Очень немногие мобильные приложения состоят только из одного экрана. На самом деле большинство приложений состоят из нескольких экранов, по которым пользователь перемещается с помощью экранных жестов, щелчков кнопок и выбора пунктов меню. Соответственно навигация является одним из ключевых моментов при построении приложения под Android.

Каждый экран обычно представляет собой отдельный компонент или объект Activity. Подобные экраны Google еще называет термином **destination**, то пункт назначения, к которому может перейти пользователь. Каждое приложение имеет главный экран (home screen), который появляется после запуска приложения и после появления любого экрана-заставки. На этом главном экране пользователь обычно выполняет задачи, которые приводят к появлению других экранов или пунктов назначения. Эти экраны обычно принимают форму других компонентов проекта. Например, приложение для обмена сообщениями может иметь главный экран со списком текущих сообщений, с которого пользователь может перейти на другой экран для доступа к списку контактов или экрану настроек. Экран списка контактов, в свою очередь, может позволить пользователю переходить к другим экранам, где можно добавлять новых пользователей или обновлять существующие контакты.

Для отслеживания переходов по различным экранам Android использует стек навигации. При первом запуске приложения начальный экран помещается в стек. Когда пользователь переходит к другому экрану, то этот экран помещается на верхушку стека. Когда пользователь переходит к другим экранам, они также помещаются в стек. Когда пользователь перемещается обратно по экранам с помощью системной кнопки "Назад", каждый компонент извлекается из стека до тех пор, пока главный экран не окажется единственным элементом в стеке. Извлекать элементы из стеа можно также программно.

Подключение функционала навигации в проект

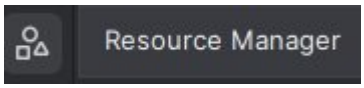
По умолчанию проект не содержит функционала навигации, и нам надо добавить все необходимые зависимости. Для этого можно вручную надо добавить соответствующую зависимость в файл **build.gradle**. Мы можем сделать это напрямую, указав в секции **dependencies** название зависимости:

```
dependencies {

implementation libs.androidx.navigation.fragment
implementation libs.androidx.navigation.ui

//ваши зависимости
}
```

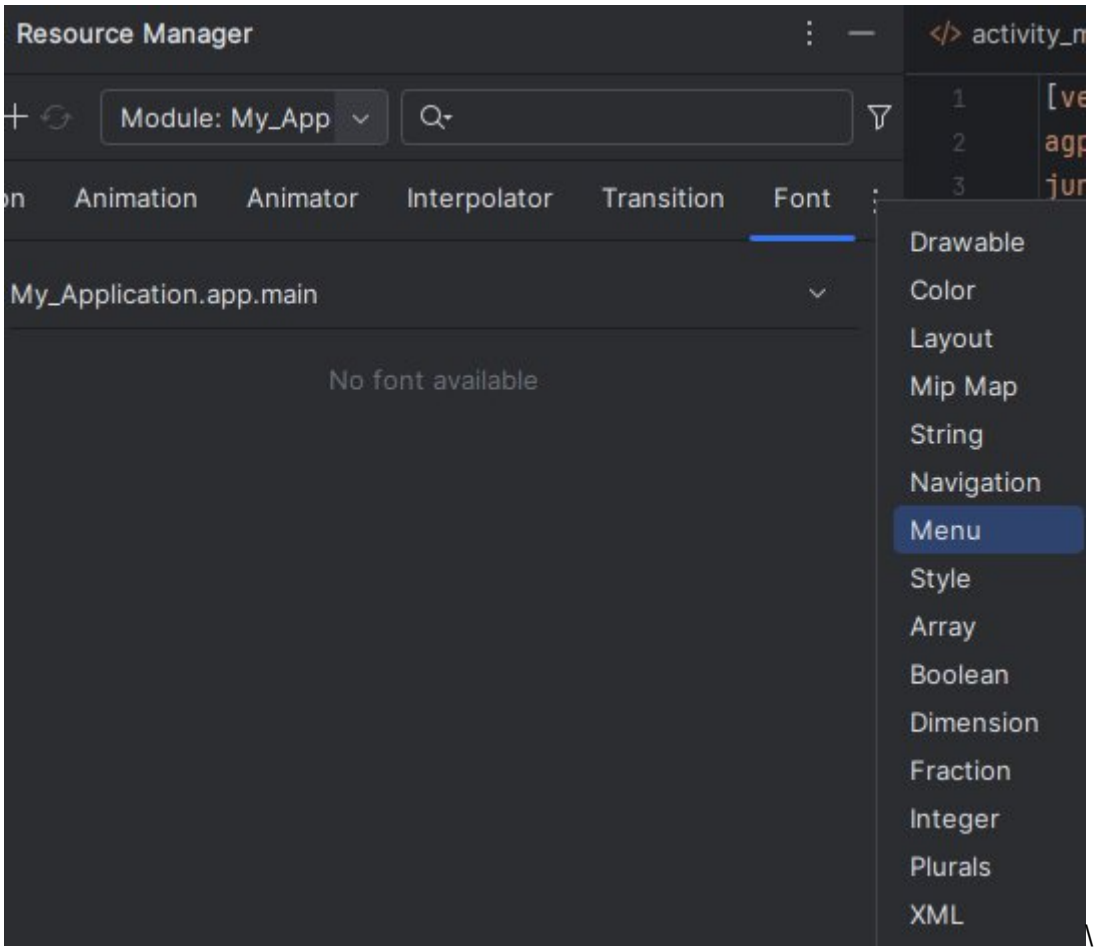
В файле **libs.version.toml** добавим в секцию `[versions]` версию подключаемой зависимости, а в секцию `[libraries]` имя подключаемой библиотеки:

```
[versions]
navigationUi = "2.5.3"
navigationFragment = "2.8.0"

[libraries]
androidx-navigation-fragment = { module = "androidx.navigation:navigation-fragment",
version.ref = "navigationFragment" }
androidx-navigation-ui = { module = "androidx.navigation:navigation-ui", version.ref =
"navigationUi" }
androidx-appcompat-v151 = { module = "androidx.appcompat:appcompat", version.ref =
"androidxAppcompat" }
```

Далее, перейдём в ResourceManager в боковой(справа) панели. В новой версии он выглядит так:

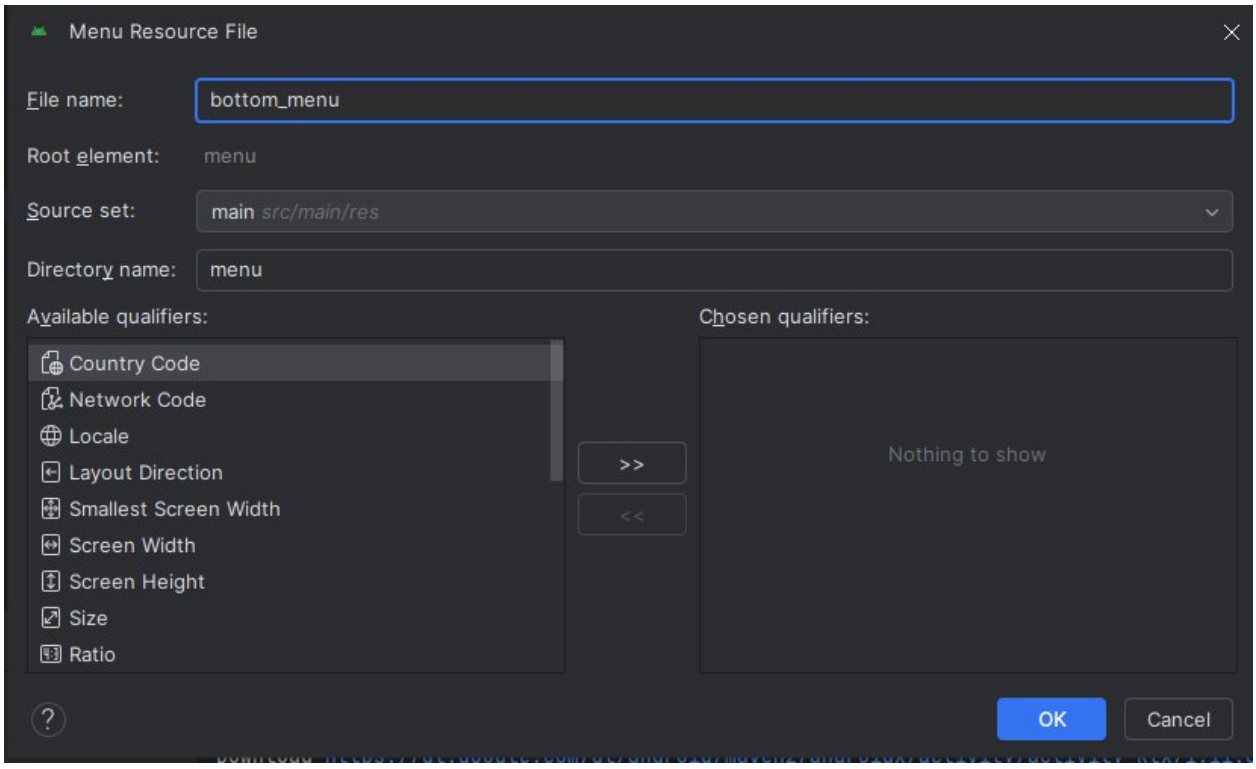


И найдём меню для создания меню:



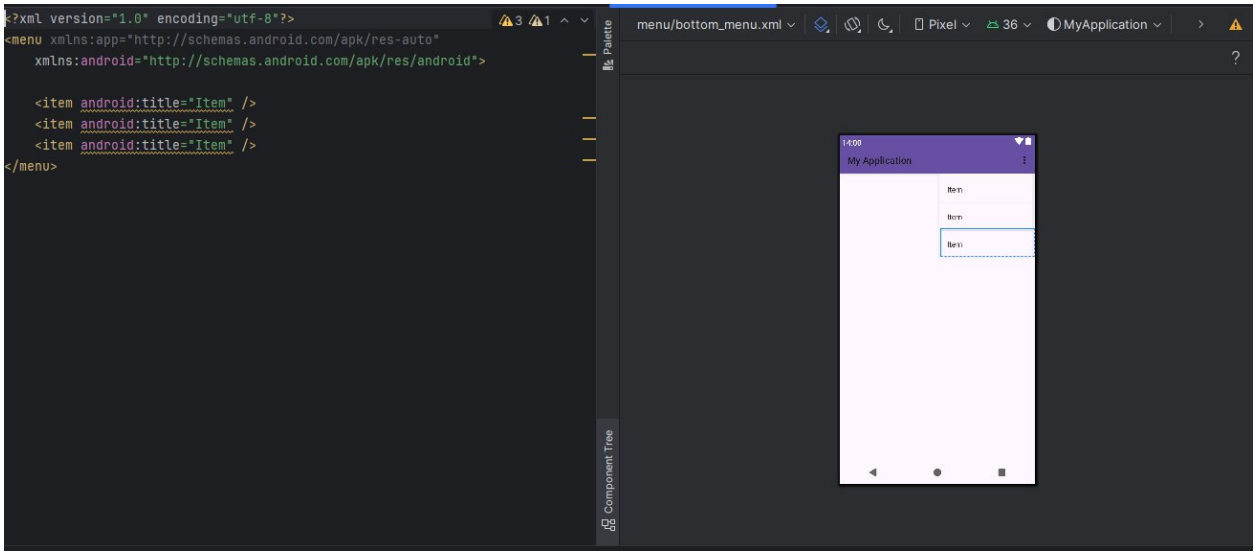
Нажмите + рядом с модулем вашего приложения и выберите **Menu Resource File.**

Создайте меню с каким-нибудь названием. К примеру:



И добавьте в него 3 **Menu item** в дизайнере, просто перетащив их на экран.

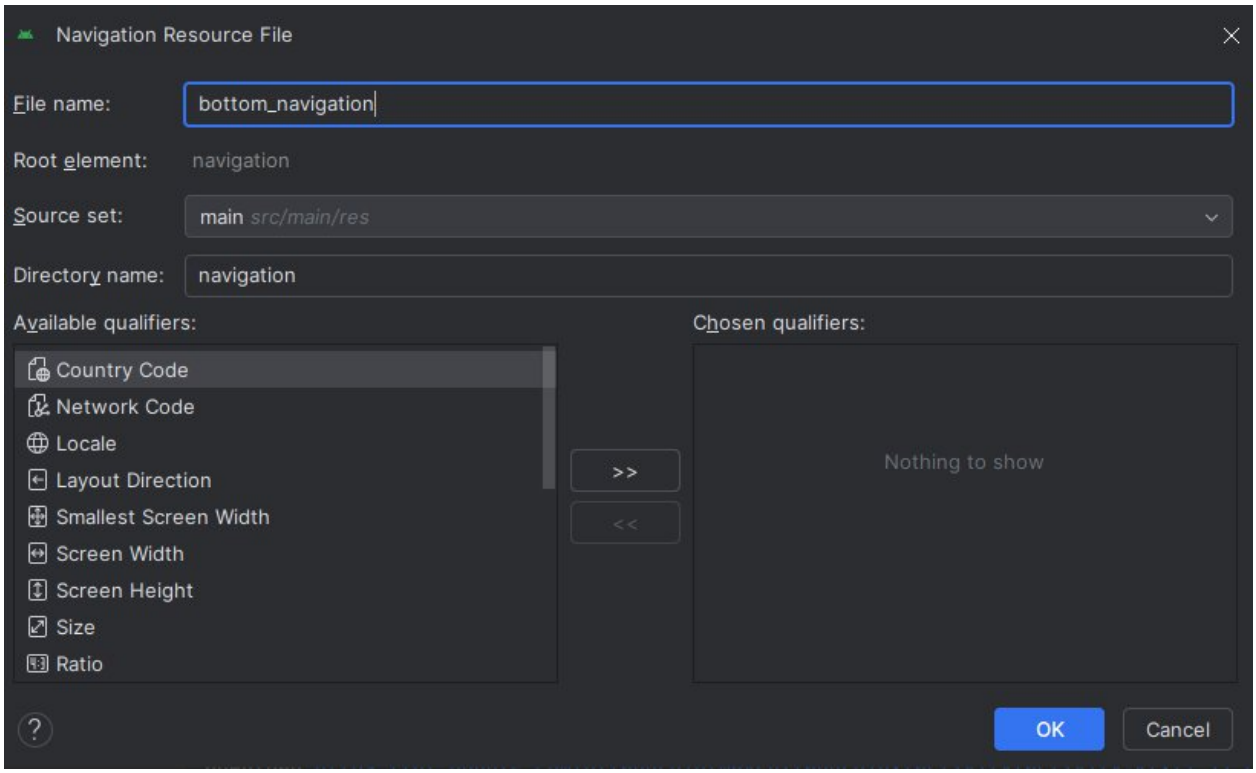
У вас получится вот так:



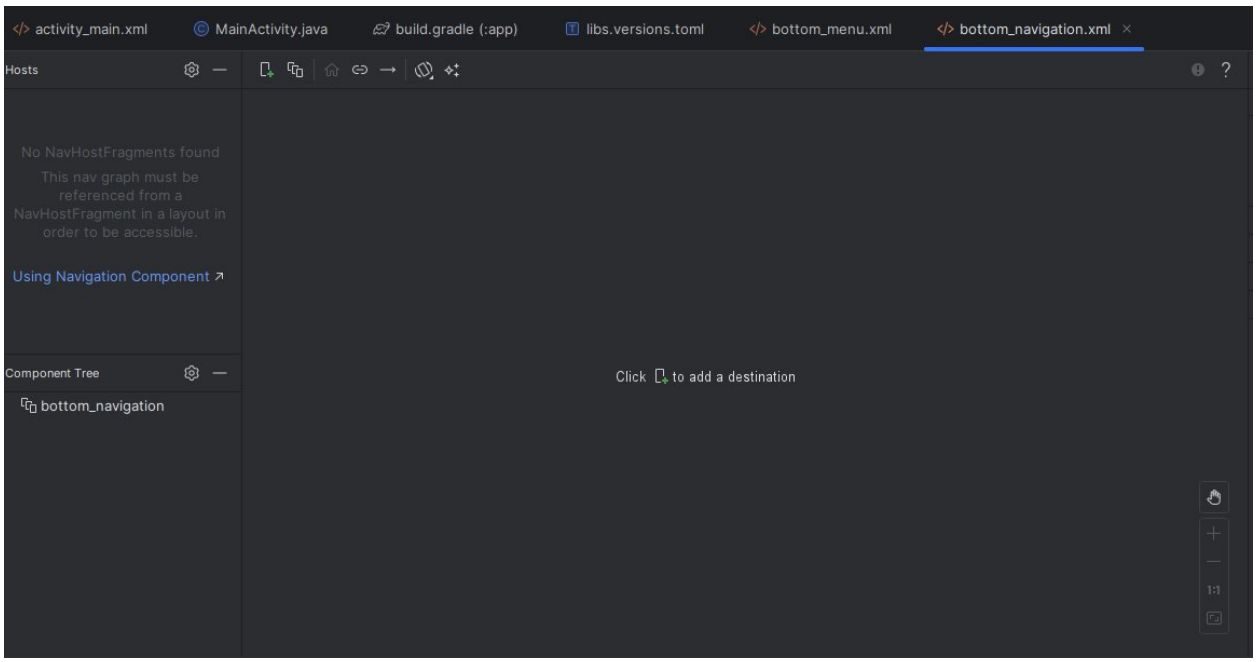
Элементам вашего меню можно проставить их название, которое будет выводится, а также иконку, с помощью **android:icon.** Об этом файле не забываем, к нему мы ещё вернёмся.

Перейдите обратно в Resource Manager и найдите меню с названием **Navigation.**

Создайте навигацию тем же способом, как и меню. У меня к примеру так:

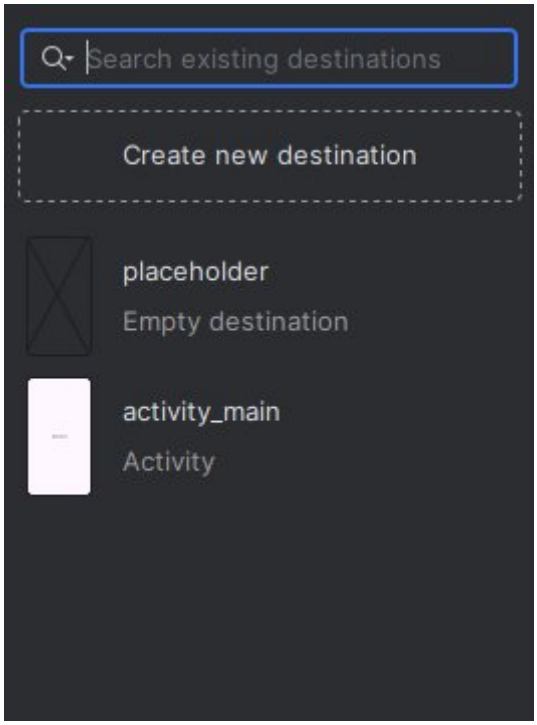


У вас получится примерно так:

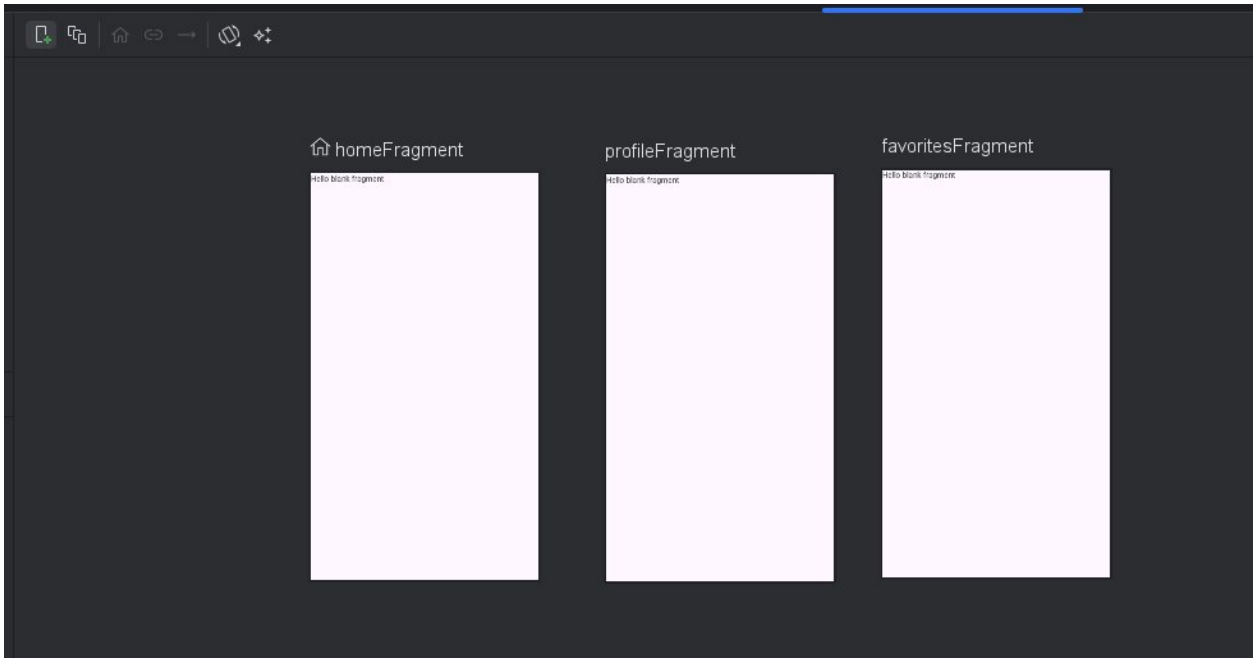


По нажатию

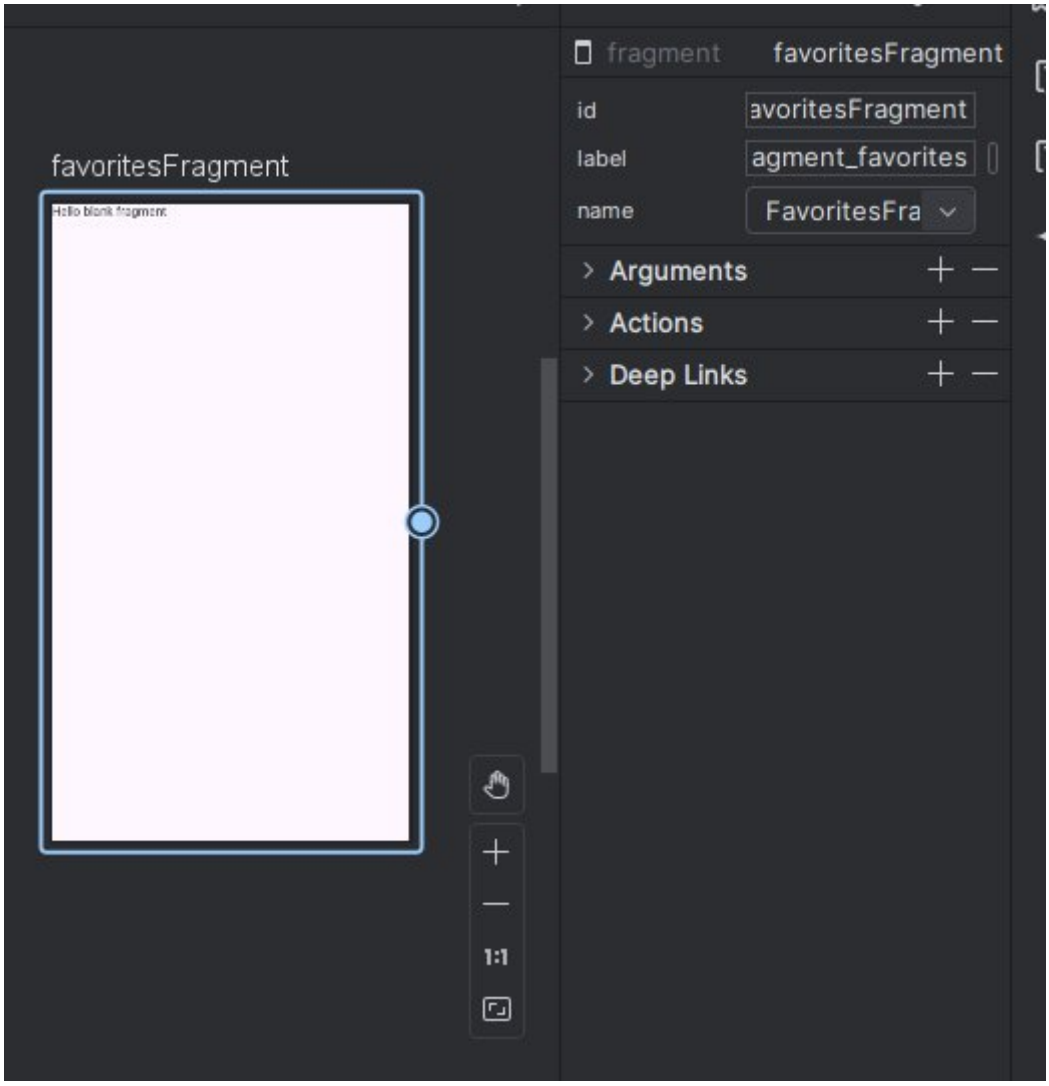
На левый элемент в верхнем меню откроется такая панель:



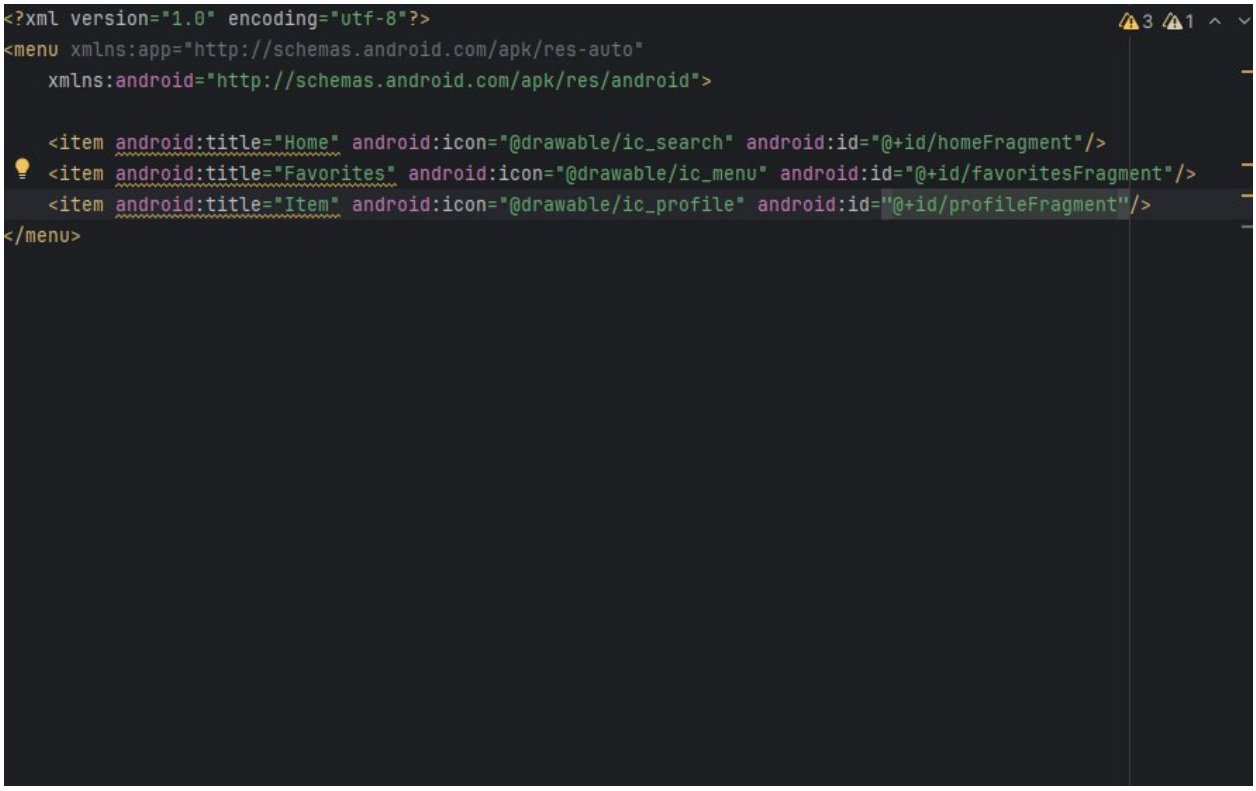
У вас там будут ваши фрагменты и Activity. Добавьте сюда уже существующие фрагменты или создайте новые. У меня их будет 3:



Кликнув на фрагменты вы справа сможете увидеть панель для управления ими:



Нам понадобятся их id. Вернитесь на файл с меню и добавьте их id с помощью **android:id** в каждый из подходящих элементов:



Вернёмся в **activity_main.xml**, чтобы при запуске приложения было видно нашу навигацию. И пропишем следующий код:

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/fragmentContainerView"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    app:defaultNavHost="true"
    app:layout_constraintBottom_toTopOf="@+id/bottomNavigationView"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:navGraph="@navigation/bottom_navigation" />
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/bottomNavigationView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHeight_percent="0.07"
    app:layout_constraintHorizontal_bias="0.497"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0"
    app:menu="@menu/bottom_menu" />
```

В **app:navGraph** вы помещаете свой файл навигации. В **android:name** вы обязаны прописать **androidx.navigation.fragment.NavHostFragment**, иначе ничего не запустится. И не забудьте **app:defaultNavHost="true"**, тоже важный элемент навигации.В BottomNavigationView в **app:menu** вы прописываете название вашего файла меню.

Далее, переходим в файл **MainActivity.java.** И в нём добавляем следующий код:

```
public class MainActivity extends AppCompatActivity {
    private NavController navController;
    protected BottomNavigationView bottomNavigationView;
    protected FragmentContainerView fragmentContainerView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        bottomNavigationView = findViewById(R.id.bottomNavigationView);
        fragmentContainerView = findViewById(R.id.fragmentContainerView);
        NavHostFragment navHostFragment = (NavHostFragment)
```

```
getSupportFragmentManager()
                .findFragmentById(R.id.fragmentContainerView);
        navController = navHostFragment.getNavController();
        NavigationUI.setupWithNavController(bottomNavigationView, navController);

          }

}
```

МОЙ КОД НЕ ОБЯЗАН СОВПАДАТЬ С ВАШИМ. Данные элементы должны будут у вас присутствовать для работы. Можете запустить ваше приложение и увидите работающее меню.

Также теперь мы можем не использовать громоздкий код для перехода из фрагмента в фрагмент.

Вместо вот этого:

```
getActivity().getSupportFragmentManager().beginTransaction()
        .replace(R.id.Layout_container, nextFrag, "findThisFragment")
        .addToBackStack(null)
        .commit()
```

Мы можем использовать NavController. Примерно следующим образом:

```
NavController navController = Navigation.findNavController(requireView());
navController.navigate(
        R.id.profileFragment,
        null,
        new NavOptions.Builder()
                .setPopUpTo(R.id.loginFragment, true)
                .build(),
        null
);
```

Стоит упомянуть, что NavController может прыгать только по фрагментам, которые вы добавили в файл навигации.

В navController.navigate() вы прописываете параметры перемещения. Сначала идёт название фрагмента, потом Bundle с данными, которые вы хотите ему передать, после NavOptions и navExtras для специфического поведения.

NavOptions, в данном случае, опциональны. С помощью них можно очистить backStack, т.е при нажатии на кнопку возвращения, вас не вернёт на предыдущий фрагмент, а оставит на нынешнем. Можно установить анимацию перехода с фрагмента на фрагмент и т.д.