Министерство образования, науки и молодежной политики
Краснодарского края
Государственное бюджетное профессиональное образовательное учреждение
Краснодарского края «Ейский полипрофильный колледж»

РАЗРАБОТКА МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

ЛАБОРАТОРНАЯ РАБОТА №9

по теме:

Навигационные UI-компоненты. Панель приложения

Выполнил:
студент ЕПК, группа
ФИО

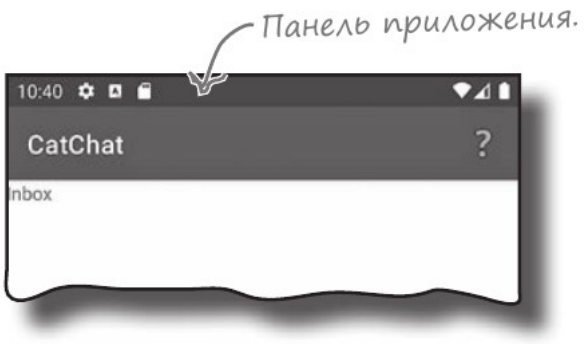
Проверил:
преподаватель дисциплины
«Разработка мобильных
приложений»
Фомин А. Т.

Ейск 2024

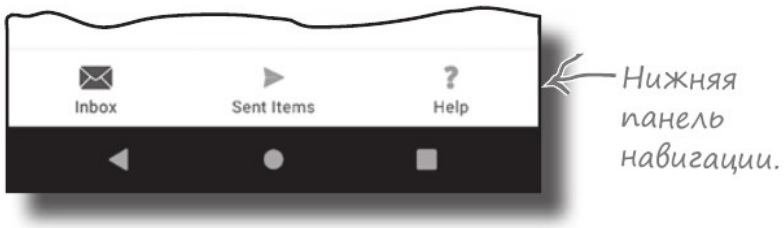
1 Описание проекта

Если в приложении есть экраны, которые должны быть доступны из любой точки приложения, то можно для навигации применять один из навигационных UI-компонентов Android:

1. **Панель приложения.** Это панель, которая отображается у верхнего края экрана. Android обычно включает такую панель по умолчанию. На нее можно добавлять элементы, по щелчку на которых происходит переход к целям.



2. **Нижняя панель навигации.** Отображается у нижнего края экрана. На ней размещаются элементы, которые могут использоваться для навигации.



3. **Выдвижная панель навигации.** Панель, которая выдвигается от стороны экрана. Такие панели используются во многих приложениях, так как они обладают очень гибкими возможностями.

В приложении, которое будет разработано, использоваться одна активность с именем MainActivity и три фрагмента: InboxFragment, SentItemsFragment, HelpFragment. Каждый фрагмент будет отображаться в MainActivity при переходе к нему.

Приложение будет включать панель приложения, на которой размещается элемент для открытия меню справки. Когда пользователь щелкает на нем, приложение переходит к фрагменту HelpFragment и отображает его в MainActivity.

2 Указания по выполнению работы

1. Создайте новый проект по известной схеме. Выберите вариант Empty Views Activity, введите имя «lab9».
2. Стандартная панель приложения добавляется в приложение путем применения темы. Тема обеспечивает целостность оформления приложения на разных экранах. Она управляет внешним видом приложения и наличием у него панели приложения. Android включает ряд тем, которые могут использоваться в ваших приложениях. По умолчанию Android Studio применяет тему, включающую панель приложения. В приложении будет использоваться тема Material Design.

   Система Material Design была разработана компанией Google. Она помогает строить качественные приложения и веб-сайты с целостным оформлением. Идея заключалась в том, чтобы пользователь, переключающийся с приложения Google (такого, как Play Store) на приложение, созданное сторонним разработчиком, мгновенно чувствовал себя в знакомой обстановке и знал, что делать.

   В основу Material Design изначально были заложены принципы дизайна печатных материалов, отражающие внешний вид и поведение реальных объектов (например, карточек для записей и листков бумаги.) Новейшей стадией ее эволюции стала система Material You, предоставляющая пользователю более динамичный опыт взаимодействия с персонализированной цветовой палитрой.

   Темы Material хранятся в отдельной библиотеке, которая должна быть включена в приложение. Для этого следует добавить зависимость для библиотеки com. google.android.material в файл build.gradle приложения.

   Так как мы собираемся использовать тему Material, необходимо проверить, что эта библиотека включена в приложение. Откройте файл app/Gradle Scripts/build.gradle и убедитесь в том, что он включает следующую строку:

```
    implementation
    ("com.google.android.material:material:1.12.0")
```

3. Нажмите на кнопку Sync Now, для начала синхронизации сборки.
4. Тема применяется в файле AndroidManifest.xml приложения. Этот файл предоставляет информацию о конфигурации приложения. В нем содержатся атрибуты — включая тему, — непосредственно влияющие на панель

приложения. Атрибут android:theme определяет тему. В приведенном выше коде он задается директивой:

```
    android:theme="@style/Theme.Lab9"
```

Это значение сообщает, что тема определяется как стилевой ресурс (обозначаемый @style) с именем Theme.Lab9.
Когда мы создается приложение, среда Android Studio создает два файла стилевых ресурсов за нас. Обоим файлам присваивается имя themes.xml, и они размещаются в папках app/src/main/res/values и app/src/main/res/values-night.
Файл в папке values содержит файл стилевых ресурсов приложения, используемый по умолчанию, а файл в папке values-night используется ночью. Содержимое файла по умолчанию:

```
    <resources xmlns:tools="http://schemas.android.com/tools">
        <!-- Base application theme. -->
        <style name="Base.Theme.Lab9"
    parent="Theme.Material3.DayNight.NoActionBar">
            <!-- Customize your light theme here. -->
            <!-- <item
    name="colorPrimary">@color/my_light_primary</item> -->
        </style>

        <style name="Theme.Lab9" parent="Base.Theme.Lab9" />
    </resources>
```

Стиль также включает атрибут parent, который указывает, на основе какой темы должен определяться стиль.
Любые свойства родительской темы (например, цветовую схему) можно переопределить , для этого следует добавить в стиль элементы <item>. Например, в приложении включены элементы для переопределения основного и дополнительного цвета. Основной цвет используется приложением для прорисовки таких объектов, как панель приложения. Дополнительный цвет используется некоторыми представлениями для создания контраста.
Ниже приведен код, используемый приложением для переопределения основных цветов в файле themes.xml из папки values:

```
    <item name="colorPrimary">@color/purple_500</item>
```

```
    <item name="colorPrimaryVariant">@color/purple_700</item>
    <item name="colorOnPrimary">@color/white</item>
```

У каждого элемента имеется атрибут name, а обозначение @color ссылается на цвет из файла цветовых ресурсов.

Когда вы создается новый проект, среда Android Studio обычно включает стандартный файл цветовых ресурсов с именем colors.xml. Он находится в папке app/res/values, а содержащиеся в нем цвета могут использоваться в вашем приложении.

Типичный файл цветовых ресурсов выглядит примерно так:

```
    <?xml version="1.0" encoding="utf-8"?>
    <resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    ...
    </resources>
```

Чтобы изменить цветовую схему приложения, включите нужные цвета в файл цветовых ресурсов и обращайтесь к ним из файлов стилевых ресурсов.

5. Чтобы заменить панель приложения по умолчанию, следует удалить исходную панель приложения, включить панель инструментов в ваш макет, а затем приказать активности использовать панель инструментов как его панель приложения.

   В отличие от панели приложения по умолчанию, панель инструментов является разновидностью представления, которая добавляется в макет. А раз это представление, это означает, что вы можете в полной мере управлять его размером и позицией.

   В приложении мы будем использовать панель инструментов Material. Эта разновидность панели инструментов хорошо работает в сочетании с темами Material.

   Добавим панель инструментов в макет MainActivity, чтобы она отображалась у верхнего края экрана. Обновите код в activity_main.xml и приведите его к виду (Листинг 1.)

6. Установленная панель еще не обладает никакой функциональностью, например, имя приложения не выводится на панели инструментов, как на стандартной панели.

Чтобы панель инструментов вела себя как привычная панель приложения, необходимо сообщить об этом MainActivity. Для этого следует вызвать в коде активности метод setSupportActionBar() и передать ему ссылку на панель инструментов.

Листинг 1. Файл activity_main.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"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.MaterialToolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        style="@style/Widget.MaterialComponents.Toolbar.Primary"
        />

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"

android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph" />

</LinearLayout>
```

Добавьте этот код в метод onCreate() актвности MainActivity:

```
    val toolbar = findViewById<MaterialToolbar>(R.id.toolbar)
```

```
    setSupportActionBar(toolbar)
```

Обновите код MainActivity.kt, добавив также импорт класса MarerialToolbar, как показано в Листинге 2.

Листинг 2. Файл MainActivity.kt

```
1.  package com.example.lab9
2.
3.  import android.os.Bundle
4.  import androidx.activity.enableEdgeToEdge
5.  import androidx.appcompat.app.AppCompatActivity
6.  import androidx.core.view.ViewCompat
7.  import androidx.core.view.WindowInsetsCompat
8.  import com.google.android.material.appbar.MaterialToolbar
9.
10. class MainActivity : AppCompatActivity() {
11.     override fun onCreate(savedInstanceState: Bundle?) {
12.         super.onCreate(savedInstanceState)
13.         enableEdgeToEdge()
14.         setContentView(R.layout.activity_main)
15.         setContentView(R.layout.activity_main)
16.         val toolbar =
    findViewById<MaterialToolbar>(R.id.toolbar)
17.         setSupportActionBar(toolbar)
18.
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)
    ) { v, insets ->
19.             val systemBars =
    insets.getInsets(WindowInsetsCompat.Type.systemBars())
20.             v.setPadding(systemBars.left, systemBars.top,
    systemBars.right, systemBars.bottom)
21.             insets
22.         }
23.     }
24. }
```

7. Создадим два новых фрагмента InboxFragment и HelpFragment, которые будут отображаться в MainActivity при переходе к ним пользователя. Фрагмент

InboxFragment будет отображаться при запуске приложения, а HelpFragment — при переходе к нему.

Выделите пакет com.example.lab9 в папке app/kotlin+java и выберите команду File→New→Fragment→Fragment (Blank). Введите имя фрагмента «InboxFragment» и имя макета «fragment_inbox»; убедитесь в том, что выбран язык Kotlin. Затем обновите код InboxFragment.kt и приведите его к виду Листинг 3.

8. Код фрагмента fragment_inbox.xml обновите, как показано в Листинге 4.

Листинг 3. Файл InboxFragment.kt

```
1.  package com.example.lab9
2.
3.  import android.os.Bundle
4.  import androidx.fragment.app.Fragment
5.  import android.view.LayoutInflater
6.  import android.view.View
7.  import android.view.ViewGroup
8.
9.  class InboxFragment : Fragment() {
10.     override fun onCreateView(
11.         inflater: LayoutInflater, container: ViewGroup?,
12.         savedInstanceState: Bundle?,
13.     ): View? {
14.         // Inflate the layout for this fragment
15.         return inflater.inflate(R.layout.fragment_inbox,
    container, false)
16.     }
17. }
```

Листинг 4. Файл fragment_inbox.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3.      xmlns:tools="http://schemas.android.com/tools"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".InboxFragment">

    <!-- TODO: Update blank fragment layout -->
    <TextView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:text="Inbox" />

</FrameLayout>
```

9. Фрагмент HelpFragment отображается в MainActivity, когда пользователь щелкает на элементе Help панели инструментов. Выделите пакет com.example.lab9 в папке app/kotlin+java и выберите команду File→New→Fragment→Fragment (Blank). Введите имя фрагмента «HelpFragment» с именем макета «fragment_help» и убедитесь в том, что выбран язык Kotlin. Код HelpFragment.kt должен выглядеть так, как показано в Листинге 5. Файл fragment_help.xml должен содержать следующую разметку, как показано в Листинге 6.

Листинг 5. Файл HelpFragment.kt

```
package com.example.lab9

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup

class HelpFragment : Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?,
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_help,
container, false)
```

```
16.      }
17.  }
```

Листинг 6. Файл fragment_help.xml

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
3.      xmlns:tools="http://schemas.android.com/tools"
4.      android:layout_width="match_parent"
5.      android:layout_height="match_parent"
6.      tools:context=".HelpFragment">
7.
8.      <!-- TODO: Update blank fragment layout -->
9.      <TextView
10.         android:layout_width="match_parent"
11.         android:layout_height="match_parent"
12.         android:text="Help" />
13.
14.  </FrameLayout>
15.
```

10. Добавим компонент Navigation средствами Gradle. Для этого, в файле build.gradle приложения добавим новую переменную с версией компонента Navigation, а затем необходимо добавить две зависимости компонента Navigation так, как показано в Листинге 7.

Листинг 7. Файл build.gradle.kts

```
dependencies {
    val nav_version = "2.8.2"
    ...
    implementation ("androidx.navigation:navigation-fragment-ktx:
$nav_version")
    implementation ("androidx.navigation:navigation-ui-ktx:
$nav_version")
    ...
    }
```

Нажмите кнопку Sync Now (в верзней части редактора) для синхронизации системы сборки.

11. Создадим новый граф навигации и добавим в него два фрагмента. Чтобы создать граф навигации, выделите папку app/res на панели проекта, а затем выберите команду File→New→Android Resource File. Введите имя файла «nav_graph», выберите тип ресурса «Navigation» и щелкните на кнопке OK. (Рис. 1).
12. Затем откройте граф навигации (файл nav_graph.xml), переключитесь в режим Code и обновите файл, чтобы он соответствовал приведенному коду, приведенному в Листинге 8. Приведенный код добавляет фрагменты InboxFragment и HelpFragment в граф навигации и назначает каждому из них метку, удобную для пользователя.

Листинг 8. Файл nav_graph.xml

```
<?xml version="1.0" encoding="utf-8"?>
<navigation
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/nav_graph"
    app:startDestination="@id/inboxFragment">

   <fragment
       android:id="@+id/inboxFragment"
       android:name="com.example.lab9.InboxFragment"
       layout="@layout/fragment_inbox"
       android:label="Inbox" />
    <fragment
       android:id="@+id/helpFragment"
       android:name="com.example.lab9.HelpFragment"
       layout="@layout/fragment_help"
       android:label="Help"/>

</navigation>
```

13. Как вы уже знаете, чтобы в приложении отображались цели, к которым вы переходите, следует включить в макет активности хост навигации. Для этого используется элемент FragmentContainerView, определяющий тип применяемого хоста навигации и имя графа навигации.

Обновите код activity_main.xml и включите в него изменения, кк показано в Листинге 9.

Листинг 9. Файл activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
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:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <com.google.android.material.appbar.MaterialToolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        style="@style/Widget.MaterialComponents.Toolbar.Primary"
        tools:ignore="MissingConstraints" />

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"

android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

14. Чтобы указать Android, какие элементы должны размещаться на панели инструментов, следует определить меню. Каждое меню определяется в XML-файле ресурсов меню. Мы создадим новый файл ресурсов меню с именем menu_toolbar.xml, который будет использоваться для добавления элемента Help на панель инструментов.

Выделите папку app/res, откройте меню File, выберите команду New, а затем — вариант создания нового файла ресурсов Android. Введите имя «menu_toolbar», укажите тип ресурса «Menu» и убедитесь в том, что выбрано имя каталога menu. При нажатии кнопки OK среда Android Studio создает файл и добавляет его в папку app/res/menu.

15. Перейти к фрагменту HelpFragment с панели инструментов, необходимо в файл ресурсов меню добавить элемент Help. Это будет сделано прямым редактированием кода XML. Переключитесь в режим Code для файла menu_toolbar.xml и обновите код, как показано в Листинге 10.

Листинг 10. Файл menu_toolbar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/helpFragment"
        android:icon="@android:drawable/ic_menu_help"
        android:title="Help"
        app:showAsAction="always" />

</menu>
```

Примечание. Каждый файл ресурсов меню, включая приведенный выше, содержит корневой элемент <menu>. Он сообщает Android, что в файле определяется меню. Внутри элемента <menu> обычно находится группа элементов <item>, каждый из которых описывает отдельный элемент меню. В нашем конкретном примере определяется один элемент Help с заголовком «Help»

- *Элемент <item> содержит ряд атрибутов, управляющих внешним видом элемента меню.*
- *Атрибут android:id назначает идентификатор элемента. Идентификатор используется компонентом Navigation для перехода к цели; он должен совпадать с идентификатором цели, к которой происходит переход на графе навигации.*
- *Атрибут android:icon указывает, какой значок должен отображаться для элемента (и должен ли). В Android есть много встроенных значков, и IDE выводит список доступных вариантов, когда вы начинаете вводить имя значка.*
- *Атрибут android:title определяет текст элемента.*
- *Атрибут app:showAsAction указывает, когда элемент должен появляться на панели инструментов. Значение «always» означает, что он всегда должен отображаться в основной области панели инструментов.*

16. После определения файла ресурсов меню необходимо добавить элементы на панель инструментов. Для этого в коде активности реализуется метод onCreateOptionsMenu(). Этот метод вызывается в тот момент, когда активность

готова к добавлению элементов на панель инструментов. Он заполняет файл ресурсов меню и добавляет каждый описанный в нем элемент на панель инструментов.

В приложении элемент, определенный в menu_toolbar.xml, должен быть добавлен на панель инструментов MainActivity.

Полный код MainActivity.kt

```
1.  package com.example.lab9
2.
3.  import android.os.Bundle
4.  import androidx.activity.enableEdgeToEdge
5.  import androidx.appcompat.app.AppCompatActivity
6.  import androidx.core.view.ViewCompat
7.  import androidx.core.view.WindowInsetsCompat
8.  import com.google.android.material.appbar.MaterialToolbar
9.  import android.view.Menu
10. import android.view.MenuItem
11. import androidx.navigation.findNavController
12. import androidx.navigation.fragment.NavHostFragment
13. import androidx.navigation.ui.AppBarConfiguration
14. import androidx.navigation.ui.onNavDestinationSelected
15. import androidx.navigation.ui.setupWithNavController
16.
17. class MainActivity : AppCompatActivity() {
18.     override fun onCreate(savedInstanceState: Bundle?) {
19.         super.onCreate(savedInstanceState)
20.         enableEdgeToEdge()
21.         setContentView(R.layout.activity_main)
22.         setContentView(R.layout.activity_main)
23.         val toolbar =
    findViewById<MaterialToolbar>(R.id.toolbar)
24.         setSupportActionBar(toolbar)
25.         setSupportActionBar(toolbar)
26.         val navHostFragment = supportFragmentManager
27.                 .findFragmentById(R.id.nav_host_fragment) as
    NavHostFragment
28.         val navController = navHostFragment.navController
```

```
29.         val builder =
    AppBarConfiguration.Builder(navController.graph)
30.         val appBarConfiguration = builder.build()
31.         toolbar.setupWithNavController(navController,
    appBarConfiguration)
32.
33.
    ViewCompat.setOnApplyWindowInsetsListener(findViewById(R.id.main)
    ) { v, insets ->
34.             val systemBars =
    insets.getInsets(WindowInsetsCompat.Type.systemBars())
35.             v.setPadding(systemBars.left, systemBars.top,
    systemBars.right, systemBars.bottom)
36.             insets
37.         }
38.     }
39.     override fun onCreateOptionsMenu(menu: Menu): Boolean {
40.         menuInflater.inflate(R.menu.menu_toolbar, menu)
41.         return super.onCreateOptionsMenu(menu)
42.     }
43.     override fun onOptionsItemSelected(item: MenuItem): Boolean
    {
44.         val navController =
    findNavController(R.id.nav_host_fragment)
45.         return item.onNavDestinationSelected(navController)
46.                 || super.onOptionsItemSelected(item)
47.     }
48. }
```

17.

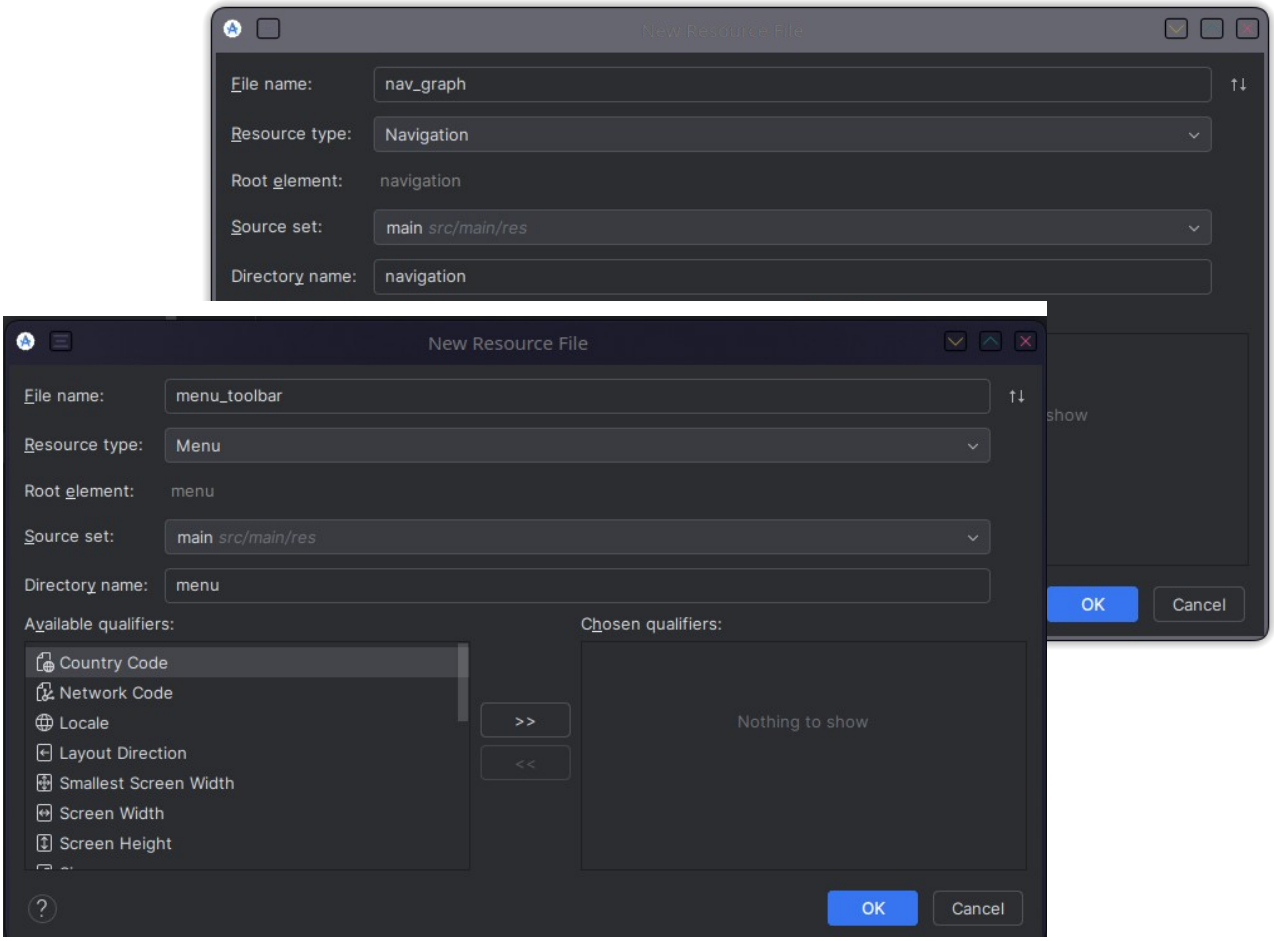


Рис. 2

3 Содержание отчета

Файл отчета должен содержать:

1. Краткое описание проекта;
2. Ход выполнения работы;
3. Листинги кода, с которым выполнялась работа;

4. Скриншоты приложения и формы.
5. В качестве вывода, опишите основные шаги по созданию многооконного мобильного приложения.