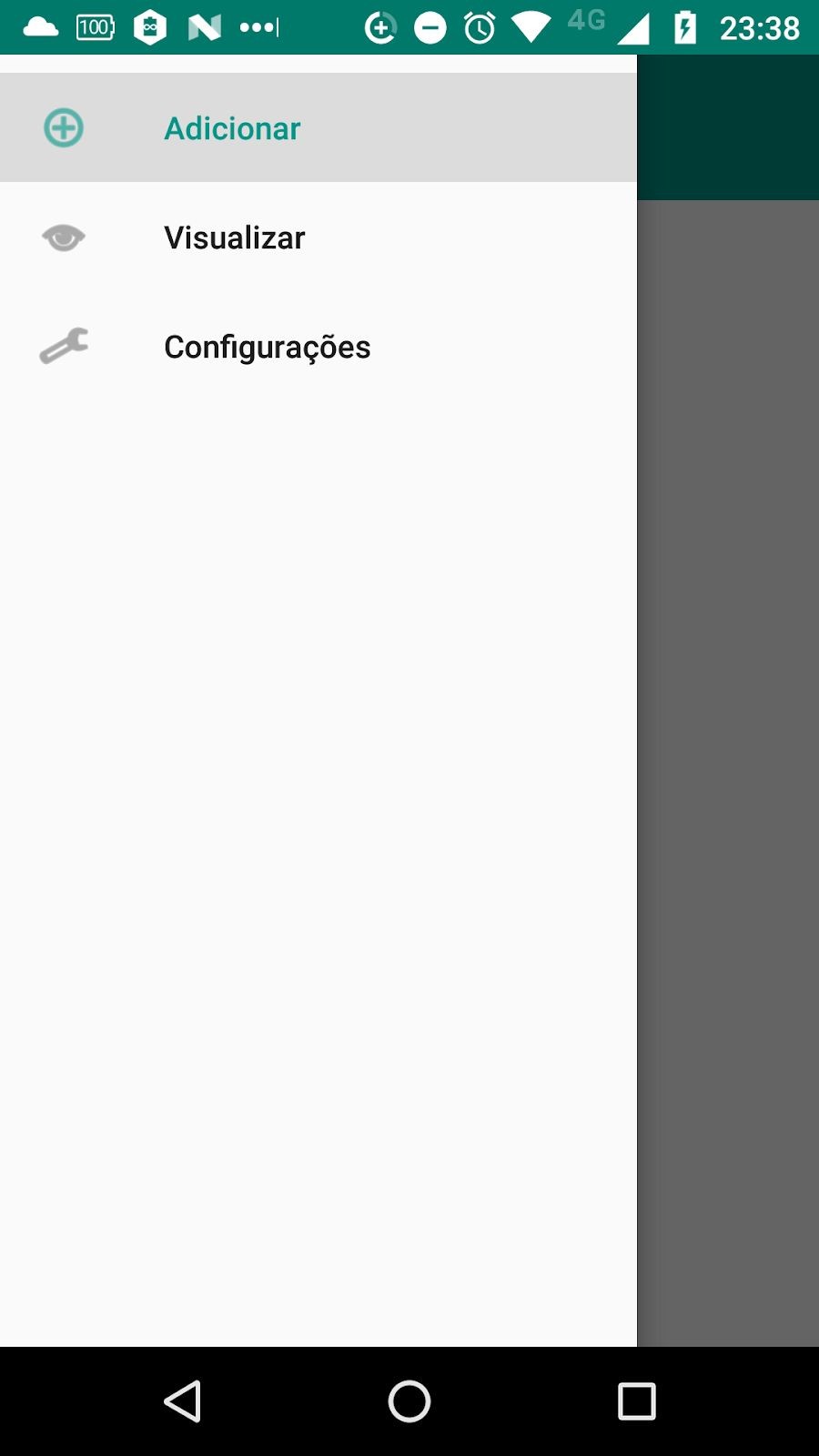
# CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

**CURSO: Técnico em Informática Integrado - 3º Ano**

**DISCIPLINA: Aplicações Móveis / Laboratório de Aplicações Móveis PROFESSOR: Wendell Fioravante da Silva Diniz**

**Tutorial 7 - Toolbar com Drawer**

Uma das maneiras de se usar a Toolbar é usando-a para fazer o papel da ActionBar. Existem abordagens diferentes, que variam quanto ao design e funcionalidades do app. Uma das formas de se oferecer um controle personalizado da navegação entre as diferentes views de uma app, é usando um elemento de interface chamado drawer ou gaveta. Quando clicarmos sobre o ícone da aplicação na Toolbar, um menu lateral será mostrado, com as opções de ações possíveis na app.

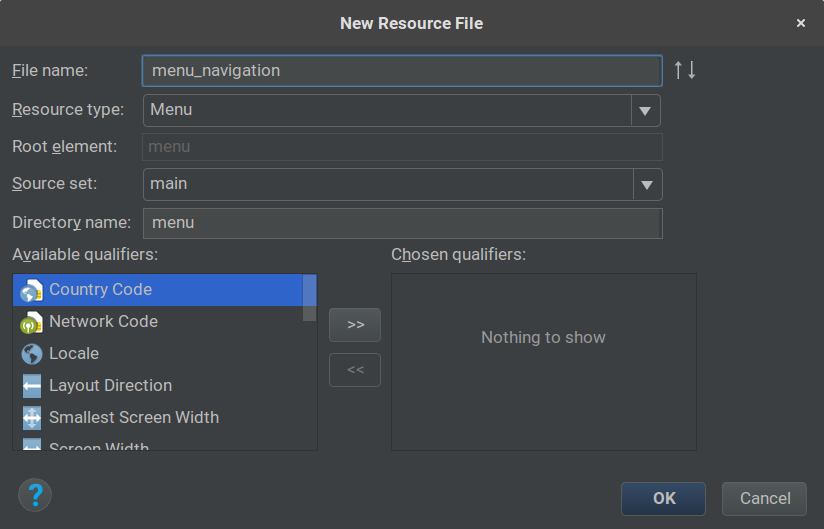


Primeiro, crie um novo projeto com o nome Android Toolbar, com uma atividade vazia como atividade principal. Abra o arquivo de layout da atividade principal e apague o TextView que é colocado por padrão. Agora, mude para o modo texto, clicando no botão *Text* localizado abaixo da paleta de widgets, para exibir o XML da interface. Mude o widget raiz de android.support.constraint.ConstraintLayout para android.support.v4.widget.DrawerLayout. O arquivo deve ficar assim:

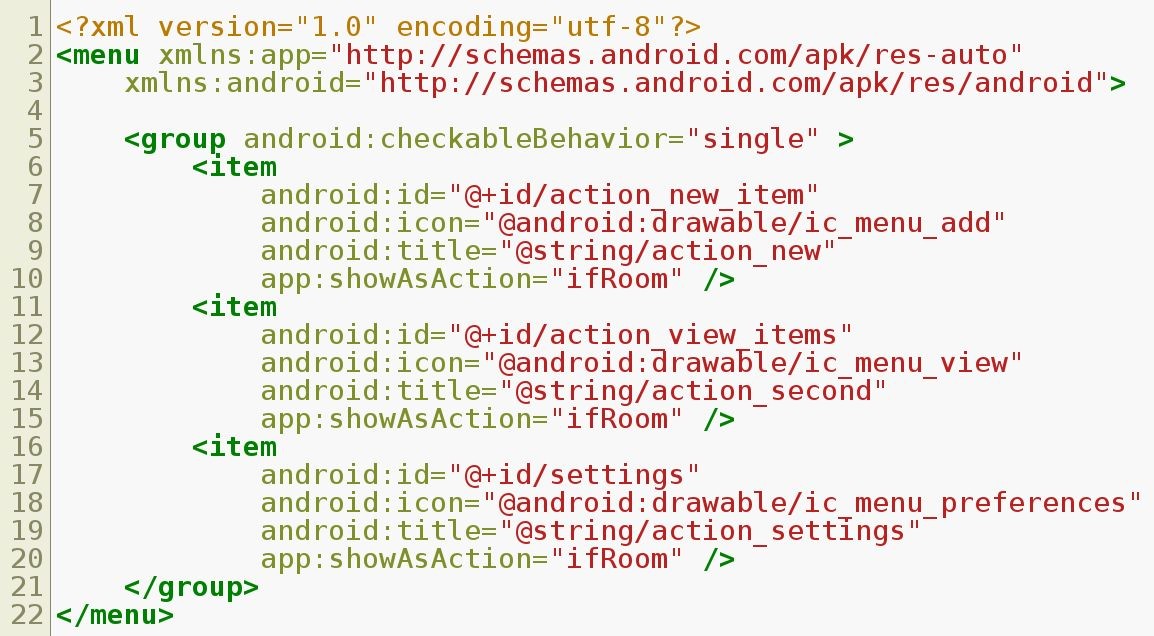


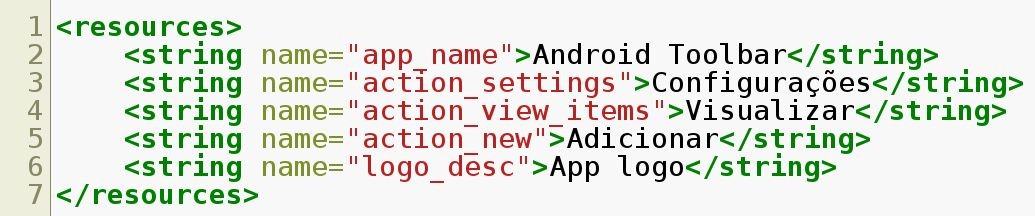
Vamos modificar o tema da app, para que ele não use a ActionBar do sistema, usaremos a Toolbar que vamos criar em seu lugar. Para isso, abra o arquivo *styles.xml*, que encontra-se na pasta res/values. Mude o valor do atributo parent da tag style de Theme.AppCompat.Light.DarkActionBar para Theme.AppCompat.Light.NoActionBar.

Agora, vamos criar uma listagem de itens de menu, para exibirmos quando clicarmos no ícone de navegação da app. Clique com o botão direito sobre a pasta res e selecione a opção New.. -> Android Resource File. Na janela que se abre, coloque menu\_navigation como nome do arquivo e na caixa Resource Type, selecione menu.

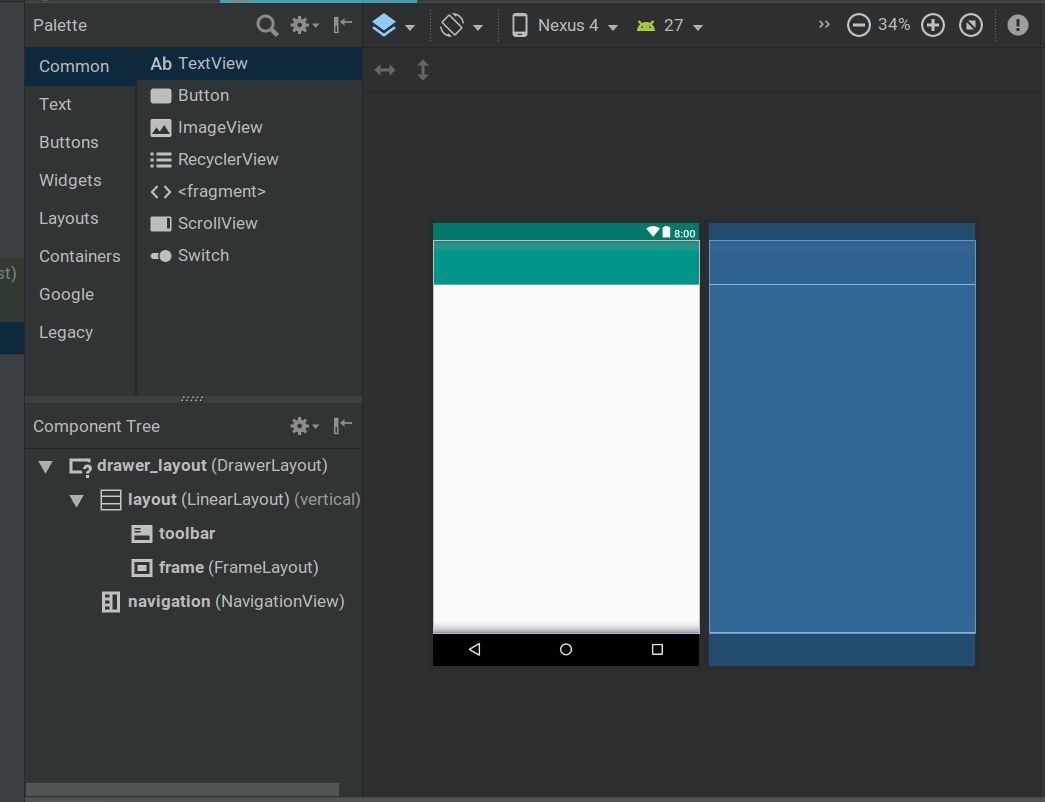


Abra o arquivo, mude para o modo texto e digite o código abaixo:



Após criar o menu, abra o arquivo strings.xml e modifique-o para que fique assim:

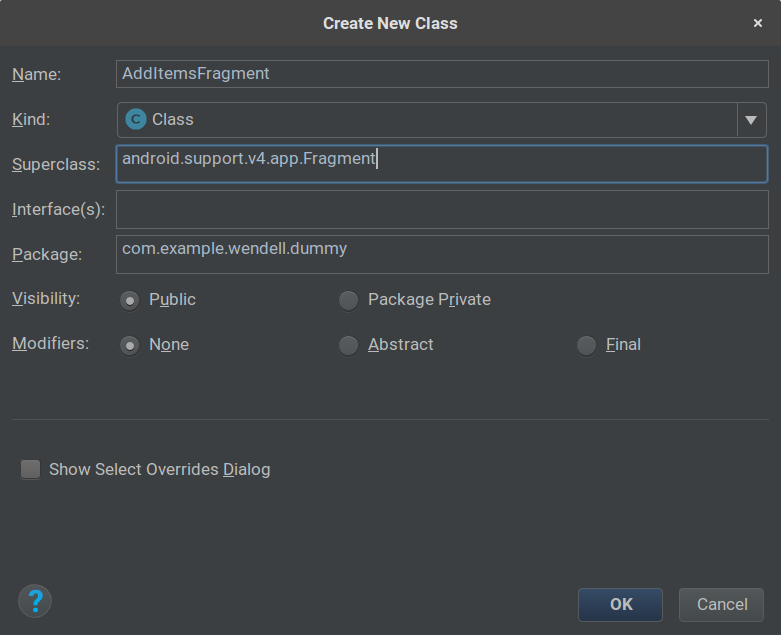
Agora, você pode voltar para a janela de design. Selecione o DrawerLayout e mude seu ID para drawer\_layout. Agora, vamos adicionar os outros elementos. O DrawerLayout deve ter exatamente dois filhos. Para o primeiro, colocaremos um LinearLayout, com orientação vertical e ID layout. Dentro do LinearLayout, acrescente um Toolbar, com o ID toolbar e abaixo dele, um FrameLayout, com o ID frame. Agora, acrescente um NavigationView como o segundo filho do DrawerLayout (tome cuidado para não acrescentar ao LinearLayout). Use o ID navigation para ele. Selecione a NavigationView e na área de propriedades, no lado direito da visualização, procure a propriedade menu. Coloque o valor @menu/menu\_navigation, que criamos anteriormente nela. Por fim, mude para a visualização de texto e acrescente a propriedade android:layout\_gravity="start" ao NavigationView. A ordem dos widgets deve ficar como na figura:

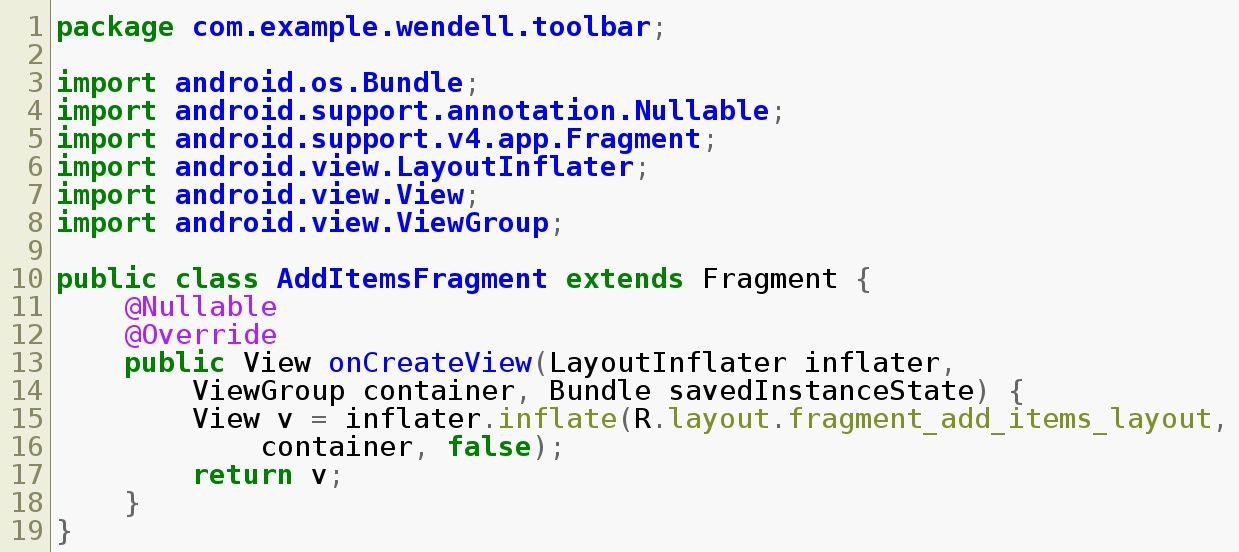


Vamos agora criar o código da atividade principal. Para exibir os diferentes itens da navegação, usaremos Fragments. Fragments são uma espécie de “subatividade”, que contém seu próprio layout e comportamentos, mas ainda faz parte de uma atividade. Vamos estudá-los mais profundamente nas próximas aulas. Por enquanto, vamos usá-los de forma bem básica. Digite o código a seguir na MainActivity:





Agora, adicione uma nova classe Java ao projeto, clicando com o botão direito sobre a pasta do pacote (com.example.alunos.nome-da-app, por exemplo). Dê o nome de AddItemsFragment e mude a superclasse para android.support.v4.Fragment:

Digite o seguinte código para ela:

Crie também um novo arquivo de layout com o nome fragment\_add\_items\_layout e coloque um TextView com um texto qualquer, para ser exibido quando carregarmos o fragmento.

Repita o processo de criar uma classe e um arquivo de layout, respectivamente, para ViewItensFragment e fragment\_view\_items\_layout e SettingsFragment e fragment\_settings\_layout. Para encerrar, crie um novo menu com o nome *empty\_menu*. A aplicação deve rodar neste ponto.

Agora é com você. Crie um novo projeto que deverá exibir um menu parecido com o do exemplo, mas com duas opções: Visualizar lista e Adicionar pessoa. Usando as classes de dados dos exercícios anteriores, construa o app, de modo que no fragmento Adicionar pessoa, seja exibido os campos para adicionar uma nova pessoa à lista. A lista atualizada deve ser exibida no outro fragmento, usando um ListView.

Dica: como o fragmento é iniciado pela MainActivity, é possível acessar métodos e atributos públicos dentro do Fragment! Isso facilita bastante a passagem de dados entre eles. Faça assim: dentro do seu Fragment, declare um objeto do tipo MainActitivity e capture-o com o método getActivity():

MainActivity atividade = (MainActivity) getActivity();

Programe um método, digamos getLista(), na sua MainActivity, que retornará a lista de

pessoas:

public ArrayList<Pessoa> getLista() {

…

}

No Fragmento, chame o método através do objeto que você declarou anteriormente:

ArrayList<Pessoa> lista = atividade.getLista();

Agora, basta popular o ListView, usando o adaptador personalizado. Para adicionar, repita

o procedimento no outro Fragment, agora capturando os valores e adicionando-os. Não se preocupe em salvar os dados entre seções do aplicativo, aprenderemos a fazê-lo quando estudarmos manipulação de banco de dados. Bom trabalho.