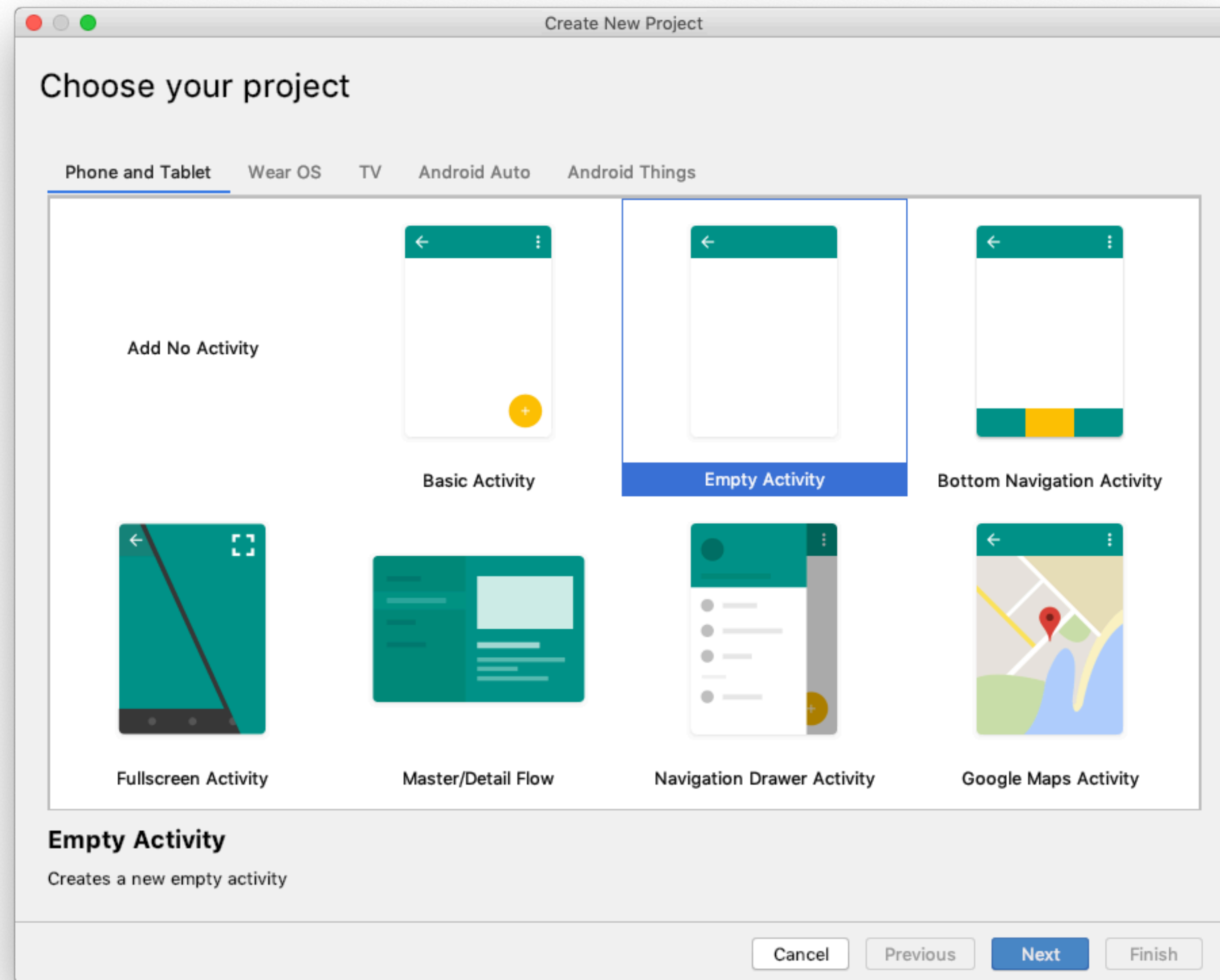


Navigation Components

Jetpack Architecture

Features of Navigation Components

- Handling fragment transaction
- Handling Up and Back actions correctly
- Providing standardized resources for animations and transitions
- Including Navigation UI patterns
- Providing type safety when passing information
- Visualizing and editing navigation from Android Studio Navigation Editor



Create New Project

Configure your project

←

Empty Activity

Creates a new empty activity

Name

Shop

Package name

at.htl.shop

Save location

droid.kotlin/2019.jetpack.udemy/03.Navigation.Components/Shop

Language

Kotlin

Minimum API level

API 23: Android 6.0 (Marshmallow)

i

Your app will run on approximately 62.6% of devices.

[Help me choose](#)

☐

This project will support instant apps

☒

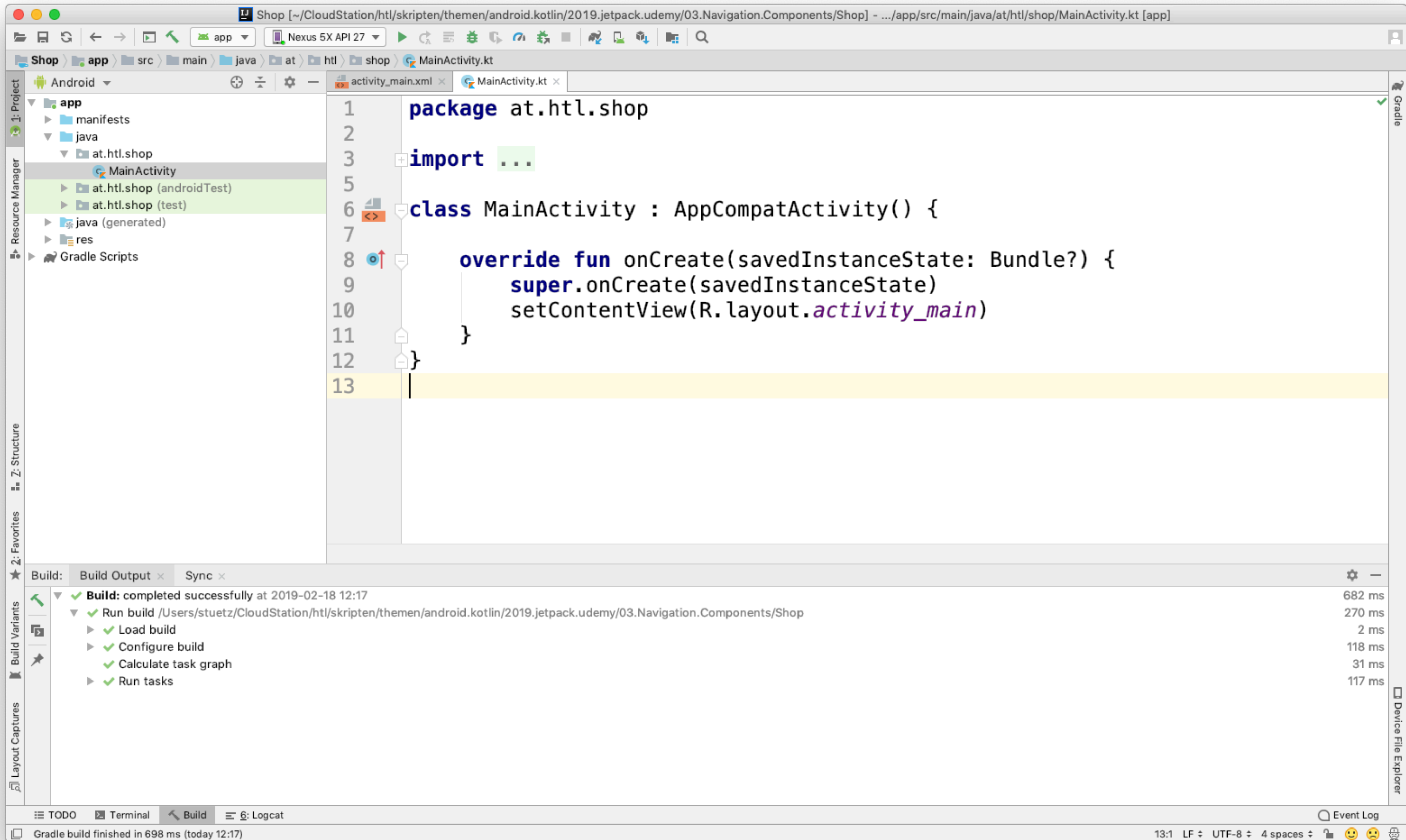
Use AndroidX artifacts

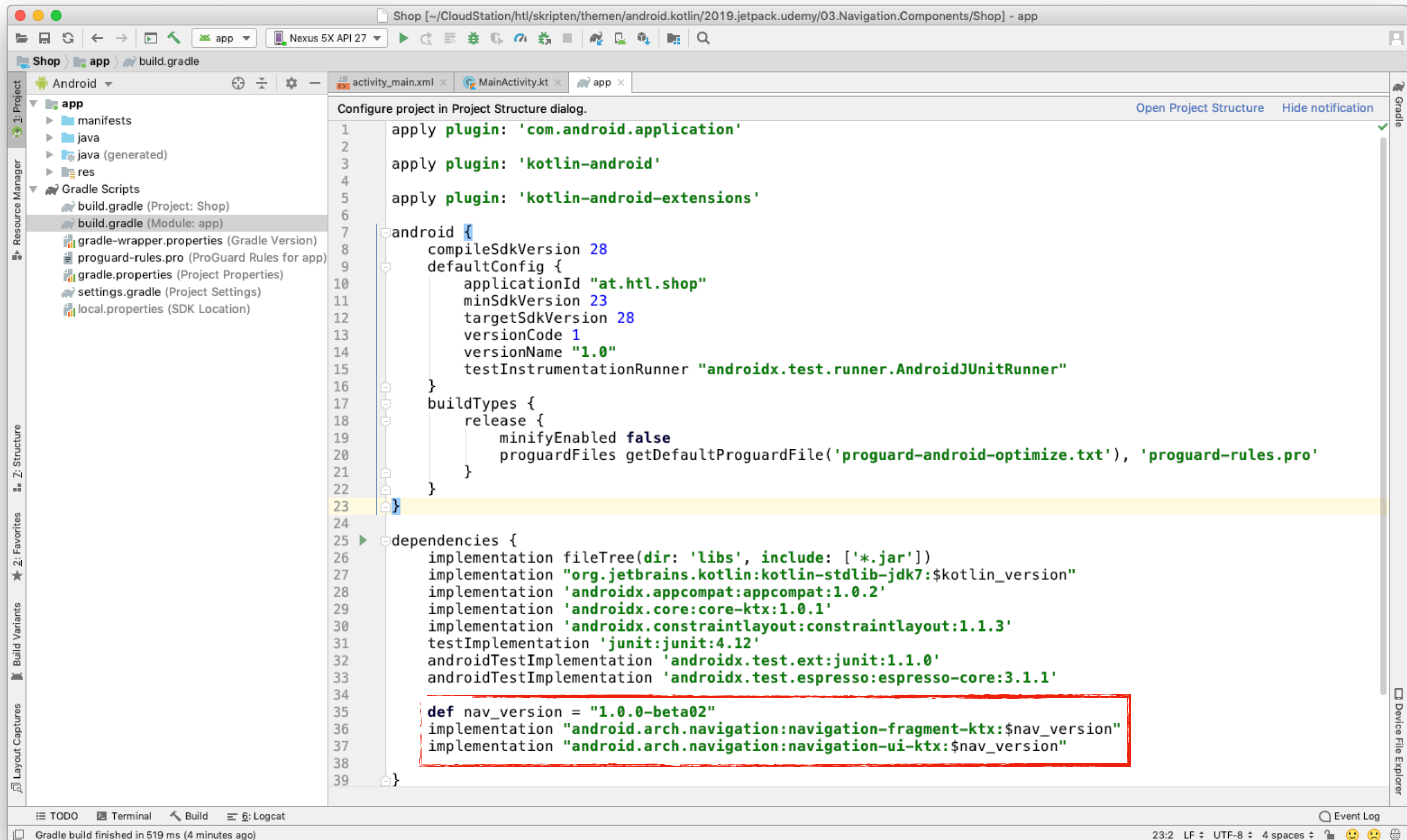
Cancel

Previous

Next

Finish





<https://developer.android.com/jetpack/androidx/releases/navigation>

values/strings.xml

```
<resources>
  <string name="app_name">Shop</string>
  <string name="home">Home</string>
  <string name="cart">Cart</string>
  <string name="hello_blank_fragment">Hello blank fragment</string>
  <string name="detail">We empower small and medium-sized businesses 1</string>
  <string name="about_us">About Us</string>
  <string name="shop">Shop</string>
</resources>
```

We empower small and medium-sized businesses to reach millions of customers with a number of programmes that help boost their revenue, reach and productivity.

values/styles.xml

```
<resources>
```

```
    <!-- Base application theme. -->
```

```
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
```

```
        <!-- Customize your theme here. -->
```

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

```
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
```

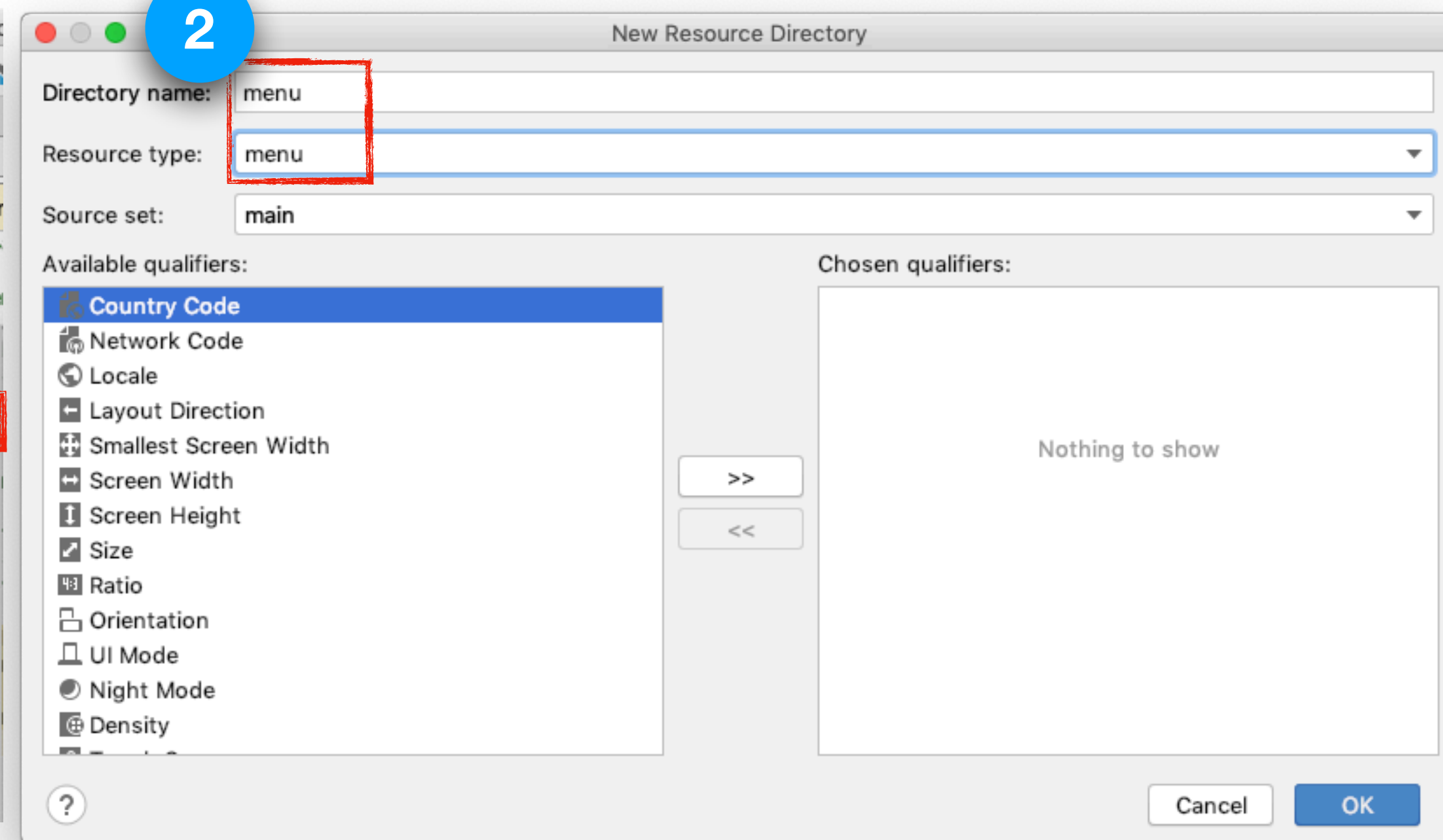
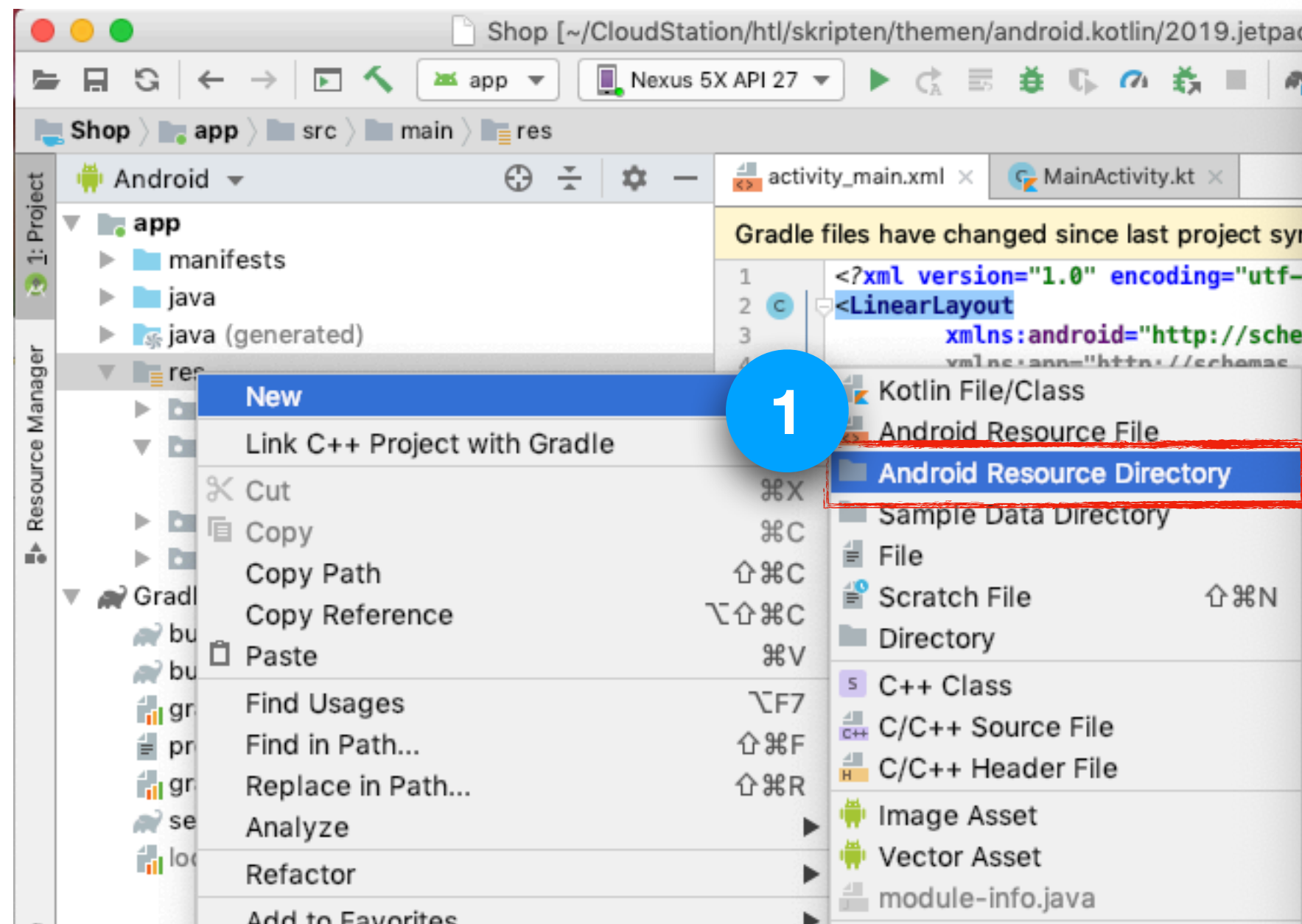
```
        <item name="colorAccent">@color/colorAccent</item>
```

```
    </style>
```

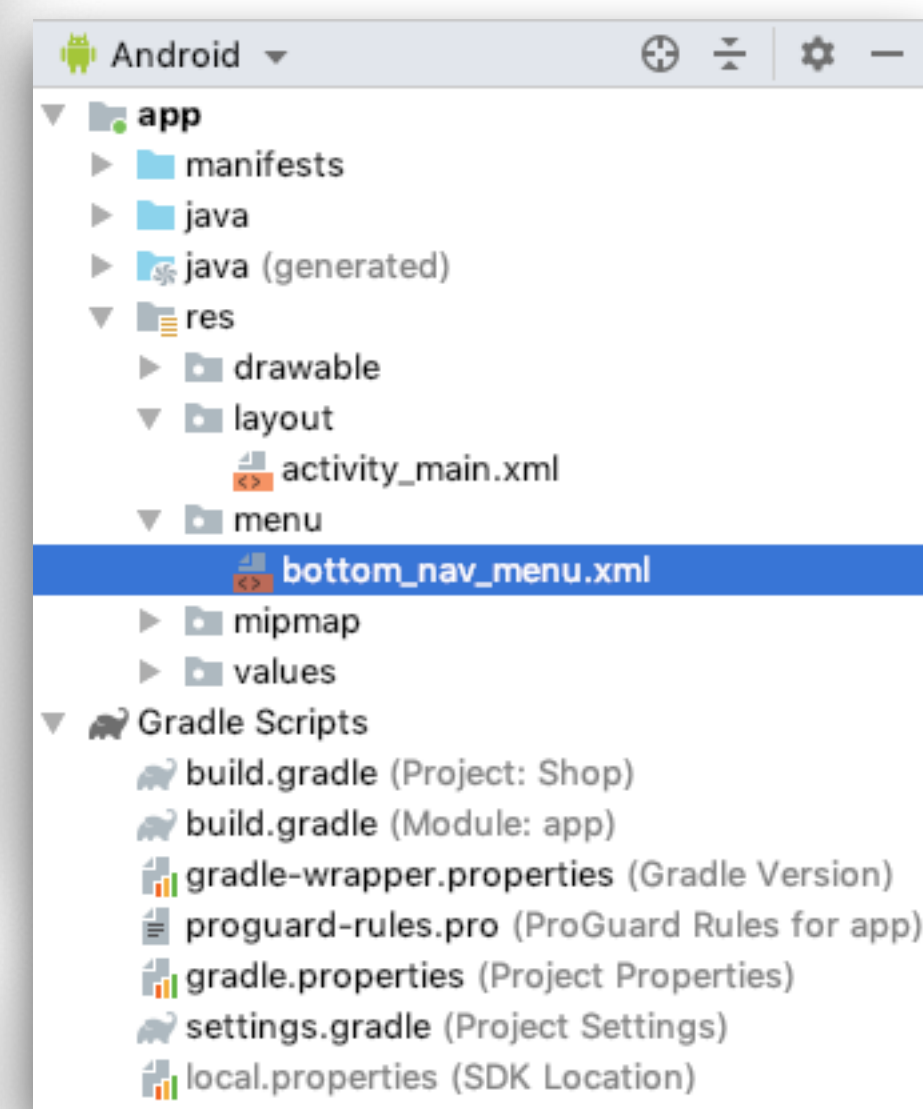
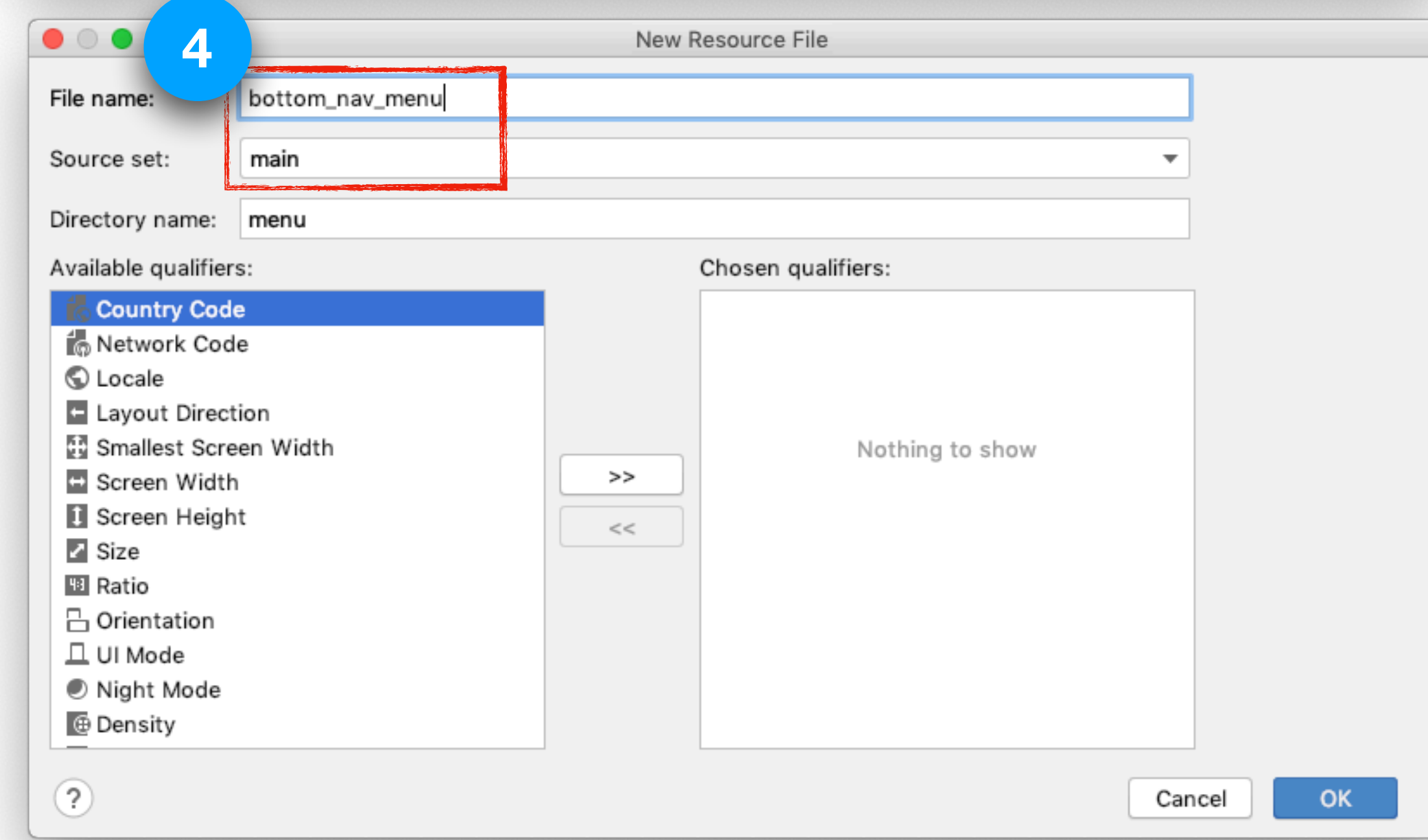
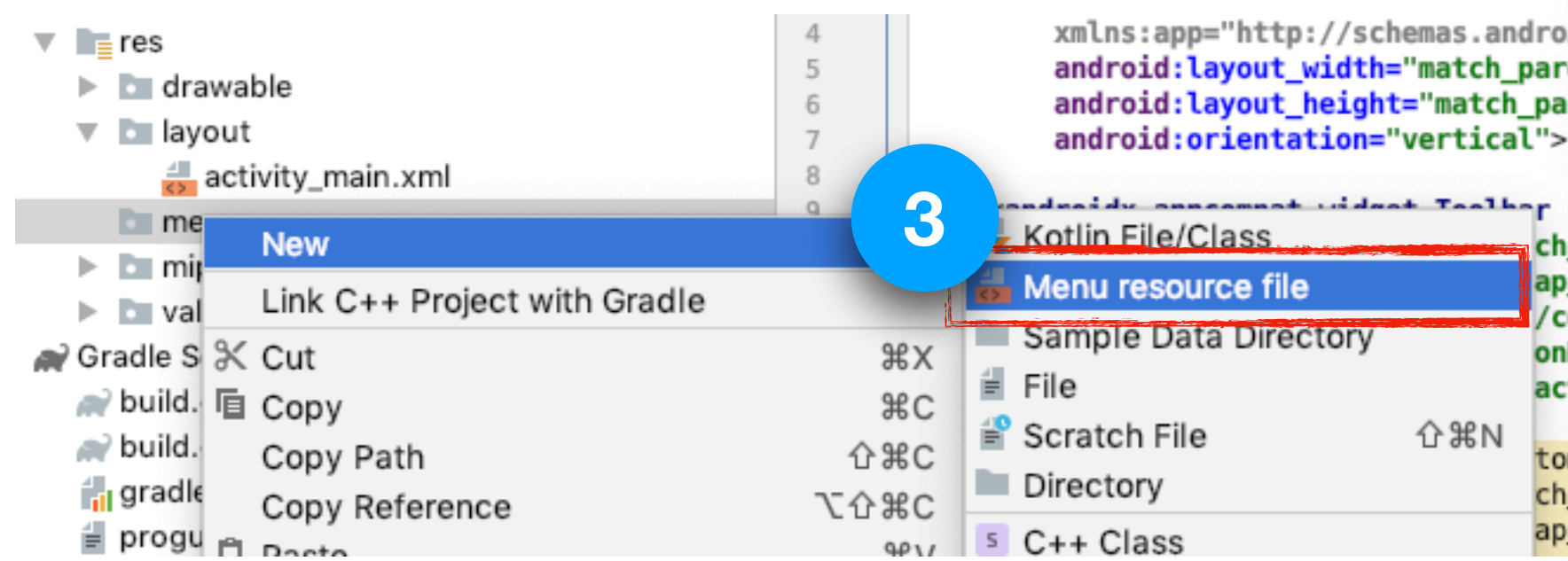
```
</resources>
```

values/colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#D81B60</color>
</resources>
```

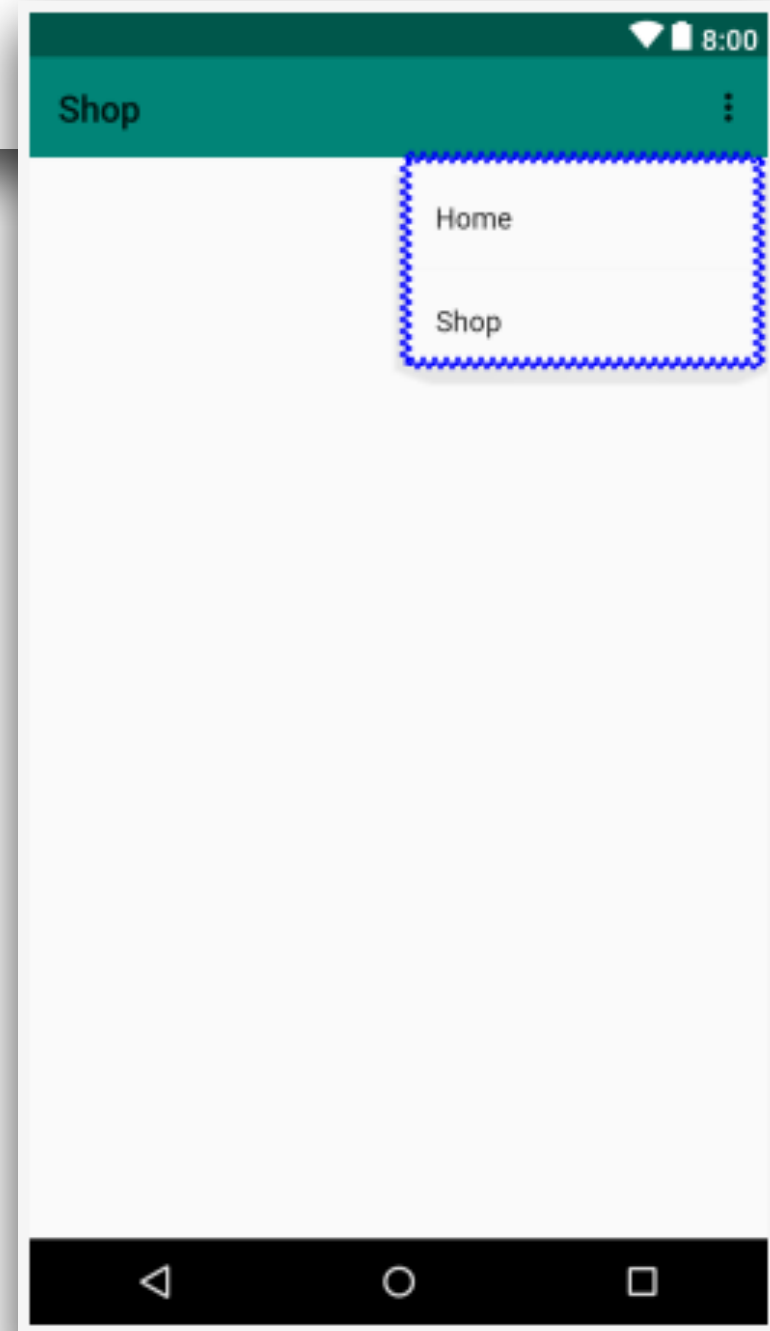



Erstellen des Menü-Ressourcenordners und eines Menüs



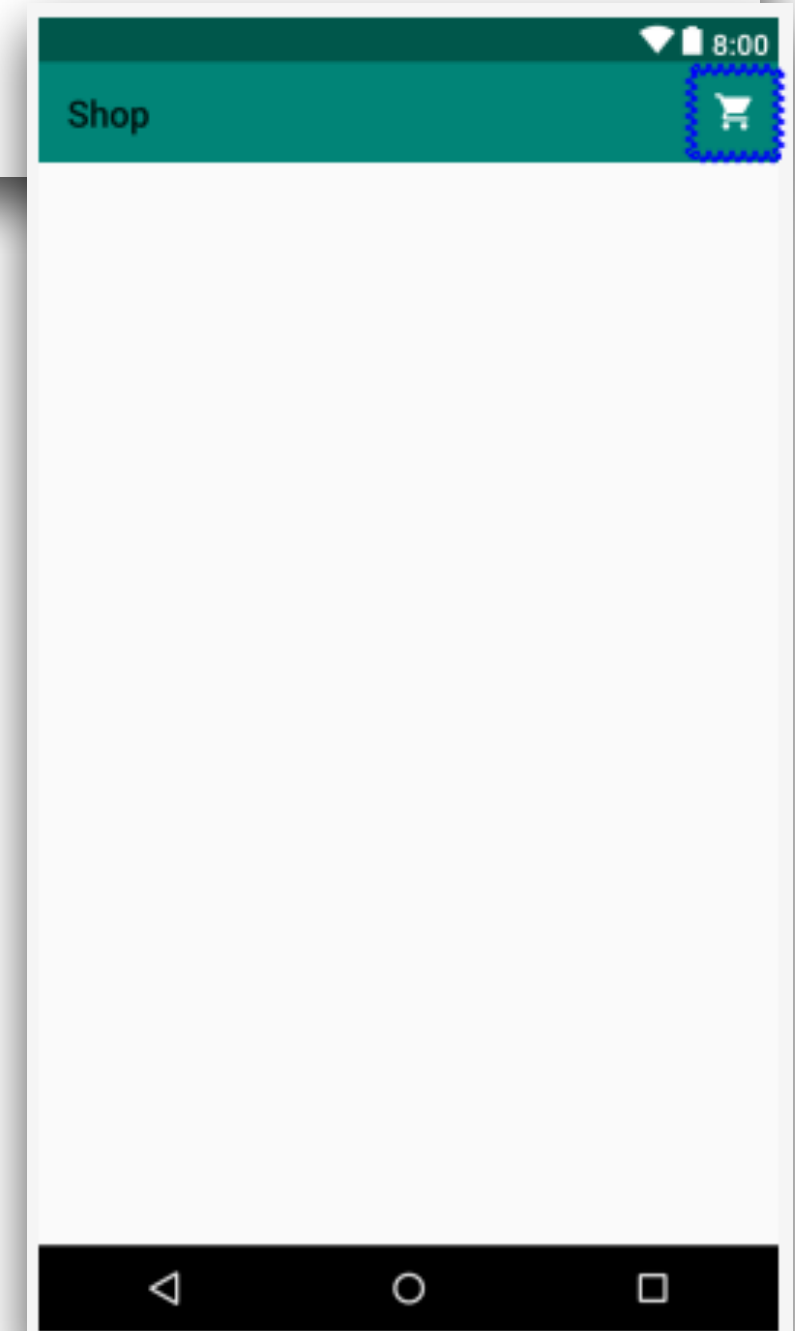
Menüs

```
bottom_nav_menu.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android">
3
4      <item
5          android:id="@+id/home_destination"
6          android:title="Home"
7          android:icon="@drawable/ic_home"
8      />
9
10     <item
11         android:id="@+id/shop_destination"
12         android:title="Shop"
13         android:icon="@drawable/ic_shop"
14     />
15 </menu>
```

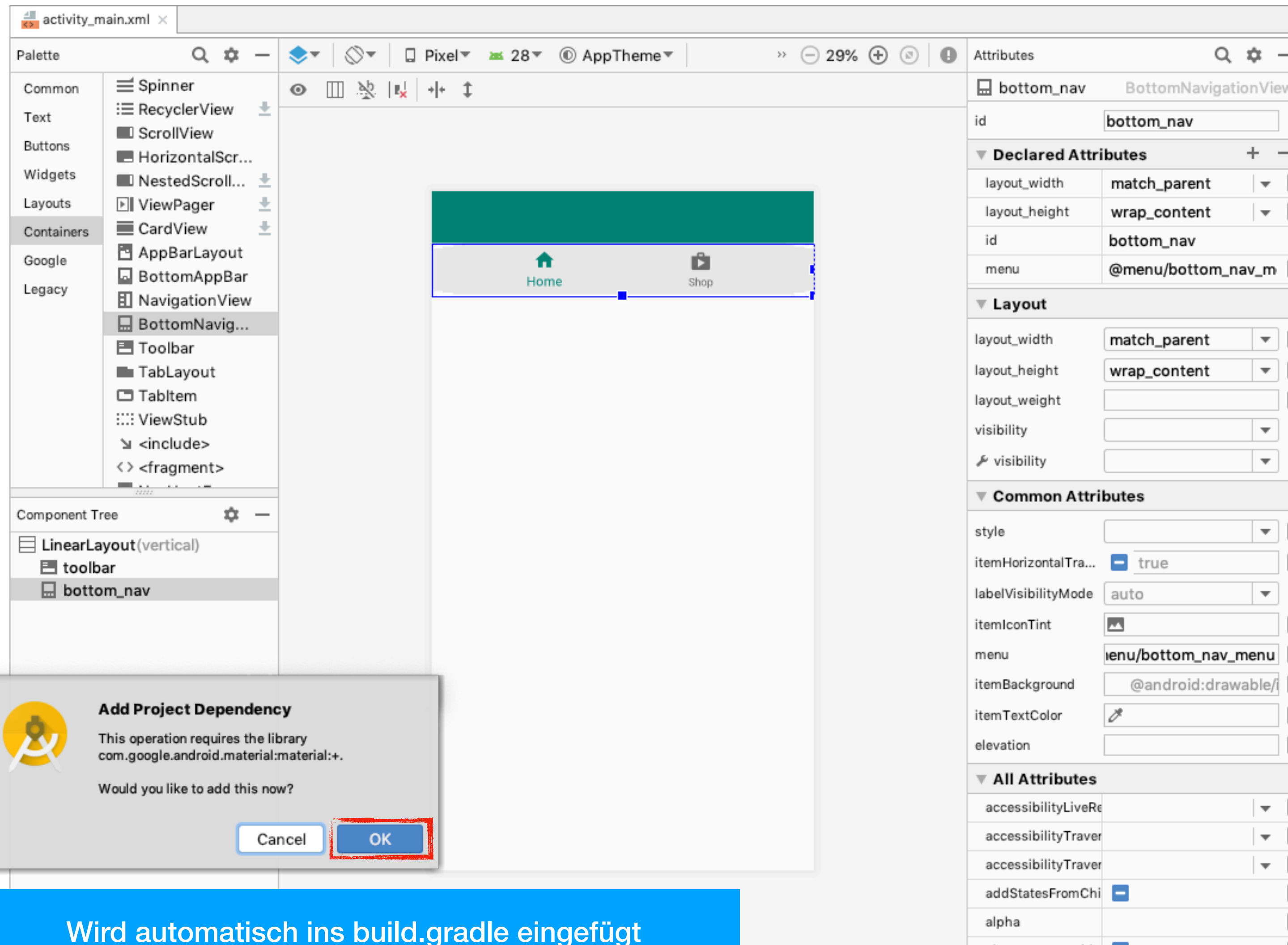


Erstellen Sie ein weiteres Menü: toolbar_menu.xml

```
toolbar_menu.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android"
3        xmlns:app="http://schemas.android.com/apk/res-auto">
4
5      <item
6          android:id="@+id/cart_destination"
7          android:icon="@drawable/ic_shopping_cart"
8          android:title="Cart"
9          app:showAsAction="ifRoom"
10     />
11 </menu>
```



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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

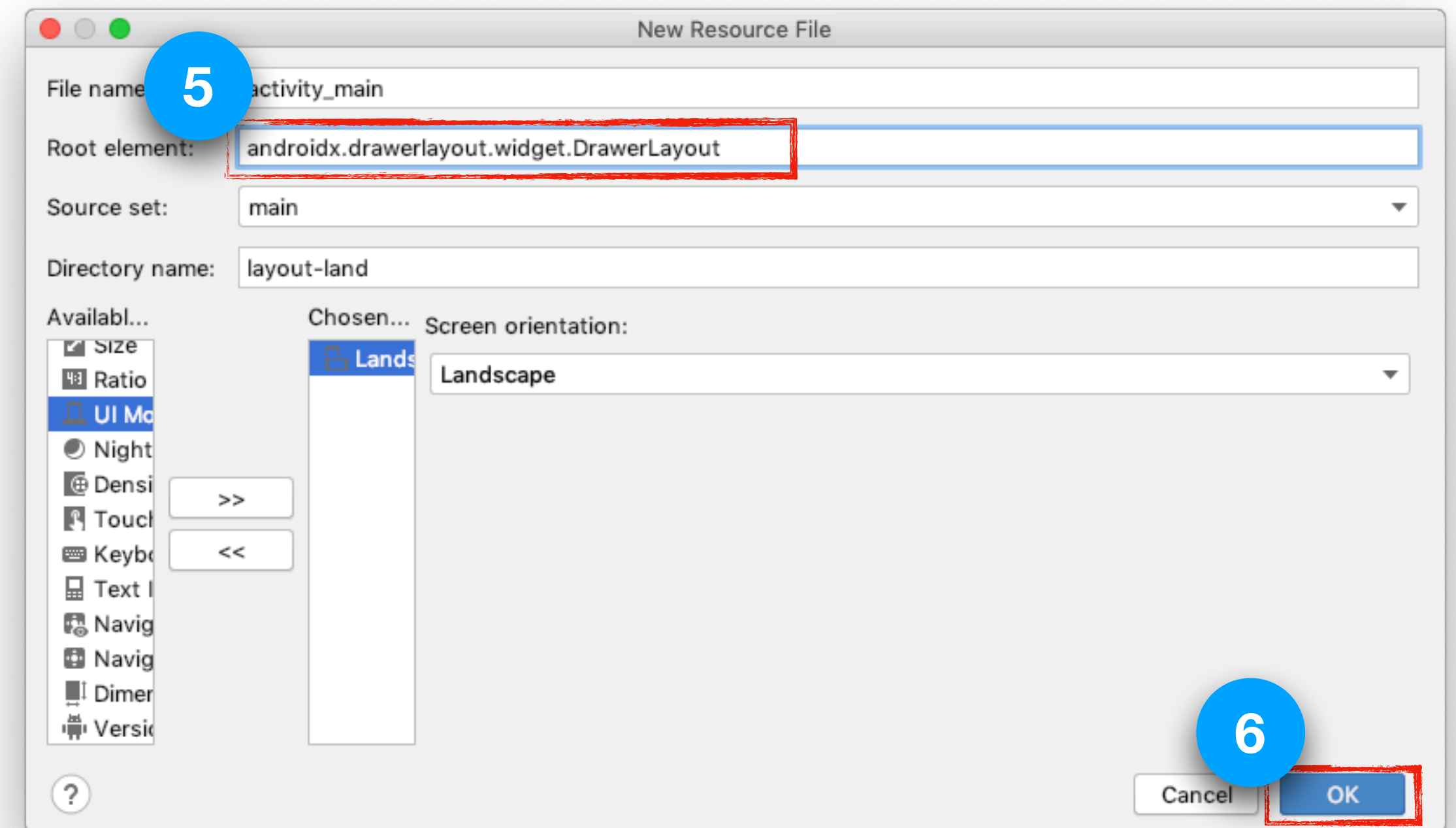
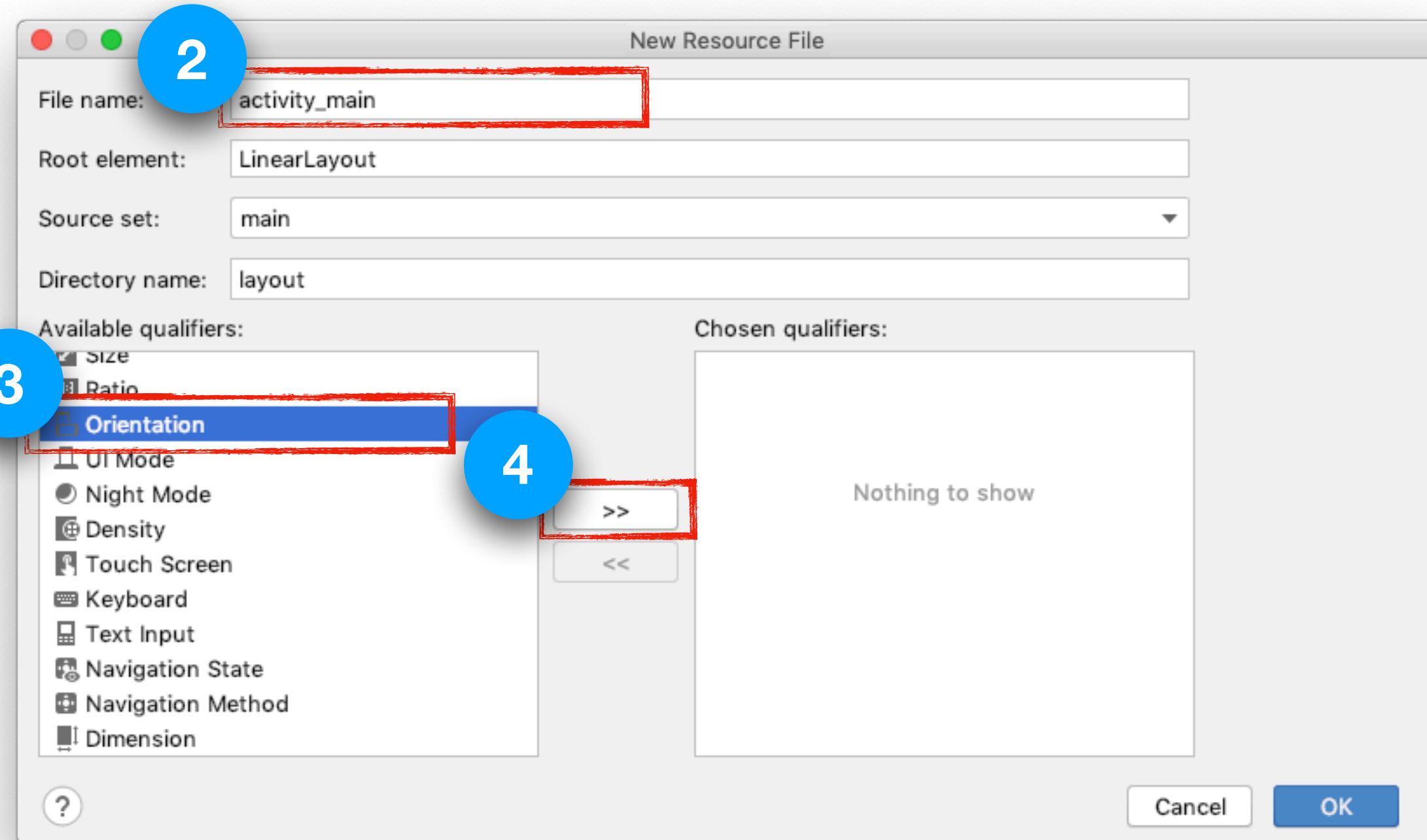
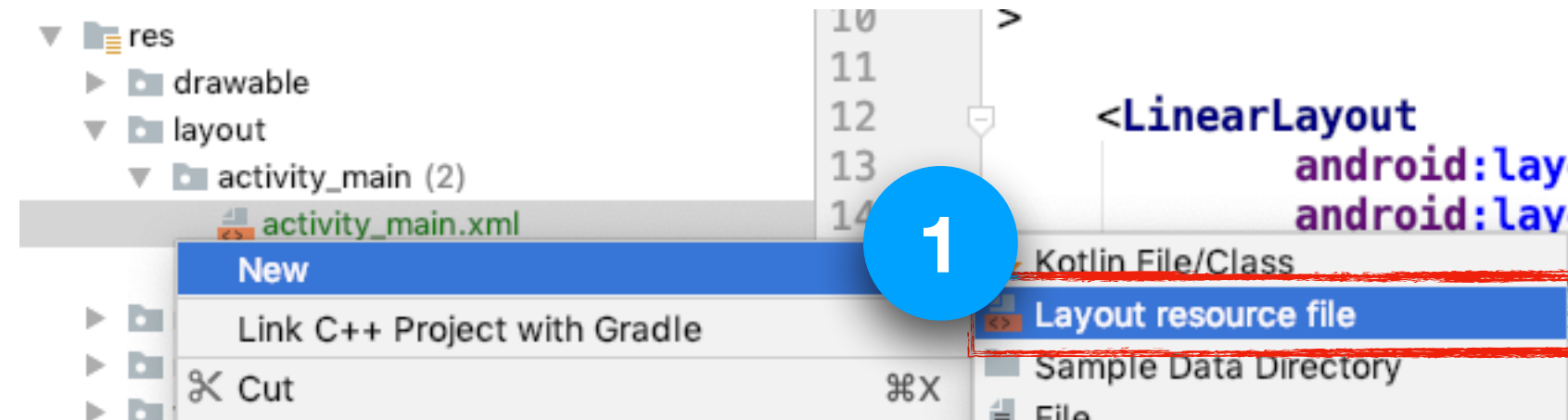
    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimary"
        android:theme="@style/ThemeOverlay.AppCompat.Dark"/>

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_nav"
        app:menu="@menu/bottom_nav_menu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

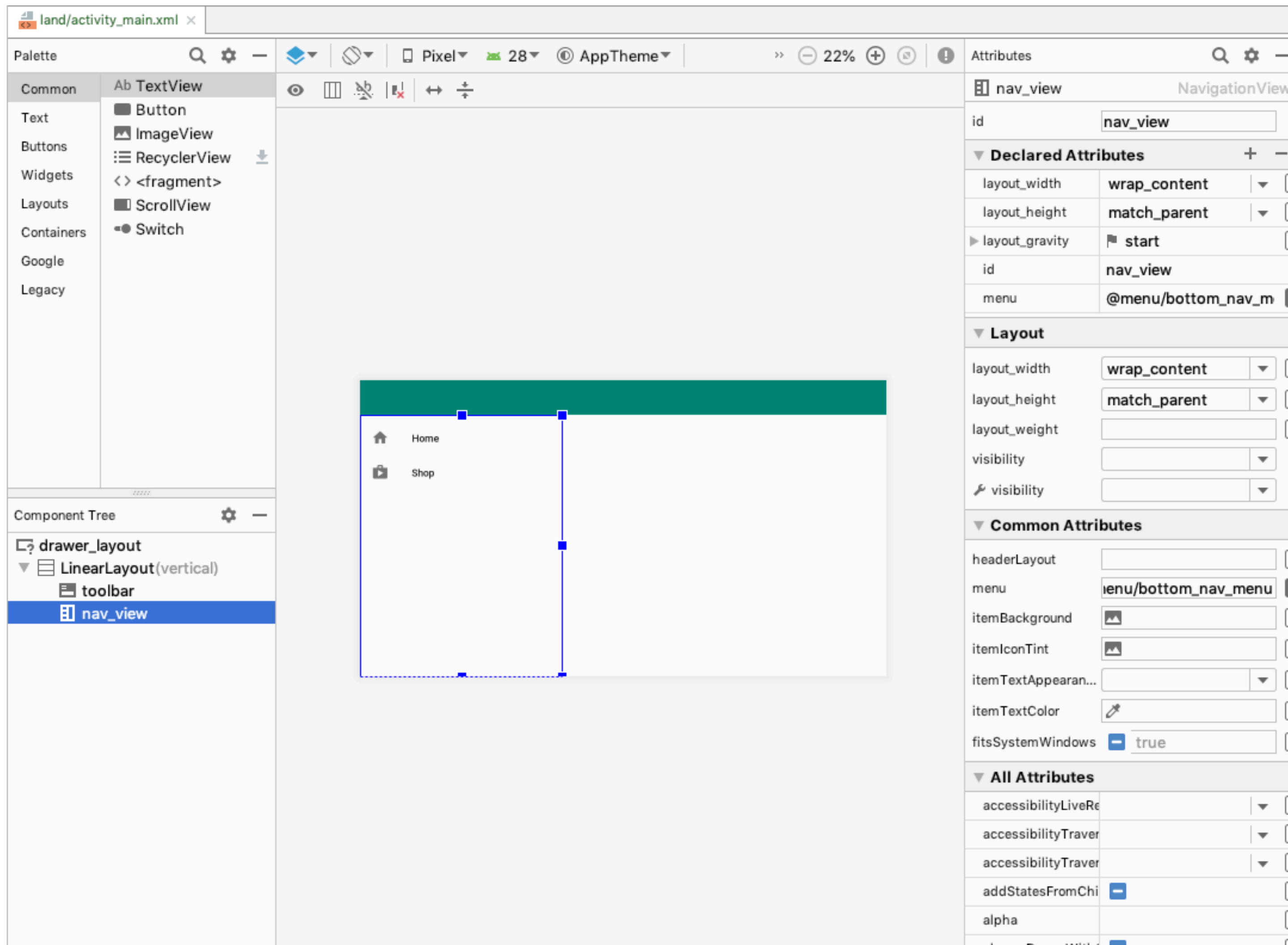
</LinearLayout>
```

implementation 'com.google.android.material:material:1.0.0'

activity_main für Landscape erstellen



land/activity_main.xml



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

```
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/colorPrimary"
            android:theme="@style/ThemeOverlay.AppCompat.Dark"/>

        <com.google.android.material.navigation.NavigationView
            android:id="@+id/nav_view"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="start"
            app:menu="@menu/bottom_nav_menu"/>

    </LinearLayout>
```

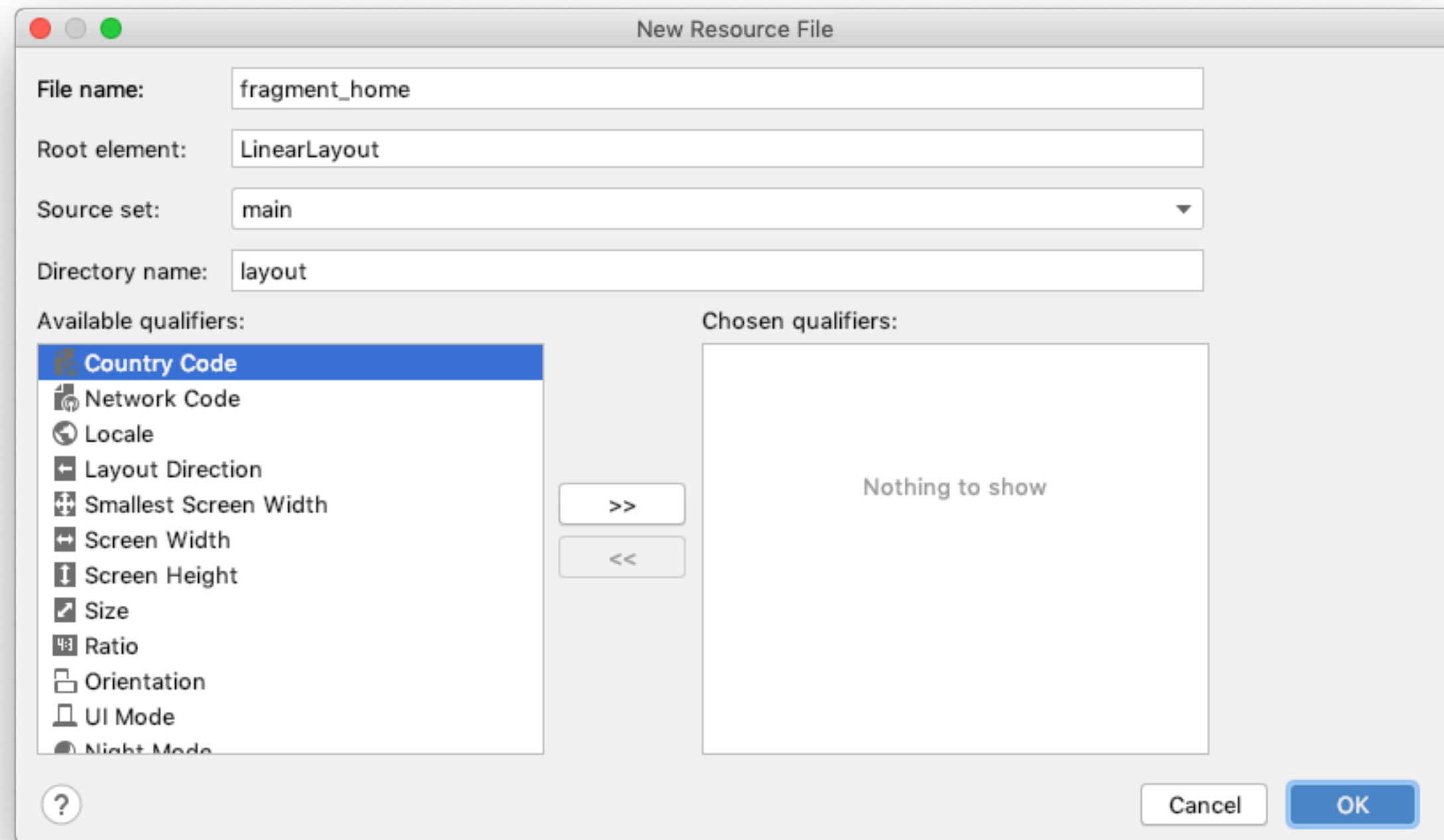
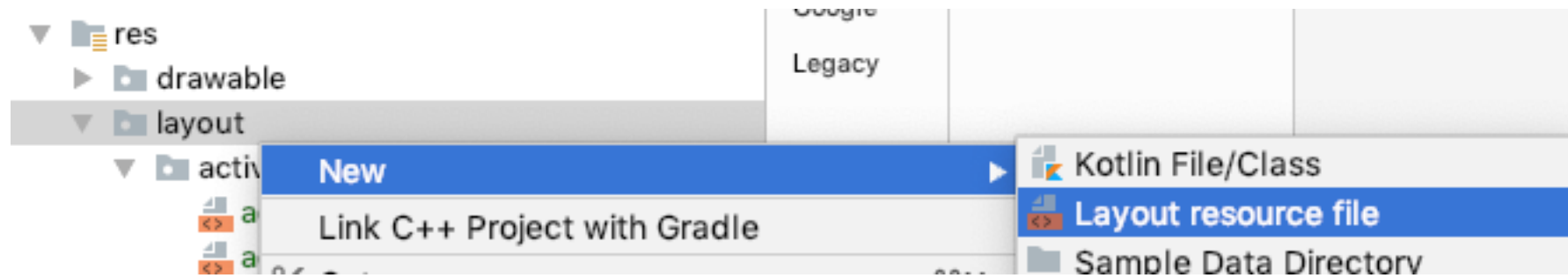
```
</androidx.drawerlayout.widget.DrawerLayout>
```

Was sind DrawerLayouts?

<https://youtu.be/DkT0vS14Um0>

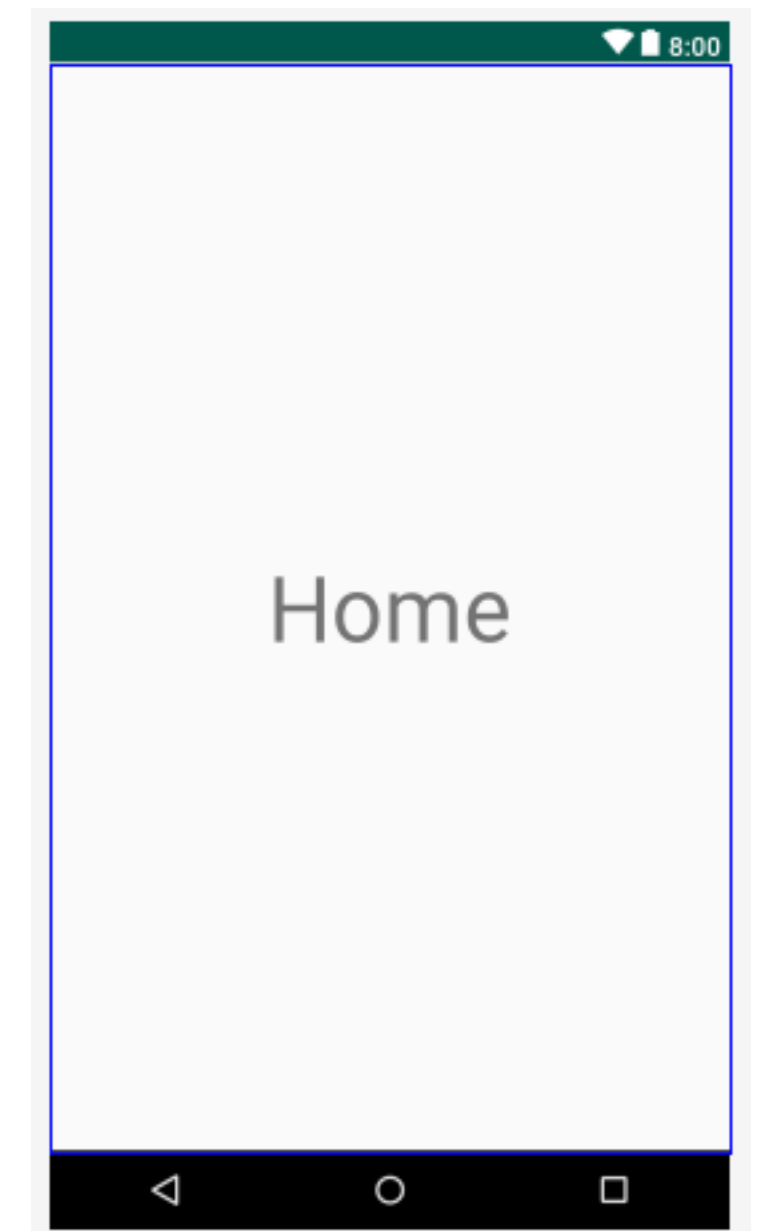
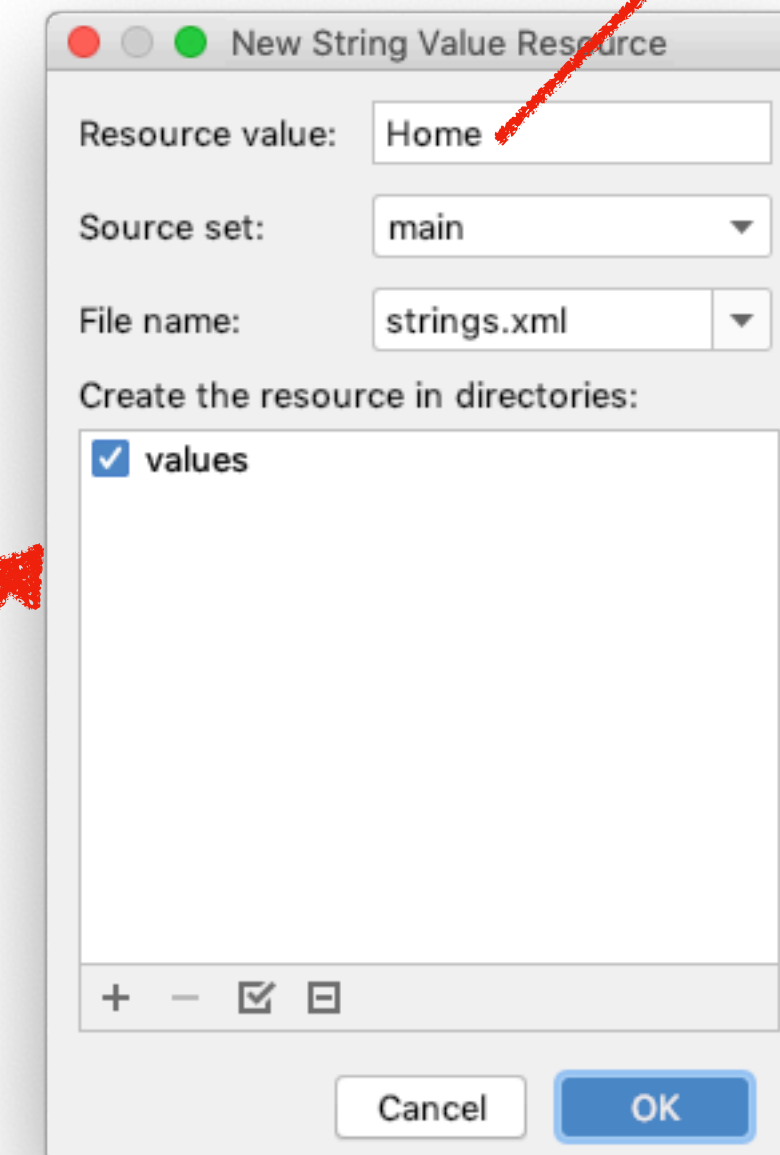
[Implementing an Android Navigation Drawer in Kotlin](#)

fragment_home.xml



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center">
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="55sp"
        android:text="@string/home" />
</LinearLayout>
```



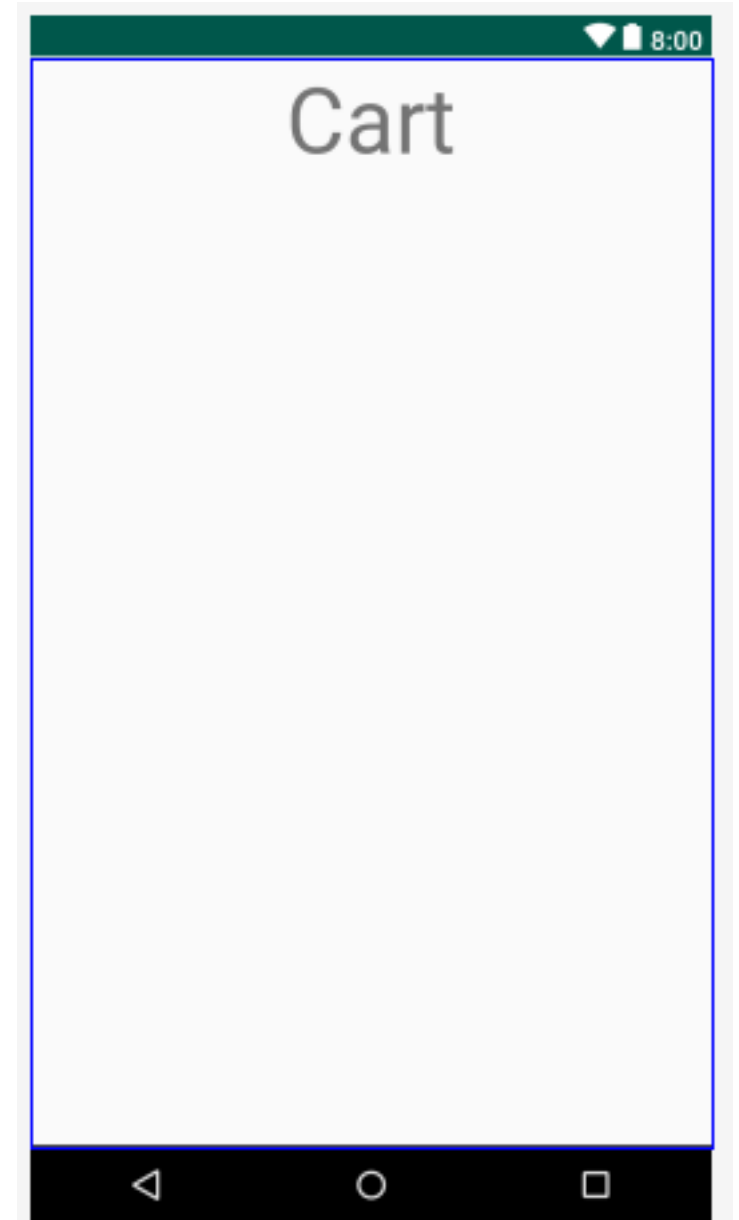
Show Intention Actions via `⌘⇧` (Alt+Enter for Win/Linux)

fragment_cart.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/cart"
        android:textSize="55sp"/>

</LinearLayout>
```



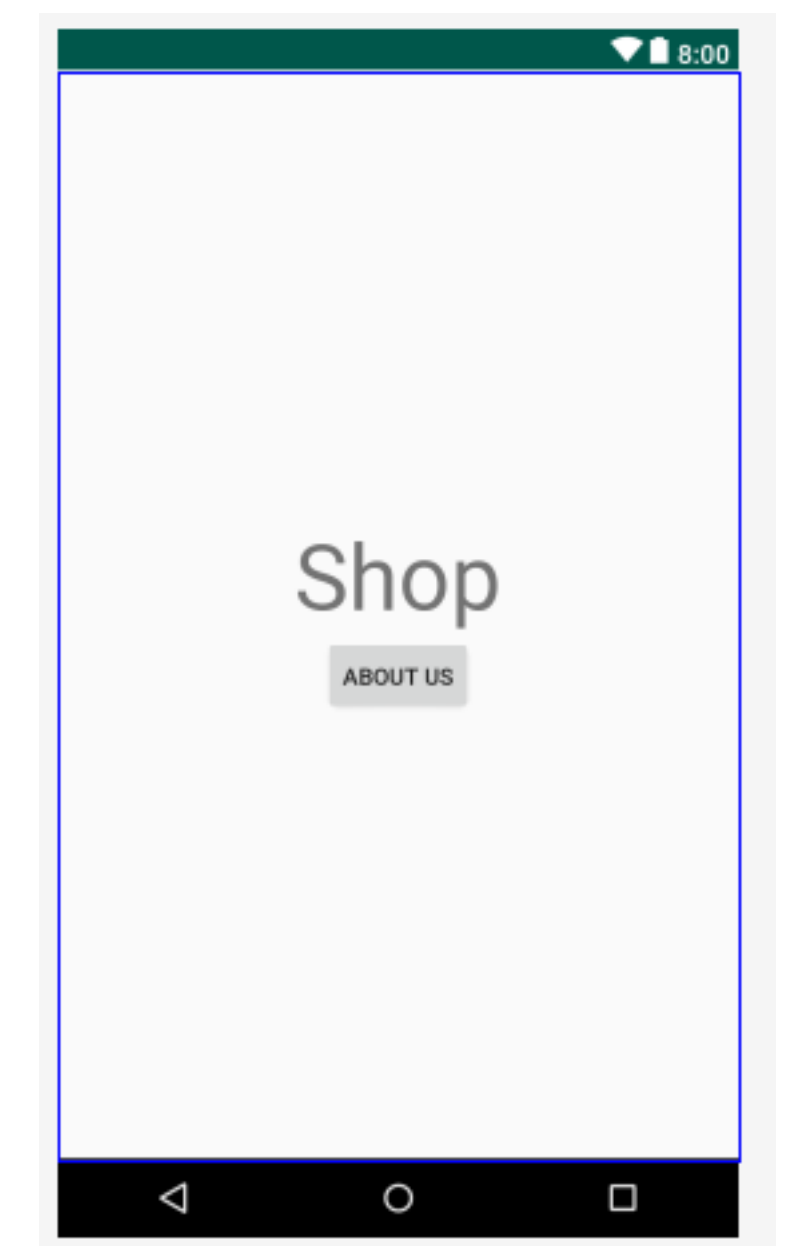
fragment_shop.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Shop"
        android:textSize="55sp"/>

    <Button
        android:id="@+id/btn_about"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="About Us"/>

</LinearLayout>
```



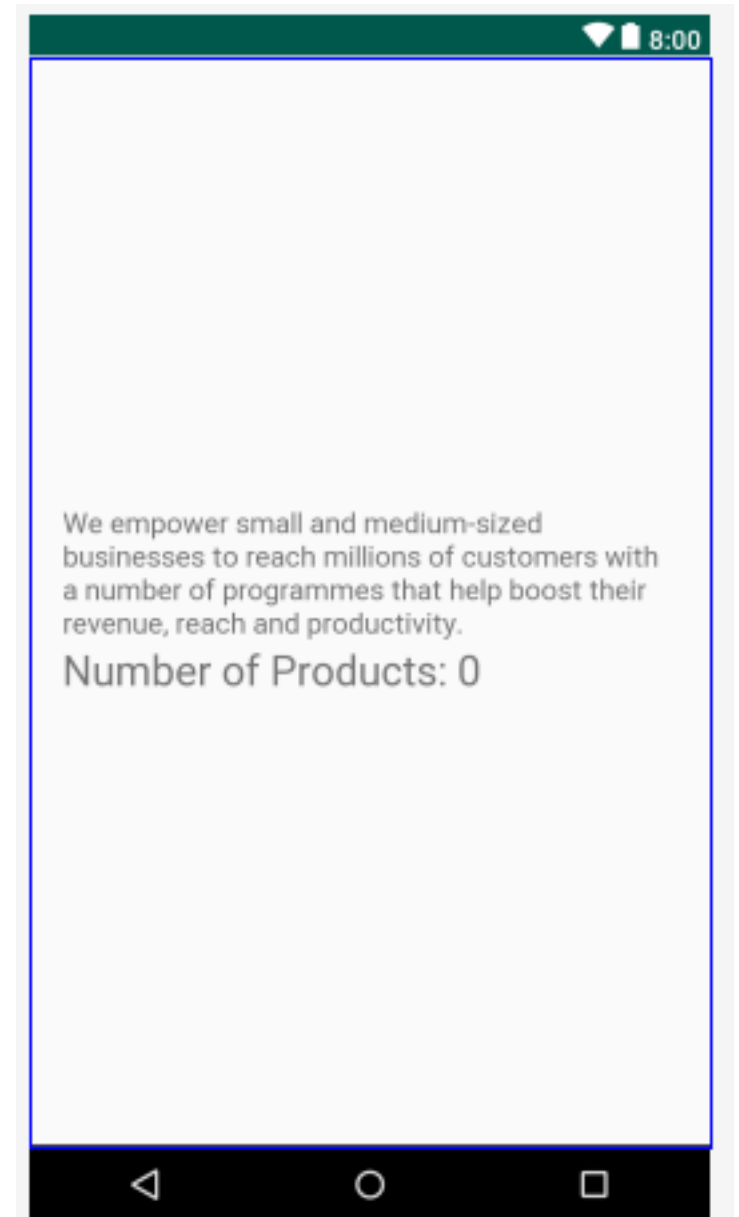
fragment_about.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    android:padding="20dp">

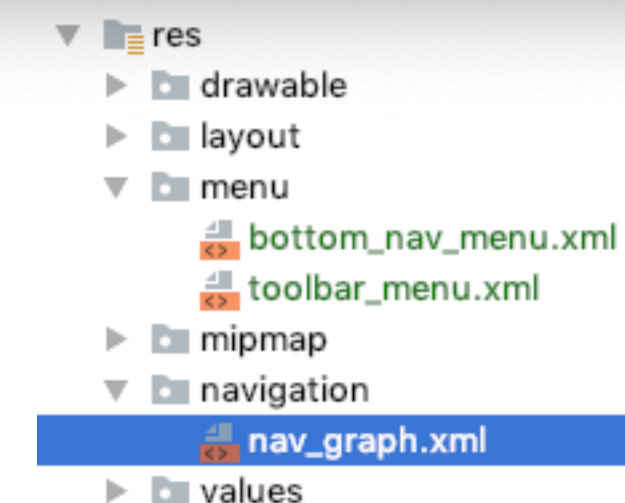
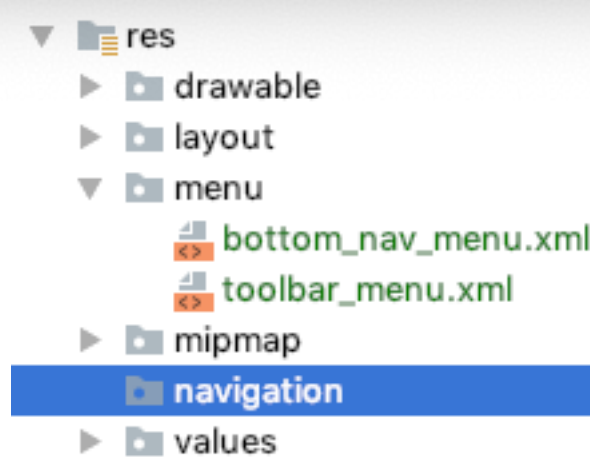
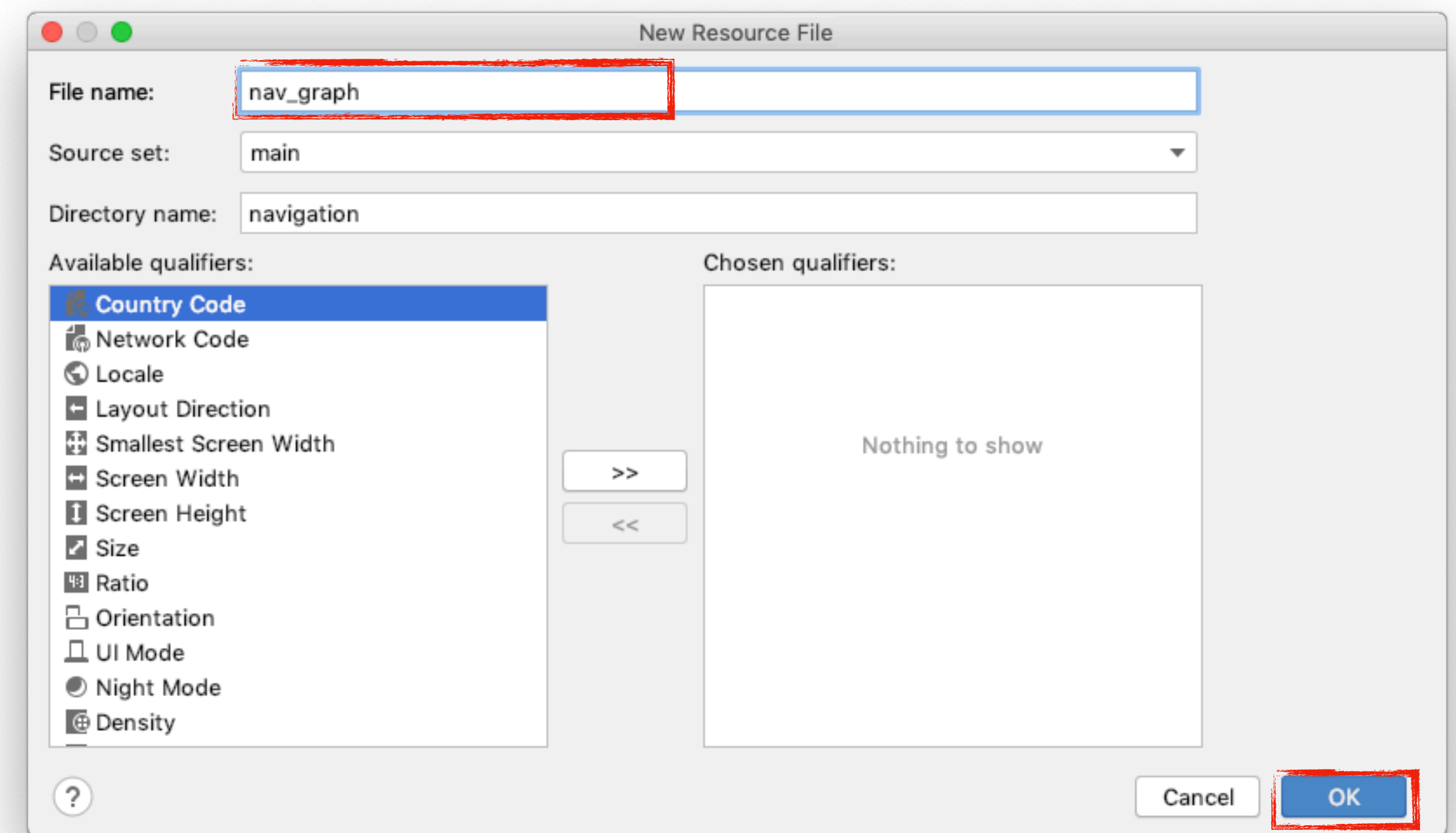
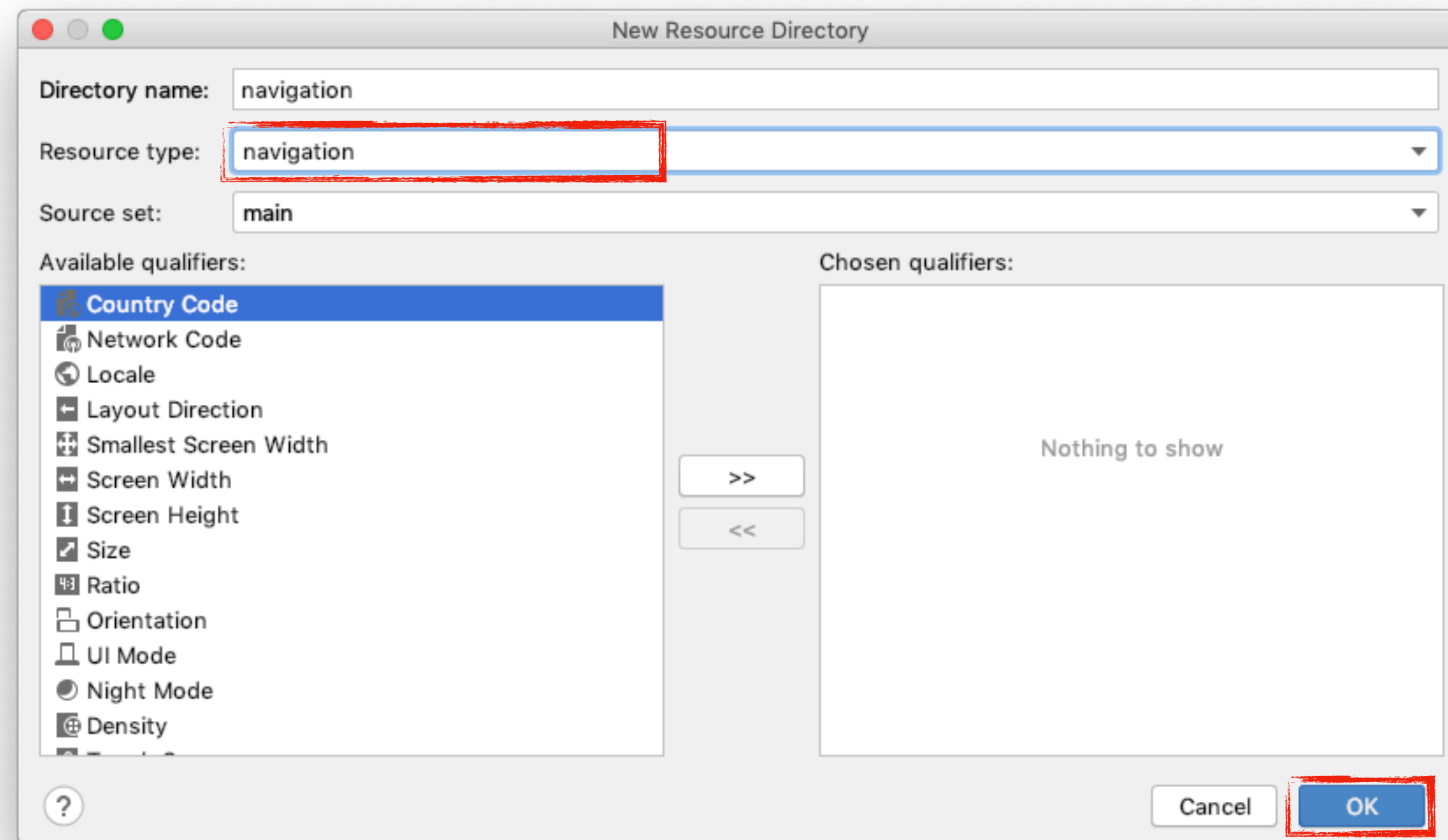
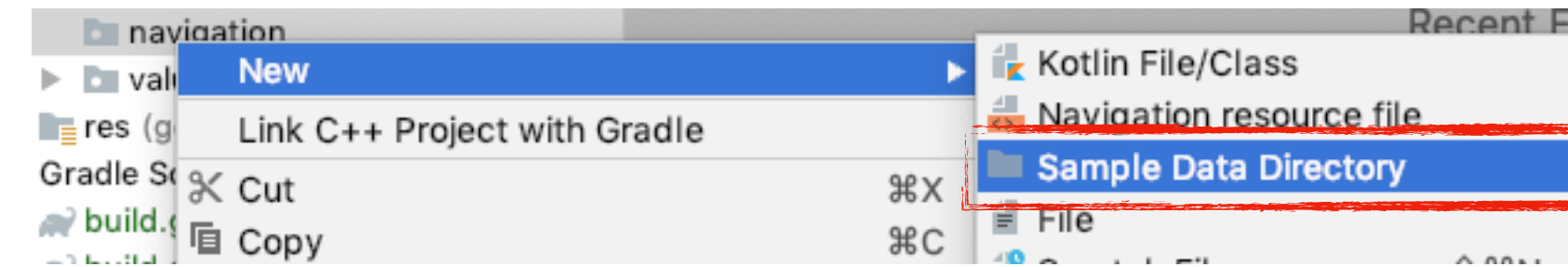
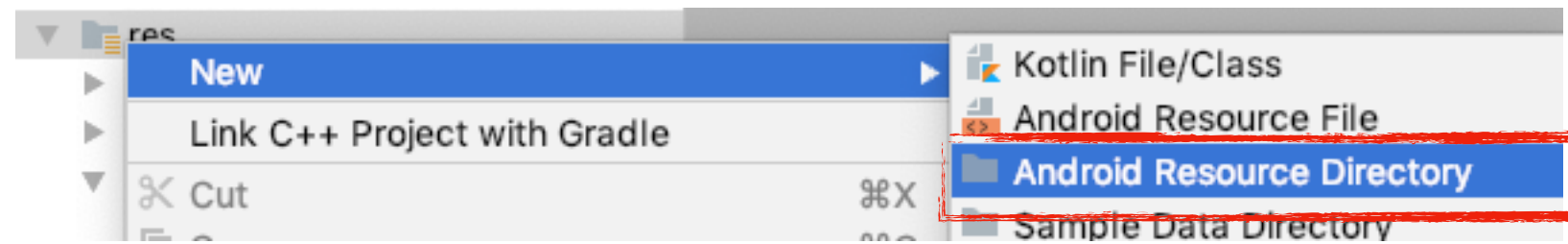
    <TextView
        android:id="@+id/tv_details"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="start"
        android:text="@string/detail"
        android:textSize="17sp"/>

    <TextView
        android:id="@+id/tv_product_count"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="start"
        android:text="Number of Products: 0"
        android:textSize="25sp"/>

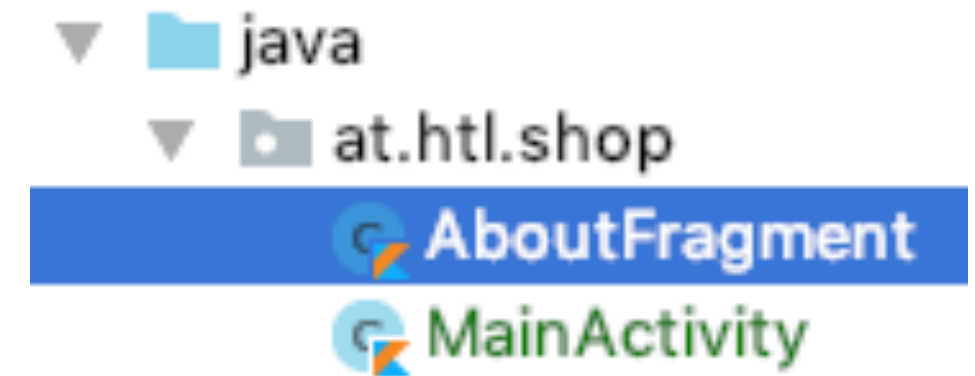
</LinearLayout>
```



Erstellen des Navigations-Files



AboutFragment.kt



```
package at.htl.shop
```

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

```
class AboutFragment : Fragment() {
```

```
    override fun onCreateView(
```

```
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
```

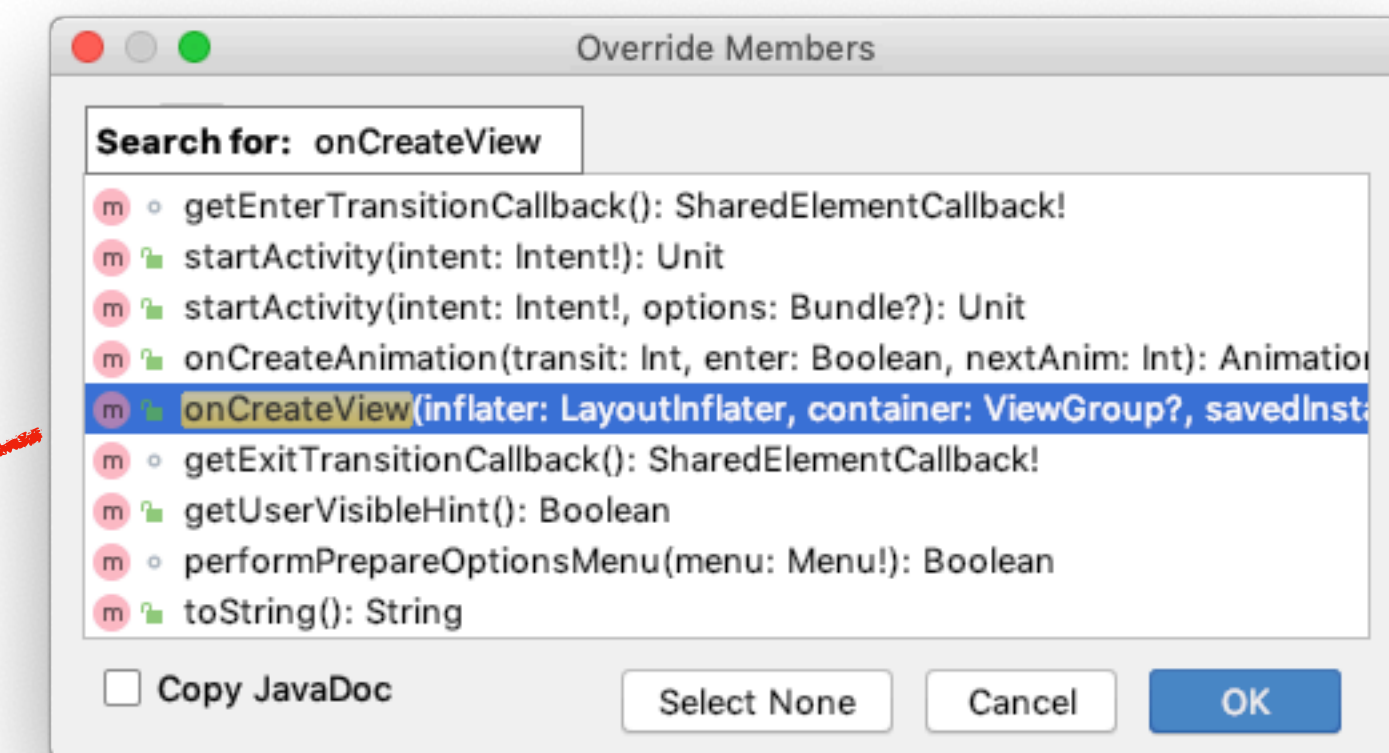
```
    ): View? {
```

```
        // inflate the layout for this fragment
```

```
        return inflater.inflate(R.layout.fragment_about, container, false)
```

```
    }
```

```
}
```



CartFragment.kt



```
package at.htl.shop

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

class CartFragment : Fragment() {

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

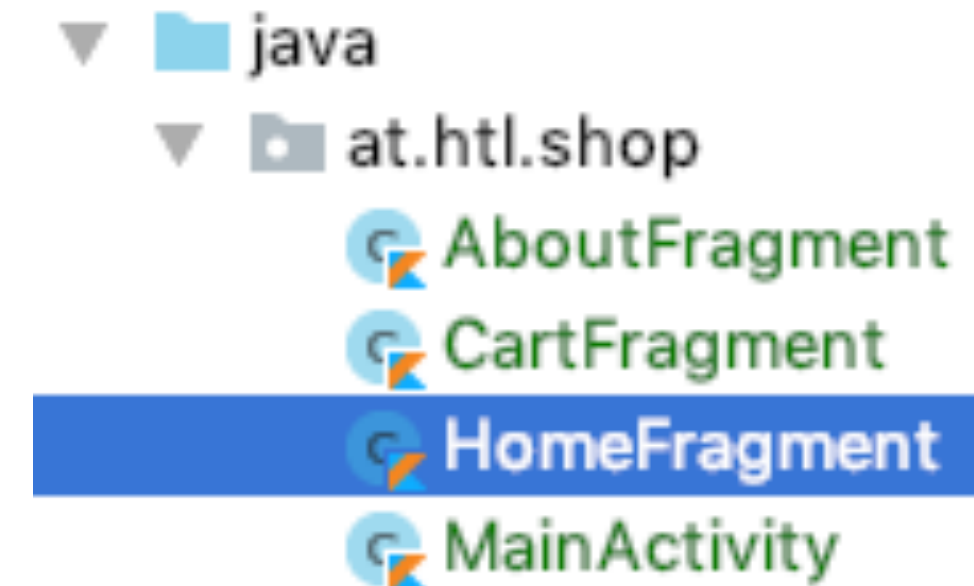
HomeFragment.kt

```
package at.htl.shop

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

class HomeFragment : Fragment() {

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



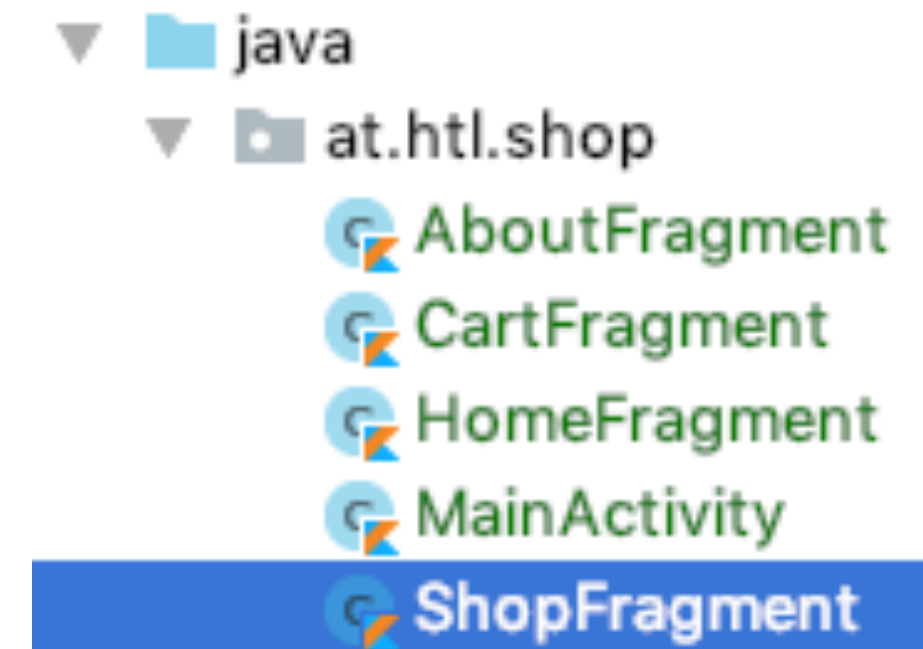
ShopFragment.kt

```
package at.htl.shop

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

class ShopFragment : Fragment() {

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



Navigation Components

- Navigation graph – contains all navigated information in centralized location
- NavHostFragment – special widget to add in layout
- NavController – Kotlin or Java object that keeps track of current position

1 NavHostFragment hinzufügen

2

Das Fragment soll über der bottom_nav eingefügt werden

4

app:defaultNavHost: A boolean that if set true will allow the Navigation Host to intercept when the system back button is pressed.

5

Im nav_graph sieht man nun die main_activity

3

```
<?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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/colorPrimary"
        android:theme="@style/ThemeOverlay.AppCompat.Dark"/>

    <fragment
        android:id="@+id/nav_host_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        app:defaultNavHost="true"
        app:navGraph="@navigation/nav_graph"/>

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_nav"
        app:menu="@menu/bottom_nav_menu"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"/>

</LinearLayout>
```

Navigation Graphs

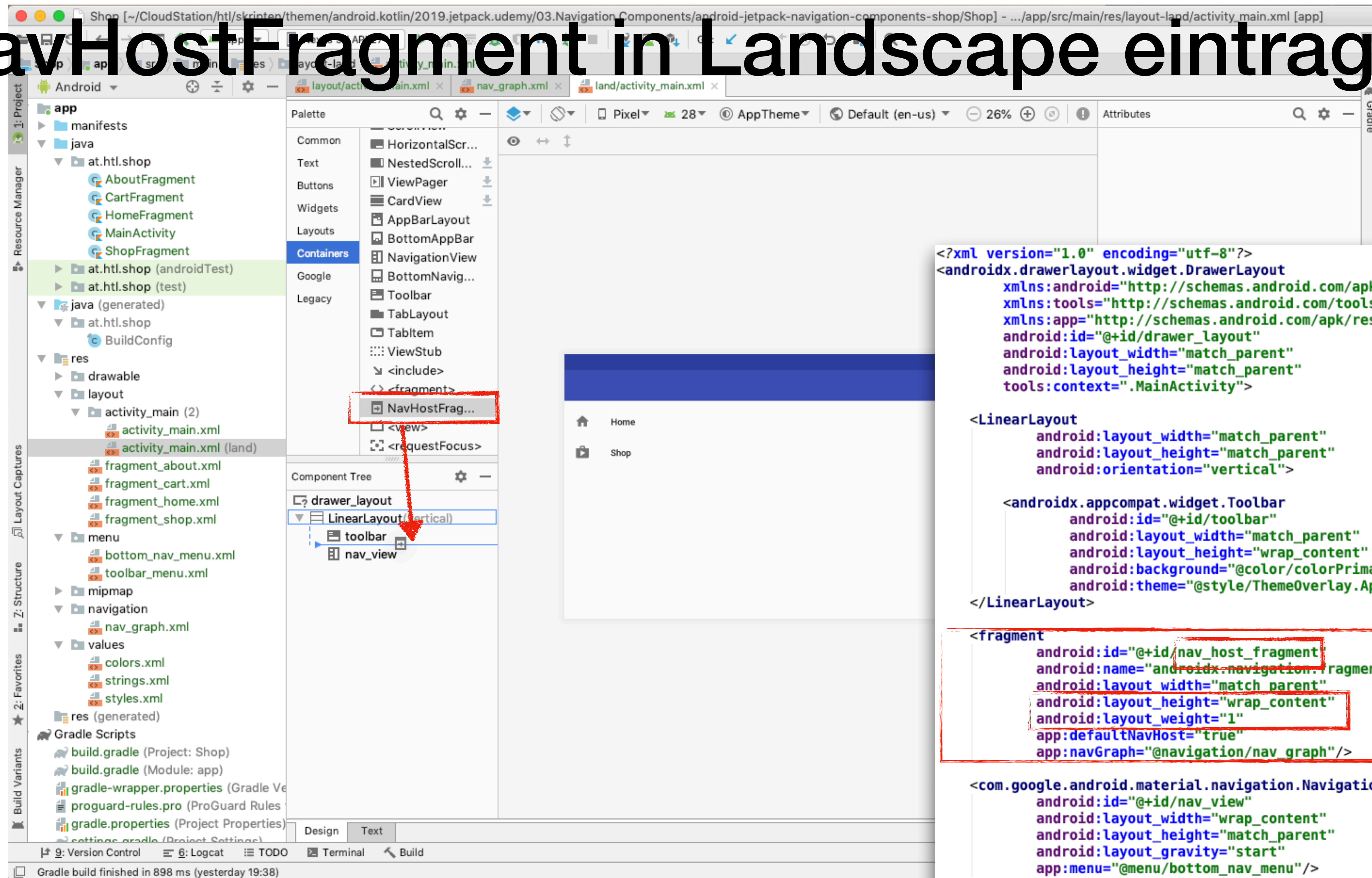
Project	Path
nav_graph	/Users/stuetz/CloudStation/ht...

Destinations

HOST	GRAPH
activity_main (fragment)	

Cancel OK

NavHostFragment in Landscape eintragen



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

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <androidx.appcompat.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:background="@color/colorPrimary"
            android:theme="@style/ThemeOverlay.AppCompat.Dark"/>

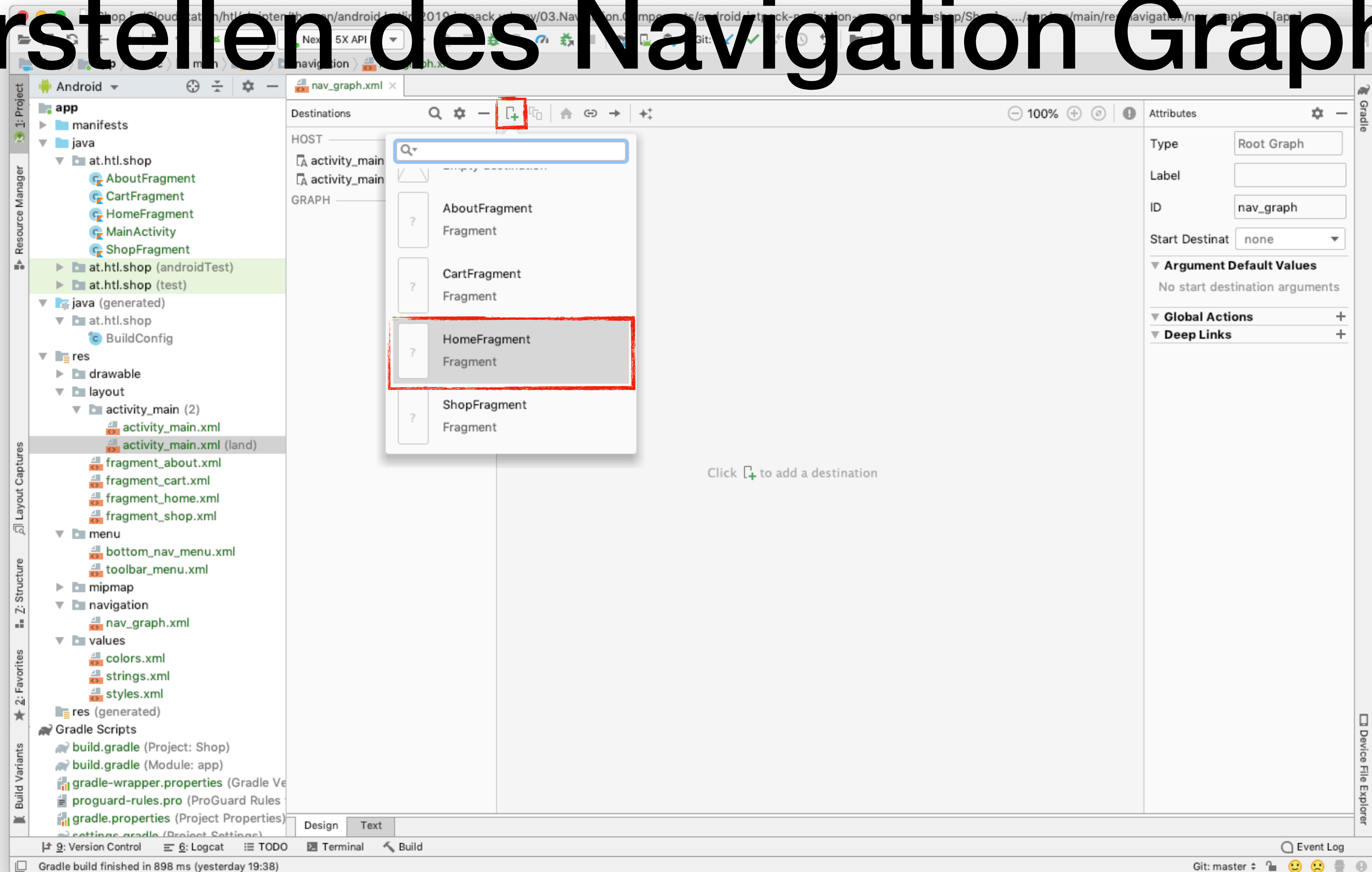
        </LinearLayout>

        <fragment
            android:id="@+id/nav_host_fragment"
            android:name="androidx.navigation.fragment.NavHostFragment"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            app:defaultNavHost="true"
            app:navGraph="@navigation/nav_graph"/>

        <com.google.android.material.navigation.NavigationView
            android:id="@+id/nav_view"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="start"
            app:menu="@menu/bottom_nav_menu"/>

    </androidx.drawerlayout.widget.DrawerLayout>
```

Erstellen des Navigation Graphs



Shop [~/CloudStation/htl/skripten/themen/android.kotlin/2019.jetpack.udemy/03.Navigation.Components/android-jetpack-navigation-components-shop/Shop] - .../app/src/main/res/navigation/nav_graph.xml [app]

Shop > app > src > main > res > navigation > nav_graph.xml

Android > app > manifests > java > at.htl.shop > AboutFragment > CartFragment > HomeFragment > MainActivity > ShopFragment > at.htl.shop (androidTest) > at.htl.shop (test) > java (generated) > at.htl.shop > BuildConfig > res > drawable > layout > activity_main (2) > activity_main.xml > activity_main.xml (land) > fragment_about.xml > fragment_cart.xml > fragment_home.xml > fragment_shop.xml > menu > bottom_nav_menu.xml > toolbar_menu.xml > mipmap > navigation > nav_graph.xml > values > colors.xml > strings.xml > styles.xml > res (generated) > Gradle Scripts > build.gradle (Project: Shop) > build.gradle (Module: app) > gradle-wrapper.properties (Gradle Wrapper) > proguard-rules.pro (ProGuard Rules) > gradle.properties (Project Properties) > settings.gradle (Project Settings)

nav_graph.xml

Destinations

HOST

activity_main (nav_host_fragment)

activity_main (nav_host_fragment)

GRAPH

home_destination - Start

home_destination

Preview Unavailable

Attributes

Type: Fragment

Label: Home

ID: home_destination

Class: HomeFragment

Arguments

Actions

Deep Links

Update Usages?

Update usages as well?
This will update all XML references and Java R field references.

☐ Don't ask again during this session

Cancel No Preview Yes

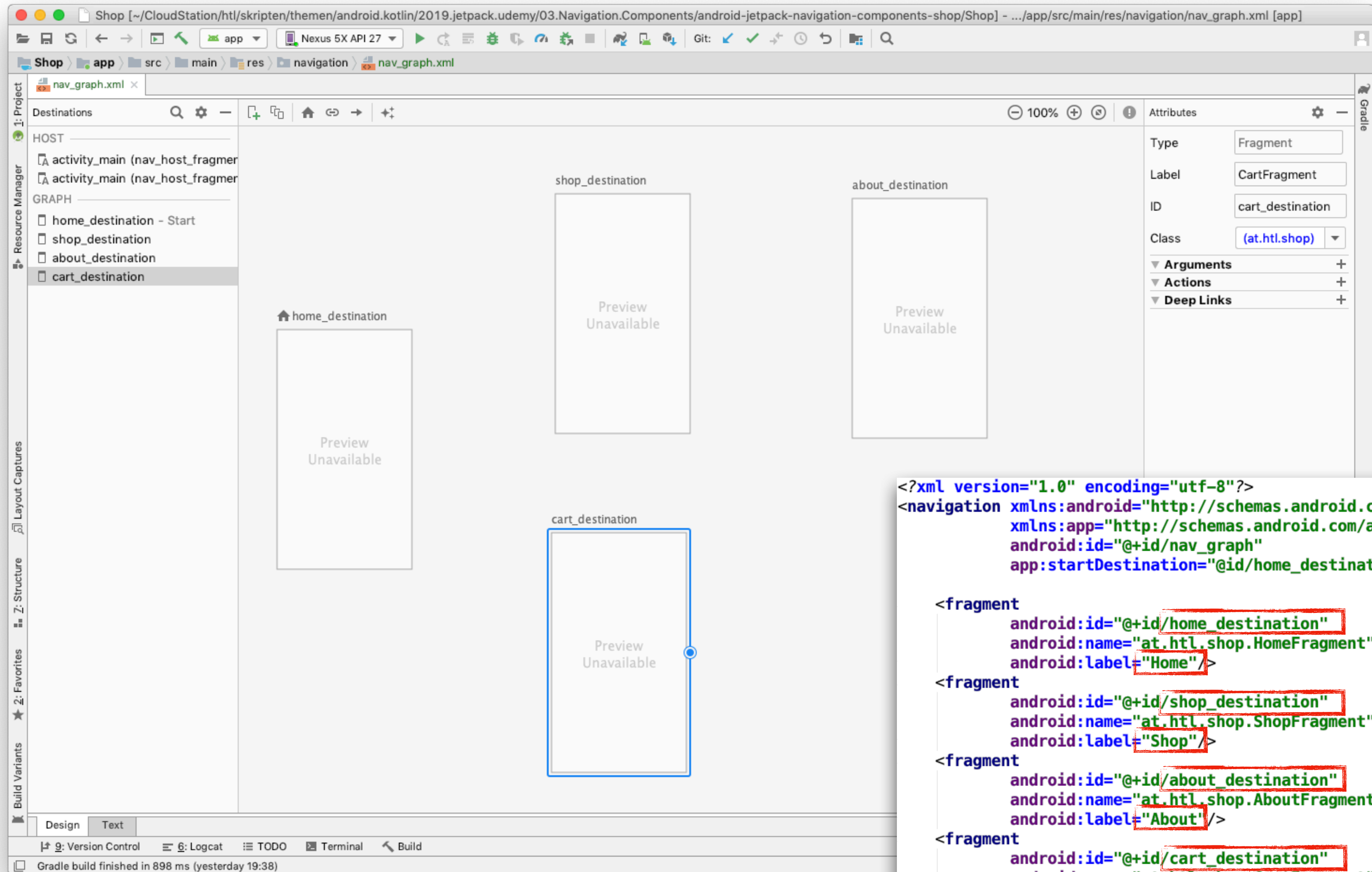
Als ID wird home_destination gewählt, da dies bereits im dazugehörigen menu „bottom_nav_menu“ eingetragen ist

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/home_destination"
        android:title="Home"
        android:icon="@drawable/ic_home"
    />

    <item
        android:id="@+id/shop_destination"
        android:title="Shop"
        android:icon="@drawable/ic_shop"
    />

</menu>
```

```
<?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/home_destination">

    <fragment
        android:id="@+id/home_destination"
        android:name="at.htl.shop.HomeFragment"
        android:label="Home"/>

    <fragment
        android:id="@+id/shop_destination"
        android:name="at.htl.shop.ShopFragment"
        android:label="Shop"/>

    <fragment
        android:id="@+id/about_destination"
        android:name="at.htl.shop.AboutFragment"
        android:label="About"/>

    <fragment
        android:id="@+id/cart_destination"
        android:name="at.htl.shop.CartFragment"
        android:label="Cart"/>

</navigation>
```

MainActivity.kt 1

In der MainActivity werden folgende
Komponenten konfiguriert:
BottomNavigationBar
Toolbar
NavigationDrawer

```
class MainActivity : AppCompatActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
  
        val navController =  
            Navigation.findNavController(this, R.id.nav_host_fragment)  
  
        setUpBottomNav(navController)  
        setUpSideNav(navController)  
        setUpActionBar(navController)  
    }  
}
```

Lassen Sie sich die Methoden generieren

```
private fun setUpBottomNav(navController: NavController) {  
}  
  
private fun setUpSideNav(navController: NavController) {  
}  
  
private fun setUpActionBar(navController: NavController) {  
}  
}
```

MainActivity.kt 2

activity_main.xml

```
<com.google.android.material.bottomnavigation.BottomNavigationView  
    android:id="@+id/bottom_nav"  
    app:menu="@menu/bottom_nav_menu"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"/>
```

Der safe-call-Operator '?' wird
verwendet, da das Landscape-Layout
kein bottom_nav enthält

<https://kotlinlang.org/docs/reference/null-safety.html#safe-calls>

NavigationUI

Class which hooks up elements of your application
such as global navigation patterns like a navigation
drawer or bottom nav bar with your NavController.

```
private fun setUpBottomNav(navController: NavController) {  
    bottom_nav?.let {  
        NavigationUI.setupWithNavController(it, navController)  
    }  
}
```

```
private fun setUpSideNav(navController: NavController) {  
    nav_view?.let {  
        NavigationUI.setupWithNavController(it, navController)  
    }  
}
```

```
private fun setUpActionBar(navController: NavController) {  
    NavigationUI.setupActionBarWithNavController(  
        this,  
        navController,  
        drawer_layout  
    )  
}
```


let { ... }

2. This vs. it argument

If we look at `T.run` and `T.let`, both functions are similar except for one thing, the way they accept the argument. The below shows the same logic for both functions.

```
stringVariable?.run {  
    println("The length of this String is $length")  
}  
  
// Similarly.  
  
stringVariable?.let {  
    println("The length of this String is ${it.length}")  
}
```

If you check the `T.run` function signature, you'll notice the `T.run` is just made as extension function calling `block: T.()`. Hence all within the scope, the `T` could be referred as `this`. In programming, `this` could be omitted most of the time. Therefore in our example above, we could use `$length` in the `println` statement, instead of `${this.length}`. I call this as sending in *this as argument*.

However for `T.let` function signature, you'll notice that `T.let` is sending itself into the function i.e. `block: (T)`. Hence this is like a lambda argument sent it. It could be referred within the scope function as `it`. So I call this as sending in *it as argument*.

MainActivity.kt 3

Erstellen des Option-Menüs

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {  
    menuInflater.inflate(R.menu.toolbar_menu, menu)  
    return true  
}
```

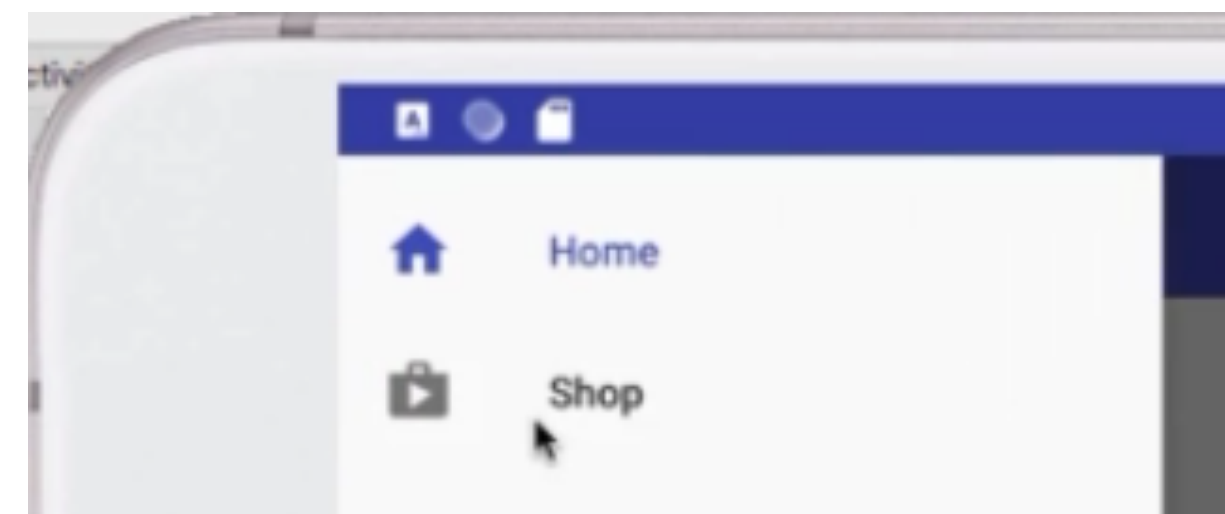
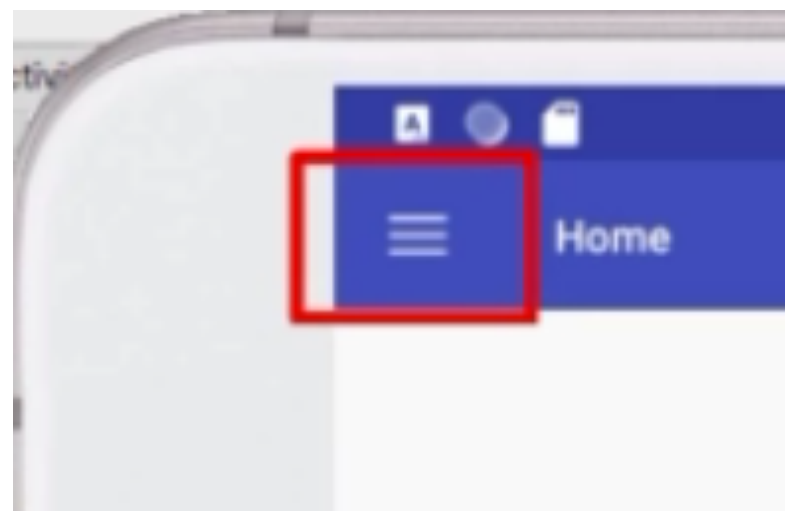
„item is not null“
Es wird ein NPE geworfen, falls item
doch null ist

<https://kotlinlang.org/docs/reference/null-safety.html#the-null-operator>

```
override fun onOptionsItemSelected(item: MenuItem?): Boolean {  
    val navController = Navigation.findNavController(this, R.id.nav_host_fragment)  
    val navigated = NavigationUI.onNavDestinationSelected(item!!, navController)  
    return navigated || super.onOptionsItemSelected(item)  
}
```

navigateUp sorgt dafür, dass
der Hamburger durch einen
Pfeil (icon 'up') ersetzt wird,
wenn zu einem anderen
Fragment gewechselt wird

```
override fun onSupportNavigateUp(): Boolean {  
    return NavigationUI.navigateUp(  
        Navigation.findNavController(this, R.id.nav_host_fragment), drawer_layout  
    )  
}
```



MainActivity.kt 4

```
package at.htl.shop
```

```
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.Menu
import android.view.MenuItem
import androidx.navigation.NavController
import androidx.navigation.Navigation
import androidx.navigation.ui.NavigationUI
import kotlinx.android.synthetic.main.activity_main.*
```

```
class MainActivity : AppCompatActivity() {

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        setSupportActionBar(toolbar)

        val navController =
            Navigation.findNavController(this, R.id.nav_host_fragment)

        setUpBottomNav(navController)
        setUpSideNav(navController)
        setUpActionBar(navController)
    }
```

```
private fun setUpBottomNav(navController: NavController) {
    bottom_nav?.let {
        NavigationUI.setupWithNavController(it, navController)
    }
}
```

```
private fun setUpSideNav(navController: NavController) {
    nav_view?.let {
        NavigationUI.setupWithNavController(it, navController)
    }
}
```

```
private fun setUpActionBar(navController: NavController) {
    NavigationUI.setupActionBarWithNavController(
        this,
        navController,
        drawer_layout
    )
}
```

```
override fun onCreateOptionsMenu(menu: Menu?): Boolean {
    menuInflater.inflate(R.menu.toolbar_menu, menu)
    return true
}
```

```
override fun onOptionsItemSelected(item: MenuItem?): Boolean {
    val navController = Navigation.findNavController(this, R.id.nav_host_fragment)
    val navigated = NavigationUI.onNavDestinationSelected(item!!, navController)
    return navigated || super.onOptionsItemSelected(item)
}
```

```
override fun onSupportNavigateUp(): Boolean {
    return NavigationUI.navigateUp(
        Navigation.findNavController(this, R.id.nav_host_fragment), drawer_layout
    )
}
```


Überblick

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    setSupportActionBar(toolbar)

    val navController =
        Navigation.findNavController(this, R.id.nav_host_fragment)

    setUpBottomNav(navController)
    setUpSideNav(navController)
    setUpActionBar(navController)
}
```

In der MainActivity werden
die Komponenten
konfiguriert

```
private fun setUpBottomNav(navController: NavController) {
    bottom_nav?.let {
        NavigationUI.setupWithNavController(it, navController)
    }
}
```

```
private fun setUpSideNav(navController: NavController) {
    nav_view?.let {
        NavigationUI.setupWithNavController(it, navController)
    }
}
```

```
private fun setUpActionBar(navController: NavController) {
    NavigationUI.setupActionBarWithNavController(
        this,
        navController,
        drawer_layout
    )
}
```

bezieht sich auf die Landscape
activity_main.xml

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

```
bottom_nav_menu.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android">
3
4     <item
5         android:id="@+id/home_destination"
6         android:title="Home"
7         android:icon="@drawable/ic_home"
8     />
9
10    <item
11        android:id="@+id/shop_destination"
12        android:title="Shop"
13        android:icon="@drawable/ic_shop"
14    />
15 </menu>
```

```
toolbar_menu.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <menu xmlns:android="http://schemas.android.com/apk/res/android"
3       xmlns:app="http://schemas.android.com/apk/res-auto">
4
5     <item
6         android:id="@+id/cart_destination"
7         android:icon="@drawable/ic_shopping_cart"
8         android:title="Cart"
9         app:showAsAction="ifRoom"
10    />
11 </menu>
```

option_menu

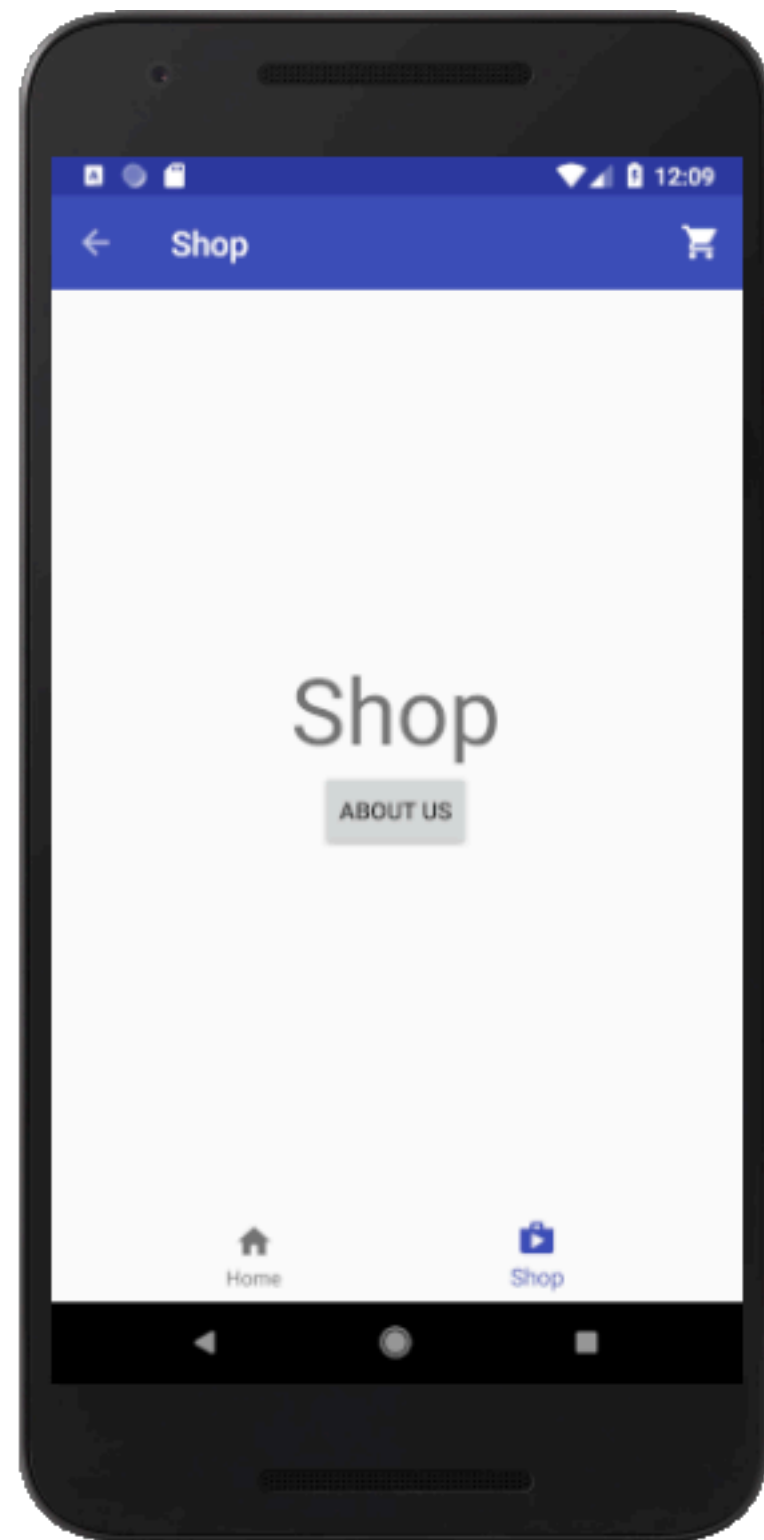
bottom_nav

```
nav_graph.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <navigation
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:id="@+id/nav_graph"
6     app:startDestination="@id/home_destination">
7
8     <fragment
9         android:id="@+id/home_destination"
10        android:name="at.htl.shop.HomeFragment"
11        android:label="Home"/>
12
13    <fragment
14        android:id="@+id/shop_destination"
15        android:name="at.htl.shop.ShopFragment"
16        android:label="Shop"/>
17
18    <fragment
19        android:id="@+id/about_destination"
20        android:name="at.htl.shop.AboutFragment"
21        android:label="About"/>
22
23    <fragment
24        android:id="@+id/cart_destination"
25        android:name="at.htl.shop.CartFragment"
26        android:label="Cart"/>
27 </navigation>
```

```
class CartFragment : Fragment() {
    override fun onCreateView(...): View? {
        return inflater.inflate(R.layout.fragment_cart, container, false)
    }
}
```

Im CartFragment werden die Fragmente am Screen dargestellt
(inflated)

onClickListener implementieren



```
package at.htl.shop

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.Navigation
import kotlinx.android.synthetic.main.fragment_shop.*

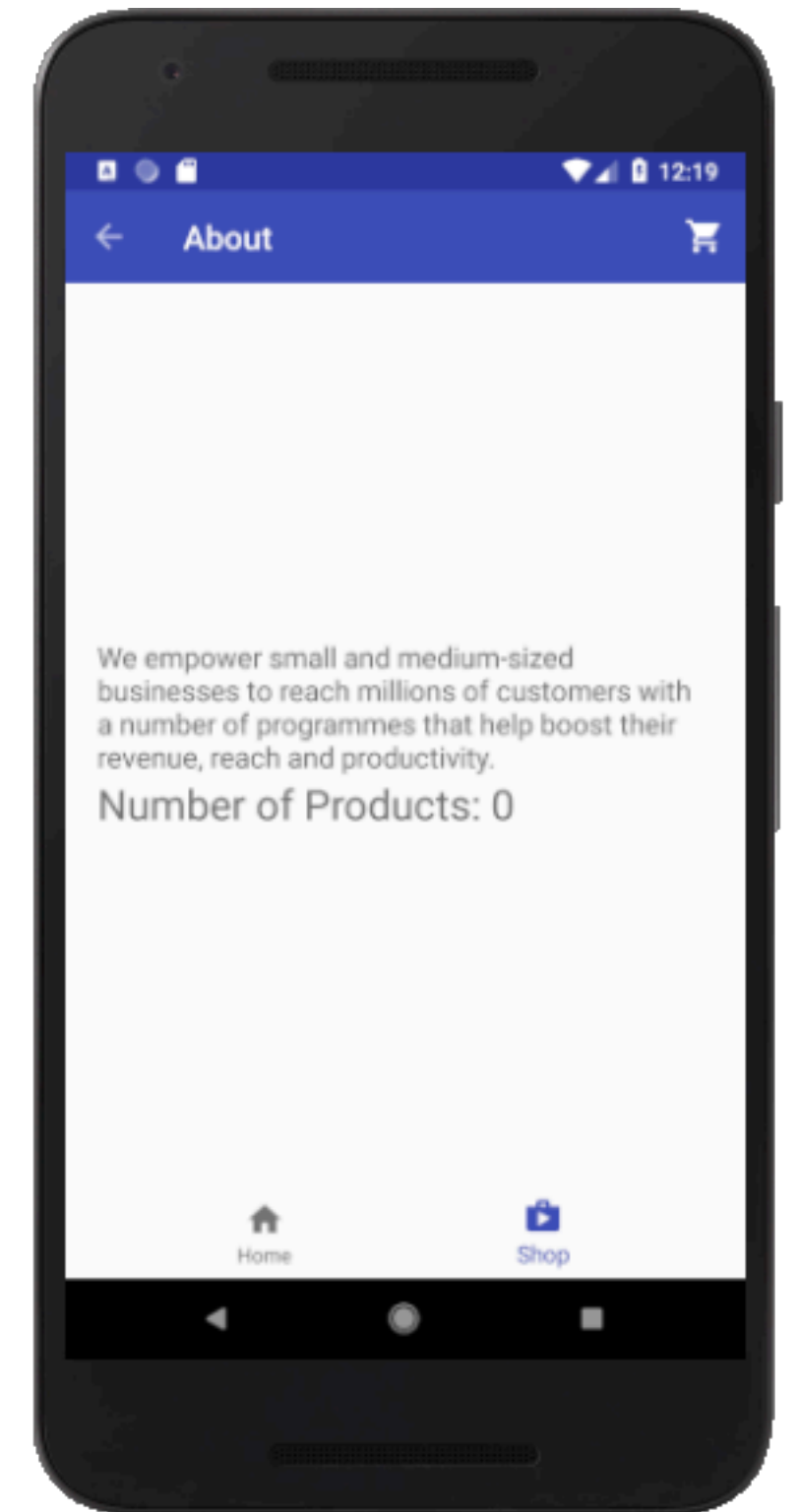
class ShopFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_shop, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

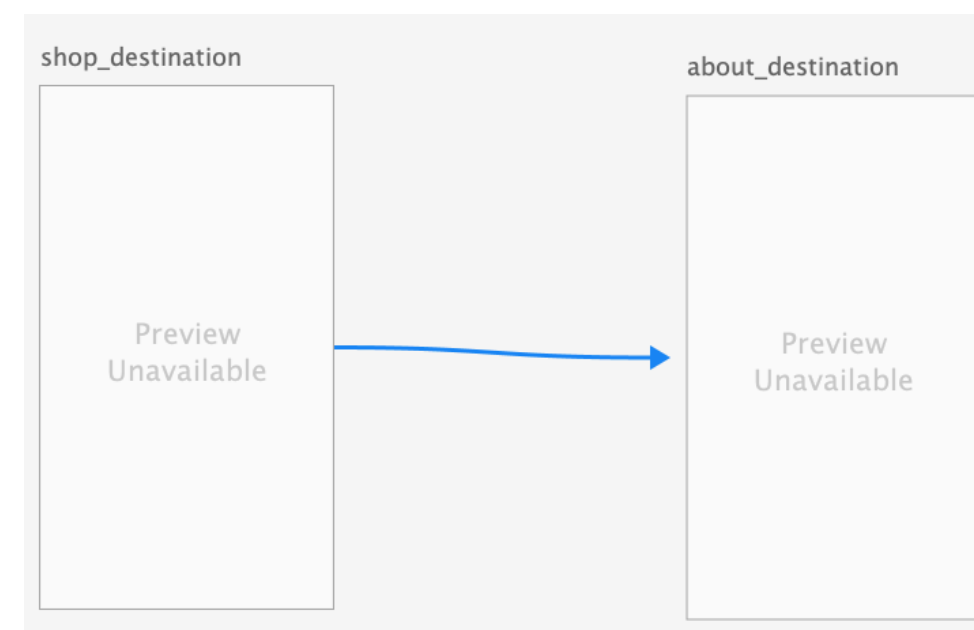
        btn_about.setOnClickListener {
            Navigation.findNavController(it).navigate(R.id.about_destination)
        }
    }
}
```

Um zum About Fragment zu gelangen, müssen wir den Button konfigurieren



Ergebnis

Navigation Actions



Setzen der ersten Action

The screenshot shows the Android Studio interface with the navigation graph editor open. The graph shows four destinations: home_destination, shop_destination, about_destination, and cart_destination. A blue arrow points from shop_destination to about_destination. The right-hand pane shows the attributes for the selected action, with the ID set to 'next_action' and the Destination set to 'about_destination'. The XML code for the navigation graph is displayed below the editor.

```
<?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/home_destination">

    <fragment
        android:id="@+id/home_destination"
        android:name="at.htl.shop.HomeFragment"
        android:label="Home"/>

    <fragment
        android:id="@+id/shop_destination"
        android:name="at.htl.shop.ShopFragment"
        android:label="Shop">
        <action
            android:id="@+id/next_action"
            app:destination="@id/about_destination"/>
        </fragment>
    <fragment
        android:id="@+id/about_destination"
        android:name="at.htl.shop.AboutFragment"
        android:label="About"/>
    <fragment
        android:id="@+id/cart_destination"
        android:name="at.htl.shop.CartFragment"
        android:label="Cart"/>
</navigation>
```


nav_graph.xml: Layout eintragen

```
<?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" xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/nav_graph"
    app:startDestination="@id/home_destination">

    <fragment
        android:id="@+id/home_destination"
        android:name="at.htl.shop.HomeFragment"
        android:label="Home"
        tools:layout="@layout/fragment_home"/>

    <fragment
        android:id="@+id/shop_destination"
        android:name="at.htl.shop.ShopFragment"
        android:label="Shop"
        tools:layout="@layout/fragment_shop">

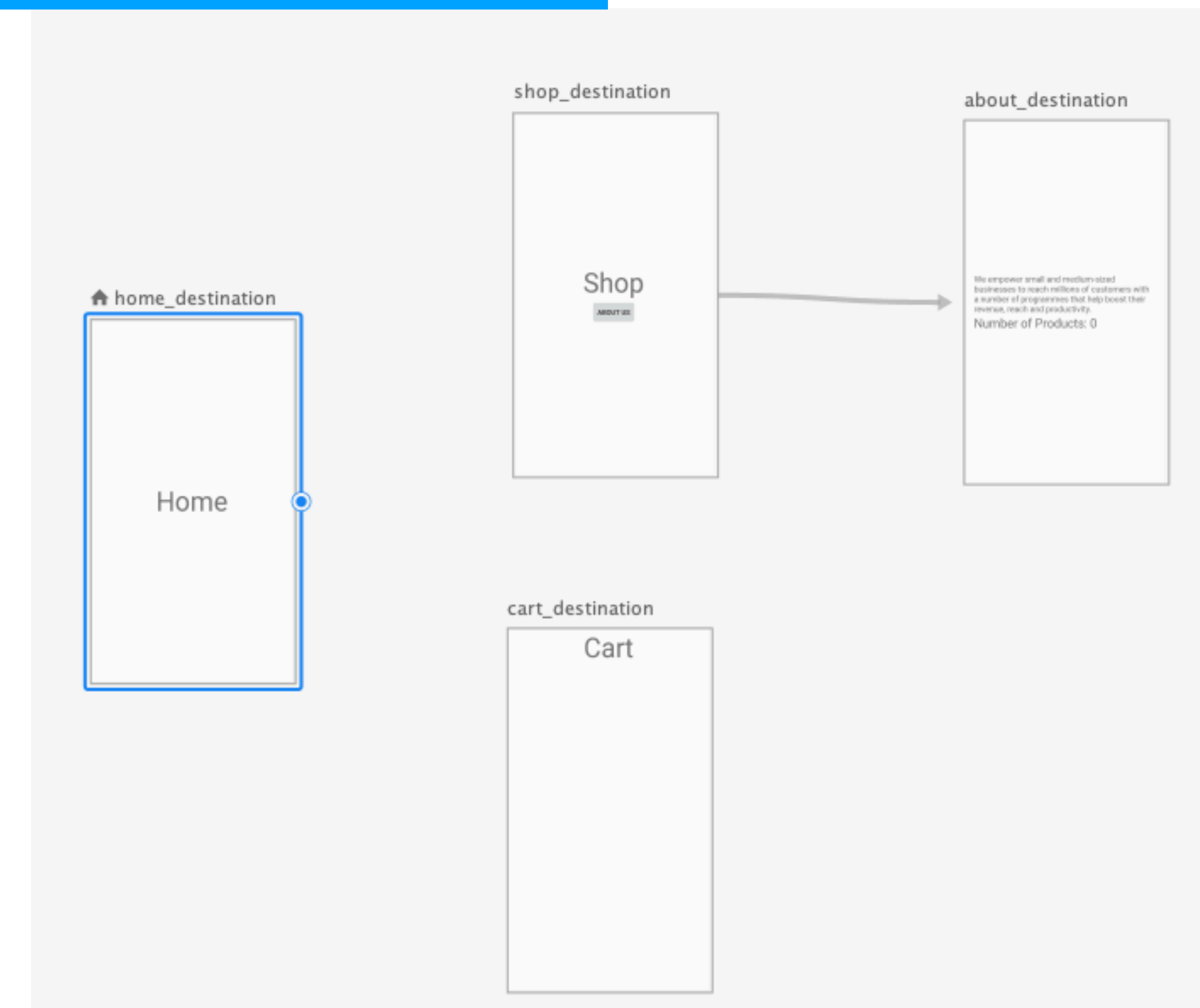
        <action
            android:id="@+id/next_action"
            app:destination="@id/about_destination"/>

        </fragment>
    <fragment
        android:id="@+id/about_destination"
        android:name="at.htl.shop.AboutFragment"
        android:label="About"
        tools:layout="@layout/fragment_about"/>

    <fragment
        android:id="@+id/cart_destination"
        android:name="at.htl.shop.CartFragment"
        android:label="Cart"
        tools:layout="@layout/fragment_cart"/>

</navigation>
```

Nun werden die fragment-Elemente mit ihren Layouts ergänzt. Dadurch sieht man im Navigations-Editor das jeweilige Layout



Verwenden der Action beim onClickListener

```
package at.htl.shop

import ...

class ShopFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_shop, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        btn_about.setOnClickListener {
            // Navigation.findNavController(it).navigate(R.id.about_destination)

            Navigation.findNavController(it).navigate(R.id.next_action)
        }
    }
}
```



funktioniert!

Pass Data between Destinations Using SafeArgs



Documentation

OVERVIEW

GUIDES

REFERENCE

SAMPLES

DESIGN & QUALITY

Adding Components to your Project

▶ Data Binding Library

Handling Lifecycles

LiveData

▼ Navigation

Overview

Implement Navigation

Update UI components with
NavigationUI

Nested graphs

[Pass data between](#)

Pass data between destinations



Navigation allows you to attach data to a navigation operation by defining arguments for a destination. For example, a user profile destination might take a user ID argument to determine which user to display.

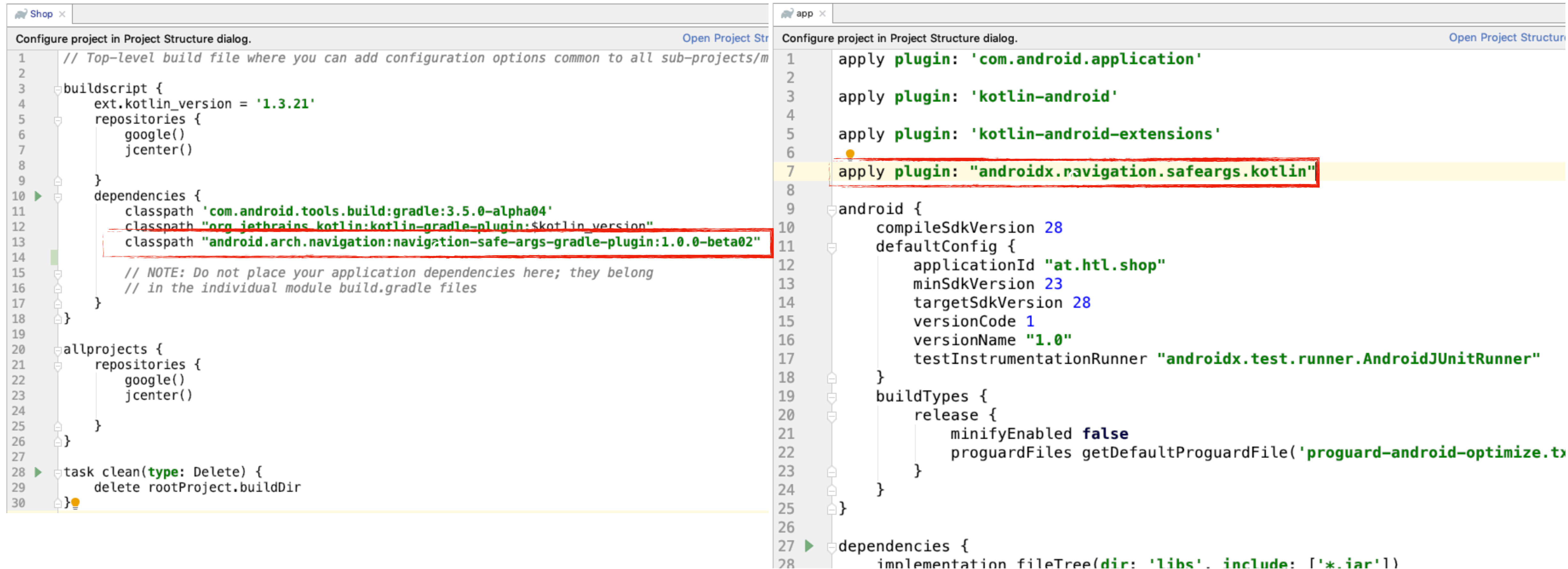
In general, you should strongly prefer passing only the minimal amount of data between destinations. For example, you should pass a key to retrieve an object rather than passing the object itself, as the total space for all saved states is limited on Android. If you need to pass large amounts of data, consider using a [ViewModel](#) as described in [Share data between fragments](#).

Contents

[Define destination arguments](#)[Override a destination argument in an action](#)[Use Safe Args to pass data with type safety](#)[Use Safe Args with a global action](#)[Pass data between destinations with Bundle objects](#)

<https://developer.android.com/topic/libraries/architecture/navigation/navigation-pass-data>

Konfigurieren von SafeArgs

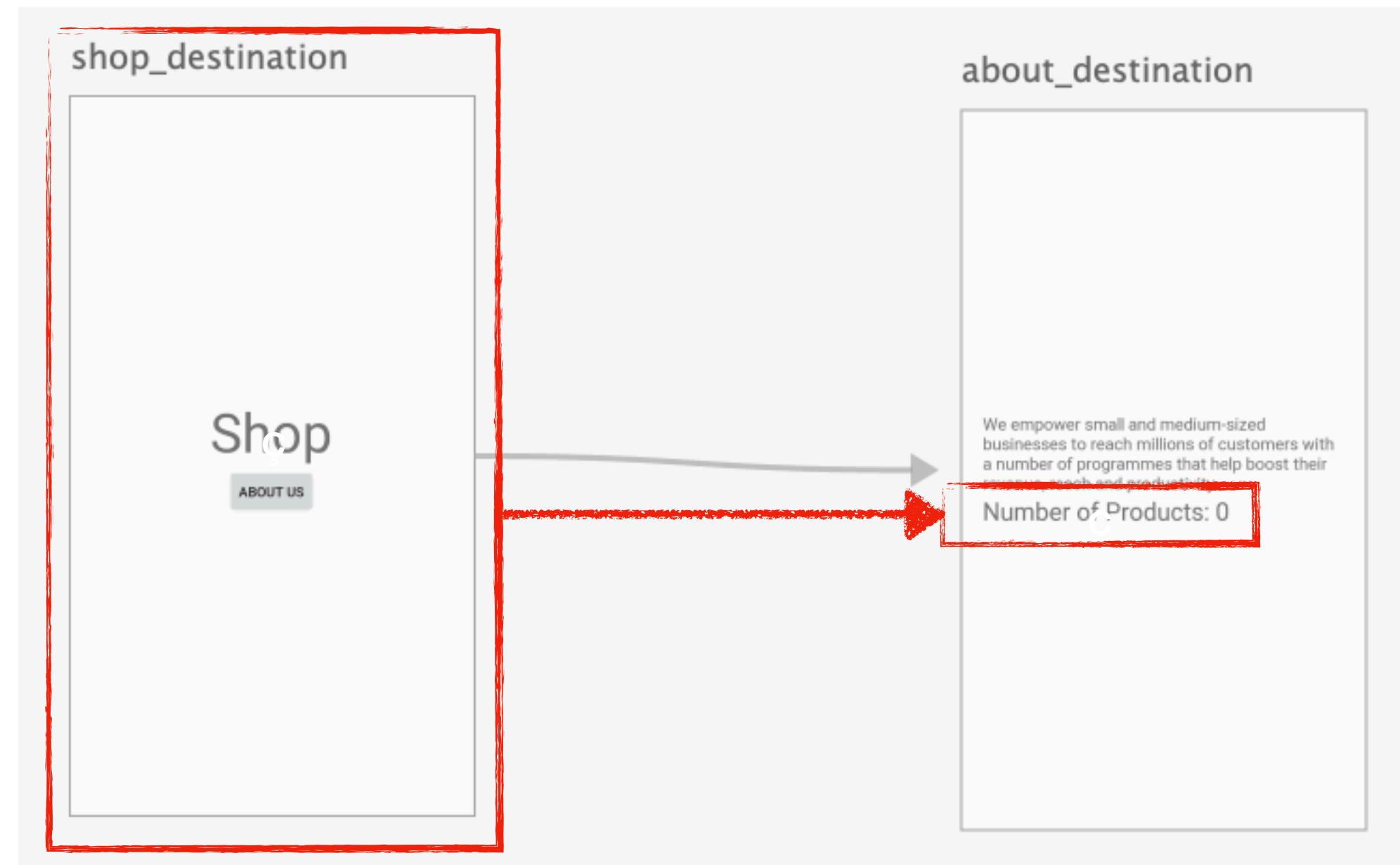


```
1 // Top-level build file where you can add configuration options common to all sub-projects/modules
2
3 buildscript {
4     ext.kotlin_version = '1.3.21'
5     repositories {
6         google()
7         jcenter()
8     }
9 }
10 dependencies {
11     classpath 'com.android.tools.build:gradle:3.5.0-alpha04'
12     classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
13     classpath "android.arch.navigation:navigation-safe-args-gradle-plugin:1.0.0-beta02"
14 }
15 // NOTE: Do not place your application dependencies here; they belong
16 // in the individual module build.gradle files
17
18 }
19
20 allprojects {
21     repositories {
22         google()
23         jcenter()
24     }
25 }
26
27 task clean(type: Delete) {
28     delete rootProject.buildDir
29 }
30 }
```

```
1 apply plugin: 'com.android.application'
2
3 apply plugin: 'kotlin-android'
4
5 apply plugin: 'kotlin-android-extensions'
6
7 apply plugin: "androidx.navigation.safeargs.kotlin"
8
9 android {
10     compileSdkVersion 28
11     defaultConfig {
12         applicationId "at.htl.shop"
13         minSdkVersion 23
14         targetSdkVersion 28
15         versionCode 1
16         versionName "1.0"
17         testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
18     }
19     buildTypes {
20         release {
21             minifyEnabled false
22             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
23         }
24     }
25 }
26
27 dependencies {
28     implementation fileTree(dir: 'libs', include: ['*.jar'])
29 }
```

https://developer.android.com/jetpack/androidx/releases/navigation#declaring_dependencies

Aufgabenstellung



Eine Zufallszahl wird vom shop- zum about-fragment übergeben

Parameter eintragen

The screenshot shows the Android Studio interface with the navigation graph editor. The graph shows a transition from 'shop_destination' to 'about_destination'. The 'about_destination' node is highlighted with a red box and a blue circle (1). The 'Add Argument Link' dialog is open, showing the 'Name' field (3) with the value 'productCount', the 'Type' field set to 'Integer', and the 'Default Value' field set to '0'. The 'Add' button (4) is highlighted. The 'Attributes' panel on the right shows the details for the 'about_destination' fragment, including its class '(at.htl.shop)' and the added argument 'productCount: integer (0)'. A code snippet at the bottom shows the XML for the fragment and argument.

```
<fragment  
    android:id="@+id/about_destination"  
    android:name="at.htl.shop.AboutFragment"  
    android:label="About"  
    tools:layout="@layout/fragment_about">  
    <argument  
        android:name="productCount"  
        app:argType="integer"  
        android:defaultValue="0"/>  
    </argument>  
</fragment>
```

ShopFragment.kt

```
package at.htl.shop

import ...

class ShopFragmentDirections private constructor() {
    private data class NextAction(val productCount: Int = 0) : NavDirections {
        override fun getActionId(): Int = at.htl.shop.R.id.next_action

        override fun getArguments(): Bundle {
            val result = Bundle()
            result.putInt("productCount", this.productCount)
            return result
        }
    }

    companion object {
        fun nextAction(productCount: Int = 0): NavDirections = NextAction(productCount)
    }
}
```

```
package at.htl.shop

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import androidx.navigation.Navigation
import kotlinx.android.synthetic.main.fragment_shop.*
import kotlin.random.Random

class ShopFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_shop, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        btn_about.setOnClickListener {
            // Navigation.findNavController(it).navigate(R.id.about_destination)

            //Navigation.findNavController(it).navigate(R.id.next_action)

            val nextAction = ShopFragmentDirections.nextAction(Random.nextInt(200))
            Navigation.findNavController(it).navigate(nextAction)
        }
    }
}
```


AboutFragment.kt

```
package at.htl.shop

import ...

data class AboutFragmentArgs(val productCount: Int = 0) : NavArgs {
    fun toBundle(): Bundle {
        val result = Bundle()
        result.putInt("productCount", this.productCount)
        return result
    }

    companion object {
        @JvmStatic
        fun fromBundle(bundle: Bundle): AboutFragmentArgs {
            bundle.setClassLoader(AboutFragmentArgs::class.java.classLoader)
            val __productCount : Int
            if (bundle.containsKey("productCount")) {
                __productCount = bundle.getInt("productCount")
            } else {
                __productCount = 0
            }
            return AboutFragmentArgs(__productCount)
        }
    }
}
```

```
package at.htl.shop

import android.os.Bundle
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import androidx.fragment.app.Fragment
import kotlinx.android.synthetic.main.fragment_about.*

class AboutFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_about, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

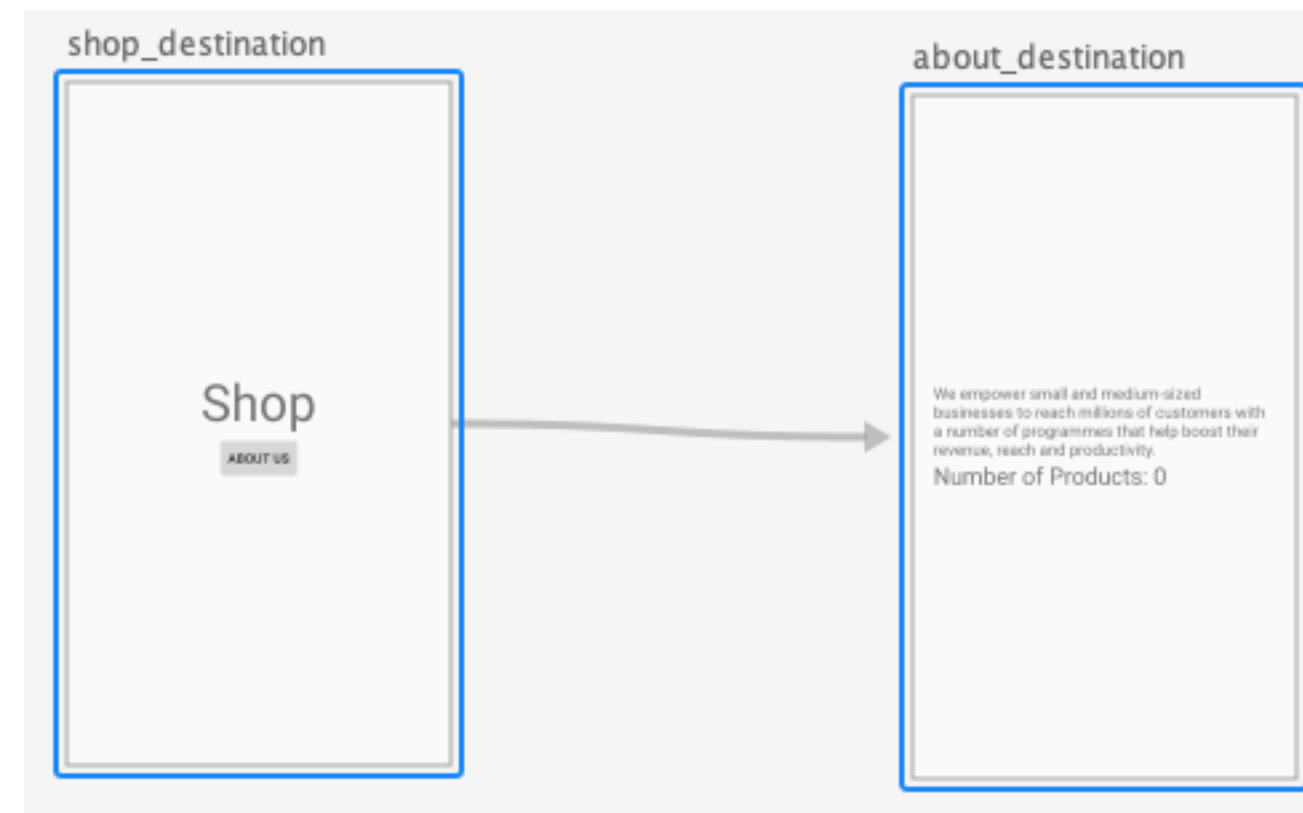
        arguments?.let {
            val safeArgs = AboutFragmentArgs.fromBundle(it)
            tv_product_count.text = "Total Products Available: ${safeArgs.productCount}"
        }
    }
}
```


SENDER

SafeArgs

EMPFÄNGER

```
class ShopFragment : Fragment() {
    override fun onCreateView(...): View? { ... }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        btn_about.setOnClickListener {
            val nextAction = ShopFragmentDirections
                .nextAction(Random.nextInt(200))
            Navigation.findNavController(it).navigate(nextAction)
        }
    }
}
```



```
class AboutFragment : Fragment() {
    override fun onCreateView(...): View? { ... }
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        arguments?.let {
            val safeArgs = AboutFragmentArgs.fromBundle(it)
            tv_product_count.text =
                „Total Products Available: ${safeArgs.productCount}“
        }
    }
}
```

automatisch generierte Klassen

ShopFragmentDirection

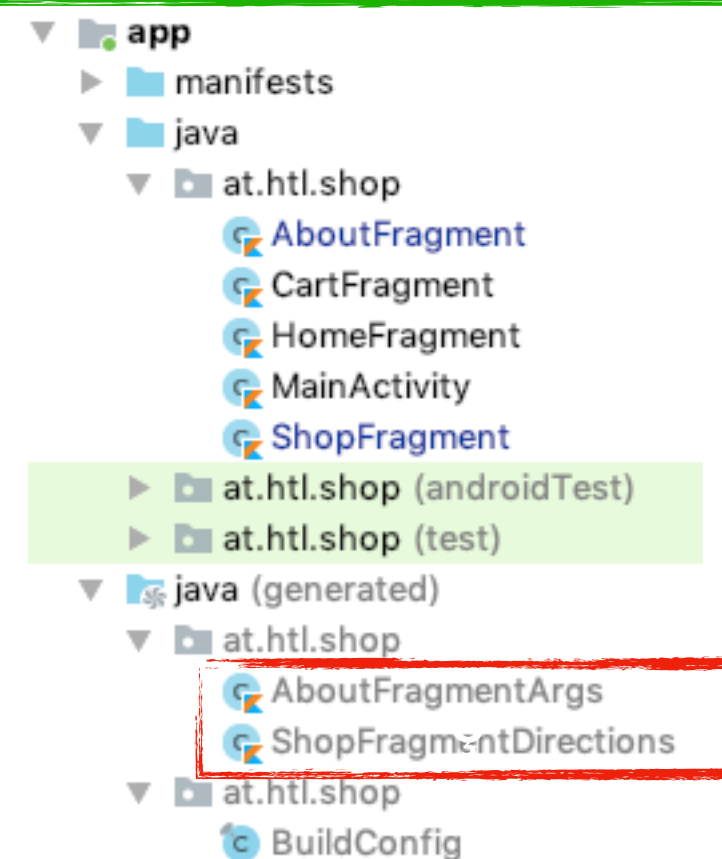
NextAction

AboutFragmentArgs

```
class ShopFragmentDirections private constructor() {
    private data class NextAction(val productCount: Int = 0) : NavDirections {
        override fun getActionId(): Int = at.htl.shop.R.id.next_action
    }

    override fun getArguments(): Bundle {
        val result = Bundle()
        result.putInt("productCount", this.productCount)
        return result
    }
}

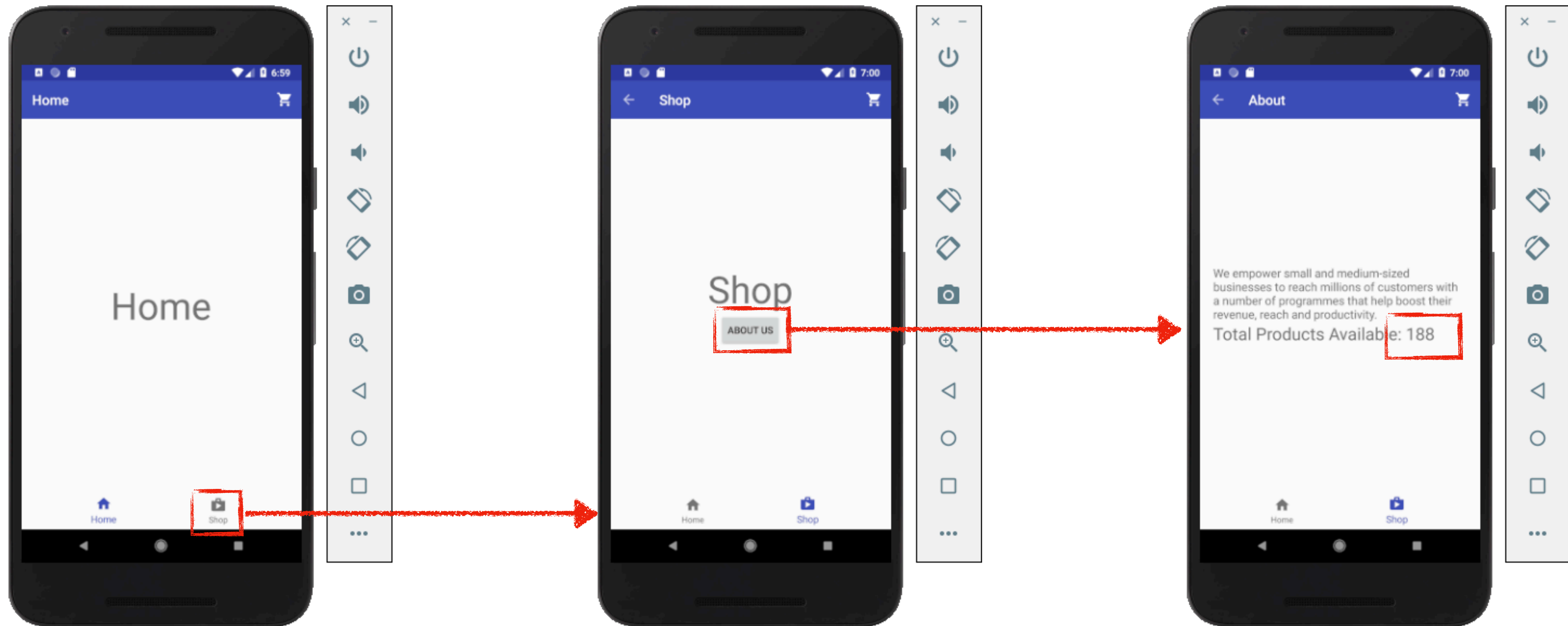
companion object {
    fun nextAction(productCount: Int = 0): NavDirections = NextAction(productCount)
}
```



```
data class AboutFragmentArgs(val productCount: Int = 0) : NavArgs {
    fun toBundle(): Bundle {
        val result = Bundle()
        result.putInt("productCount", this.productCount)
        return result
    }

    companion object {
        @JvmStatic
        fun fromBundle(bundle: Bundle): AboutFragmentArgs {
            bundle.setClassLoader(AboutFragmentArgs::class.java.classLoader)
            val __productCount : Int
            if (bundle.containsKey("productCount")) {
                __productCount = bundle.getInt("productCount")
            } else {
                __productCount = 0
            }
            return AboutFragmentArgs(__productCount)
        }
    }
}
```

Testlauf



Wie gehts weiter?


- <https://codelabs.developers.google.com/codelabs/android-room-with-a-view/#0>
-




Noch
Fragen?


Source

<https://www.udemy.com/android-jetpack-architecture-components/>

CategoriesUdemy for BusinessBecome an instructorLog InSign Up

Development > Mobile Apps > Android Game Development

 Gift This Course

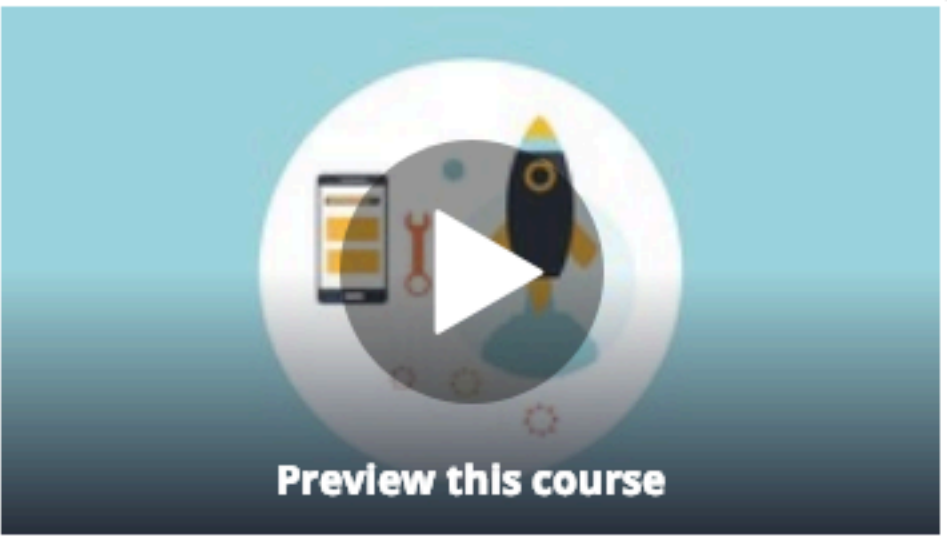
 Wishlist

Android Jetpack Architecture Components

Utilize Android Jetpack Architecture components to make your Android application development flexible and maintainable

NEW ★★★★★ 0.0 (0 ratings) 4 students enrolled

Created by Packt Publishing Last updated 2/2019 English English [Auto-generated]



Preview this course

What you'll learn

- ✓ Get introduced to Android architecture components
- ✓ Provide stability in your app by handling life cycles, view models, and live data
- ✓ Load data gradually and gracefully in RecyclerView by using the Paging library
- ✓ Explore how to perform CRUD operations in the Room database
- ✓ Use the Data Binding library to bind data to the UI
- ✓ Implement effective in-app navigation by using the Navigation architecture component
- ✓ Implement a local database to store structured data by using the Room database
- ✓ Schedule tasks asynchronously by using Work Manager

€10.99

€124.99 91% off

🕒 5 hours left at this price!

Add to cart

Buy now

30-Day Money-Back Guarantee

Includes

- 📺 3 hours on-demand video
- 📄 1 downloadable resource
- 🌐 Full lifetime access
- 📱 Access on mobile and TV
- 📜 Certificate of Completion

