

Laboratorio 05 - Navigation Component

1. Introducción

En este laboratorio se explicarán conceptos básicos sobre Navigation Component, su arquitectura y sobre los componentes que lo conforman. Además será implementado a una aplicación para poder gestionar la navegación entre fragmentos de una manera más sencilla y estructurada

1.1 ¿Qué se aprenderá?

- Uso de Navigation Component y su integración a Android Studio
- Diseño y creación de un Navigation Graph
- Implementación de la navegación entre fragmentos
- Integración de Navigation Component en una aplicación

1.2. Requisitos

- Tener previamente instalado Android Studio en tu computadora
- Conocimiento previo en construir vistas con Constraint Layout
- Ejecutar acciones al presionar un botón

2. ¿Qué es Navigation Component?

El Navigation Component es una biblioteca de Android Jetpack que simplifica el proceso de implementación de la navegación en una aplicación. Proporciona una forma declarativa de definir las rutas de navegación entre varias pantallas (llamadas "destinos" en la terminología de Navigation Component) en tu aplicación.

Con Navigation Component, es posible implementar fácilmente características como:

- Navegar entre distintos fragmentos o actividades de tu aplicación
- Mostrar un botón de retroceso en la barra de acciones o en la barra de herramientas
- Paso de datos entre pantallas
- Animación de transiciones entre pantallas
- Gestión de enlaces profundos y navegación desde fuentes externas

Además, por medio de un grafo de navegación permite describir cómo se moverá el usuario dentro de la aplicación, y Navigation Component se encargará de gestionar todo el proceso de navegación, incluyendo transiciones entre fragmentos, animaciones y navegación hacia atrás.

3. ¿Por qué usar Navigation Component?

Navigation Component proporciona una serie de beneficios, tales como:

- Forma estructurada de definir la navegación de la aplicación
- Separación de la lógica de navegación de la lógica de negocios
- Mejor integración con otros componentes de Android Jetpack (ViewModel y LiveData)
- Facilitar el proceso para crear aplicaciones más robustas y escalables

En resumen, con Navigation Component la navegación se vuelve más sencilla, se facilita la comprensión de la misma y a su vez el mantenimiento y escalabilidad de la aplicación. Además, permite al desarrollador centrarse únicamente en la lógica de negocios, lo que vuelve el proceso de desarrollo más eficiente ya que Navigation Component se encarga de la gestión de navegación.

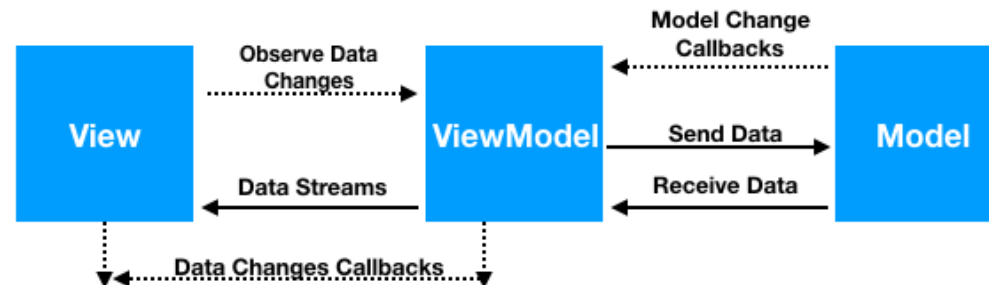
Sin olvidar que la experiencia de usuario se mejora radicalmente al brindar una navegación clara, estructurada e intuitiva, y al combinarlo con Material Design la aplicación mejora tanto en UX como en UI.

4. Arquitectura de Navigation Component

La arquitectura de Navigation Component se basa en tres componentes principales, los cuales son:

- Navigation Graph
- NavHost
- NavController

Asimismo, Navigation Component está diseñado para integrarse con la arquitectura o patrón de diseño **MVVM** (**M**odel **V**iew **V**iew**M**odel). Lo que permite modularizar la aplicación separando la capa que se encargará de gestionar y tener la lógica sobre los datos, de la parte visual de la aplicación. Lo que simplifica el código y proporciona una mayor escalabilidad.

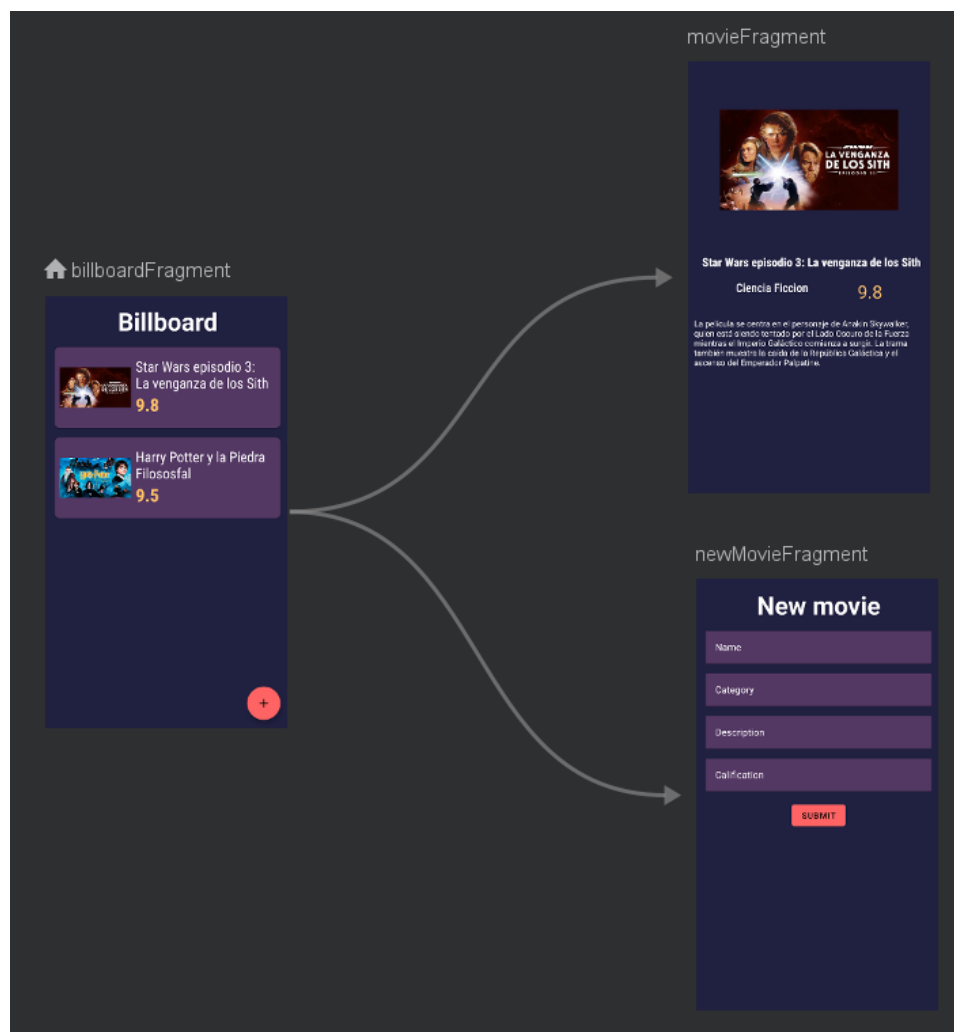


5. Componentes de Navigation Components

5.1 Navigation Graph

El Navigation Graph es un archivo de recursos **(XML)** que define la estructura de navegación de nuestra aplicación, especificando los destinos y acciones que serán utilizadas en el proceso de navegación. Cada destino representa una pantalla o una sección de la aplicación, como puede ser un fragmento o una actividad.

En la siguiente imagen es posible observar tres destinos que corresponden a los fragmentos conectados por dos acciones que indican el flujo de navegación.



5.2. NavHost

El NavHost es un contenedor que muestra los diferentes destinos (fragmentos o actividades) definidos en un Navigation Graph. Además, es el responsable de gestionar la pila de navegación y mostrar el destino apropiado en función de las acciones del usuario.

Al estar usando fragmentos y el Navigation Component la **MainActivity** ya no tiene porqué contener contenedor de diseño alguno, ahora pasa a utilizar un **FragmentContainerView** que se encargará de actuar como el NavHost para nuestros fragmentos.

5.3 NavController

El NavController es un objeto el cual está encargado de gestionar todo el proceso de navegación dentro de la aplicación. Es utilizado para navegar entre destinos, así como para procesar las acciones definidas en el Navigation Graph y mantener el estado de la navegación de nuestra aplicación.

En el siguiente ejemplo, cuando el botón llamado **btnCreateNewMovie** sea presionado se ejecutará la acción que permite moverse del **billboardFragment** hacia el fragmento llamado **newMovieFragment**

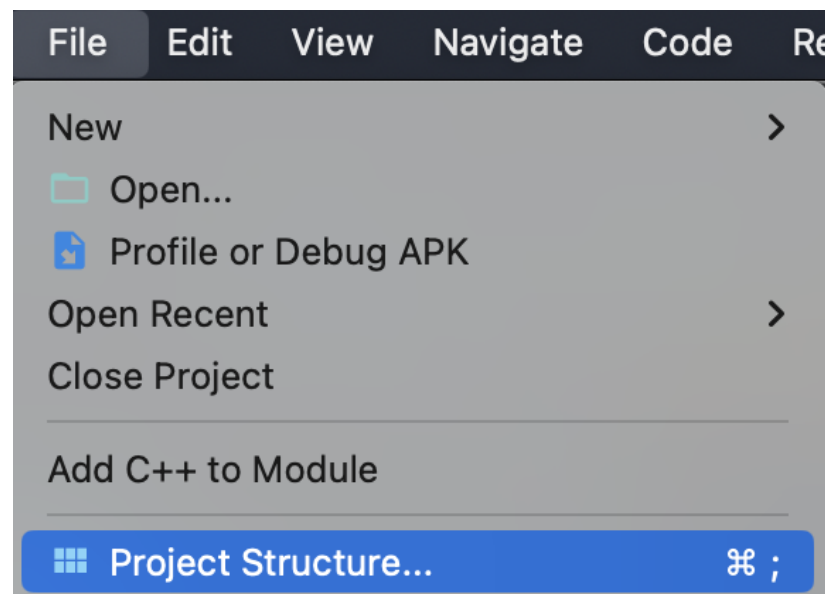
```
btnCreateNewMovie.setOnClickListener { it: View!
    it.findNavController().navigate(R.id.action_billboardFragment_to_newMovieFragment)
}
```

5.4 ¿Cómo utilizar Navigation component en Android?

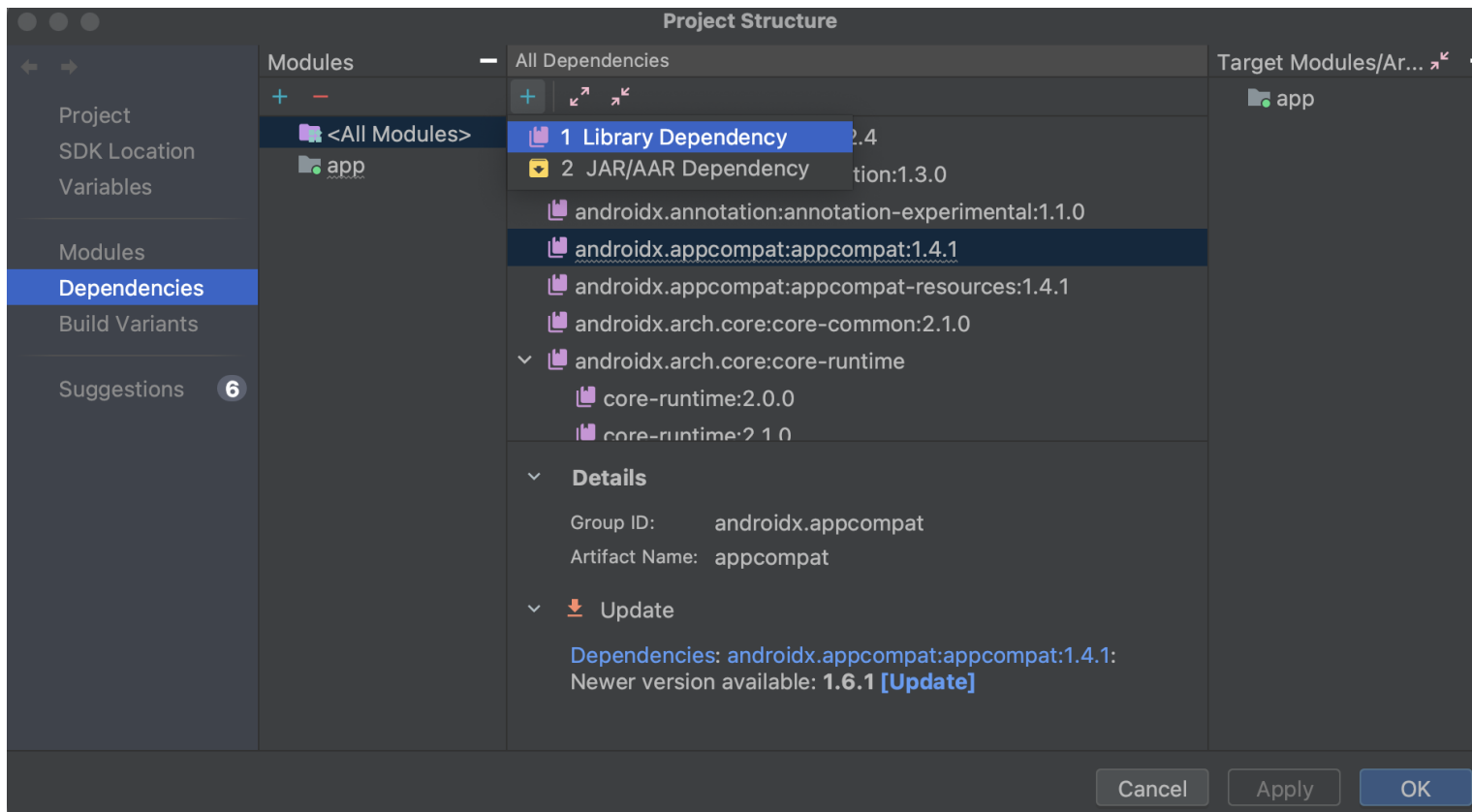
5.4.1 Agrega la dependencia de Navigation component al proyecto

Para utilizar Navigation component en una aplicación Android, es necesario añadir la biblioteca de navegación al archivo build.gradle de tu aplicación:

Dirígete a **Project Structure** dentro de **File**



En **All Dependencies** presiona el símbolo **+** y seguidamente selecciona **Library Dependency** para poder añadir una nueva dependencia



Ahora dentro de la siguiente ventana escribe **“navigation-fragment-ktx”** y presiona el botón **Search**, y dentro de las dependencias que aparezcan escoge la proveniente de **“androidx.navigation”** y selecciona la última versión para posteriormente presionar **OK**

Add Library Dependency

Module 'app'

Step 1.
Use the form below to find the library to add. This form uses the repositories specified in the project's build files (Google, Maven Central)

navigation-fragment-ktx Search

Enter a search query or fully-qualified coordinates (e.g. guava* or com.google.*:guava* or com.google.guava:guava:26.0)

Group ID	Artifact Name	Repository	Versions
android.arch.navigation	navigation-fragment-ktx	Google	2.6.0-alpha09
androidx.navigation	navigation-fragment-ktx	Google	2.6.0-alpha08
com.microsoft.device.dualscreen	navigation-fragment-ktx	Maven Central	2.6.0-alpha07
			2.6.0-alpha06
			2.6.0-alpha05

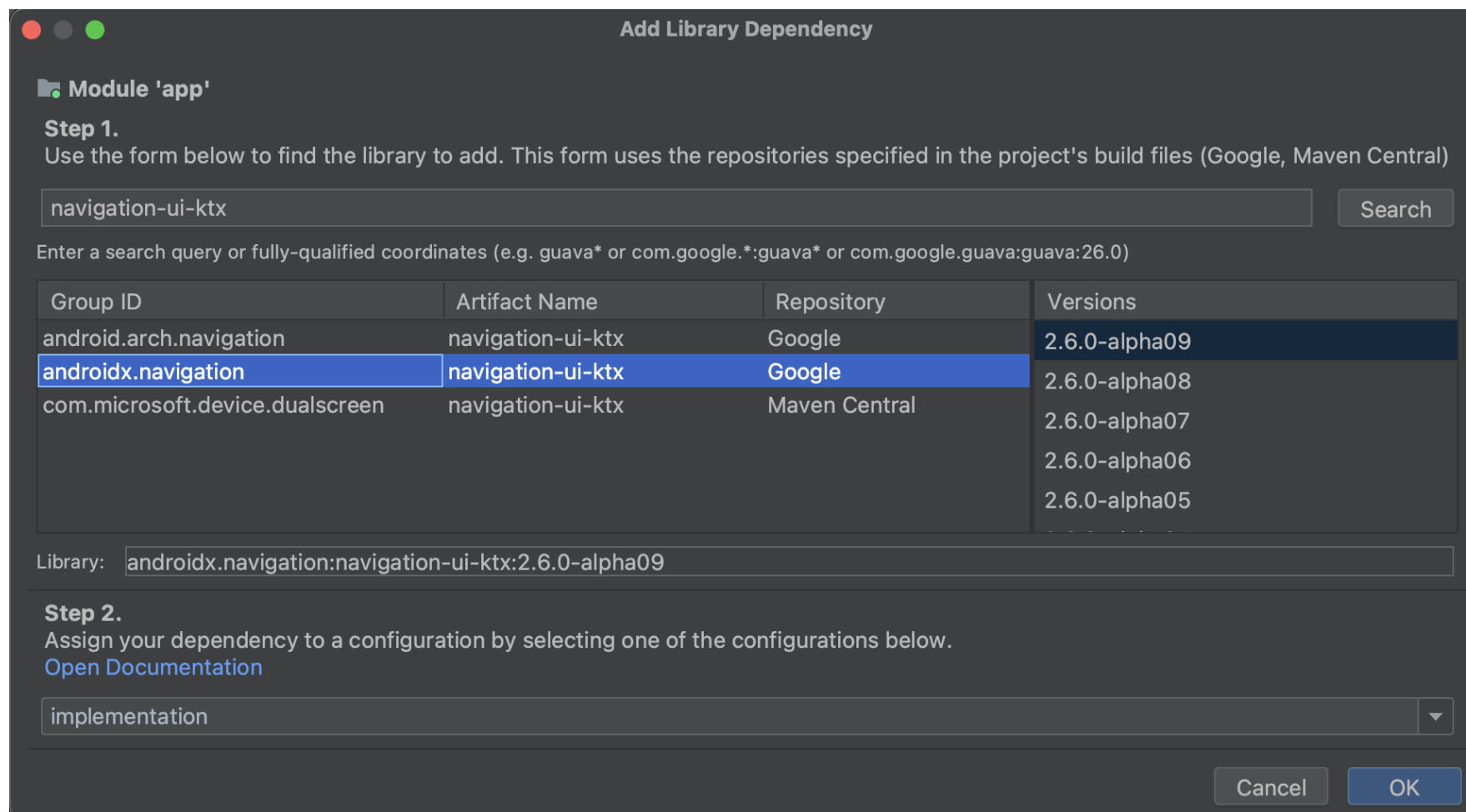
Library: androidx.navigation:navigation-fragment-ktx:2.6.0-alpha09

Step 2.
Assign your dependency to a configuration by selecting one of the configurations below.
[Open Documentation](#)

implementation ▼

Cancel OK

Realiza el procedimiento realizado previamente para implementar **“navigation-ui-ktx”** proveniente de **“androidx.navigation”** y escoge su última versión

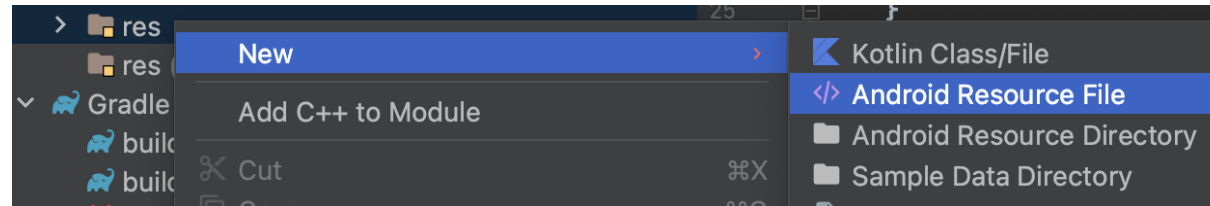


Y verifica en **build.gradle (Module: app)** que dichas dependencias hayan sido añadidas correctamente

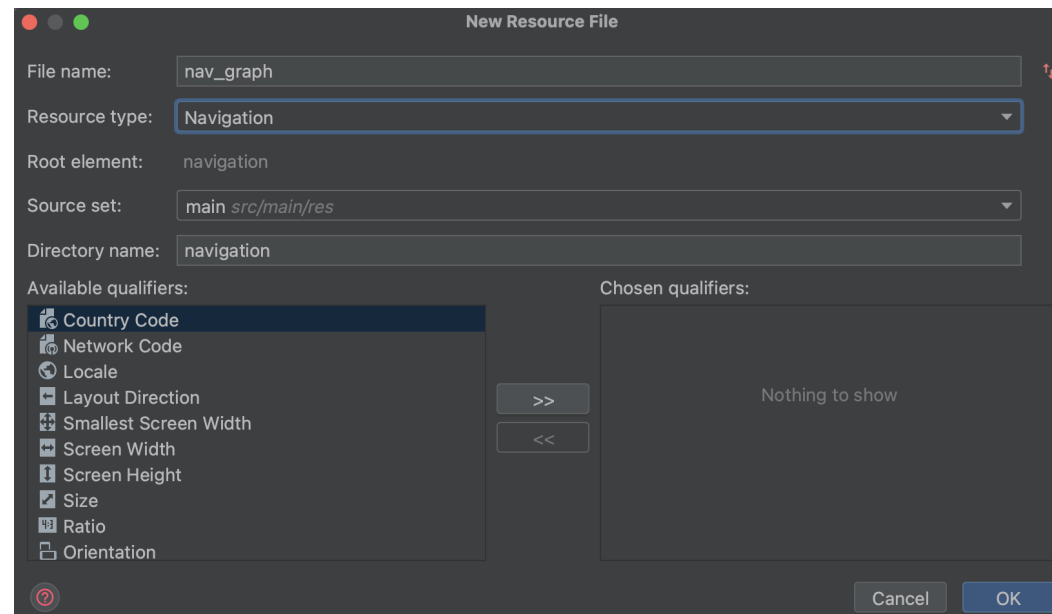
```
dependencies {  
  
    implementation 'androidx.core:core-ktx:1.8.0'  
    implementation 'androidx.appcompat:appcompat:1.4.1'  
    implementation 'com.google.android.material:material:1.5.0'  
    implementation 'androidx.constraintlayout:constraintlayout:2.1.3'  
     implementation 'androidx.navigation:navigation-fragment-ktx:2.6.0-alpha09'  
    implementation 'androidx.navigation:navigation-ui-ktx:2.6.0-alpha09'  
    testImplementation 'junit:junit:4.13.2'  
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'  
}
```

5.4.2 Configuración del Navigation Graph

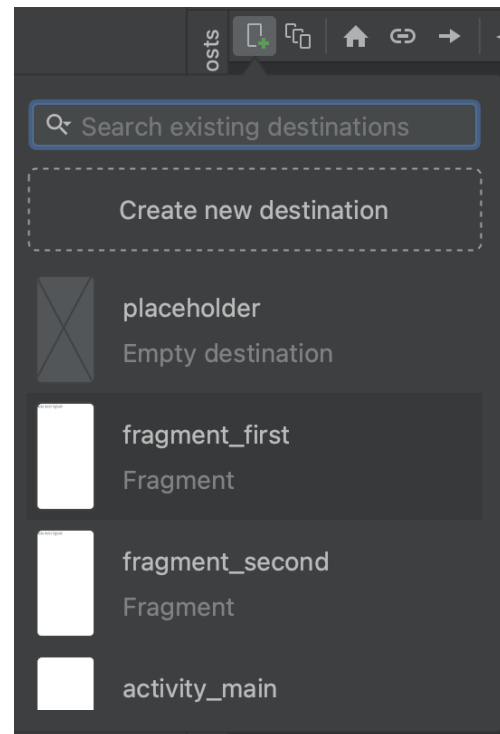
Ahora que las dependencias han sido añadidas, es necesario crear un nuevo **Android Resource File** dentro de la carpeta **res**



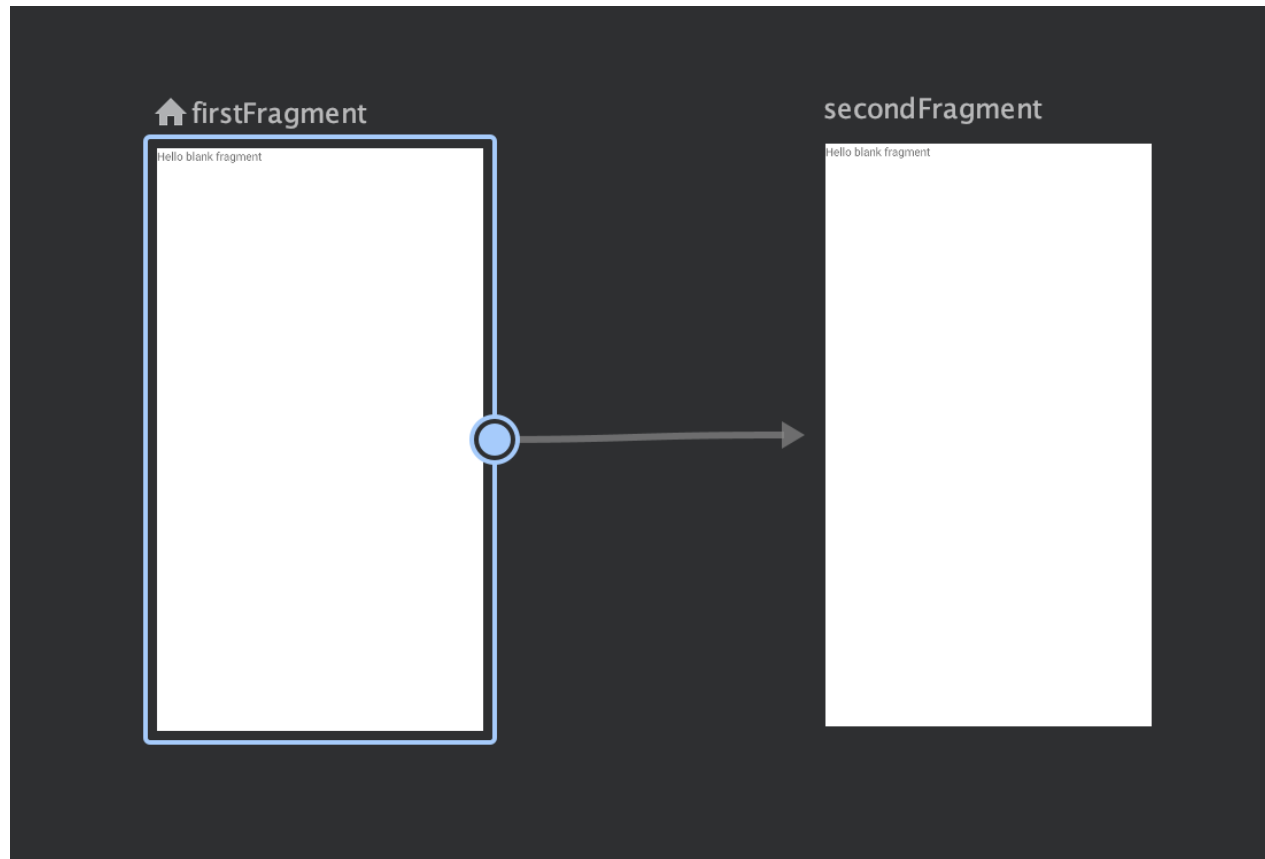
Se escoge en **Resource type** la opción de **Navigation** y como nombre de archivo **nav_graph**



Para brindar un ejemplo, se han creado dos fragmentos, llamados **FirstFragment** y **SecondFragment**. Ahora dentro del archivo **nav_graph.xml** se presiona en **New Destination** y se agregan ambos fragmentos



Ya que ambos fragmentos han sido añadidos, debes posicionar sobre el primer fragmento y crear una acción hacia el segundo fragmento arrastrando la flecha



5.4.3 Configuración del NavHost

Dentro del archivo **activity_main.xml** se procede a agregar un componente llamado **FragmentContainerView** donde se configuraba por medio de **name** a dicha actividad como el **NavHostFragment**. Además, se agrega el atributo **defaultNavHost** como **verdadero**, lo que permitirá al contenedor interactuar con la jerarquía de navegación definida en el **Navigation Graph**. Finalmente usando el atributo **navGraph** se le define la ruta donde está ubicado nuestro **nav_graph**.

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:name="androidx.navigation.fragment.NavHostFragment"
        app:navGraph="@navigation/nav_graph"/>

</FrameLayout>
```

5.4.4 Uso del NavController

Ya que ha sido configurado tanto el **Navigation Graph** que define los destinos y las acciones y el **NavHost** que funciona como contenedor y se encarga de mostrar los destinos, únicamente queda movilizarse entre dichos fragmentos. Por medio de ***findNavController().navigate()*** es posible navegar entre fragmentos, únicamente es necesario indicar la acción a ejecutar.

```
class FirstFragment : Fragment() {

    private lateinit var buttonFirstFragment: Button

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

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)
        bind()
        buttonFirstFragment.setOnClickListener { it: View!
            it.findNavController().navigate(R.id.action_firstFragment_to_secondFragment2)
        }
    }

    private fun bind() {
        buttonFirstFragment = view?.findViewById(R.id.button_first_fragment) as Button
    }
}
```



6. Paleta de colores de la aplicación

name	color
backgroundLayout	#202040
cardLayout	#543864
calification	#FFBD69
buttonColor	#FF6363


7. Aplicación a realizar

4:21

Billboard




Star Wars episodio 3:
La venganza de los Sith
9.8



Harry Potter y la Piedra
Filosofal
9.5

+

4:21



Star Wars episodio 3: La venganza de los Sith

Ciencia Ficción 9.8

La película se centra en el personaje de Anakin Skywalker, quien está siendo tentado por el Lado Oscuro de la Fuerza mientras el Imperio Galáctico comienza a surgir. La trama también muestra la caída de la República Galáctica y el ascenso del Emperador Palpatine.

4:23

New movie

Name

Category

Description

Calification

SUBMIT

8. To do:

1. ¿Qué diferencia existe entre navegar dentro de la aplicación utilizando **Navigation Component** o hacer uso de **intents**?

2. Menciona los **componentes fundamentales** de **Navigation Component**

3. Mencione algunas de las **ventajas** del **Navigation Graph**

9. Rúbrica

Actividad	Porcentaje	Nota
Elementos visuales posicionados correctamente en todos los fragmentos	10%	
Extracción de recursos (string, dimens)	10%	
Implementación de Navigation Graph	15%	
Definición de NavHost	10%	
Navegación entre fragmentos (NavController)	15%	
La aplicación se ejecuta correctamente con todas las funcionalidades requeridas	20%	
Preguntas teóricas	10%	
Puntualidad	10%	
Total	100%	