



PROGRAMAÇÃO PARA
DISPOSITIVOS MÓVEIS

MATERIAL DESIGN

PARTE I

Material Design

- É uma linguagem de design criada pelo Google;
- Baseada em “layout de cartões” (que surgiram no Google Now), o Material Design faz uso mais frequente de layouts baseados em grade, animações e transições responsivas, preenchimentos e efeitos de profundidade, como iluminação e sombras.

Material Design

- Disponível a partir da versão 5.0 do Android;
- O Google fornece bibliotecas para que aplicações baseadas em Material Design possam funcionar em versões anteriores do Android:
 - appcompat
 - design

Biblioteca “appcompat”

- Contém implementações próprias, independentes da versão do sistema operacional, para elementos padrão do Android baseados no Material Design, como atividades, fragmentos, menus e etc.

Biblioteca “design”

- Contém implementações de vários layouts, componentes visuais (views) e recursos para implementar elementos de design propostos pela guia do Material Design de forma mais simples;
- Sem esta biblioteca, a implementação de tais elementos seria a cargo do desenvolvedor.

Elementos das Bibliotecas de Suporte

- As bibliotecas de suporte “design” e “appcompat” trazem uma série de recursos visuais complexos prontos, como:
 - Navigation View (“menu sanduíche”);
 - Barras de Ferramentas e botão “Up”;
 - Tab Layout;
 - Snackbar;
 - Floating Action Button;
 - Etc

Elementos das Bibliotecas de Suporte

- Nesta aula, estudaremos dois dos principais elementos de suporte do material design, a navigation view (para exibição de menus laterais) e o botão “up” (utilizado para voltar de uma atividade “filha” para uma atividade “pai”).

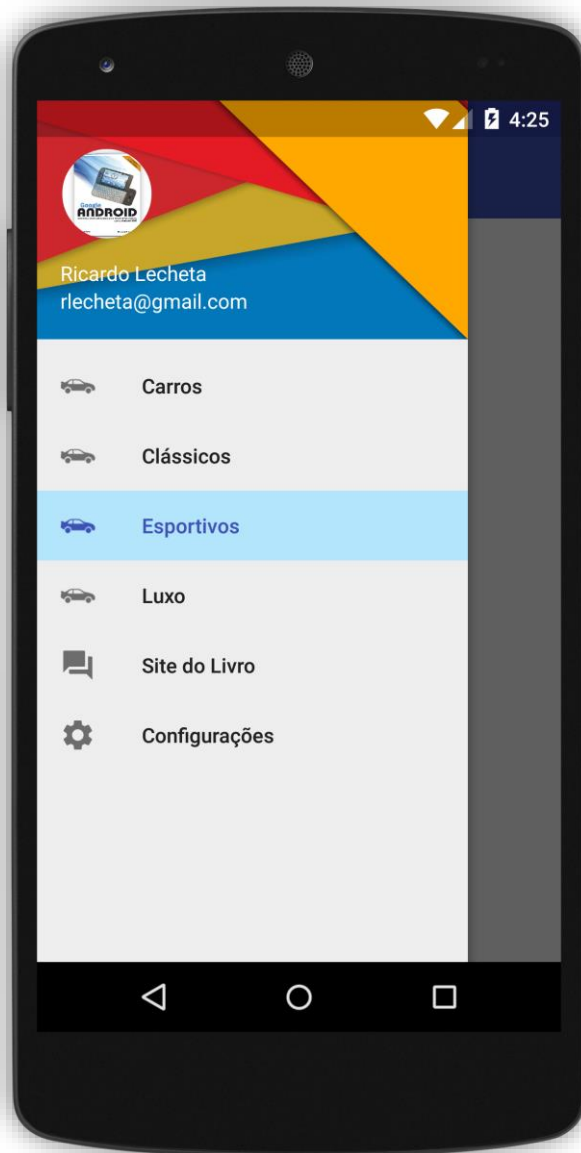
NAVIGATION VIEW (MENU SANDUÍCHE)



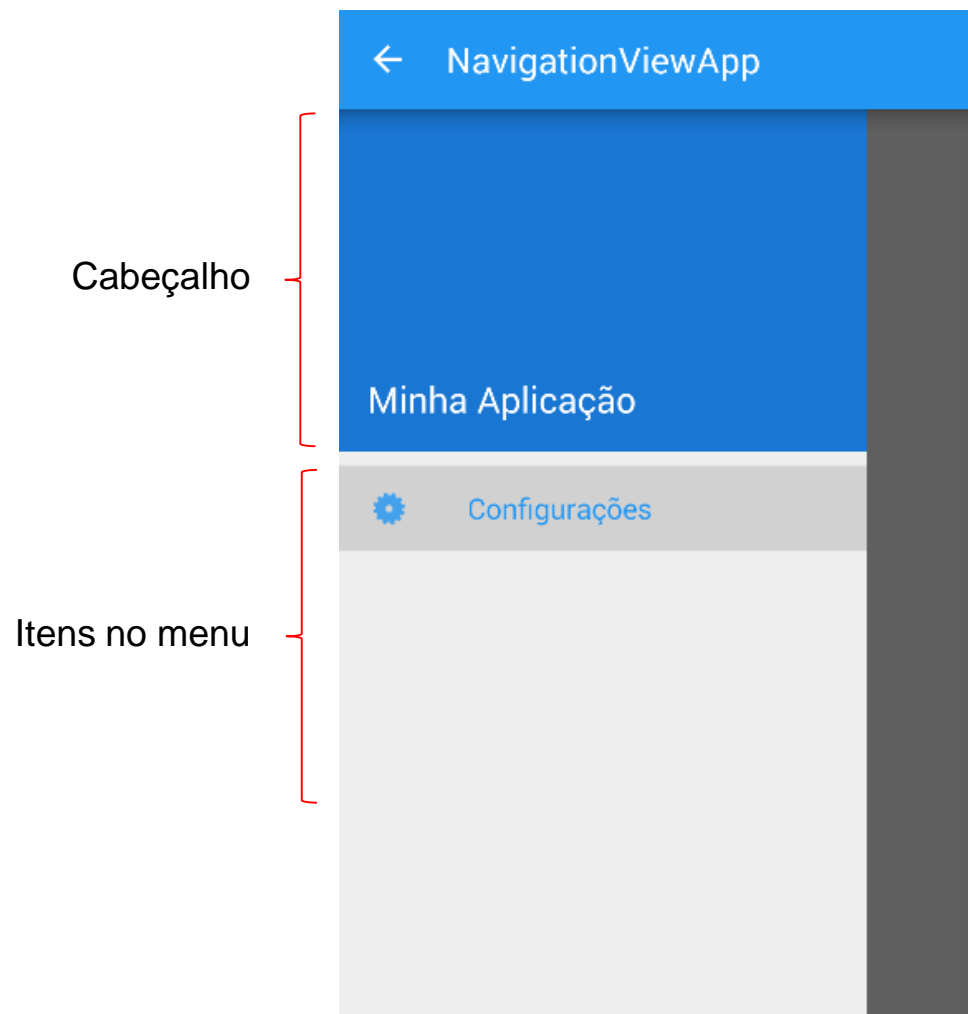
Navigation View

- A Navigation View corresponde a uma composição visual do Android baseada no Material Design;
- Utilizada para apresentação de uma tela com um menu lateral capaz de carregar múltiplos fragmentos no conteúdo da tela.

Navigation View



Navigation View



Navigation View

- O menu lateral permite navegar por itens e abri-los na view principal (como fragmentos);
- Para criar o menu lateral precisaremos de:
 - Um layout para o cabeçalho;
 - Um XML de menu;
 - Uma atividade que estenda “AppCompatActivity”.

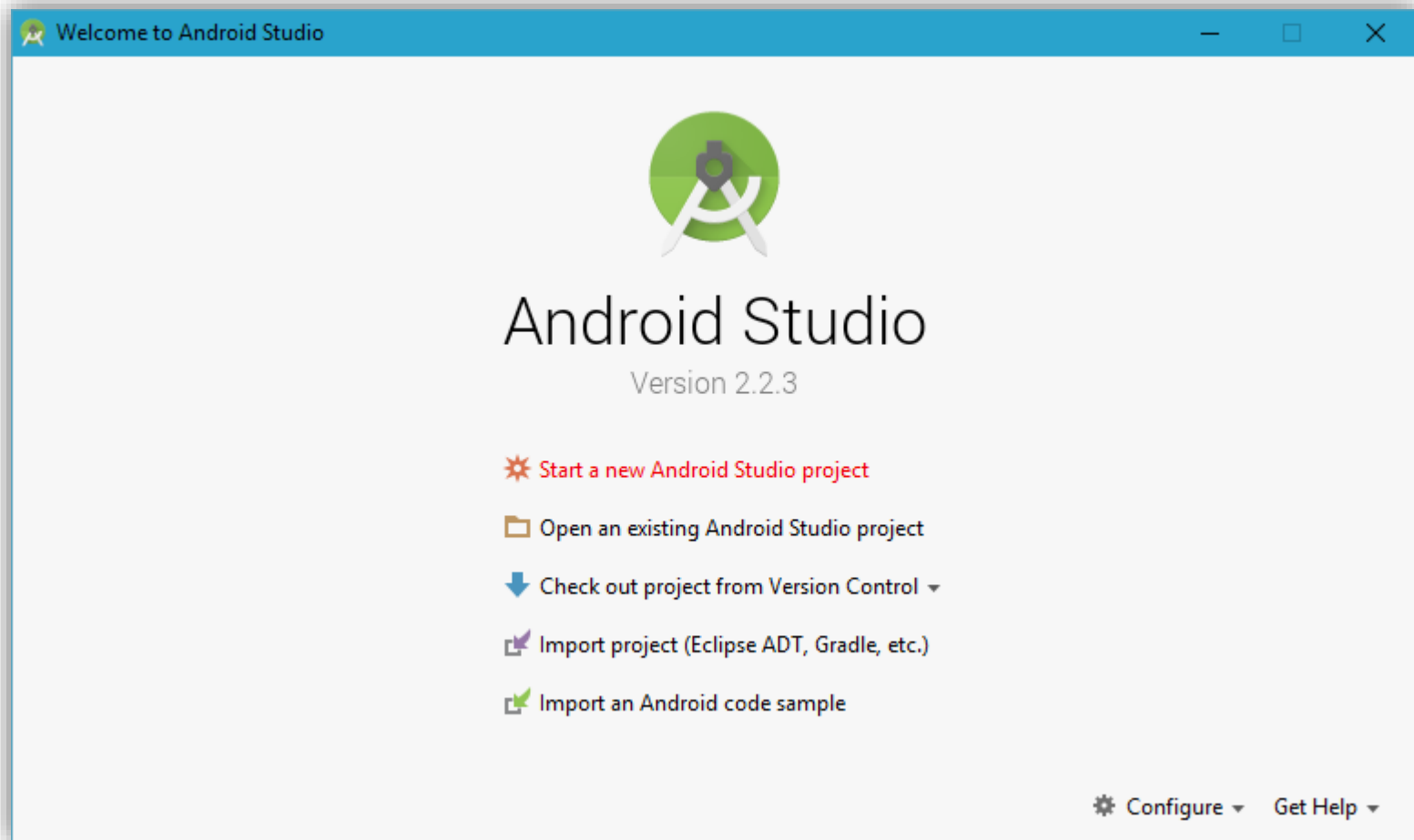
Criando Uma Navigation View

- Vamos criar um exemplo utilizando a navigation view;
- O exemplo será capaz de exibir um menu que, quando acionado, possibilite carregar um fragmento na tela principal;

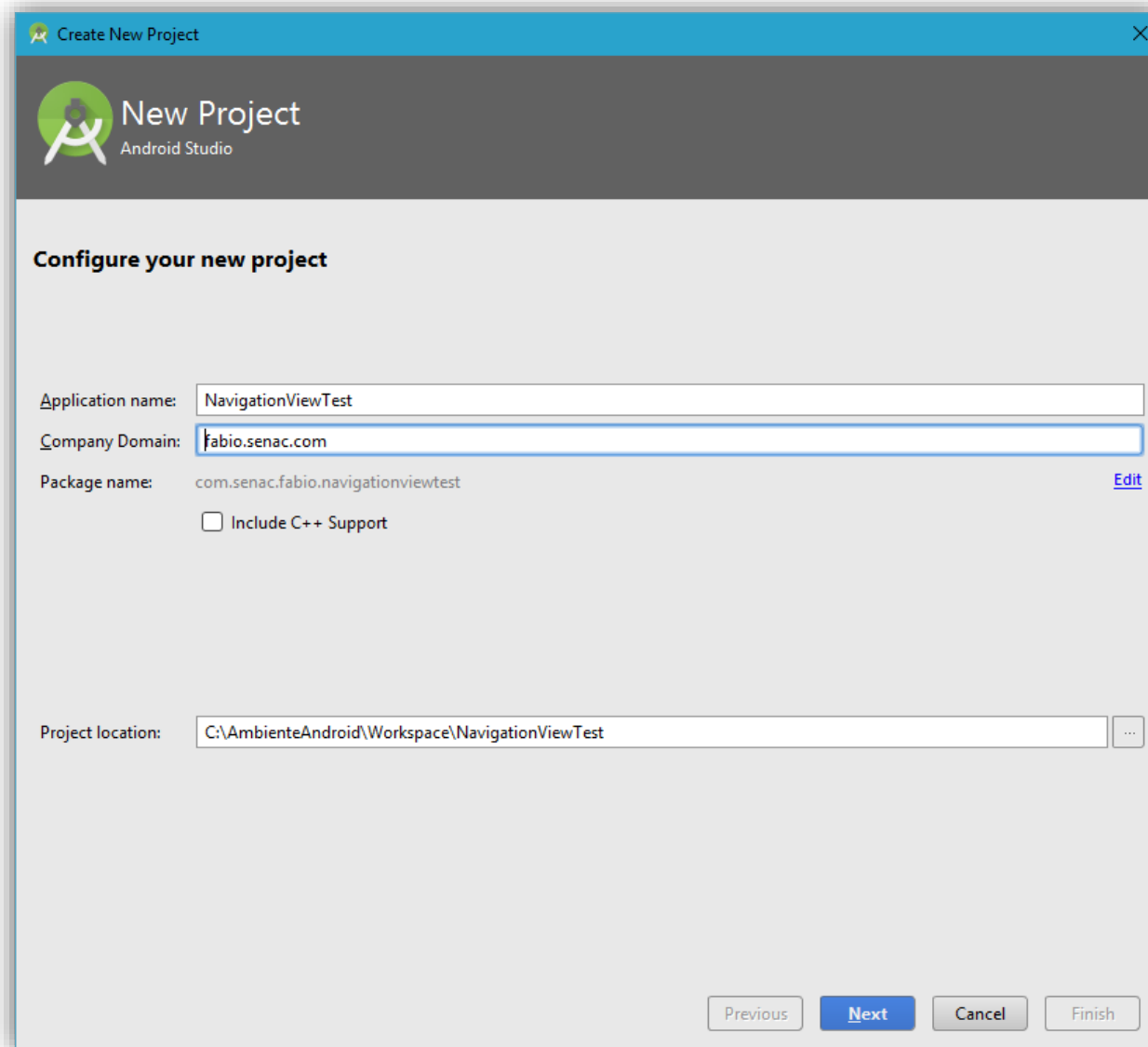
Criando um Projeto para Navigation View

- Vamos criar um novo projeto para utilizar o menu lateral;
- No Android Studio, crie um novo projeto com o nome “**NavigationViewTest**”. 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) ▼

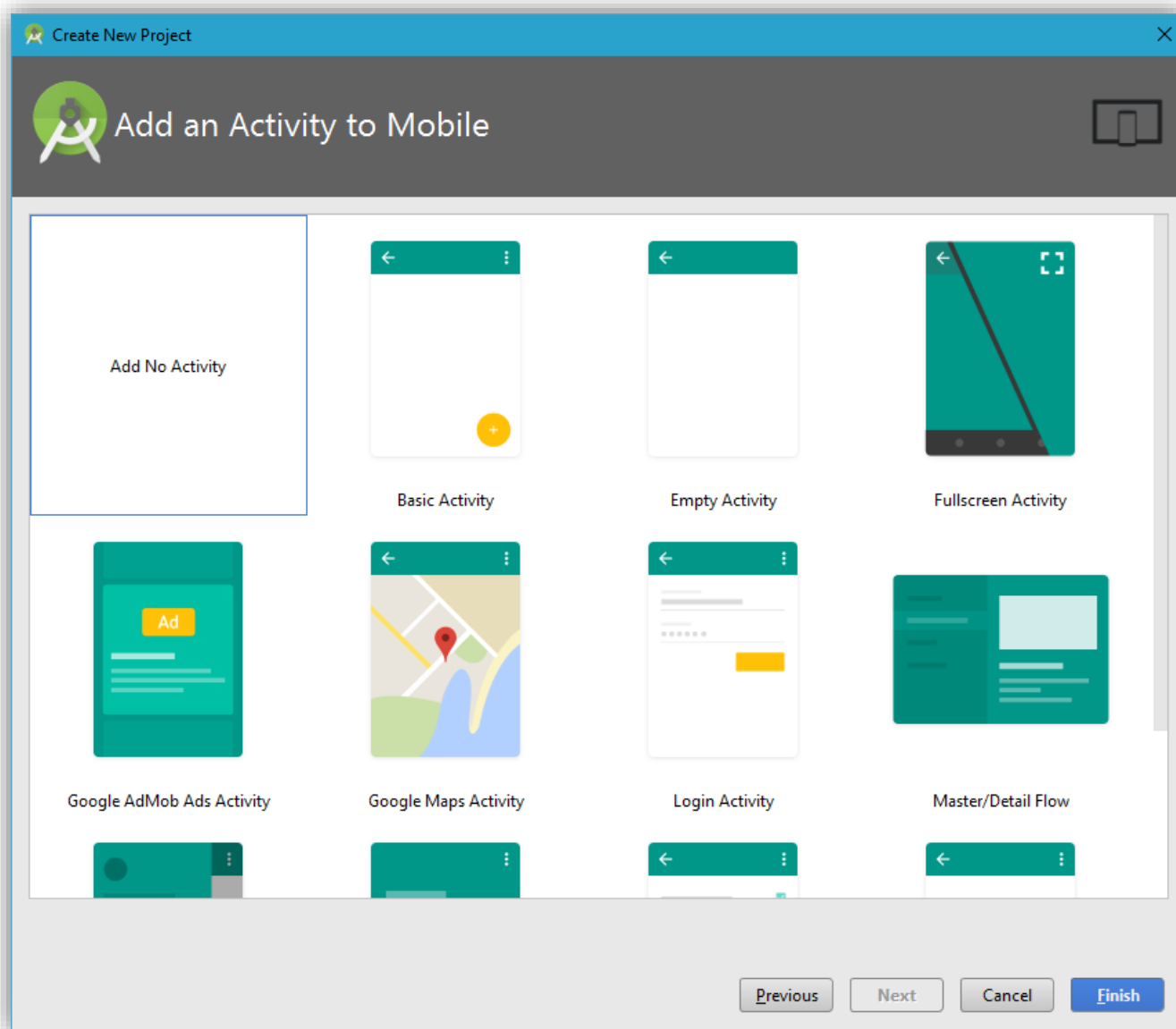
☐ Android Auto

☐ Glass (Not Available)

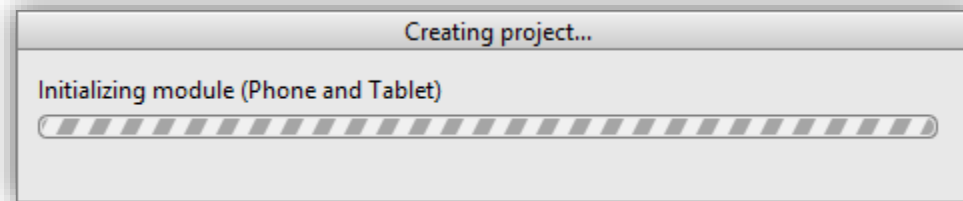
Minimum SDK ▼

Previous Next Cancel Finish

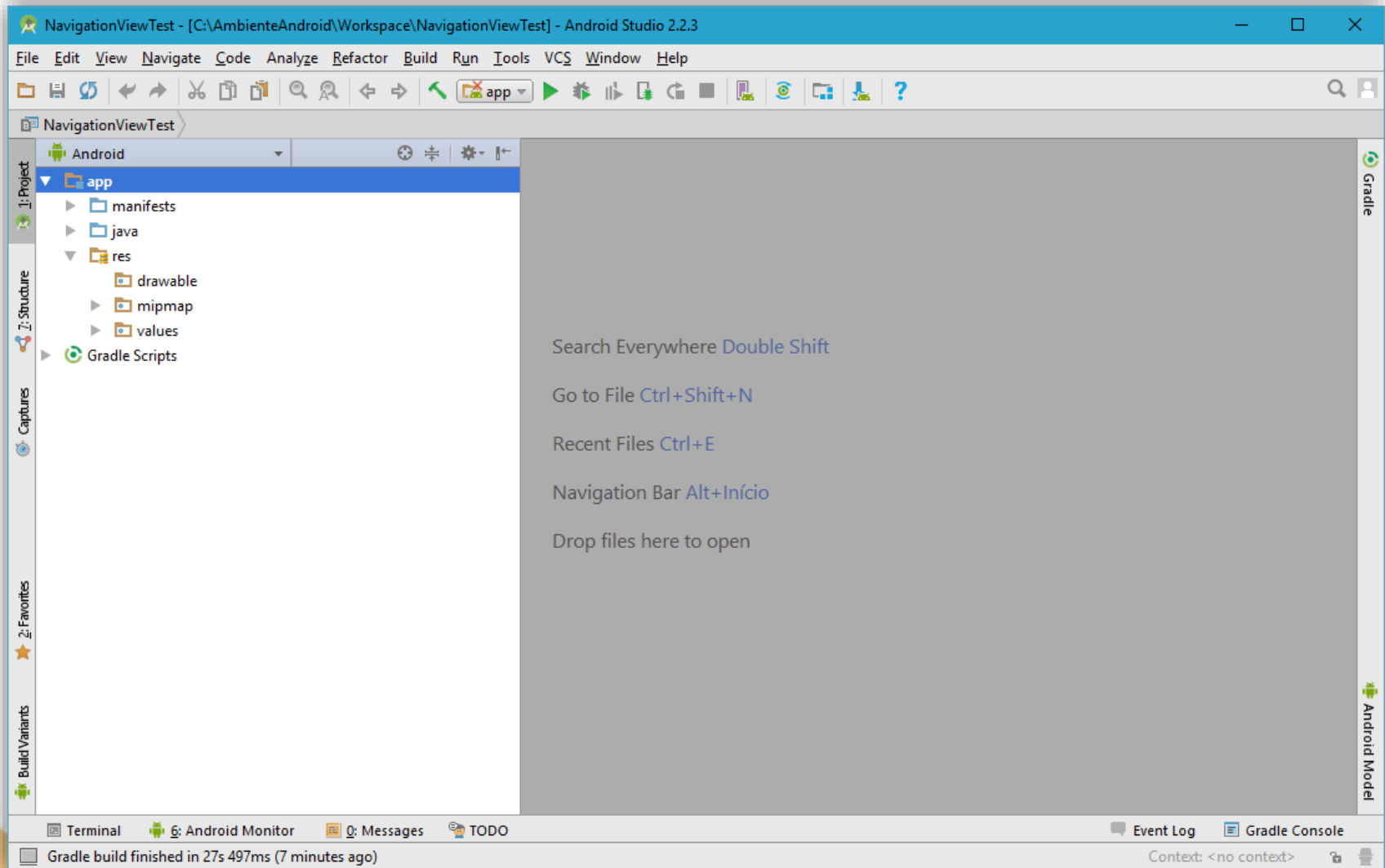
Criando um Projeto



Criando um Projeto



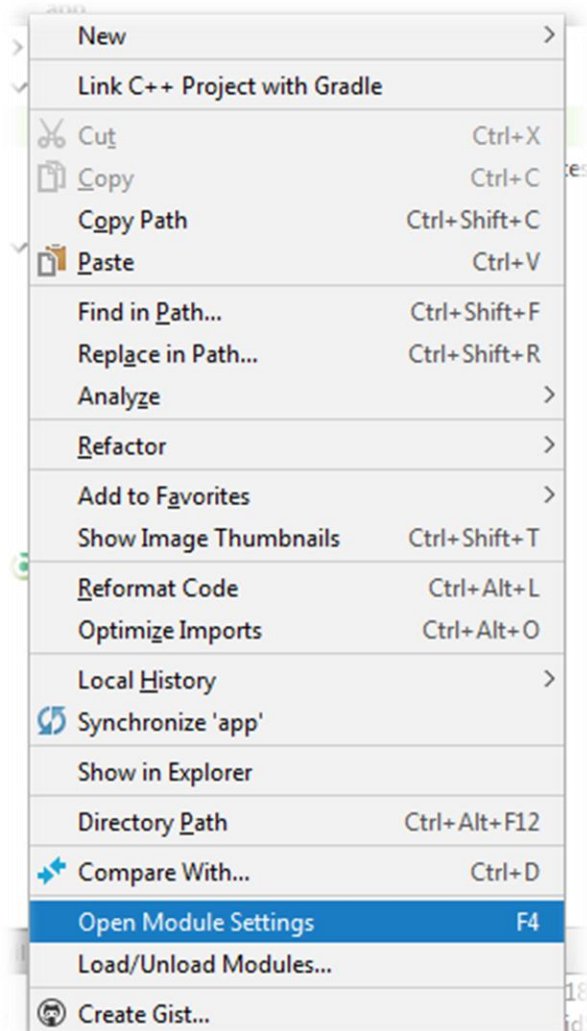
Criando um Projeto



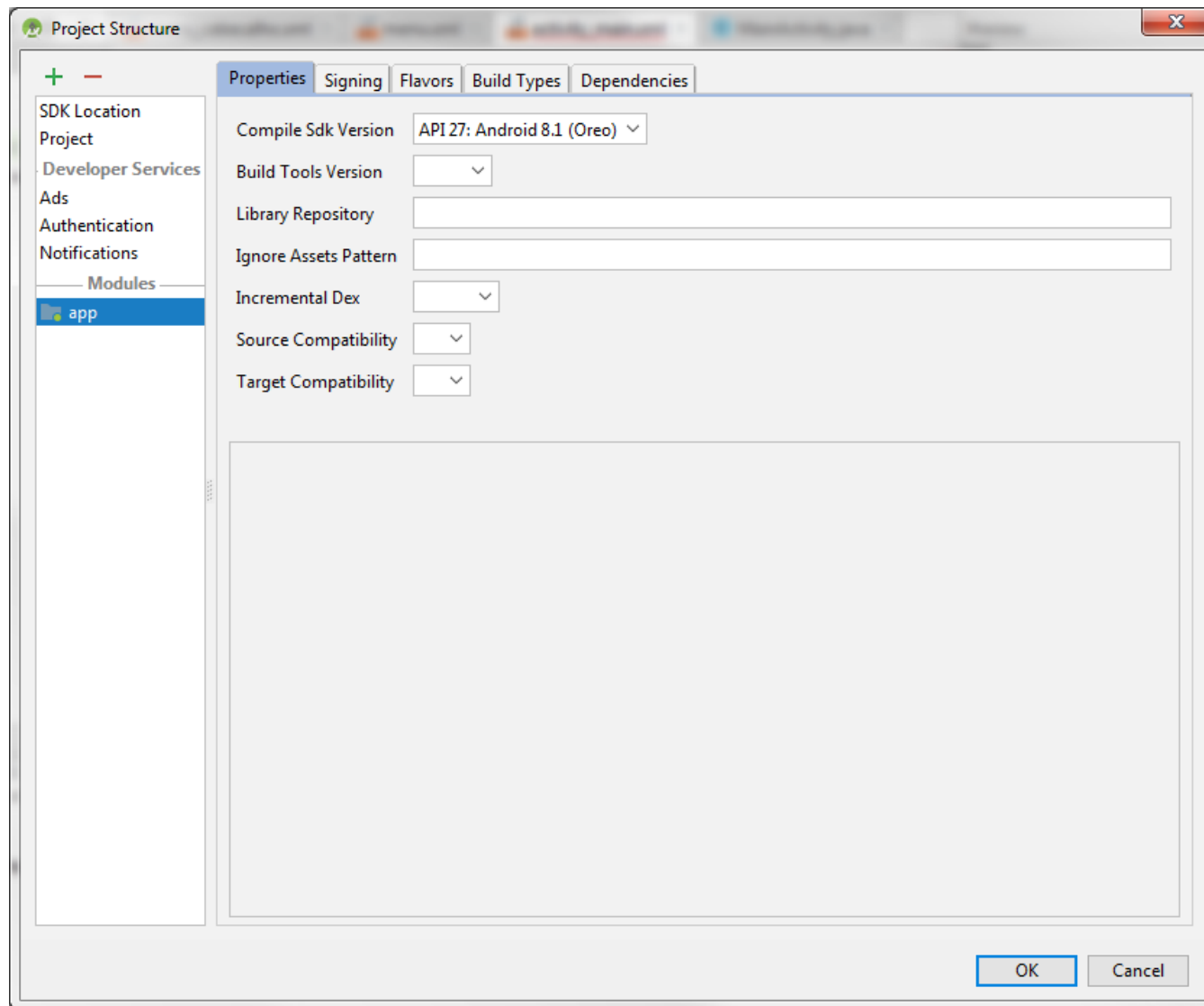
Suporte ao Material

- Primeiro, é necessário adicionar suporte aos componentes do material design na aplicação;
- Para isso, clique com o botão direito na aplicação e selecione “Open Module Settings”.

Open Module Settings



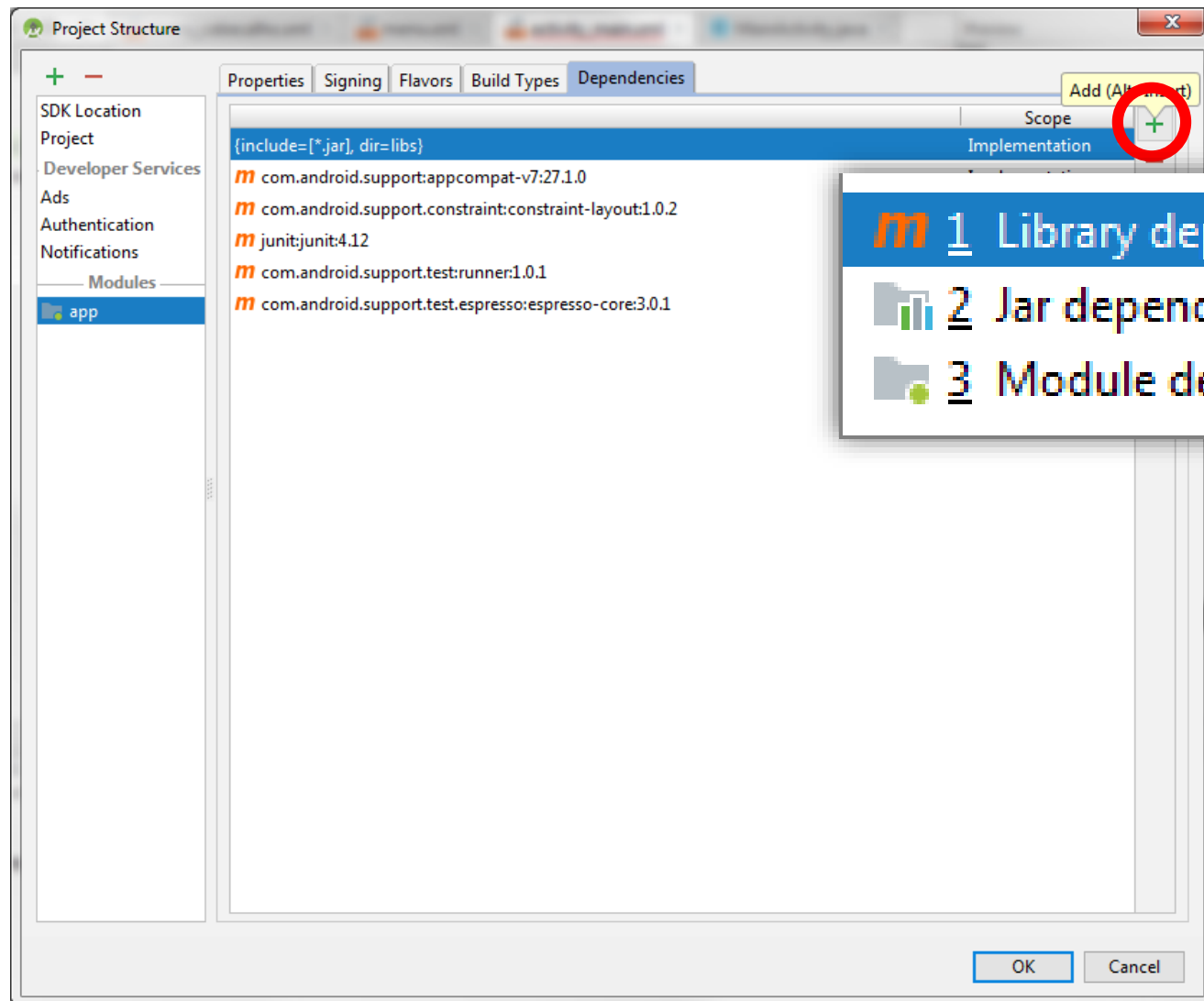
Module Settings






Adicionando Dependências

- Vá até o item “Dependencies” e clique no ícone de “+” (mais) para adicionar uma nova dependência ao projeto, selecionando “Library dependency” em seguida.

Dependencies / Add

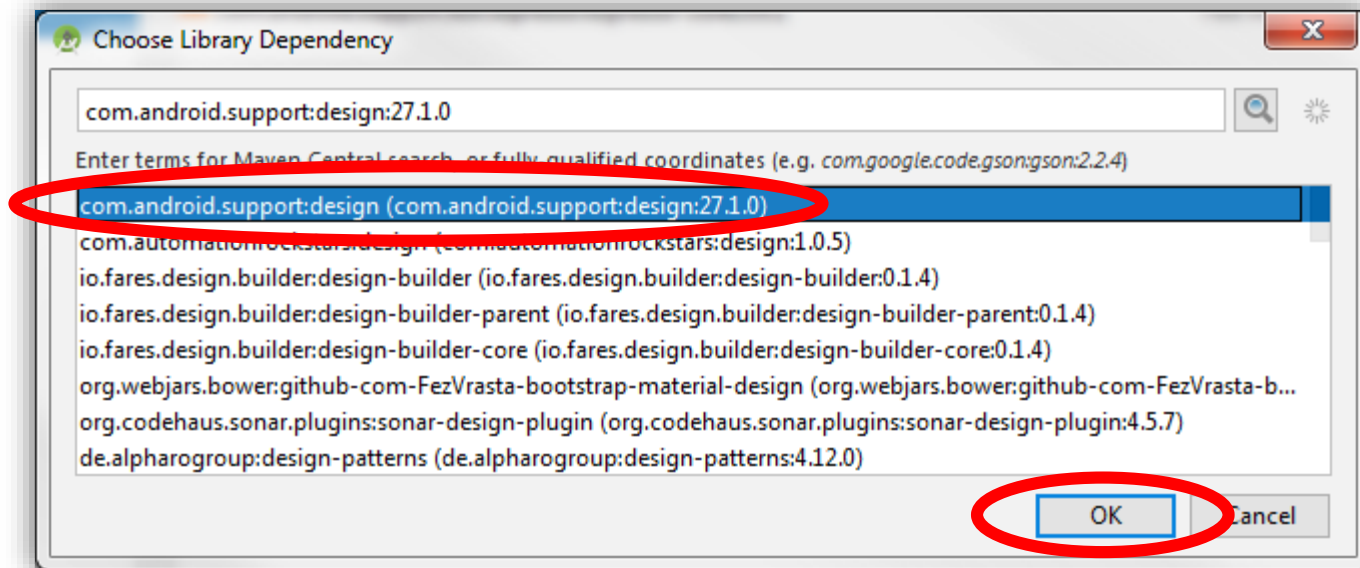


-  1 Library dependency
-  2 Jar dependency
-  3 Module dependency

Biblioteca do Material Design

- Na janela que surgir, digite “design”, pressione “Enter”, escolha o primeiro resultado, clique em “OK” na janela de pesquisa de dependências e em “OK” na janela de “Module Settings”.

Biblioteca do Material Design



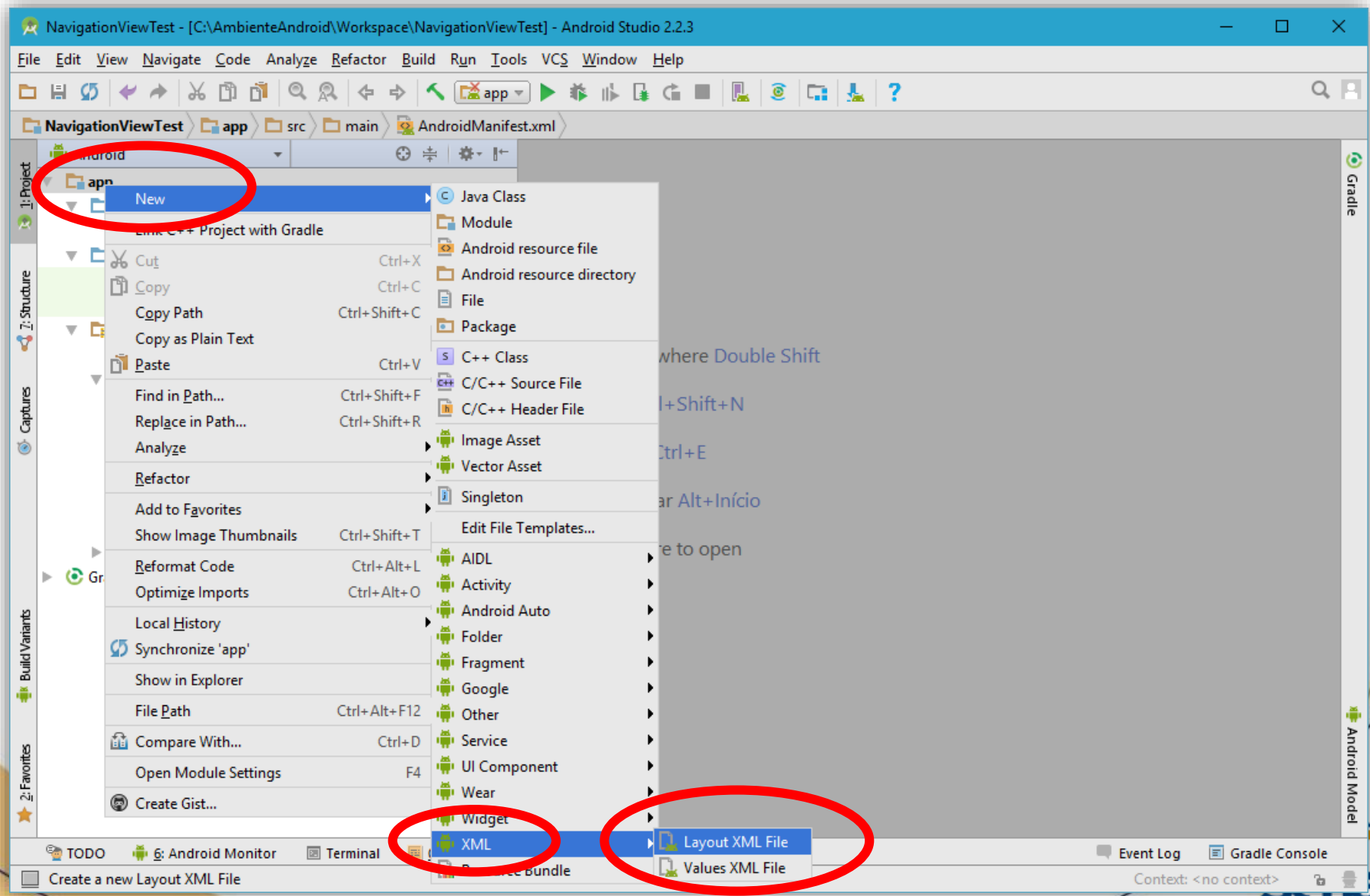
Layout do Cabeçalho

- Com o suporte ao material adicionado, é necessário criar o layout do cabeçalho do menu lateral;
- O layout pode conter um logotipo, apenas texto, ambos ou qualquer combinação de componentes de exibição suportada pelo Android;

Criando o Layout do Cabeçalho


- Para criar o novo item de layout que utilizaremos no cabeçalho, clique com o botão direito no projeto e escolha as opções “New” > “XML” > “XML Layout File”;
- Este comando abrirá o assistente de criação de layouts do Android Studio.

Criando o Layout do Cabeçalho



Assistente de Criação de Layouts

New Android Component

 **Configure Component**
Android Studio

Creates a new XML layout file.

Layout File Name:

Root Tag:

Target Source Set:


Name of the layout XML file.

Definindo o Layout do Cabeçalho

- Nomeie o layout do cabeçalho como “menu_cabecalho”, inserindo esta informação no campo “Layout File Name”;
- No caso do cabeçalho do menu, utilizaremos um layout simples, baseado no empilhamento de elementos. O “LinearLayout” é perfeito para isto, portanto, deixe este valor em “Root Tag”;
- Clique em “Finish” para criar o layout.

Definindo o Layout do Cabeçalho

New Android Component

 **Configure Component**
Android Studio

Creates a new XML layout file.

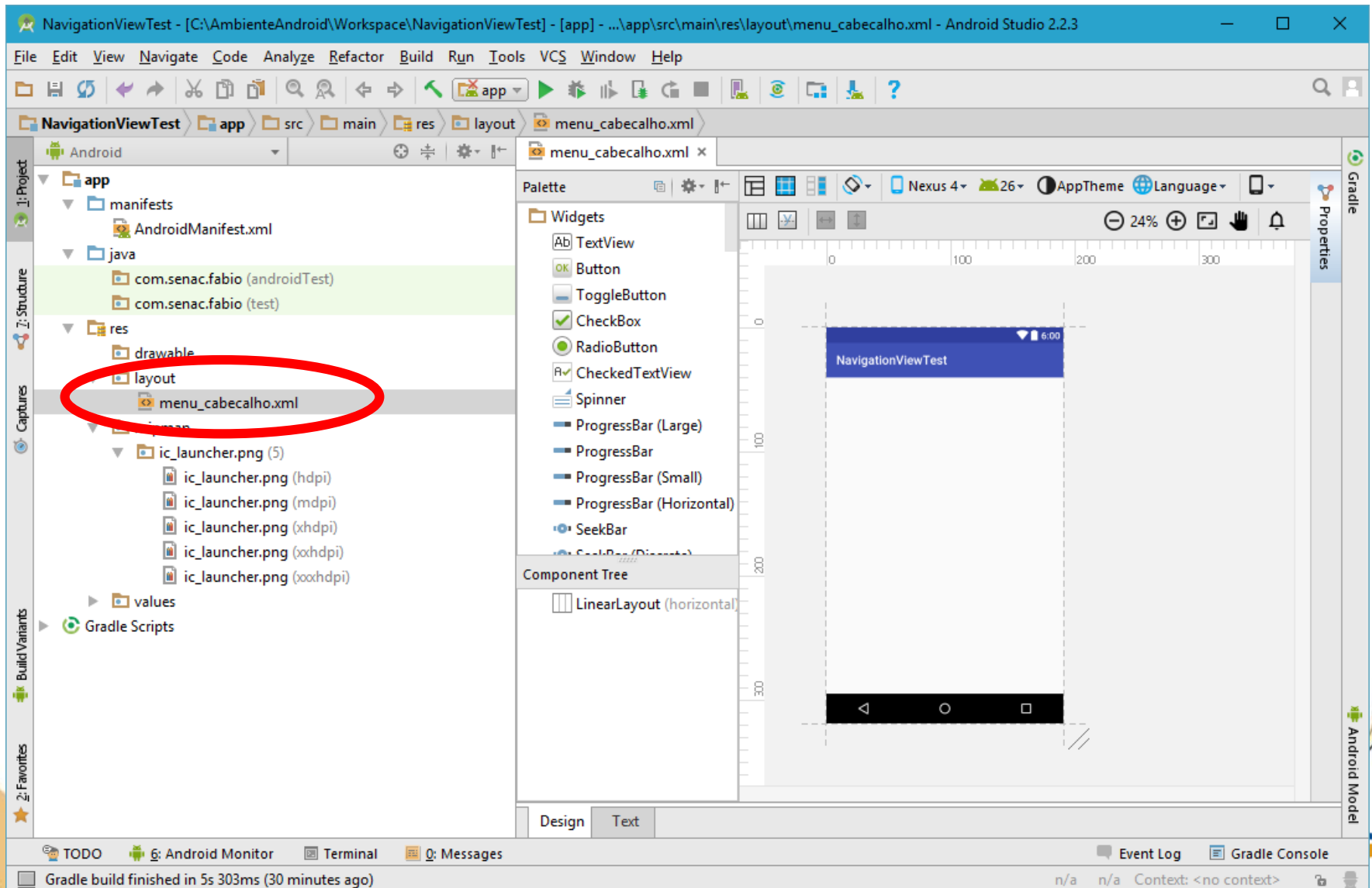
Layout File Name:

Root Tag:

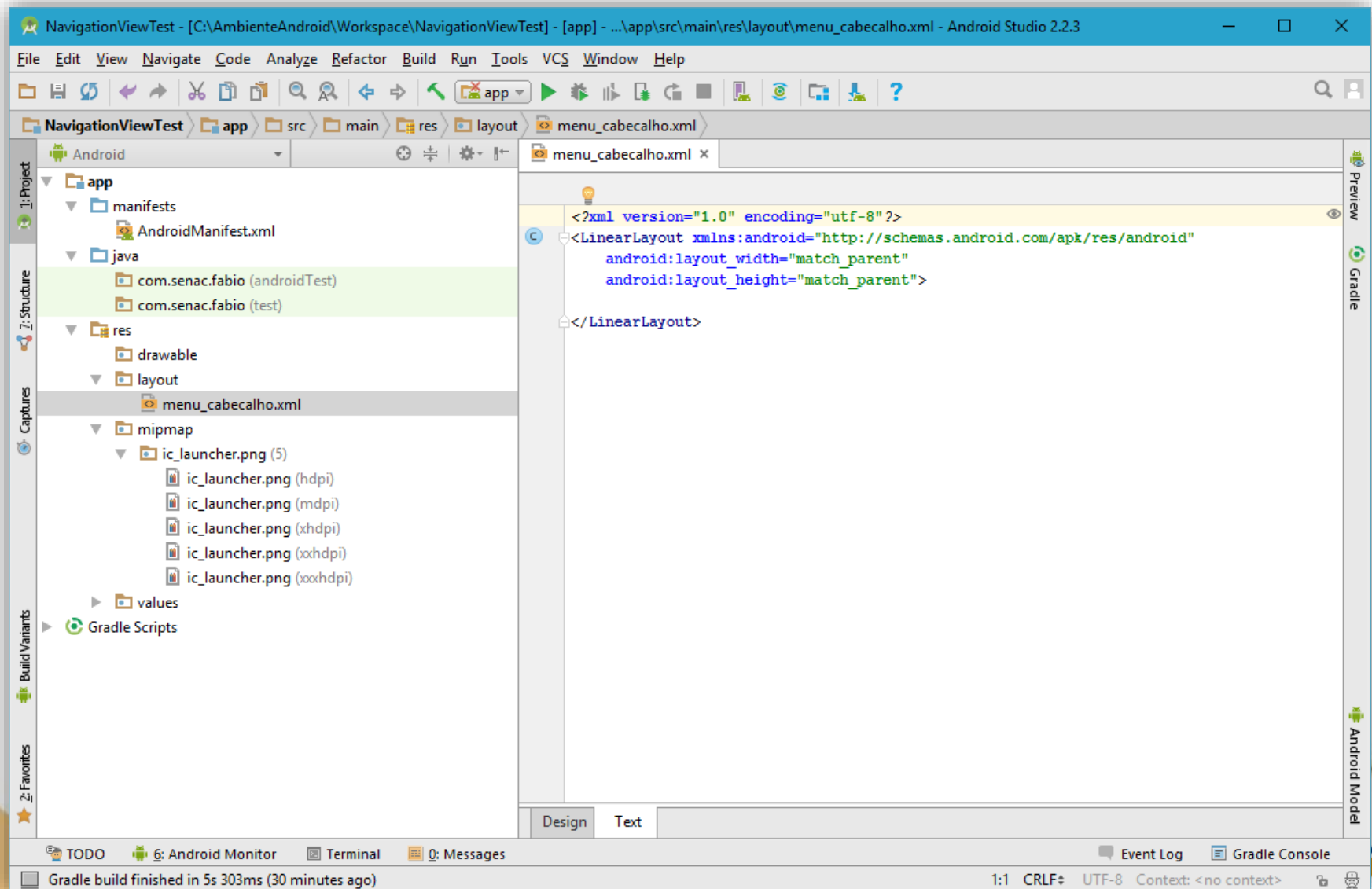
Target Source Set:

Name of the layout XML file.

Layout do Cabeçalho



Layout do Cabeçalho



Configurando o Layout do Cabeçalho

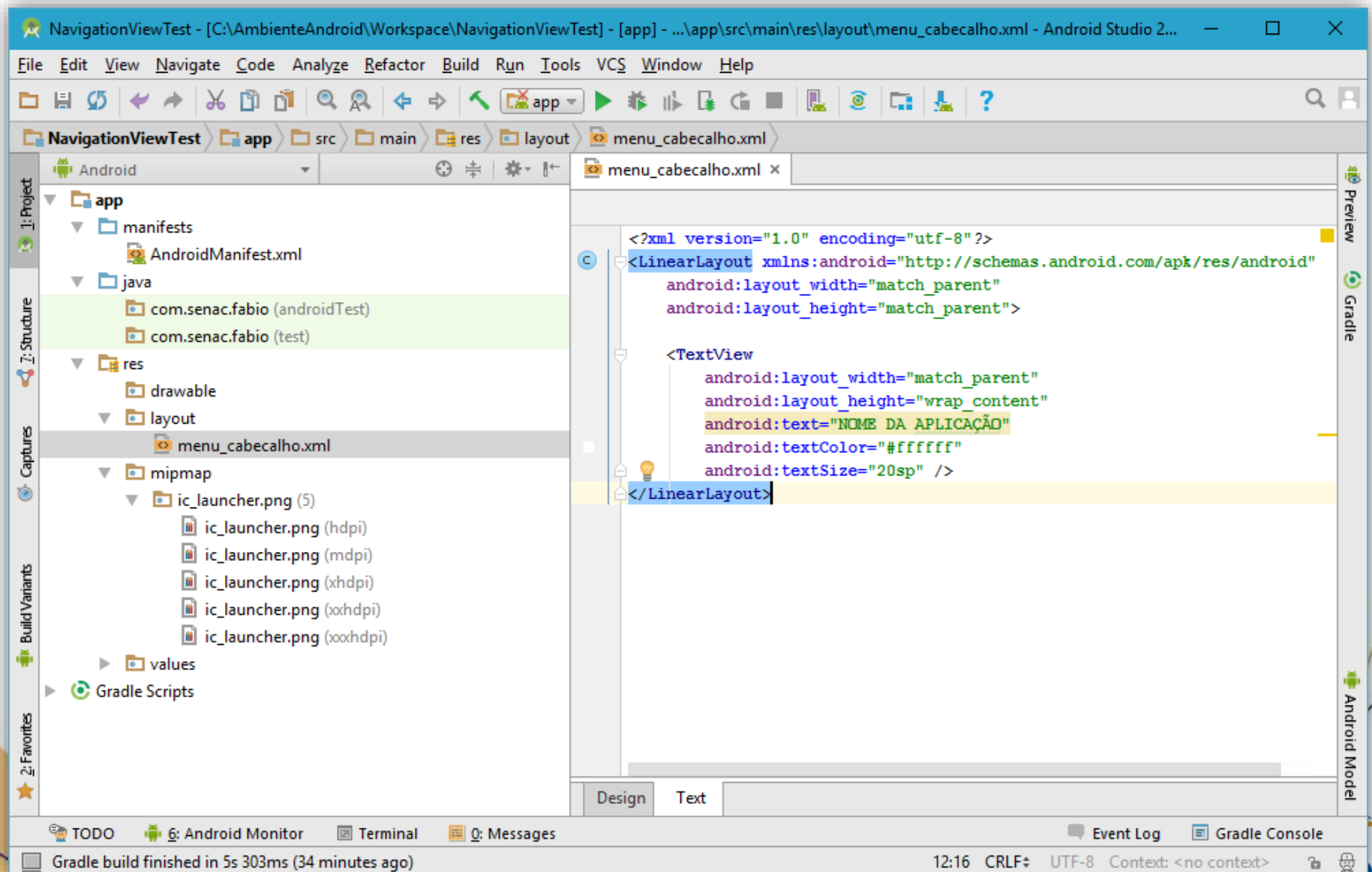
- Vamos configurar o layout do cabeçalho de forma bem simples, apenas inserindo uma view que ajude a identificar a aplicação;
- Caso desejado, outros elementos, como uma imagem ou outras views de auxílio visual, podem ser utilizados.

Configurando o Layout do Cabeçalho

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="192dp"
    android:background="@android:color/holo_blue_dark"
    android:gravity="bottom"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="APLICAÇÃO"
        android:textColor="#ffffff"
        android:textSize="20sp" />
</LinearLayout>
```

Configurando o Layout do Cabeçalho



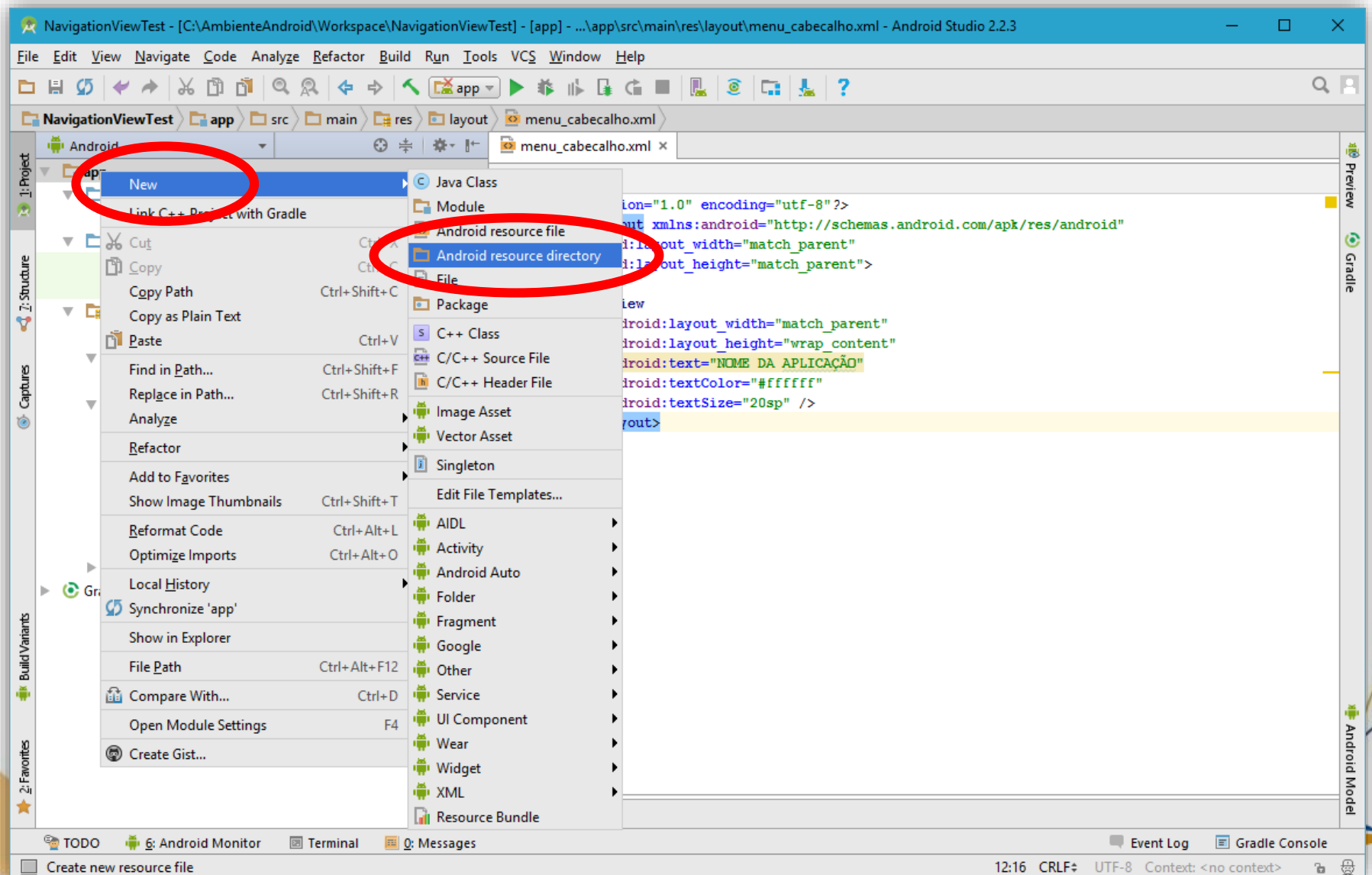
Arquivo de Menu

- Para exibir as opções disponíveis no menu, precisamos criar um arquivo de definições específico para tanto (um arquivo de definições de menu);
- As opções definidas no arquivo serão carregadas no menu lateral.

Criando a Pasta de Menu

- Para criar o novo arquivo de menu, é necessário que exista a pasta adequada na estrutura de diretórios de “res” do projeto;
- Para criar o diretório, clique com o botão direito no projeto e escolha a opção “New” > “Android resource directory”;
- Este comando abrirá o assistente de criação de novos diretórios do Android Studio.

Criando o Arquivo de Menu



Assistente de Criação de Diretórios

New Resource Directory

Directory name: values

Resource type: values

Source set: main

Available qualifiers:

- Country Code
- Network Code
- Locale
- Layout Direction
- Smallest Screen Width
- Screen Width
- Screen Height
- Size
- Ratio
- Orientation
- UI Mode
- Night Mode
- Density
- Touch Screen

Chosen qualifiers:

Nothing to show

>>

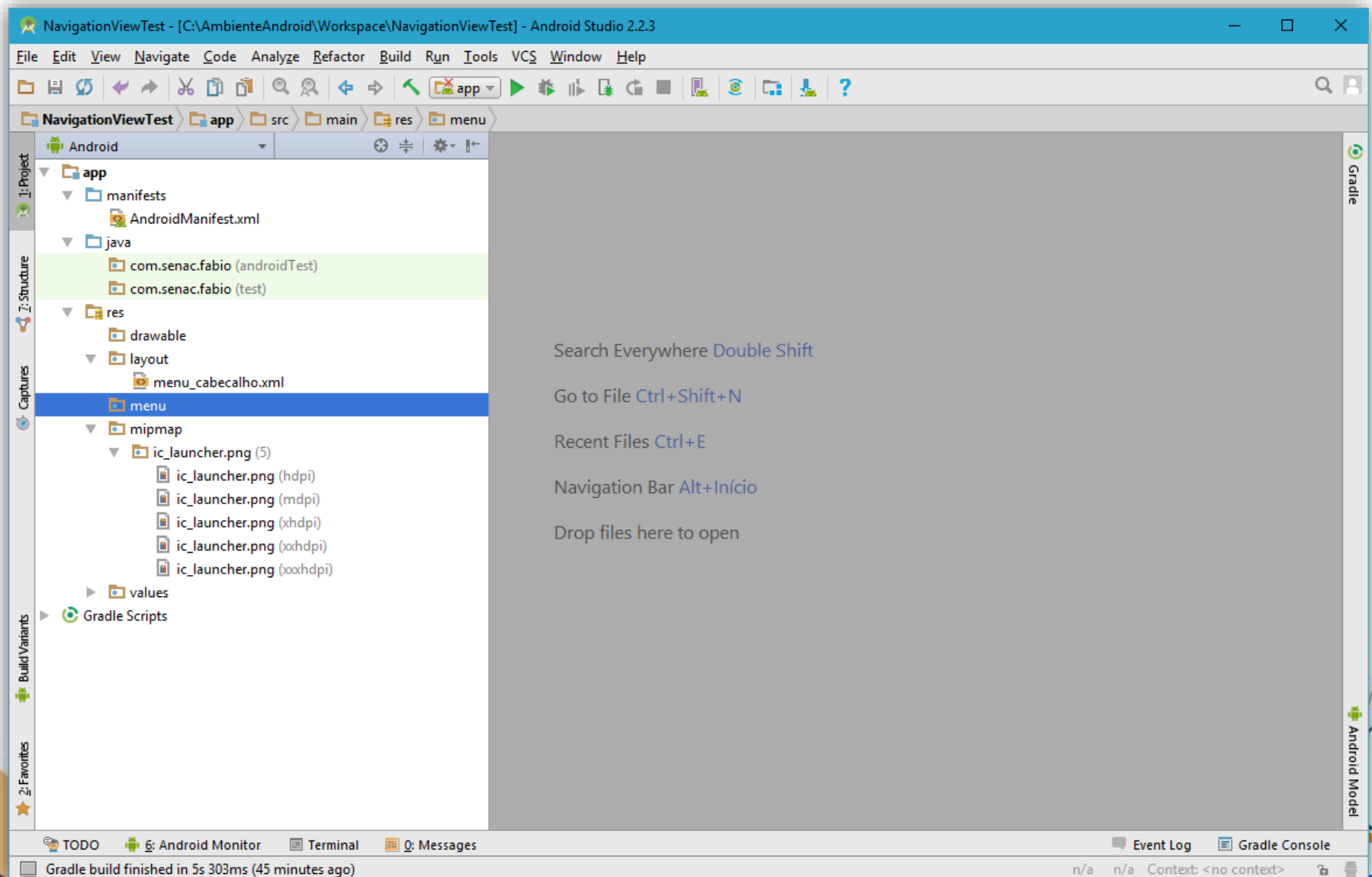
<<

OK Cancel Help

Criando a Pasta de Menu

- No assistente de criação de diretórios, dê o nome de “menu” ao novo diretório (no campo “Directory name” e clique em “OK” para criá-lo.

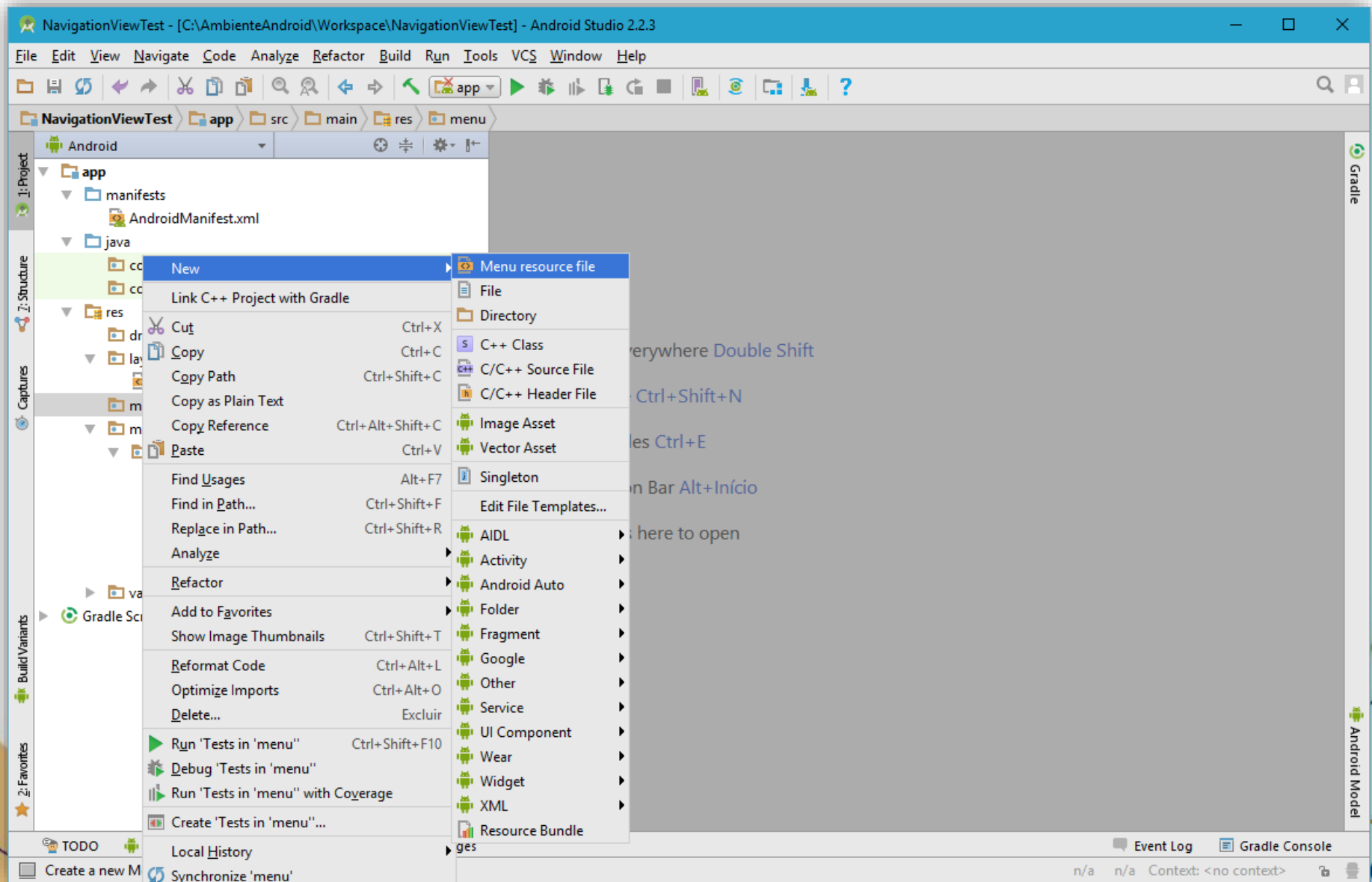
Pasta de Menu



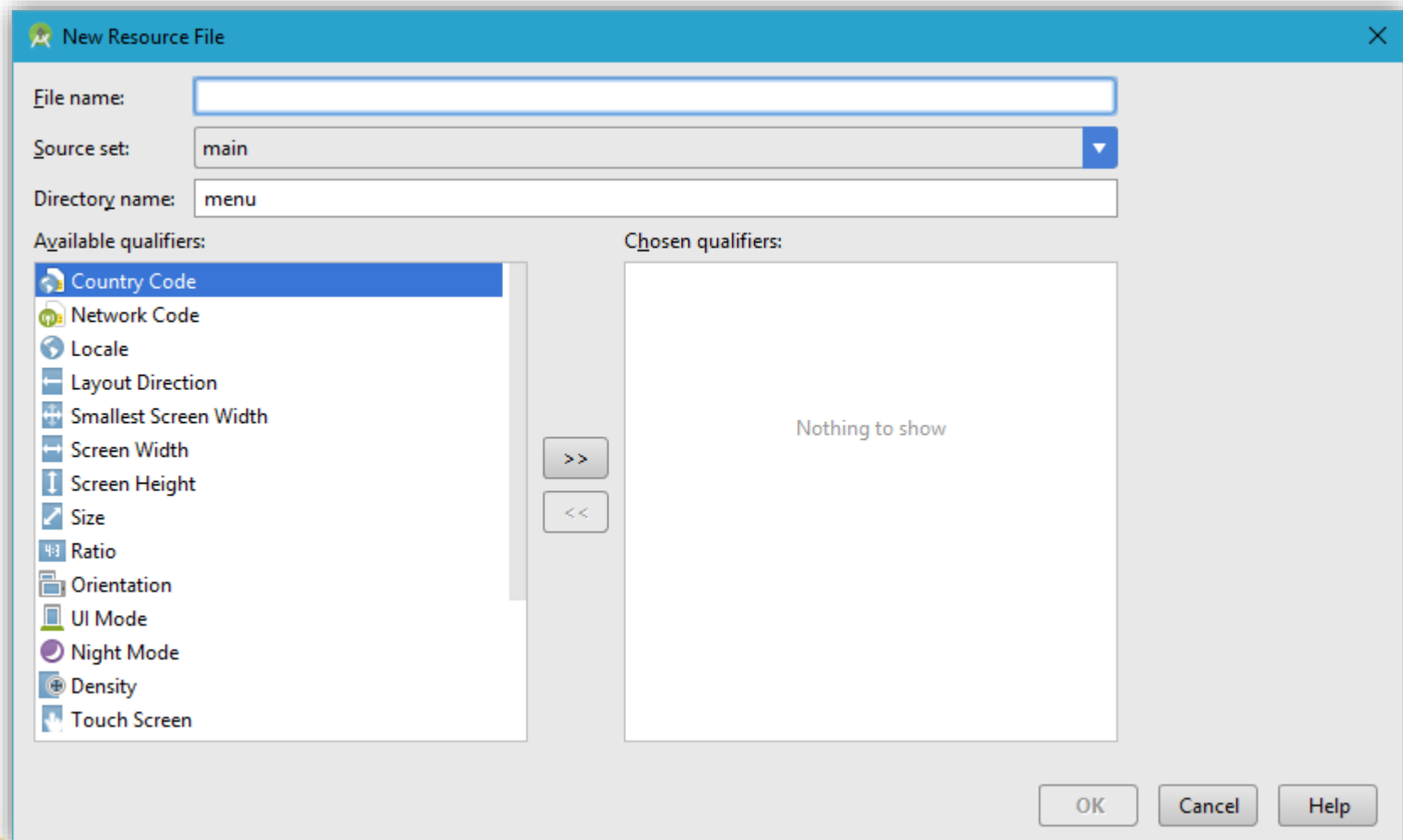
Criando o Arquivo de Menu

- Com a pasta criada, clique com o botão direito na mesma (pasta “menu”) e selecione as opções “New” > “Menu resource file” para abrir o assistente de criação de menus.

Criando o Arquivo de Menu



Criando o Arquivo de Menu



Criando o Arquivo de Menu

- Chame o arquivo de “menu” e clique em “OK”.

Criando o Arquivo de Menu

New Resource File

File name:

Source set:

Directory name:

Available qualifiers:

- Country Code
- Network Code
- Locale
- Layout Direction
- Smallest Screen Width
- Screen Width
- Screen Height
- Size
- Ratio
- Orientation
- UI Mode
- Night Mode
- Density
- Touch Screen

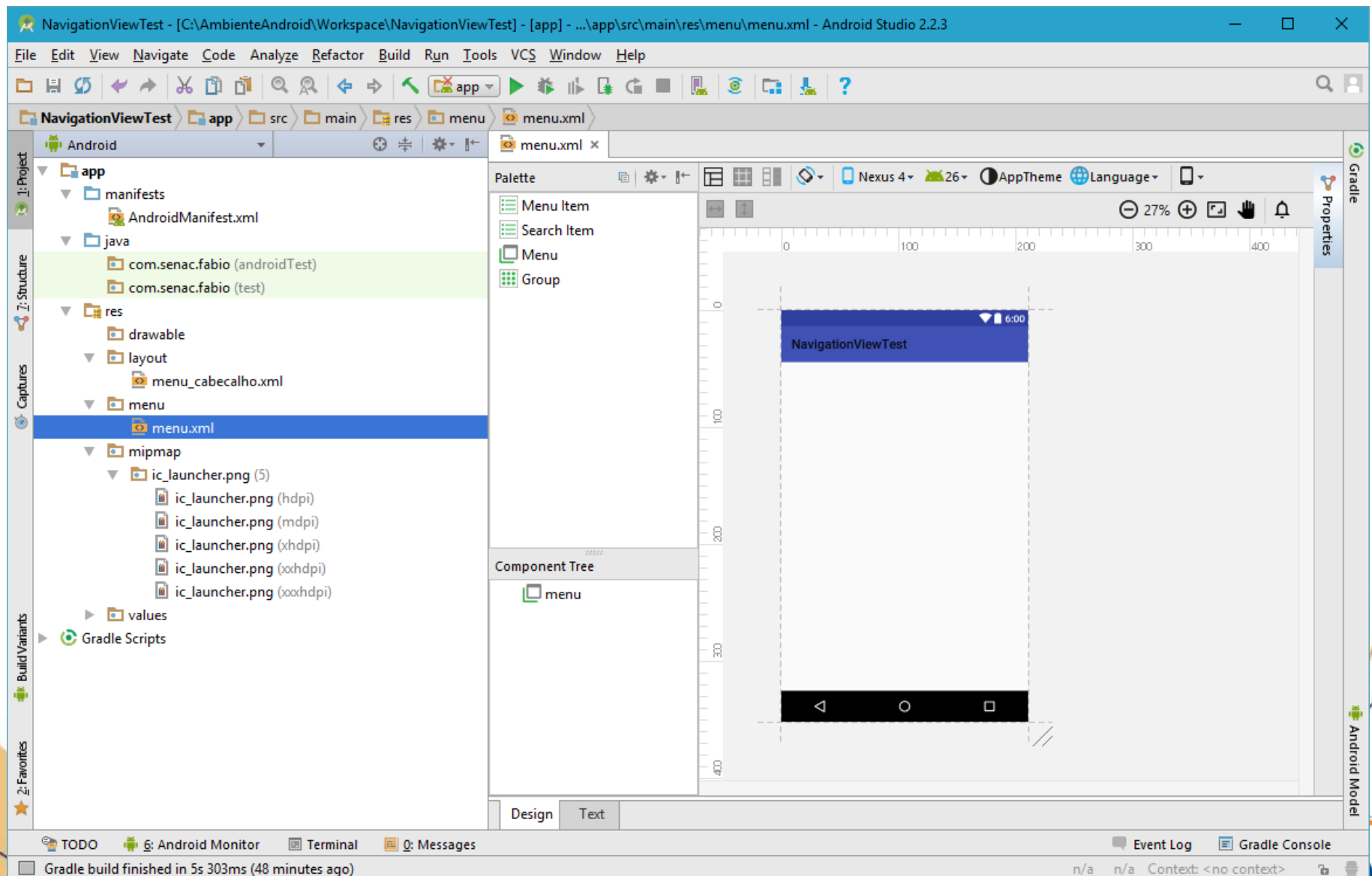
Chosen qualifiers:

Nothing to show

>> <<

OK Cancel Help

Arquivo de Menu



Definindo o Arquivo de Menu

- No arquivo de menu, vamos criar um grupo (necessário para o menu lateral) e um item;
- No grupo, configure o atributo “android:checkableBehavior” como false, para que seja possível selecionar apenas um item por vês no menu;
- Configure o menu como a seguir:

Definindo o Arquivo de Menu

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <group android:checkableBehavior="single">
    <item
      android:id="@+id/action_settings"
      android:checked="false"
      android:icon="@android:drawable/ic_menu_preferences"
      android:title="Configurações" />
  </group>
</menu>
```

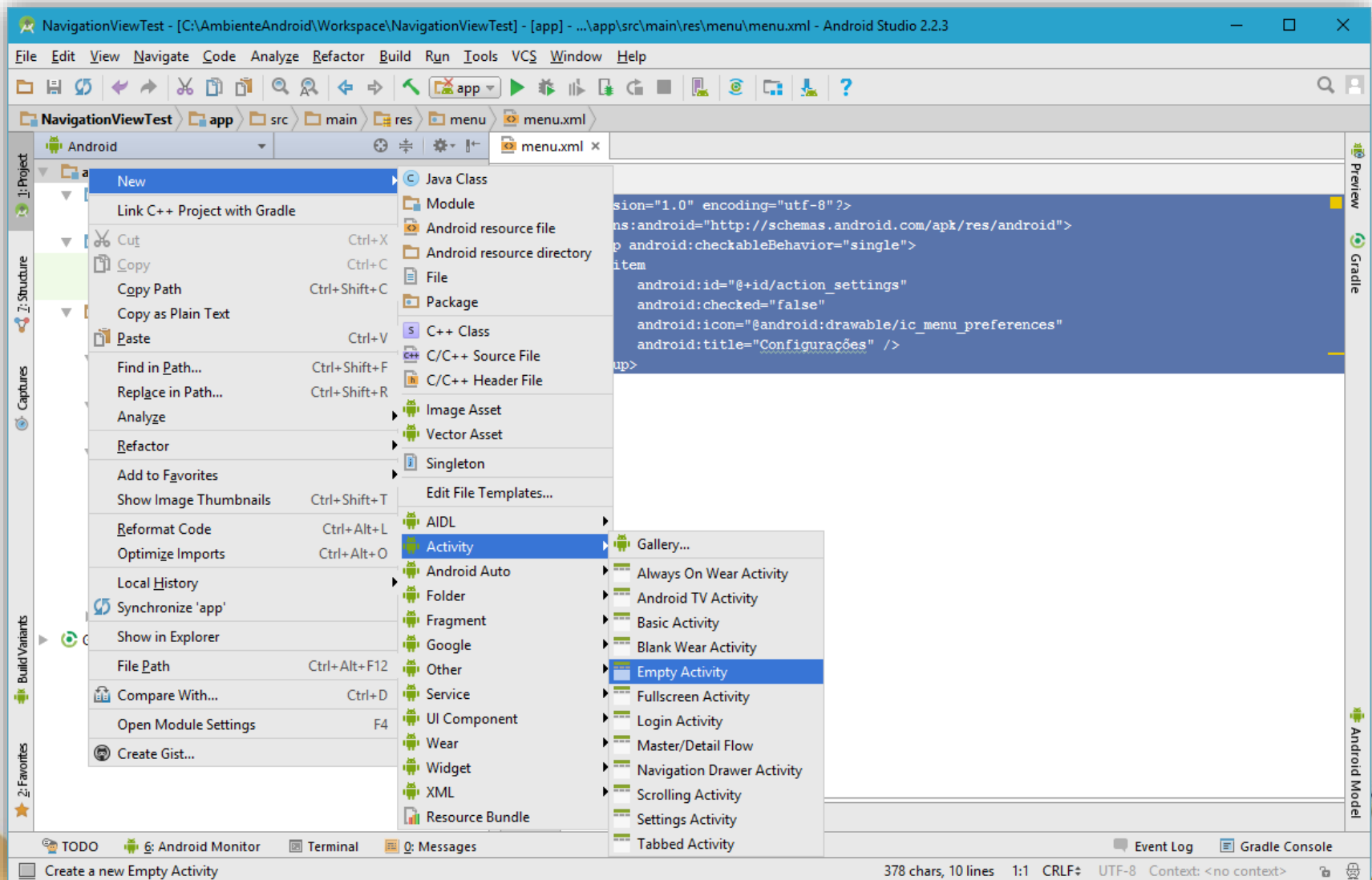
Atividade Principal

- Vamos criar uma atividade que armazenará o local para exibição das telas (fragmentos) e o menu lateral;
- Para isso, clique com o botão direito na aplicação e selecione “New” > “Activity” > “Empty Activity”.

Atividade Principal


- No assistente de criação de atividades que surgir, chame a atividade de “MainActivity”, marque a opção “Launcher Activity” e clique em finalizar para criar a classe e o XML da atividade.

Criando a Atividade Principal

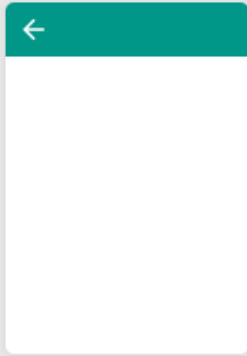


Criando a Atividade Principal

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 launch

Layout da Atividade Principal

- As atividades com navigation view precisam, primariamente, ter como elemento principal um item do tipo **“android.support.v4.widget.DrawerLayout”**;
- Este layout define uma atividade que suporta menus laterais.

Layout da Atividade Principal

- O `NavigationView` também precisa dos atributos:
 - **`app:headerLayout`**: o arquivo de layout de cabeçalho
 - **`app:menu`**: o arquivo de menu
 - **`android:layout_gravity="start"`**: Para alinhar o menu a esquerda

Layout da Atividade Principal

- Defina o layout da atividade principal (arquivo “activity_main”) como a seguir, onde:
 - **DrawerLayout** é o layout principal
 - O elemento **FrameLayout** é a área onde as telas (fragmentos) serão carregados;
 - O elemento do tipo **NavigationView** representa o menu lateral.

Layout da Atividade Principal

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v4.widget.DrawerLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/drawer"
    tools:context=".MainActivity">

    <FrameLayout android:id="@+id/frag_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    </FrameLayout>

    <android.support.design.widget.NavigationView
        android:id="@+id/navigation_view"
        android:layout_height="match_parent"
        android:layout_width="wrap_content"
        android:layout_gravity="start"
        app:headerLayout="@layout/menu_cabecalho"
        app:menu="@menu/menu">

    </android.support.design.widget.NavigationView>
</android.support.v4.widget.DrawerLayout>
```

Strings de Abertura e Fechamento

- É necessário definir strings de abertura e fechamento do menu, a serem utilizadas em elementos de acessibilidade do Android;
- Para isso, abra o arquivo “strings.xml” em “res/values” e adicione as duas strings, conforme o exemplo.

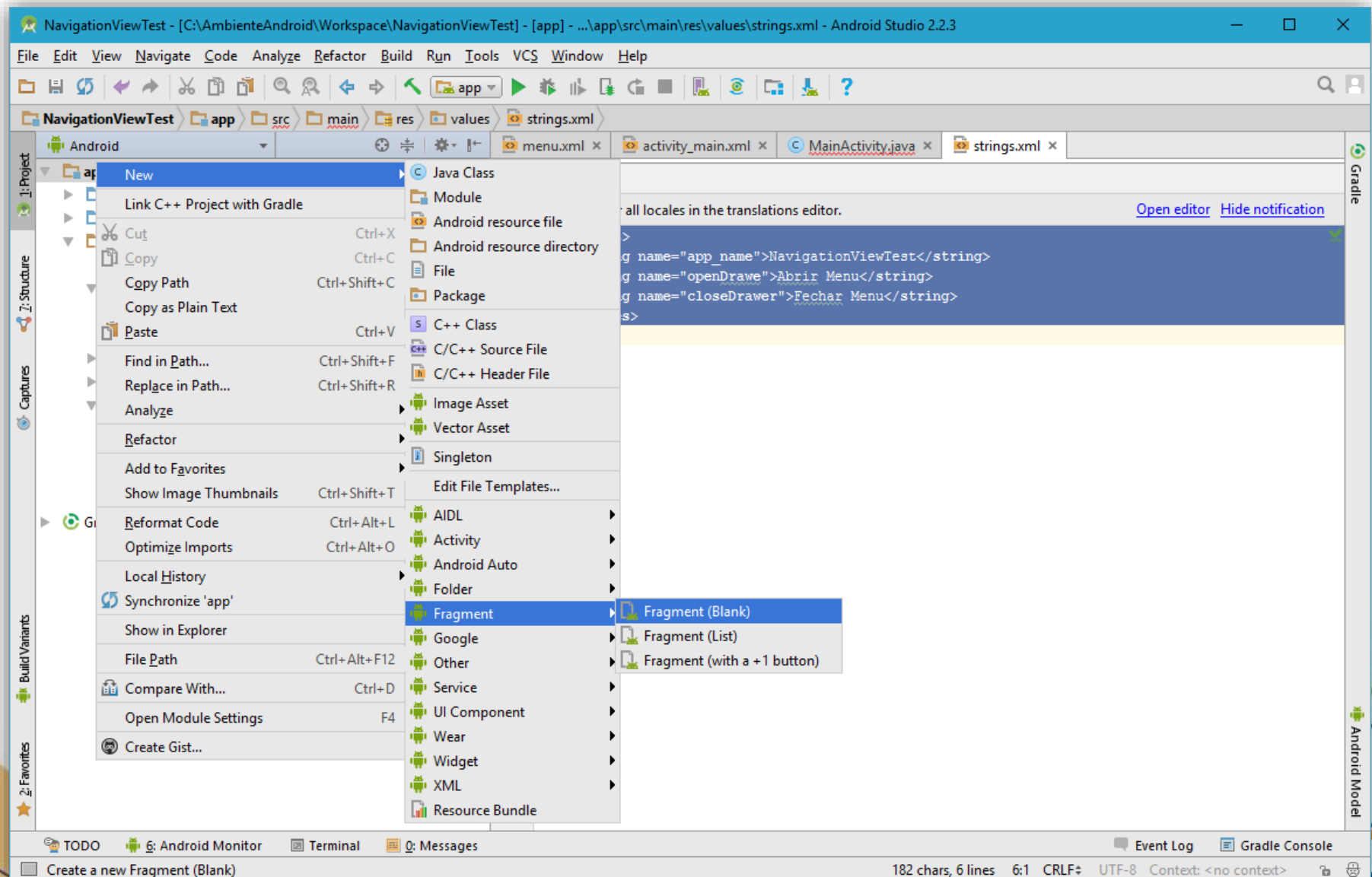
Strings de Abertura e Fechamento

```
<resources>  
  <string name="app_name">NavigationViewTest</string>  
  <string name="openDrawer">Abrir Menu</string>  
  <string name="closeDrawer">Fechar Menu</string>  
</resources>
```

Criando um Fragmento a Ser Aberto

- Vamos criar um fragmento simples, a ser aberto quando a opção de menu for selecionada;
- Para isso, clique com o botão direito sobre o projeto e selecione as opções “New” > “Fragment” > “Fragment (Blank)”.

Criando um Fragmento a Ser Aberto




Criando um Fragmento a Ser Aberto


- Chame o fragmento de “Fragmento01”, desmarque as opções de criação de métodos e clique em “Finalizar”

Criando um Fragmento a Ser Aberto

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?

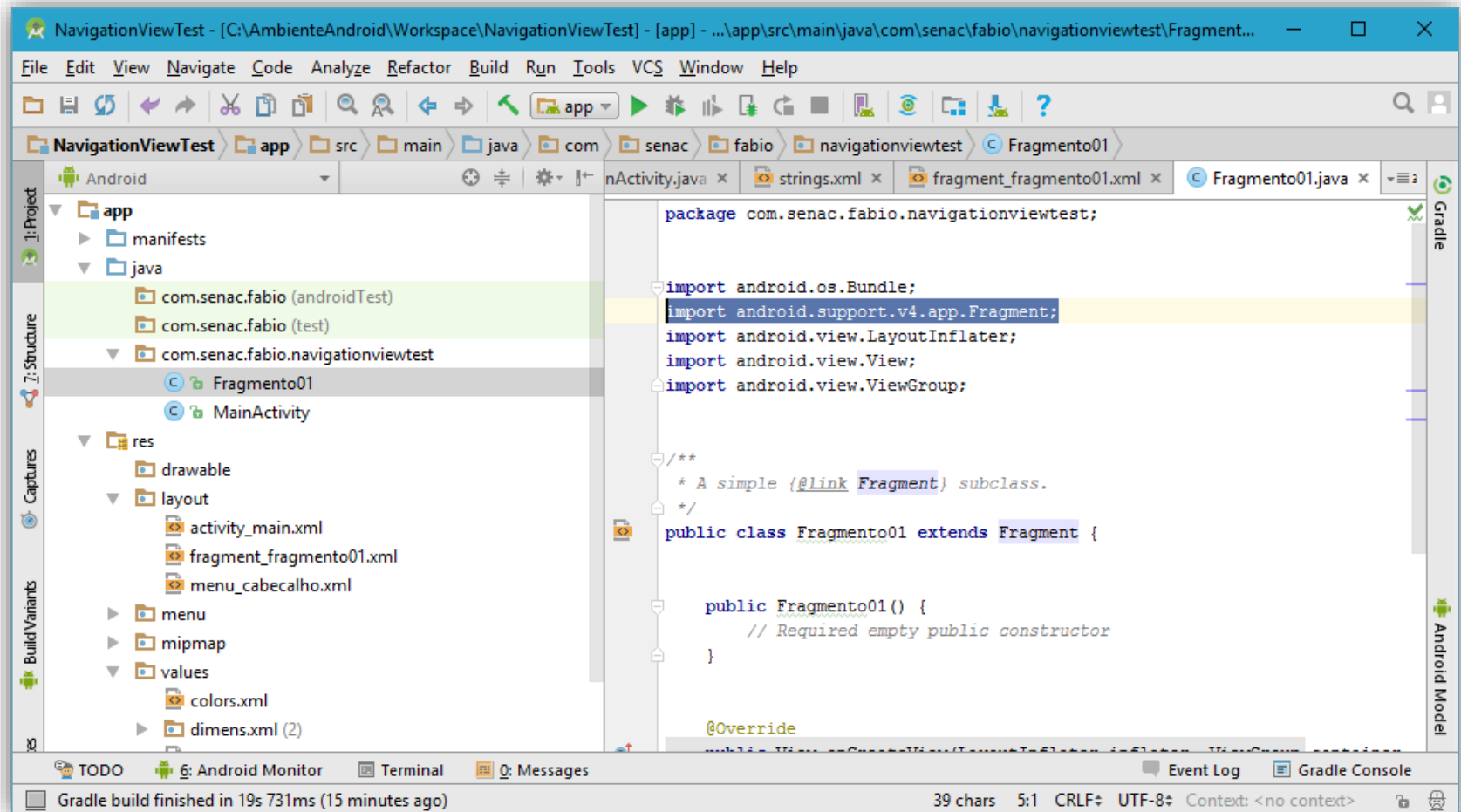
☐ Include interface callbacks?

Target Source Set:

Import do Fragmento

- Mude o import do fragmento para “import android.support.v4.app.Fragment”, editando-o diretamente.

Import do Fragmento



Classe da Atividade Principal

- No arquivo Java, vamos amarrar todos os elementos, fazendo:
 - A definição do menu lateral;
 - Um listener para quando um item do menu for selecionado
 - O controle de animação do menu lateral.

Classe da Atividade Principal

- Vá até a classe da atividade principal e defina as três principais views que utilizaremos (a `navigationView`, que corresponde ao menu, o `drawerLayout`, que corresponde a toda a tela da atividade, e o `actionBarDrawerToggle`, que representa o botão de abertura do menu

Classe da Atividade Principal

```
public class MainActivity extends AppCompatActivity {  
    private NavigationView navigationView;  
    private DrawerLayout drawerLayout;  
    private ActionBarDrawerToggle actionBarDrawerToggle;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```


Classe da Atividade Principal

- No método “onCreate”, logo abaixo dos elementos de inflação de XML, habilite o botão de abertura do menu, conforme o exemplo a seguir:

Classe da Atividade Principal

@Override

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);  
    getSupportActionBar().setHomeButtonEnabled(true);  
}
```

Classe da Atividade Principal

- Agora, carregue o componente de menu da tela, fazendo o binding do elemento da classe com o XML, conforme exemplo:

Classe da Atividade Principal

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

    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    getSupportActionBar().setHomeButtonEnabled(true);

    navigationView = (NavigationView) findViewById(R.id.navigation_view);
}
```

Classe da Atividade Principal

- Em seguida, é necessário definir e associar um listener a navigation view;
- O listener será responsável por carregar determinado fragmento sempre que uma opção for ativada;
- Desenvolva o código conforme exemplo:

Classe da Atividade Principal

...

```
navigationView = (NavigationView) findViewById(R.id.navigation_view);

navigationView.setNavigationItemSelectedListener(
    new NavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(MenuItem menuItem) {
            if (menuItem.isChecked()) {
                menuItem.setChecked(false);
            }
            else {
                menuItem.setChecked(true);
            }

            drawerLayout.closeDrawers();

            if (menuItem.getItemId() == R.id.action_settings) {
                Fragmento01 fragment = new Fragmento01();
                getSupportFragmentManager().beginTransaction().replace(
                    R.id.frag_container, fragment).commit();
                return true;
            }

            return false;
        }
    });
```

...

Classe da Atividade Principal

- Como configuração final da navigation view, precisamos associar o botão de abertura com o menu;
- Faça conforme o exemplo (logo abaixo do código do slide anterior):

Classe da Atividade Principal

```
drawerLayout = (DrawerLayout) findViewById(R.id.drawer);  
actionBarDrawerToggle =  
    new ActionBarDrawerToggle(this, drawerLayout,  
        R.string.openDrawer, R.string.closeDrawer) {  
        };  
drawerLayout.setDrawerListener(actionBarDrawerToggle);  
actionBarDrawerToggle.syncState();
```


Classe da Atividade Principal

- Por fim (UFA!) precisamos associar a seleção do menu (padrão do Android) com o navigation view, redirecionando seus comandos para o action view que carregamos anteriormente;
- Faça como o exemplo a seguir (fora do método “onCreate”:

Classe da Atividade Principal

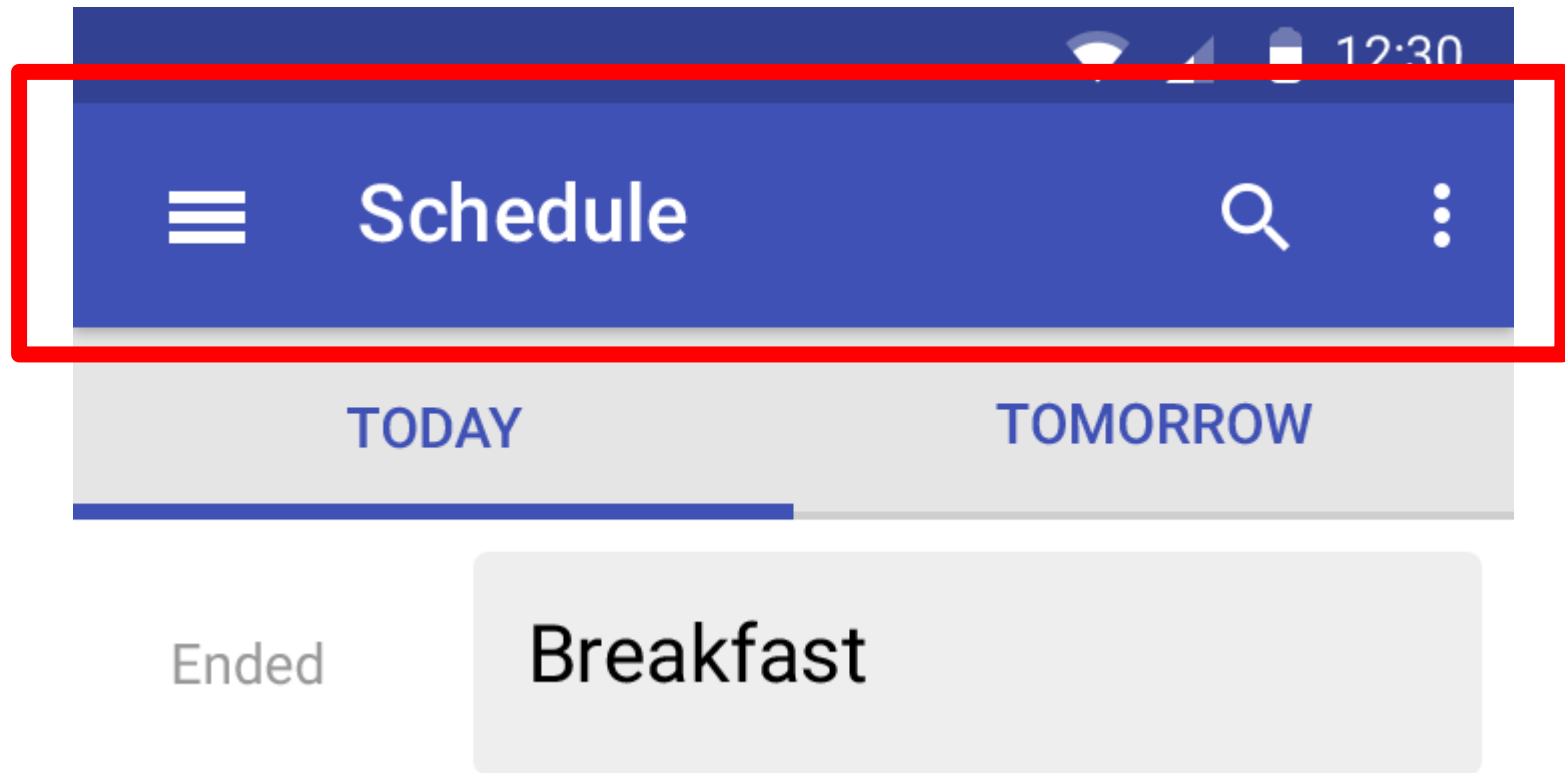
```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    if (actionBarDrawerToggle.onOptionsItemSelected(item)) {
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

BARRA DE AÇÕES E MENUS

Sobre a Barra de Menus/Ferramentas

- Até a versão 2.3.6, não existia barra de menu
 - Necessário apertar o botão de menu
- A partir da versão 3, foi introduzida a ActionBar
- Na versão 5, foi substituída pela Toolbar
- Mesmo funcionamento, com novas opções

Barra de Menus



Requisitos para a barra de menus

- Para criar um menu, são necessários:
 - Um arquivo XML com as opções do menu (em “res/menu”)
 - Implementar a função “onCreateOptionsMenu()” na atividade que infla o XML
 - Implementar “onOptionsItemSelected()” na atividade, informando o que fazer para cada opção

Arquivo XML

- O arquivo XML deve ficar na pasta “res/menu”
- É possível ter um arquivo para cada atividade, com diferentes opções

Arquivo XML

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

    <item android:id="@+id/action_messages"
          android:title="AÇÃO"
          android:icon="@android:drawable/sym_action_email"
          app:showAsAction="always" />

</menu>
```


Configurações de exibição

- O atributo “showAsAction” permite configurar como o item poderá aparecer para o usuário:
 - Sob a forma de um ícone de ação (também chamado simplesmente de “ação”)
 - Sob a forma de um item de menu, exibido ao clicar-se sob o ícone de menu correspondente

Possíveis valores de “showAsAction”

Valor	Descrição
ifRoom	Apenas mostra o item como uma ação se houver espaço disponível. Caso contrário, o item vai para o menu.
withText	Mostra o item como ação junto ao texto de título. Pode-se concatenar itens com um “ ” (pipe).
never	Sempre aparece no menu.
always	Sempre mostrar o item na barra de ações. Usar com cuidado, pois pode fazer os itens “encavalem”.
collapseActionView	Permite exibir um ícone ou item de menu que, ao clicado, expande um widget para toda a barra. Útil para composição de barras de pesquisa na barra de menus.

“Inflando” o menu

- Na atividade, é necessário inflar o menu XML no método de callback “onCreateOptionsMenu()”:

```
public boolean onCreateOptionsMenu(Menu menu) {  
    // Carrega o menu  
    getMenuInflater().inflate(R.menu.main, menu);  
    return true;  
}
```

Tratando as seleções do menu

- Também é necessário tratar cada opção do menu, através do método de callback “onOptionsItemSelected()”

Tratando as seleções do menu

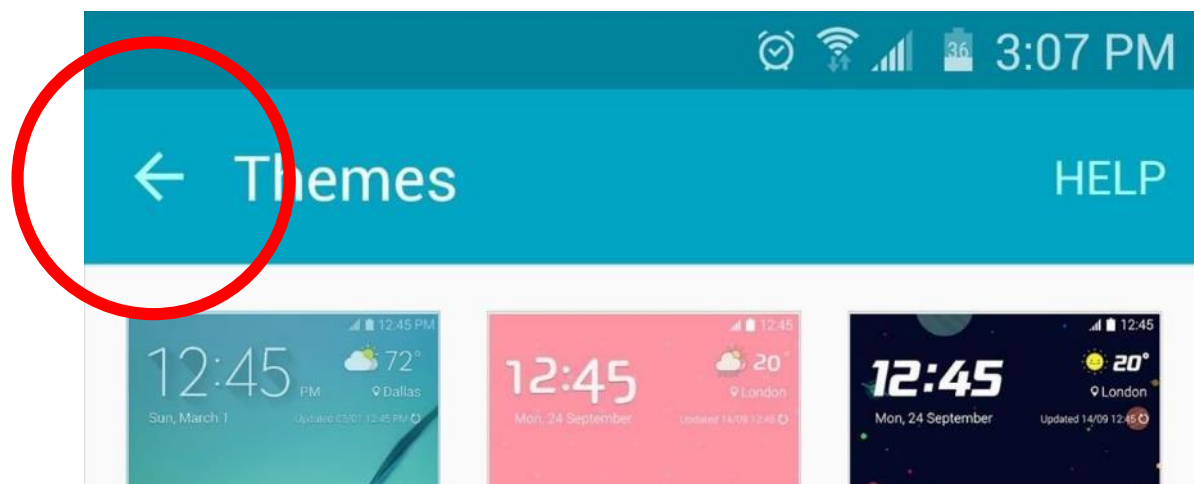
```
public boolean onOptionsItemSelected(MenuItem item) {  
    int id = item.getItemId();  
    if (id == R.id.action_messages) {  
        Intent intent = new Intent(this, MessagesActivity.class);  
        startActivity(intent);  
        return true;  
    }  
    return super.onOptionsItemSelected(item);  
}
```

BOTÃO VOLTAR (UP)

Sobre o Botão Voltar

- Partindo da versão 4.0, o Android permite exibir um botão voltar no topo das atividades com barra de menus
- O botão voltar permite retornar a atividade principal (ou anterior) da aplicação
- Também conhecido como botão “up”

Botão Voltar (“Up”)



Configurando a Atividade “Mãe”

- Para exibir o botão voltar, é primeiro necessário informar a atividade pai em cada uma das atividades cuja exibição do botão é desejada;
- Adicionar o atributo “android:parentActivityName” no elemento da atividade no “AndroidManifest.xml”

Configurando a Atividade “Mãe”

```
<activity android:name=".UpExample"  
android:parentActivityName=".MenuExample">  
</activity>
```

Habilitando o Botão “Up”

- Para habilitar a exibição do botão “up”, pode ser necessário acionar o comando abaixo no callback “onCreate” da atividade desejada:

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

    ActionBar ab = getSupportActionBar();
    ab.setDisplayHomeAsUpEnabled(true);
}
```

EXERCÍCIOS

Exercício 01

- Crie um novo projeto no Android Studio com Navigation View:
- Personalize o cabeçalho do Navigation View
- Modifique o arquivo de layout da atividade para conter um FrameLayout (será o container)

Exercício 01

- Crie 3 fragmentos com textos diferentes
- Modifique o menu para conter apenas opções para os 3 fragmentos criados
- Modifique o Java para trocar o fragmento de acordo com a opção selecionada no menu

Exercício 01

- Cada fragmento deverá exibir telas de cadastro distintas, como produto, cliente e usuário (as telas não precisam funcionar completamente, apenas ter exibição correta de alguns campos e botões referentes a respectiva funcionalidade)

Exercício 02

- Crie duas novas atividades, “Principal” (tela inicial) e “Sobre” (com informações do aplicativo)
- Adicione uma opção na toolbar da atividade principal para acessar a atividade sobre
- Atribua um ícone sempre visível a esta ação

Exercício 02

- Configure a segunda atividade (sobre) como filha da atividade principal;
- Faça com que o botão voltar (“up”) seja exibido na atividade sobre e permita retornar à atividade inicial

Exercício 02

- Adicione uma opção de menu sempre “oculta” que permita encerrar a aplicação ao ser acionada (deve ser exibida apenas na tela principal)

"That's all Folks!"