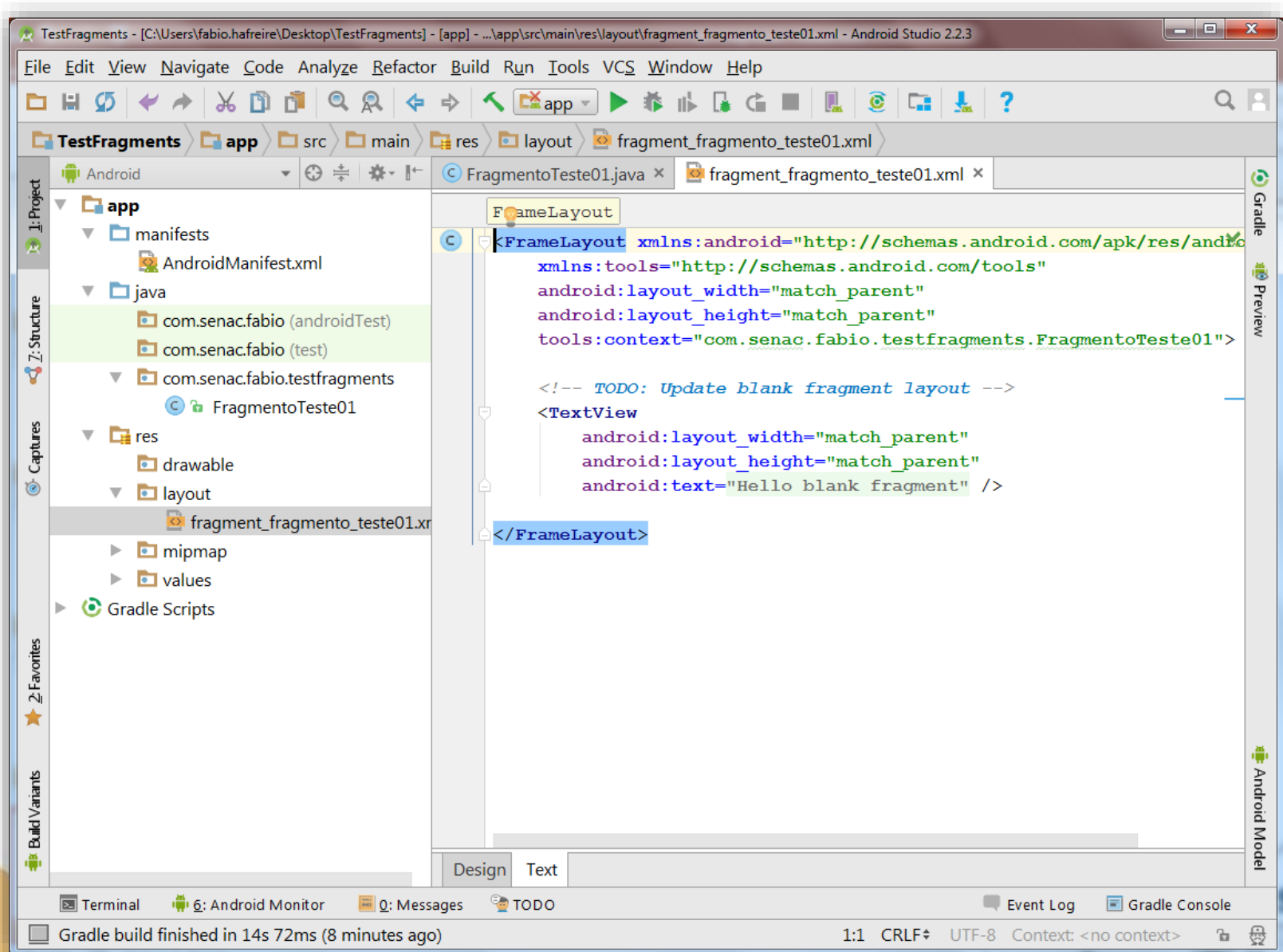
# Classe do Fragmento



# Layout do Fragmento - Design



# Layout do Fragmento - Texto

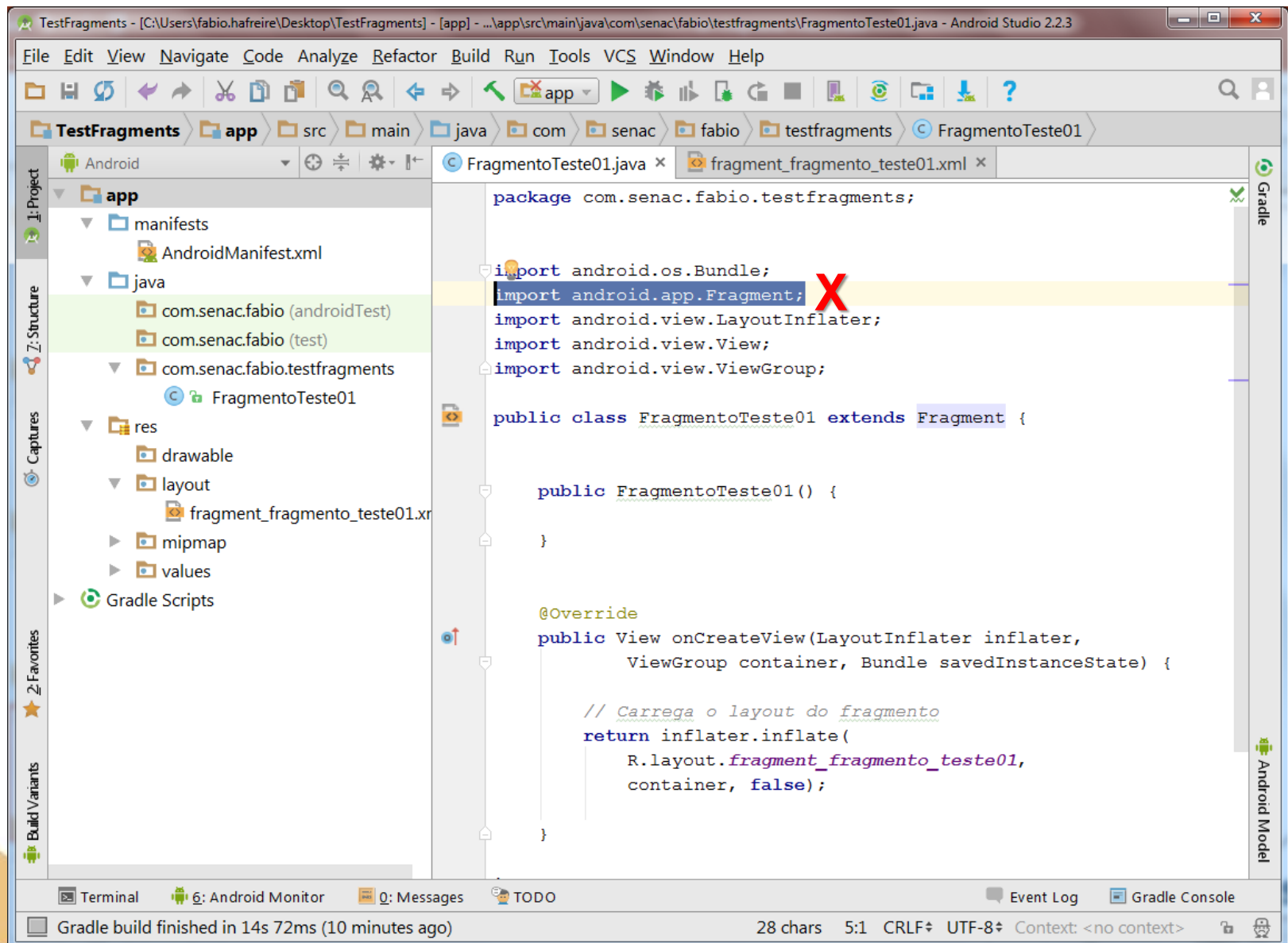


# Alterando a Extensão do Fragmento

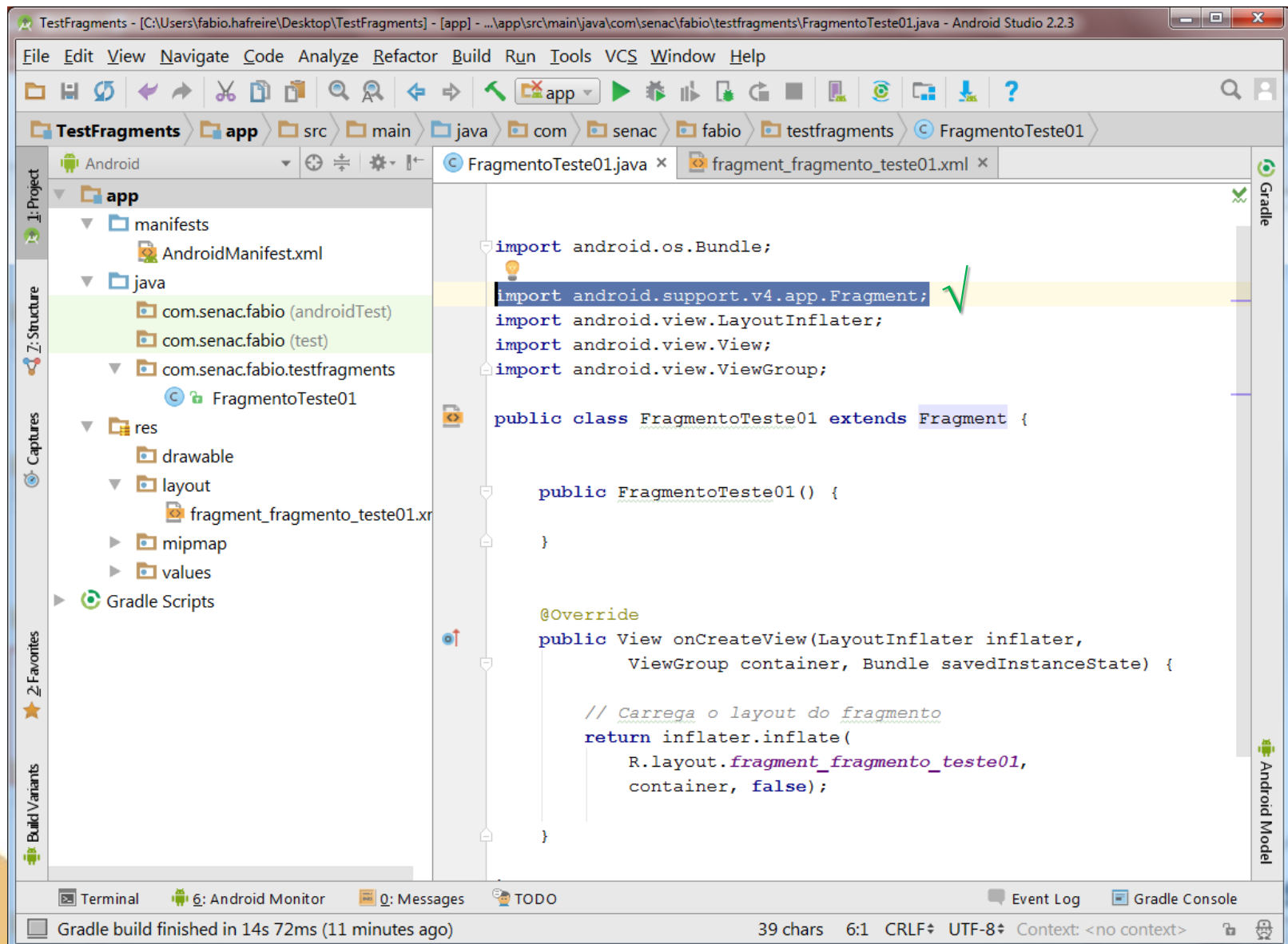
- Na classe do fragmento, ajuste o import do `Fragmento` para que estenda de `“android.support.v4.app.Fragment”` e não o padrão que é definido, `android.app.Fragment`;
- Isso evitará problemas de compatibilidade com versões antigas do Android.



# Alterando a Extensão do Fragmento



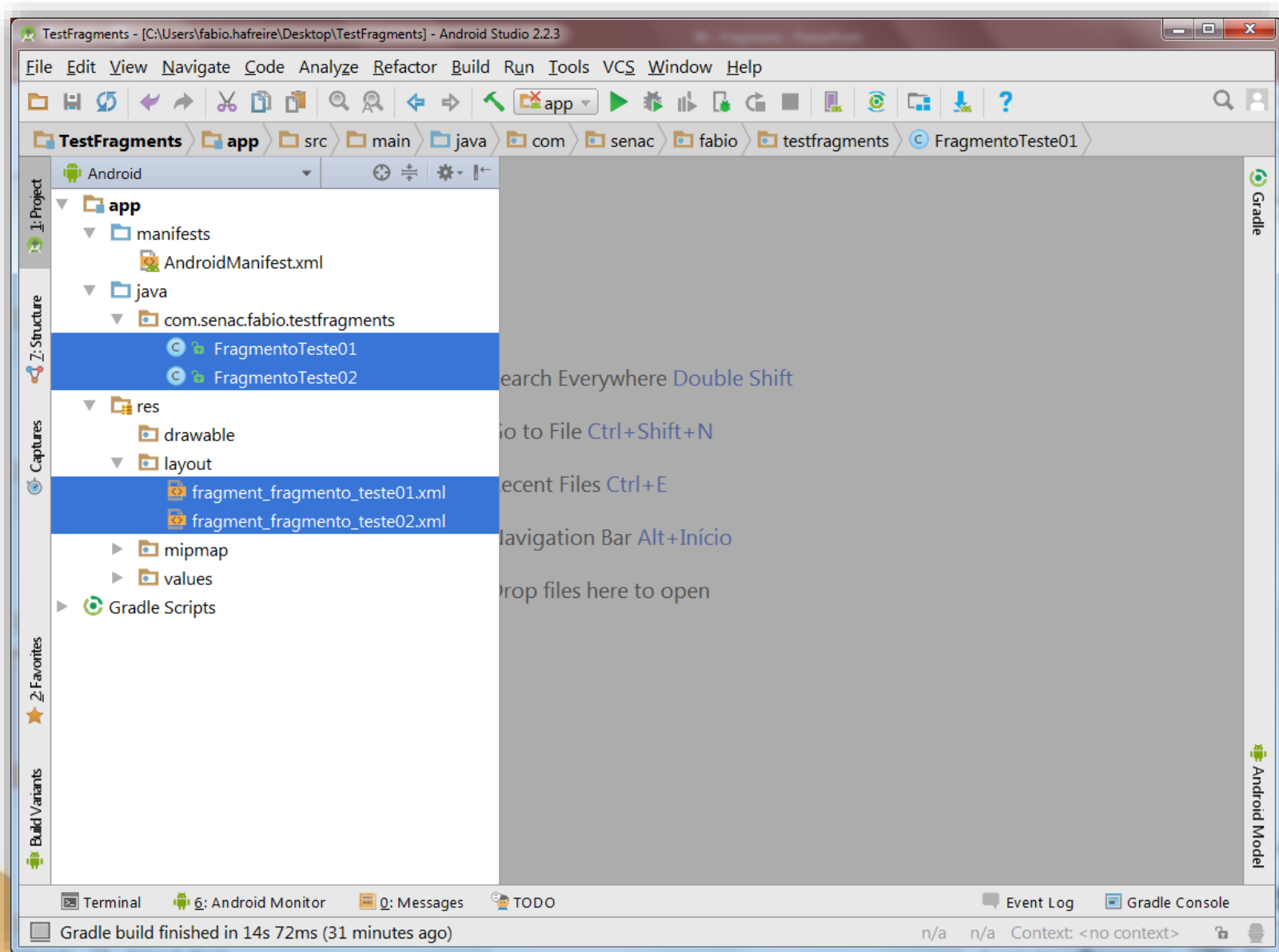
# Alterando a Extensão do Fragmento



# Criando um Segundo Fragmento

- Repita os mesmos procedimentos para criar um segundo fragmento. Chame-o de **“FragmentoTeste02”**;
- Todas as demais opções (com exceção do nome do layout) devem ser idênticas as do **“FragmentoTeste01”**

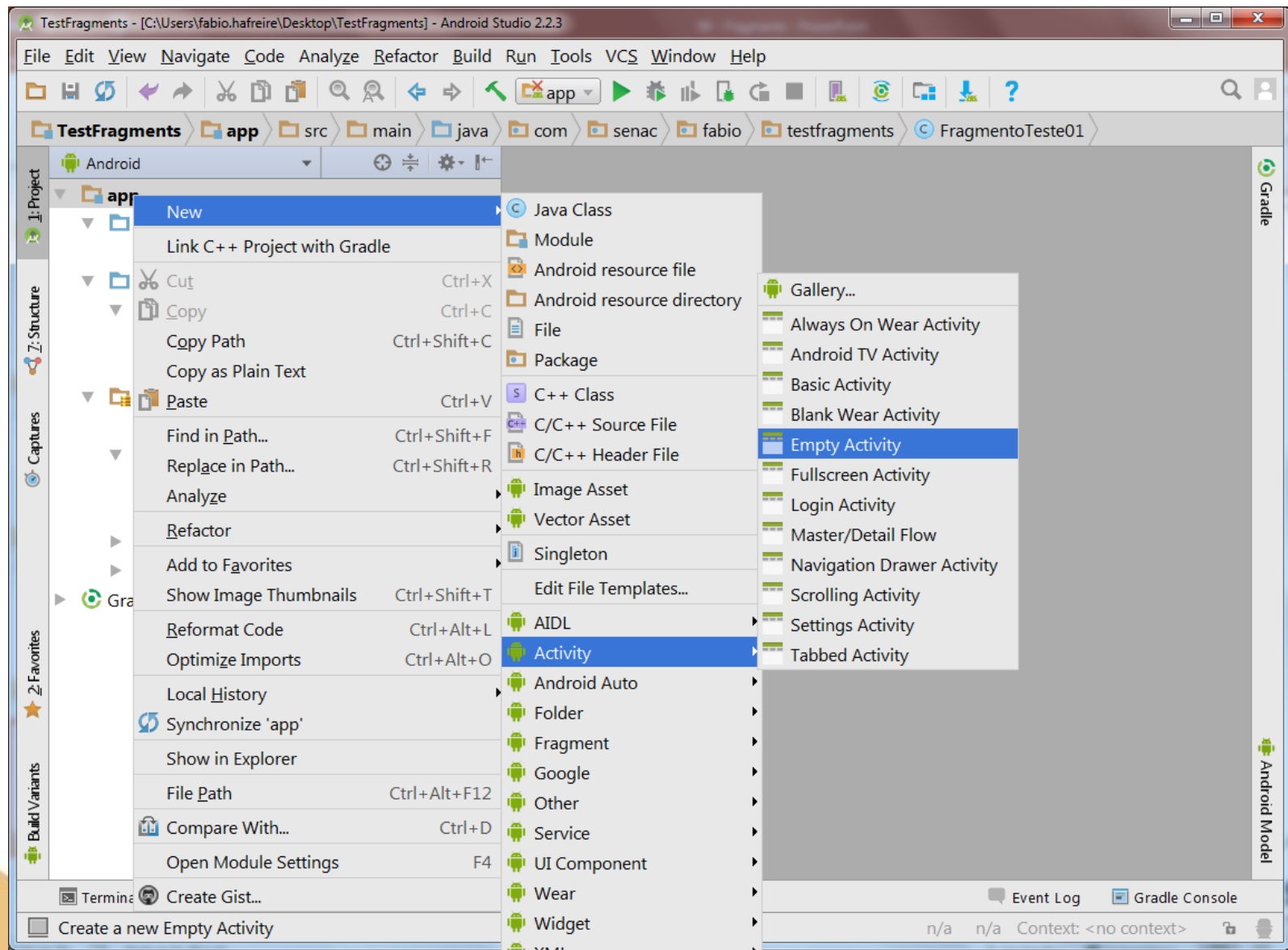
# Criando um Segundo Fragmento



# Atividade com Fragmento Estático

- Vamos criar uma atividade para exibir o “FragmentoTeste01” de forma estática;
- Para tanto, clique com o botão direito no projeto e selecione “New” > “Activity” > “Empty Activity”.

# Atividade com Fragmento Estático




# Atividade com Fragmento Estático

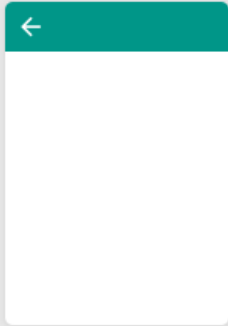
- Chame a atividade de “AtividadeFragEstatico”;
- Marque a opção “Launcher Activity”;
- Deixe as demais opções como estão;
- Clique em “Finish”.

# Atividade com Fragmento Estático

New Android Activity

 **Configure Activity**  
Android Studio

Creates a new empty activity



Activity Name:

☒ Generate Layout File

Layout Name:

☒ Launcher Activity

☒ Backwards Compatibility (AppCompat)

Package name:

Target Source Set:

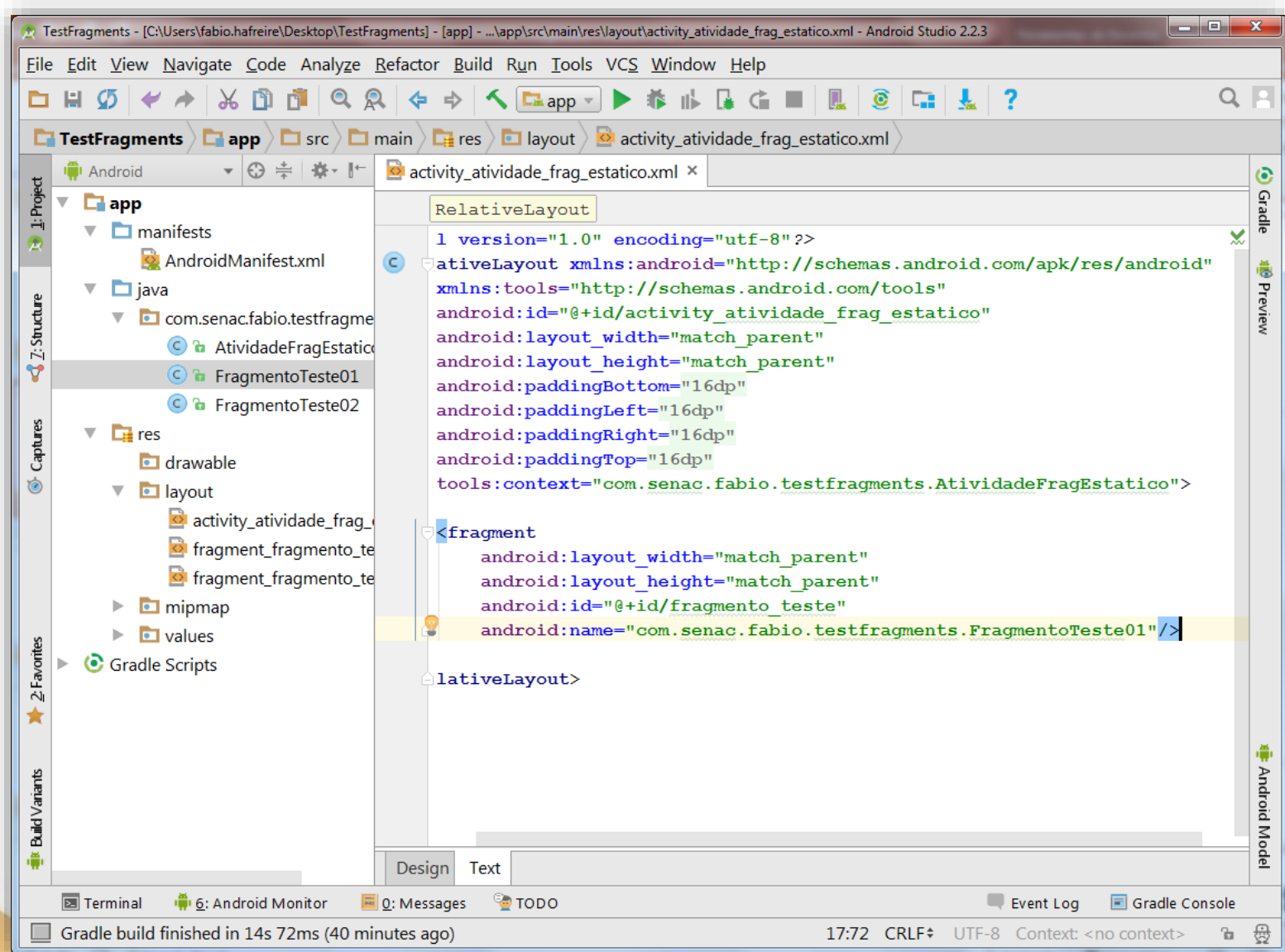
If true, this activity will have a CATEGORY\_LAUNCHER intent filter, making it visible in the launcher



# Atividade com Fragmento Estático

- No Layout da Atividade, insira um elemento do tipo fragment e posicione-o adequadamente;
- No elemento “android:name”, informe o nome completo da classe do fragmento “TesteFragmento01”, conforme o exemplo.

# Atividade com Fragmento Estático



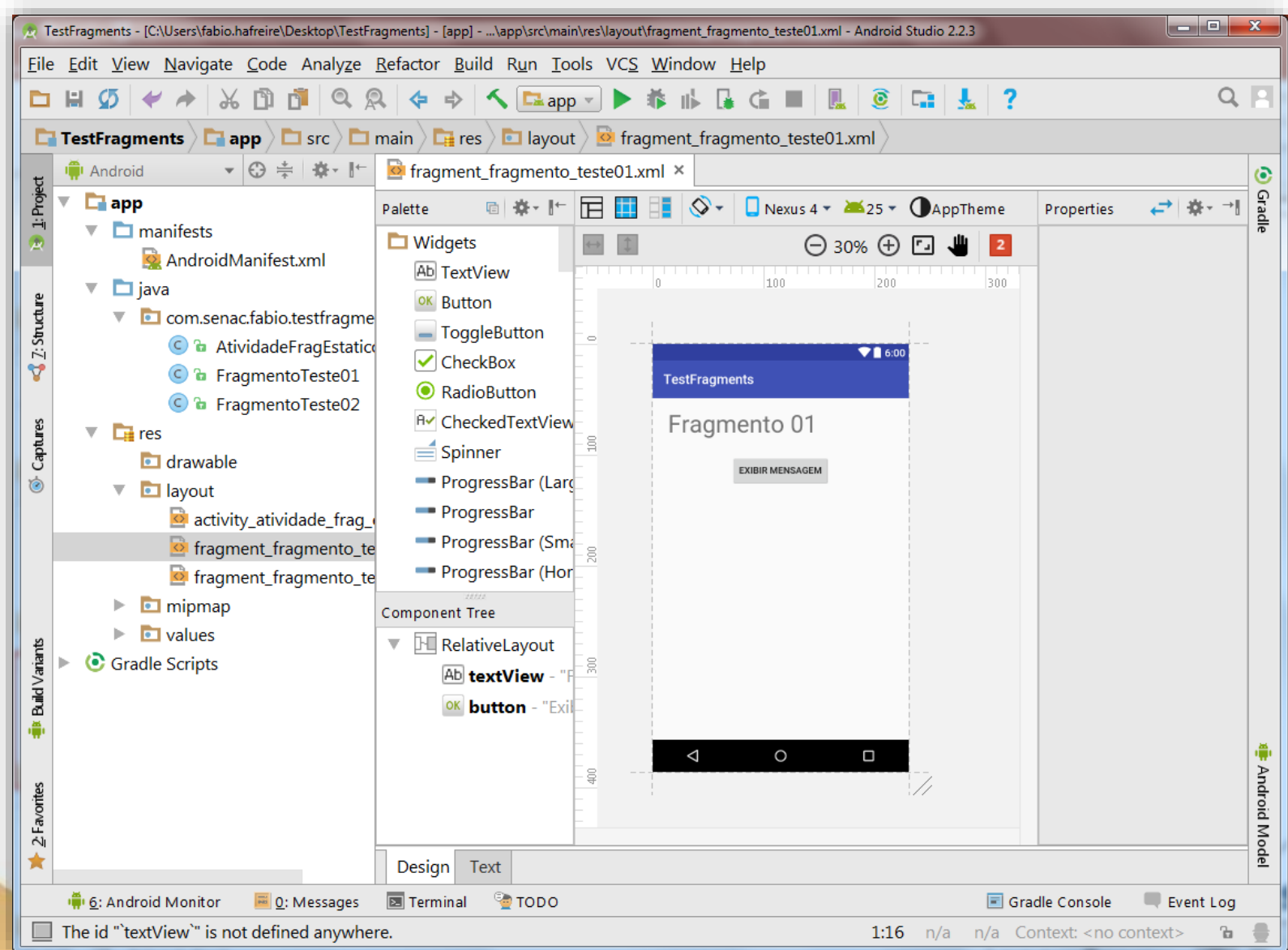
# Atividade com Fragmento Estático

```
<fragment
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/fragmento_teste"
    android:name="com.senac.fabio.testfragments.FragmentoTeste01"/>
```

# Customizando o Fragmento

- No Fragmento “TesteFragmento01”, adicione um TextView e um Button para identificar o fragmento e exibir uma mensagem, respectivamente;
- Se preciso, altere o tipo do Layout do Fragmento para “RelativeLayout”.

# Customizando o Fragmento



# Customizando o Fragmento

- Na classe do fragmento “TesteFragmento01”, adicione o código para fazer o binding do botão e um listener neste para exibir uma mensagem simples.

# Customizando o Fragmento

```
public View onCreateView(LayoutInflater inflater,
    ViewGroup container, Bundle savedInstanceState) {

    //Carrega o layout do fragmento
    View view = inflater.inflate(
        R.layout.fragment_fragmento_teste01,
        container, false);

    button = (Button) view.findViewById(R.id.button);
    View.OnClickListener listener = new View.OnClickListener() {
        public void onClick(View v) {
            AlertDialog.Builder builder =
                new AlertDialog.Builder(getActivity());
            builder.setTitle("Olá!");
            builder.setMessage("Um Olá vindo do fragmento 01!");
            builder.setPositiveButton("OK", null);
            AlertDialog dialog = builder.create();
            dialog.show();
        }
    };
    button.setOnClickListener(listener);

    return view;
}
```

# Execute a Atividade

- Execute a atividade para observar o uso do fragmento em seu interior. O botão deve funcionar normalmente.



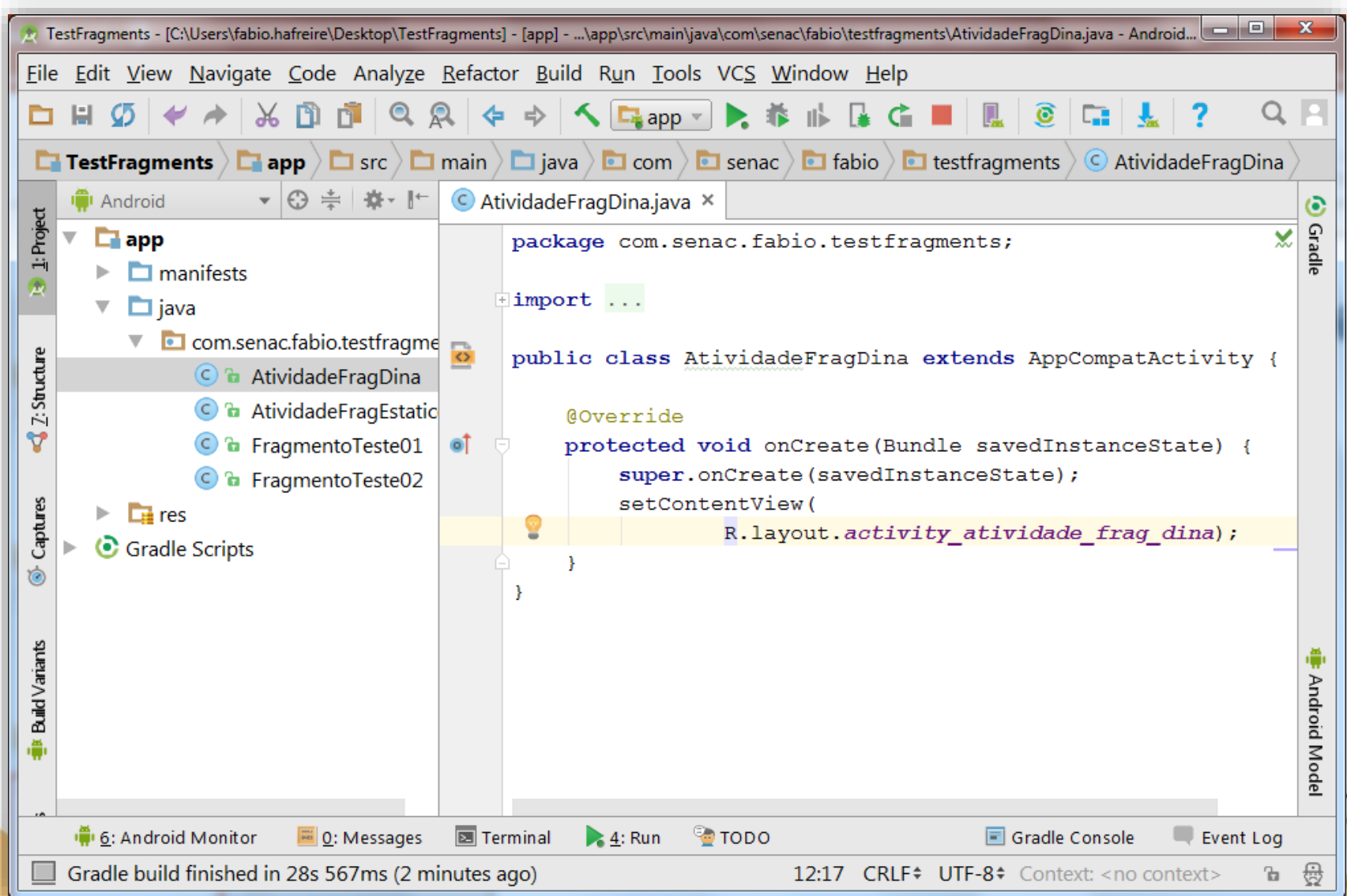
# Execute a Atividade



# Criando Atividade com Fragmento Dinâmico

- Crie outra atividade para testarmos o uso de fragmentos dinâmicos;
- Chame-a de “AtividadeFragDina” e marque-a como “Launcher Activity”.

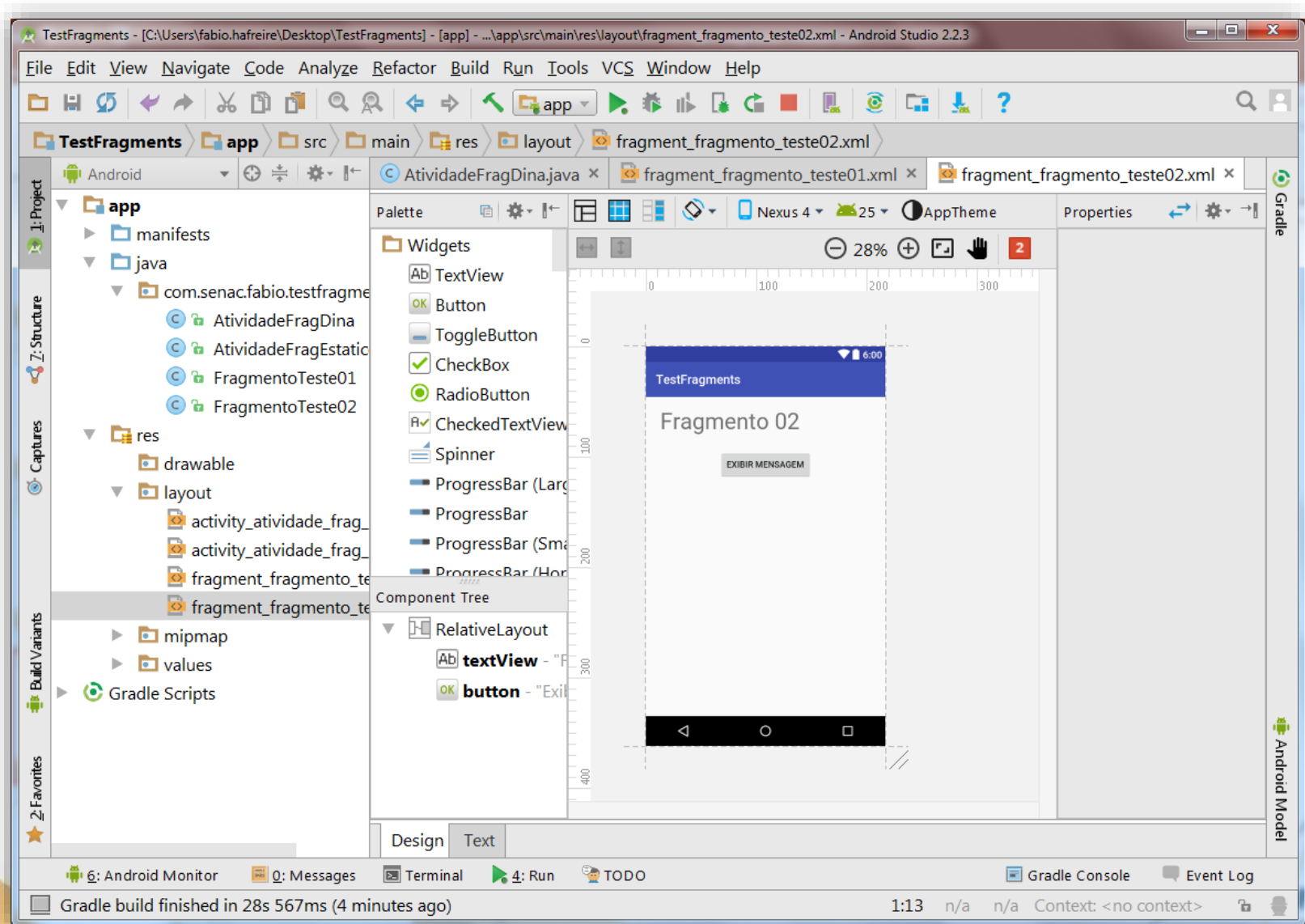
# Criando Atividade com Fragmento Dinâmico



## Definindo o Fragmento 02

- Repita o procedimento de definição do “TesteFragmento01” para o “TesteFragmento02”, criando seu layout e o listener para exibição da mensagem.

# Definindo o Fragmento 02



# Definindo o Fragmento 02

```
public View onCreateView(LayoutInflater inflater,
                        ViewGroup container, Bundle savedInstanceState)
{
    //Carrega o layout do fragmento
    View view = inflater.inflate(
        R.layout.fragment_fragmento_teste02, container, false);

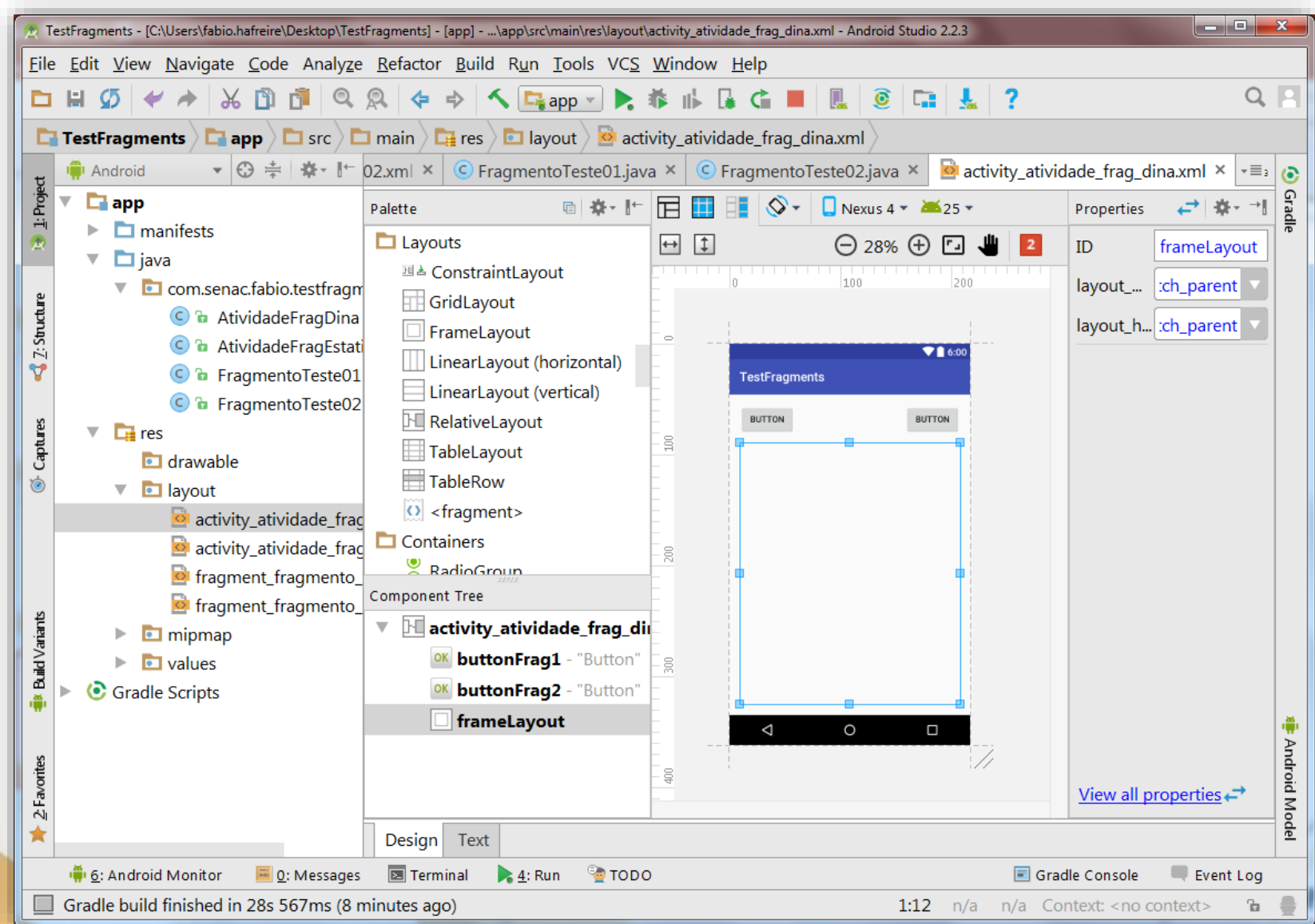
    button = (Button) view.findViewById(R.id.button);
    View.OnClickListener listener = new View.OnClickListener() {
        public void onClick(View v) {
            AlertDialog.Builder builder = new
AlertDialog.Builder(getActivity());
            builder.setTitle("Olá!");
            builder.setMessage("Um olá vindo do fragmento 02!");
            builder.setPositiveButton("OK", null);
            AlertDialog dialog = builder.create();
            dialog.show();
        }
    };
    button.setOnClickListener(listener);

    return view;
}
```

# Layout da AtividadeFragDina

- Defina o layout da atividade da seguinte forma:

# Layout da AtividadeFragDina





## Classe Java da AtividadeFragDina

- Na classe da atividade “AtividadeFragDina”, faça bindings dos botões e do fragmentos;
- Em seguida, defina dois listeners e associe-os, um com cada botão.

# Classe Java da AtividadeFragDina

```
public class AtividadeFragDina extends AppCompatActivity {  
    private Button buttonFrag1;  
    private Button buttonFrag2;  
    private FrameLayout frameLayout;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(  
            R.layout.activity_atividade_frag_dina);  
  
        buttonFrag1 = (Button) findViewById(R.id.buttonFrag1);  
        buttonFrag2 = (Button) findViewById(R.id.buttonFrag2);  
        frameLayout = (FrameLayout) findViewById(R.id.frameLayout);  
    }  
}
```

# Classe Java da AtividadeFragDina

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(
        R.layout.activity_atividade_frag_dina);

    buttonFrag1 = (Button) findViewById(R.id.buttonFrag1);
    buttonFrag2 = (Button) findViewById(R.id.buttonFrag2);
    frameLayout = (FrameLayout) findViewById(R.id.frameLayout);

    View.OnClickListener listenerButton01 = new View.OnClickListener() {
        public void onClick(View v) {

        }
    };

    View.OnClickListener listenerButton02 = new View.OnClickListener() {
        public void onClick(View v) {

        }
    };
}
```

## Classe Java da AtividadeFragDina

- Em cada listener, vamos utilizar o gestor de fragmentos para trocar o fragmento ativo quando os respectivos botões forem clicados;
- Use o método “getSupportFragmentManager” para obter o gestor de fragmentos.

# Listener do 1º Botão

```
View.OnClickListener listenerButton01 = new View.OnClickListener() {  
    public void onClick(View v) {  
        FragmentoTeste01 fragment = new FragmentoTeste01();  
        getSupportFragmentManager().beginTransaction()  
            .replace(R.id.frameLayout, fragment).commit();  
    }  
};  
buttonFrag1.setOnClickListener(listenerButton01);
```

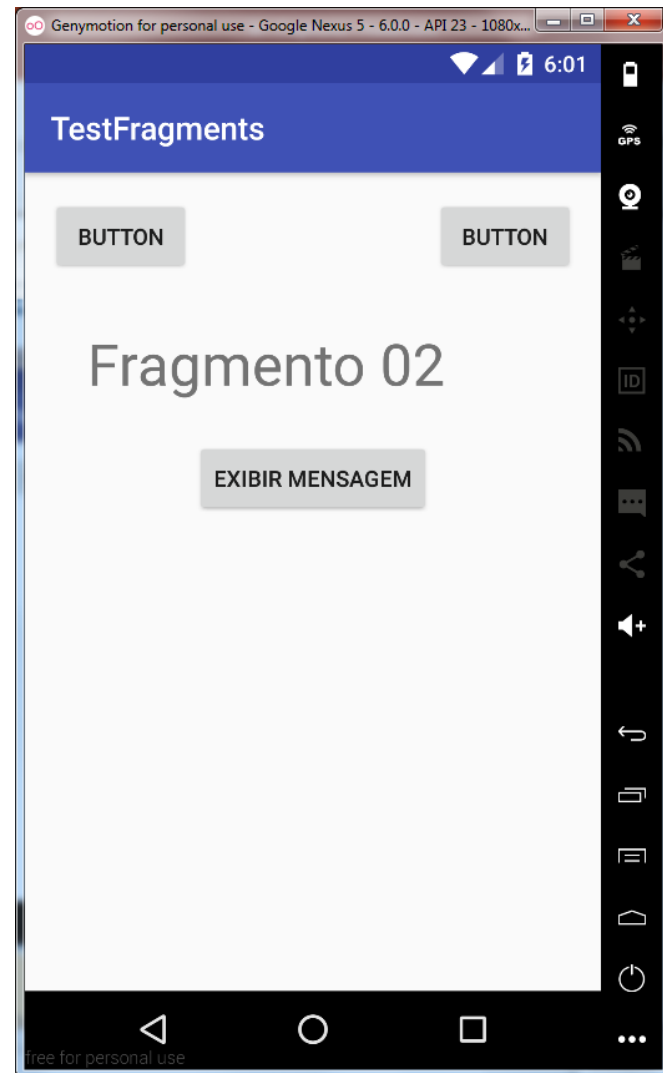
# Listener do 2º Botão

```
View.OnClickListener listenerButton02 = new View.OnClickListener() {  
    public void onClick(View v) {  
        FragmentoTeste02 fragment = new FragmentoTeste02();  
        getSupportFragmentManager().beginTransaction()  
            .replace(R.id.frameLayout, fragment).commit();  
    }  
};  
buttonFrag2.setOnClickListener(listenerButton02);
```

## Executando a AtividadeFragDina

- Execute a atividade “AtividadeFragDina” e clique nos botões para alternar entre as atividades.

# Executando a AtividadeFragDina





# Executando a AtividadeFragDina



# EXERCÍCIOS

# Exercícios

- Crie um novo projeto com uma atividade que simule um sistema de abas
- Para tanto, componha sua interface com pelo menos quatro botões na parte superior e uma área de fragmentos na parte inferior
- Ao serem clicados, os botões devem alternar os fragmentos para os elementos correspondentes

*"That's all Folks!"*