

Android Jetpack's Navigation component

1TEL05 - Servicios y Aplicaciones para IoT

Navigation in Android



La “navegación” en Android se refiere a las interacciones que permiten a los usuarios navegar a través, dentro y fuera de las diferentes partes de la aplicación.

El componente **Navigation de Android Jetpack** permite implementar la navegación, desde simples clics de botones hasta patrones más complejos, como las TopAppBar (BottomAppBar) y los paneles laterales de navegación.

El componente Navigation también garantiza una experiencia del usuario coherente y predecible, ya que se adhiere a un conjunto de principios de buenas prácticas.

Así mismo, sigue la arquitectura Single Activity.

<https://youtu.be/LmkKFCfmnhQ?si=5qJsSxQRGnC9Fo-9&t=90>

El componente Navigation

Consta de tres partes claves:

- **Navigation graph**: Es un XML que contiene toda la información relacionada con la navegación, incluye todas las áreas de contenido individuales dentro de tu app, llamadas destinos, así como las posibles rutas que un usuario puede tomar a través de tu app.
- **NavHost**: Es un contenedor vacío que muestra los destinos de tu gráfico de navegación. El componente Navigation contiene una implementación NavHost predeterminada, NavHostFragment, que muestra distintos de fragmentos.
- **NavController**: Es un objeto que administra la navegación de la app dentro de un NavHost. NavController orquesta el intercambio de contenido de destino en el objeto NavHost a medida que los usuarios se mueven a través de la app.

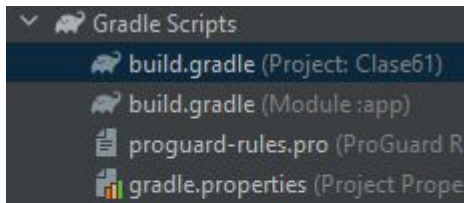
Se utiliza NavController para indicar la ruta específica del gráfico de navegación o directamente a un destino específico. Luego, NavController muestra el destino apropiado en NavHost.

Dependencias


Para utilizar navigation component, necesita:

```
dependencies {  
    implementation 'androidx.navigation:navigation-fragment:2.5.3'  
    implementation 'androidx.navigation:navigation-ui:2.5.3'
```

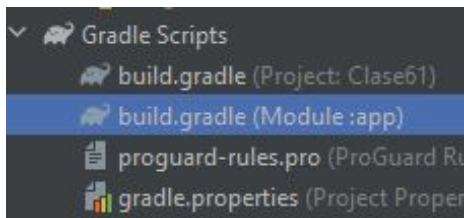
Así mismo, en las configuraciones del proyecto:




```
plugins {  
    id 'com.android.application' version '7.4.2' apply false  
    id 'com.android.library' version '7.4.2' apply false  
    id 'androidx.navigation:safeargs' version '2.5.2' apply false  
}
```



En las configuraciones de módulo:

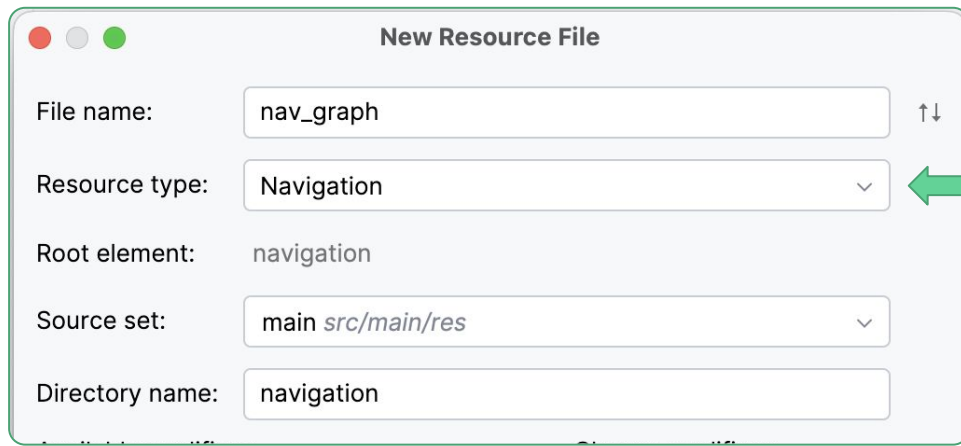


```
plugins {  
    id 'com.android.application'  
    id 'androidx.navigation:safeargs'  
}
```



Crear un Navigation Graph

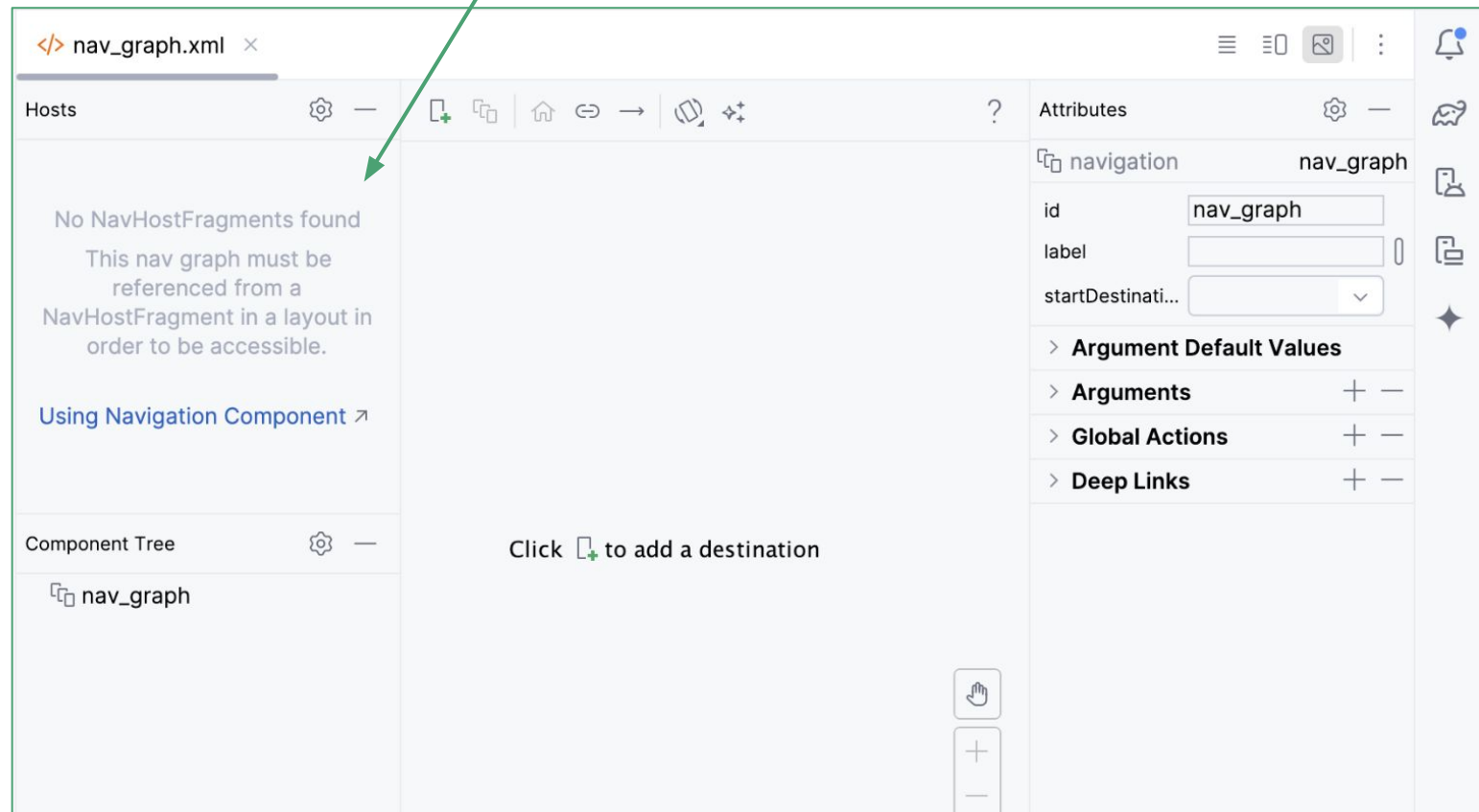
Un **Navigation Graph** es un archivo de recursos que contiene todos los destinos y acciones de la aplicación.



Seleccione Navigation

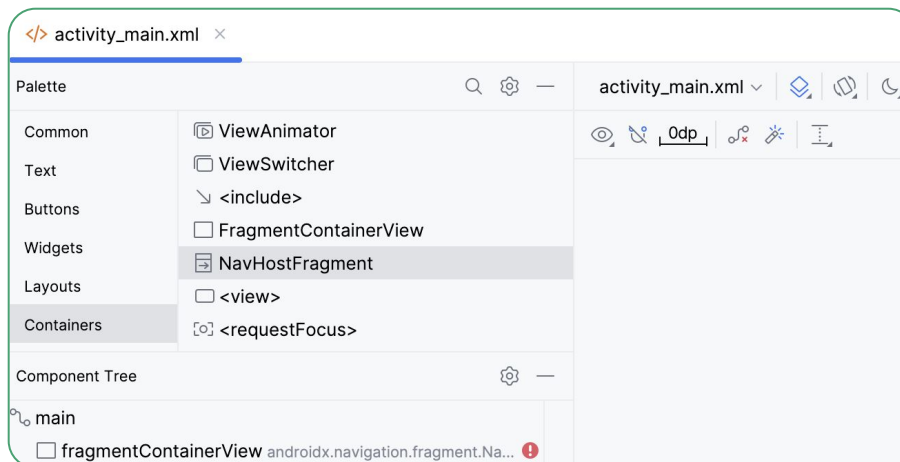
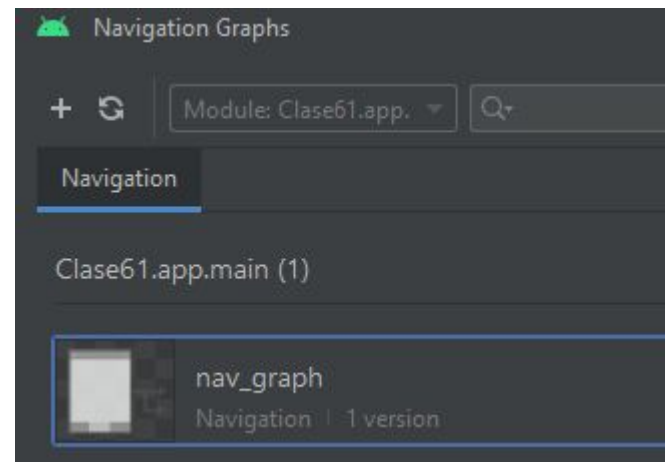
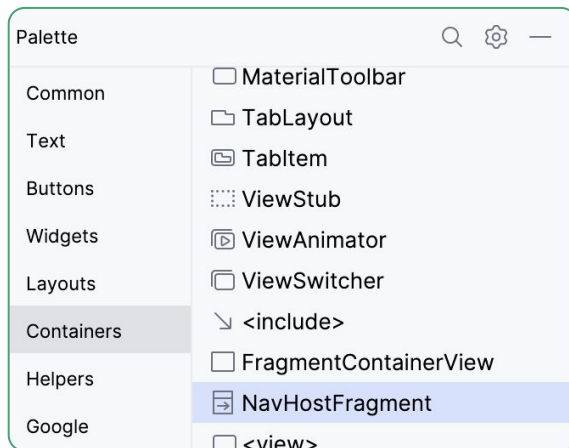
Navigation Graph

Se le indica que no hay un **NavHostFragment**, el cual es el componente donde se van a ir colocando/retirando los fragmentos de forma dinámica.



Agregando NavHostFragment

En el XML del Activity: **Containers** → **NavHostFragment** y seleccione su nav graph.



Observará que aparece un error... Falta adicionar nuestros fragmentos.

[Ver comentarios](#)

Fragmentos de prueba

Cree 3 fragmentos. FragmentA, B y C.


```
public class FragmentA extends Fragment {  
  
    FragmentABinding binding;  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
        binding = FragmentABinding.inflate(inflater, container, false);  
  
        return binding.getRoot();  
    }  
}
```

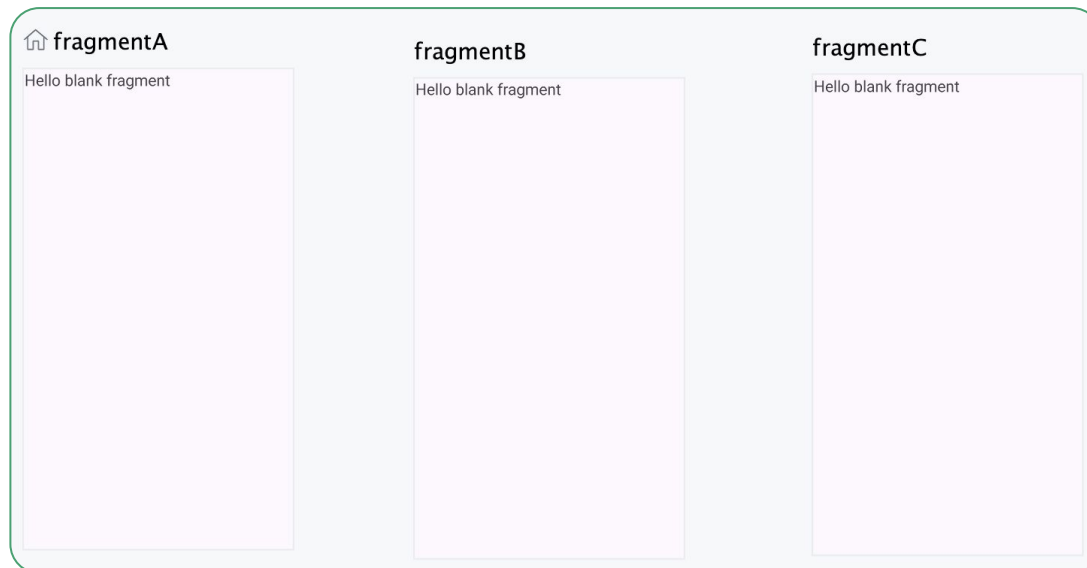
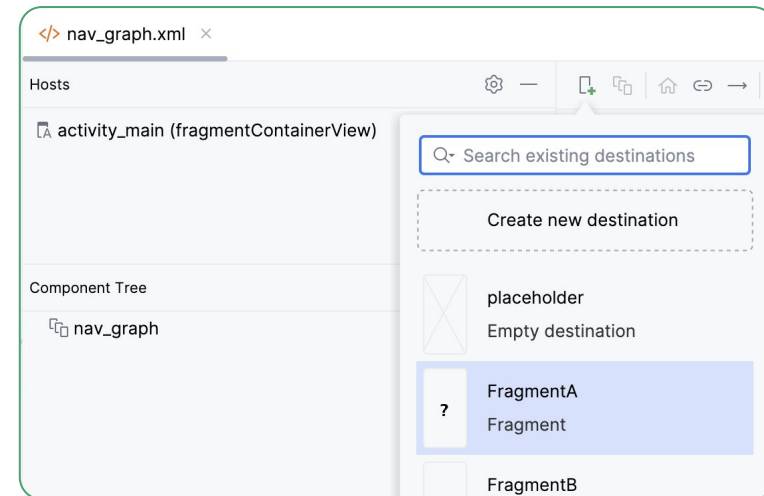
```
public class FragmentB extends Fragment {  
  
    FragmentBBinding binding;  
  
    @Override  
    public View onCreateView(LayoutInflater inflater,  
                             ViewGroup container,  
                             Bundle savedInstanceState) {  
        binding = FragmentBBinding.inflate(inflater, container, false);  
  
        return binding.getRoot();  
    }  
}
```

```
public class FragmentC extends Fragment {  
  
    FragmentCBinding binding;  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {  
        binding = FragmentCBinding.inflate(inflater, container, false);  
  
        return binding.getRoot();  
    }  
}
```


Navigation Graph

En su gráfica de navegación, adiciones sus fragmentos.

El fragmento con el ícono  a la izquierda, indica que será el primero en mostrarse (home).



Implementar navegación

En este ejemplo, se crearán botones para:

- Ir del fragmento A al fragmento B
- Del fragmento B al fragmento C
- Del fragment C al fragmento A

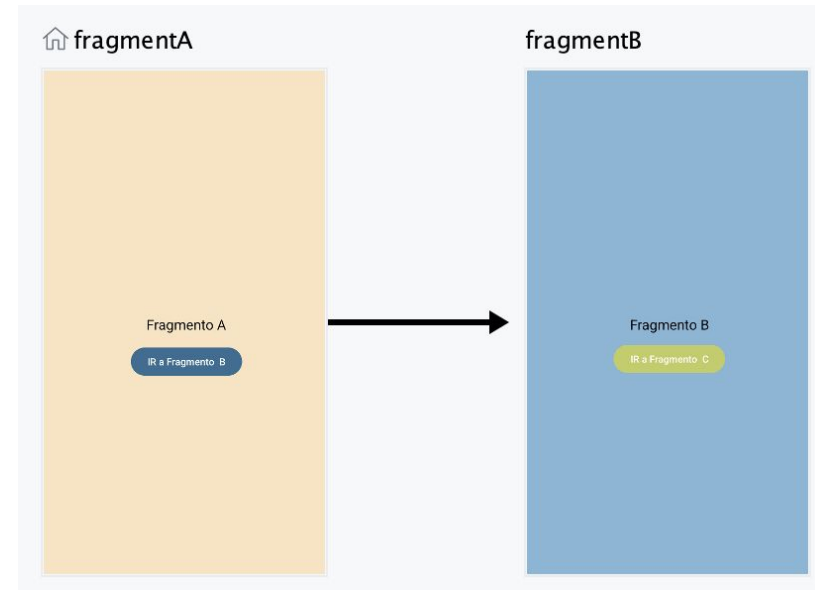
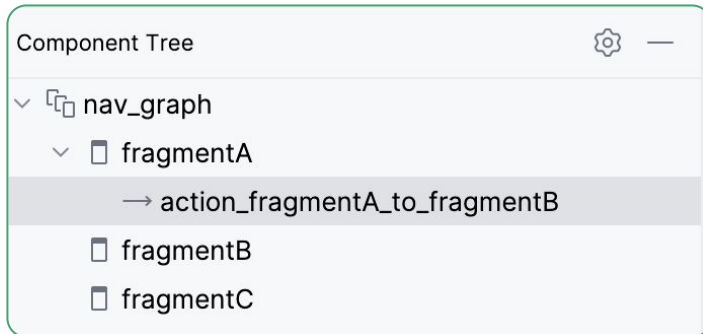


Implementar navegación → Nav graph

Arrastre el borde del fragmento A al B.

Esto creará un Action, el cual representa un origen y un destino, y su ID es:

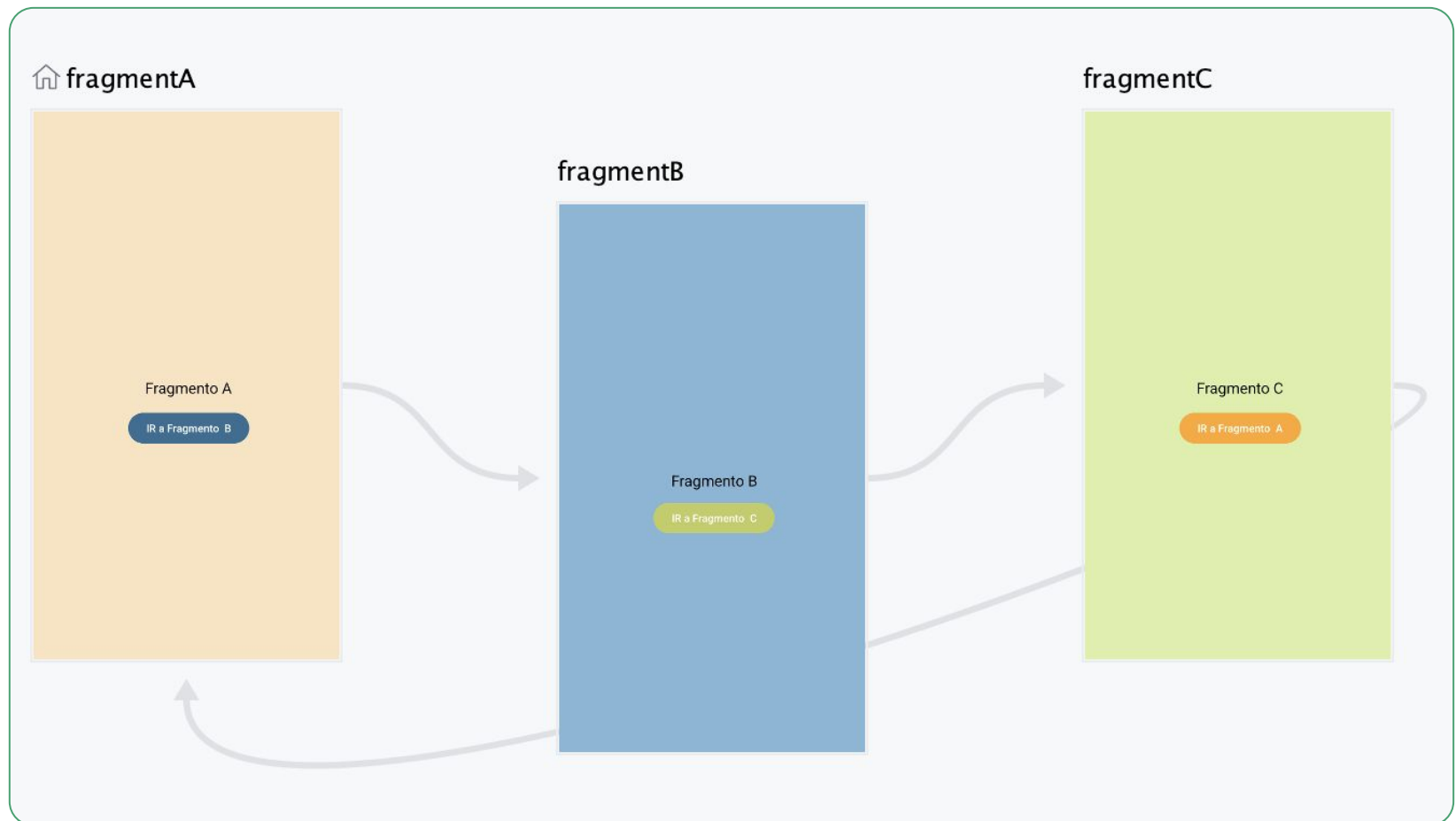
action_fragmentA_to_fragmentB



```
<fragment
  android:id="@+id/fragmentA"
  android:name="com.example.clase61.FragmentA"
  android:label="fragment_a"
  tools:layout="@layout/fragment_a" >
  <action
    android:id="@+id/action_fragmentA_to_fragmentB"
    app:destination="@id/fragmentB" />
</fragment>
```

Implementar navegación → Nav graph

Termine de implementar la navegación entre todos los fragmentos.



Navegar a un destino

La navegación a un destino se realiza con el ***NavController***. Puede obtener una instancia con cualquiera de los siguientes métodos:

- `NavHostFragment.findNavController(Fragment)`
- `Navigation.findNavController(Activity, @IdRes int viewId)`
- `Navigation.findNavController(View)`

FragmentA.java

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                          Bundle savedInstanceState) {
    binding = FragmentABinding.inflate(inflater, container, false);

    NavController navController = NavHostFragment.findNavController(FragmentA.this);

    binding.button.setOnClickListener( view -> {

    });
}
```

Navegar a un destino

Con el NavController disponible, puede utilizar el método navigate para ir a un destino (previamente creado). Puede ir de dos formas:

- Opción 1: Usando el ID del destino:

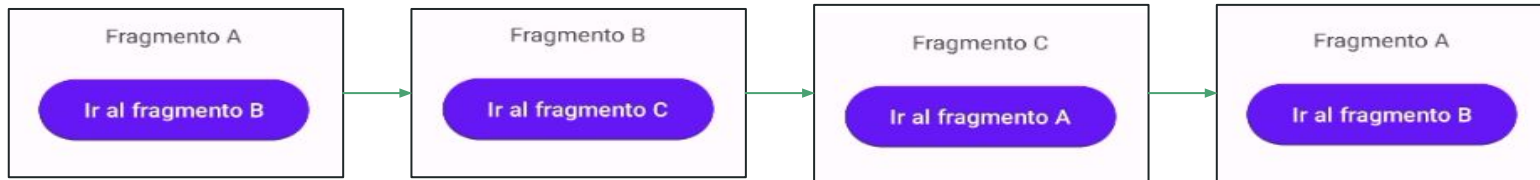
```
binding.button.setOnClickListener(view -> {  
    navController.navigate(R.id.action_fragmentA_to_fragmentB);  
})
```

- Opción 2: Usando la clase Directions (creada automáticamente) con el “action” correspondiente.

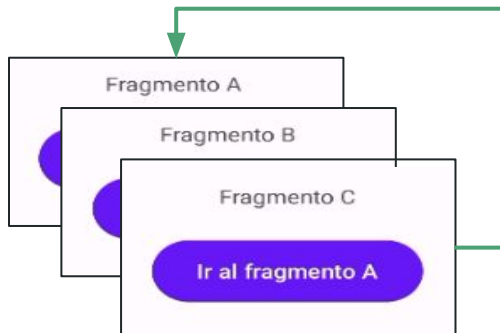
```
binding.button.setOnClickListener(view -> {  
    navController.navigate(FragmentADirections.actionFragmentAToFragmentB());  
})
```

Implementar todas las navegaciones

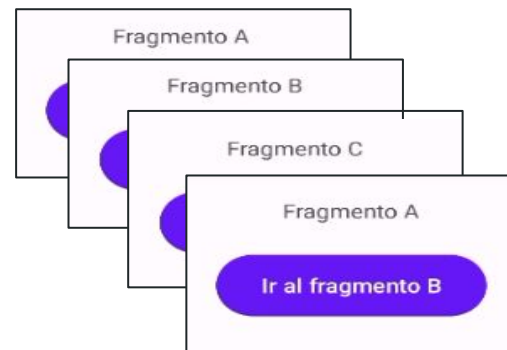
Si prueba la navegación observará que:



Lo que parece que sucede en el backstack:



Lo que en realidad está sucediendo:



Navigation y el backstack

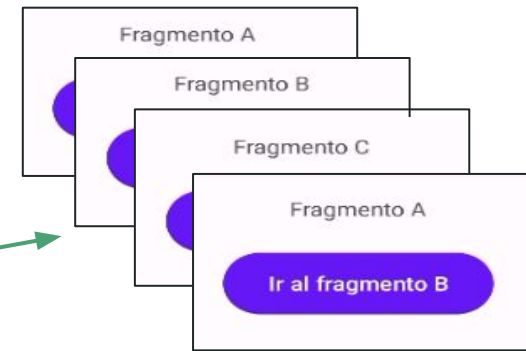
Android mantiene un Backstack que incluye los destinos a los que se navegó. Cuando el usuario abre la app, se coloca el primer destino de la app en la pila.

Cada llamada al método **navigate()** ubica otro destino sobre la pila. Cuando se presiona *Up* o *Back*, se llama a los métodos **NavController.navigateUp()** o **NavController.popBackStack()**, respectivamente, para quitar el destino superior de la pila.

Suponga que luego del login va a la pantalla principal. Si el usuario presiona atrás no debería mostrar el login, sino, cerrar la app.

Navigation y el backstack

Para solucionar el problema de:



En el action que va del fragmento C al fragmento A debemos incluir:

- **app:popUpTo="X"** → indica que debe sacar del backstack todos los fragmentos hasta "X". Si indicamos **app:popUpTo="@+id/FragmentA"**, el backstack quedará:



Adicionalmente, agregar:

- **app:popUpToInclusive="true"** → indica que debe sacar del backstack el fragmento indicado en popUpTo. En este caso el primer Fragmento A.



XML

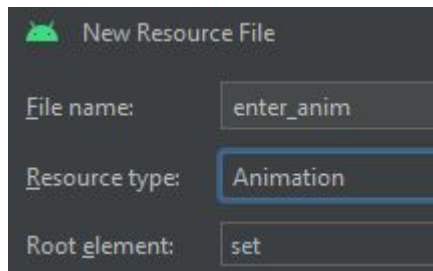
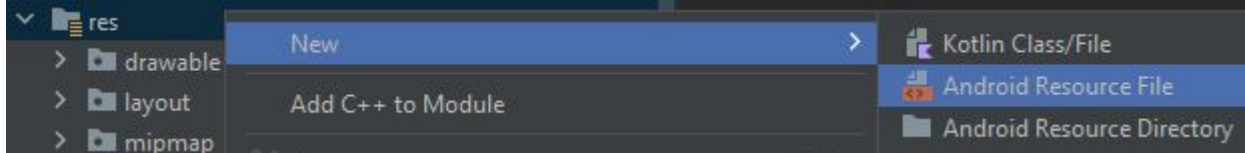
Quedando la navegación en C:

```
<fragment
    android:id="@+id/fragmentC"
    android:name="com.example.clase61.FragmentC"
    android:label="fragment_c"
    tools:layout="@layout/fragment_c" >
    <action
        android:id="@+id/action_fragmentC_to_fragmentA"
        app:destination="@id/fragmentA"
        app:popUpTo="@id/fragmentA"
        app:popUpToInclusive="true" />
</fragment>
```

Animar transiciones

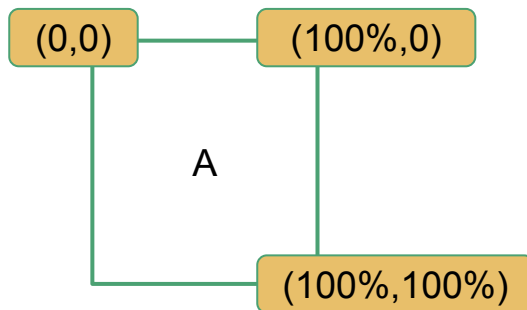
Para animar el cambio de fragmento debe crear 4 animaciones:

- **enterAnim**: Entrada del fragmento
- **exitAnim**: salida del fragmento
- **popEnterAnim**: Entrada del fragmento via popAction
- **popExitAnim**: salida del fragmento via popAction

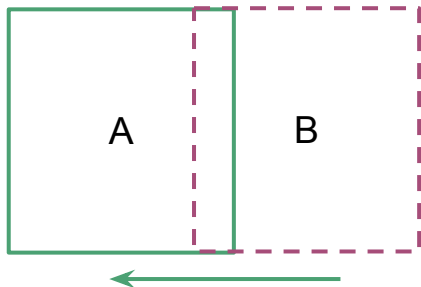


X,Y para layouts

En un Layout en Android, el punto 0,0 es la esquina superior izquierda:



Si se desea animar la “entrada” de un layout (B), entonces habría que moverlo de la derecha a la izquierda.



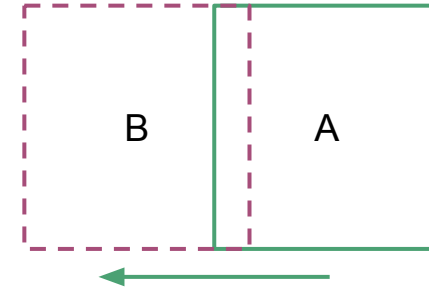
Se podría indicar que se mueva 100% a 0%, en 200ms.

```
enter_anim.xml
<set
xmlns:android="http://schemas.android.com/apk/res/android"
  <translate android:duration="200"
    android:fromXDelta="100%"
    android:toXDelta="0%"
  />
</set>
```

Otras animaciones

Para simular la salida: debería ir de 0% a -100%

```
exit_anim.xml x
<set xmlns:android="http://schemas.android.com/apk/res/android"
  <translate
    android:fromXDelta="0%"
    android:toXDelta="-100%"
    android:duration="200" />
  </set>
```



Entrada del fragmento via popAction:

```
pop_enter_anim.xml x
<set xmlns:android="http://schemas.android.com/apk/res/android"
  <translate
    android:fromXDelta="-100%"
    android:toXDelta="0%"
    android:duration="200" />
  </set>
```

popExitAnim: salida del fragmento via popAction:

```
pop_exit_anim.xml x
<set xmlns:android="http://schemas.android.com/apk/res/android"
  <translate
    android:fromXDelta="0%"
    android:toXDelta="100%"
    android:duration="200" />
  </set>
```

Usando animaciones

Las animaciones definidas se definen para cada acción:

▼ Animations		
enterAnim	@anim/enter_anim	<input type="checkbox"/>
exitAnim	@anim/exit_anim	<input type="checkbox"/>
popEnterAnim	@anim/pop_enter_anim	<input type="checkbox"/>
popExitAnim	@anim/pop_exit_anim	<input type="checkbox"/>

Enviar data entre fragmentos

Utilice Live Data pues todo está con una misma actividad.

Si desea enviar datos pequeños como un ID o unos cuantos valores, puede utilizar safe args: <https://developer.android.com/guide/navigation/navigation-pass-data>

```
FragmentA.java ×  
navController.navigate(FragmentADirections.actionFragmentAToFragmentB("Juan"));
```

```
FragmentB.java ×  
FragmentBArgs bundle = FragmentBArgs.fromBundle(getArguments());  
String name = bundle.getName();  
Log.d("msg-test", "Name: " + name);
```

¿Preguntas?

Muchas gracias
