

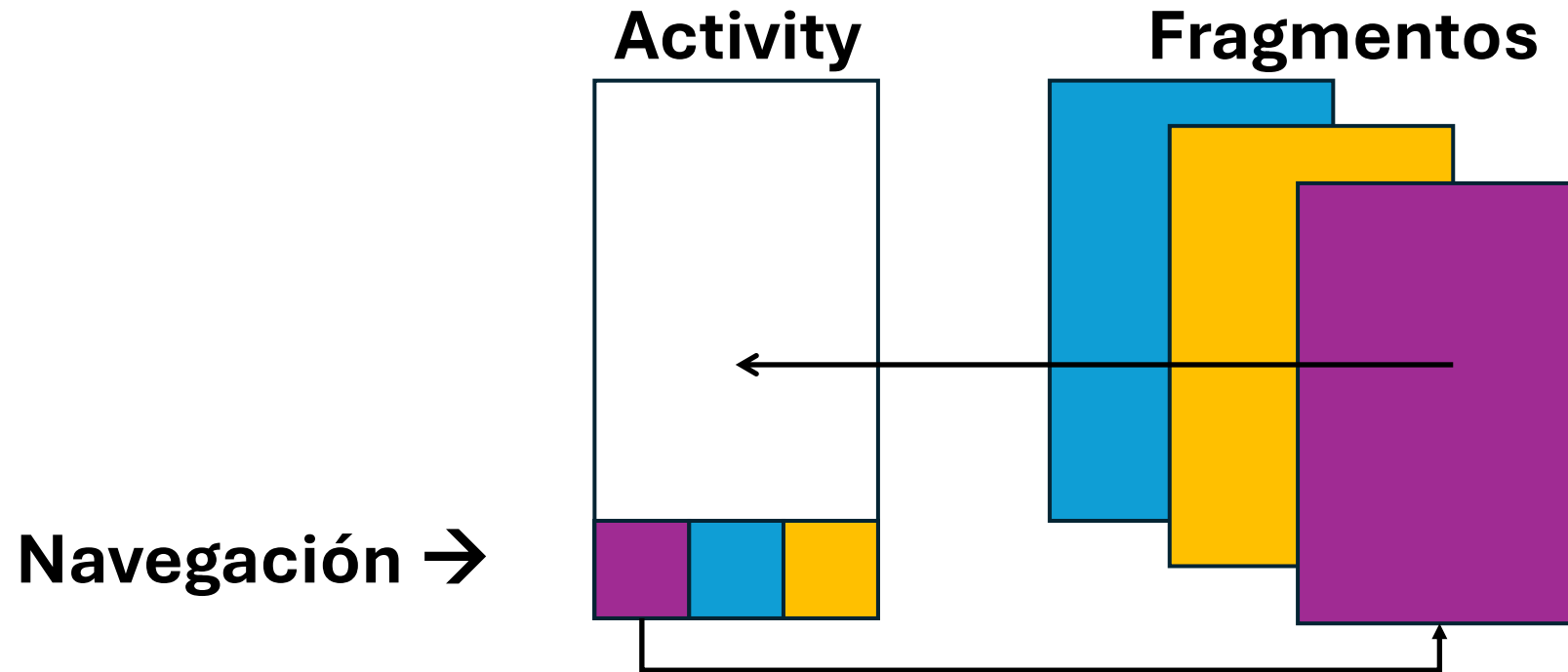
Software para Dispositivos Móviles

Miguel Sánchez Santillán
sanchezsmiguel@uniovi.es

Single Activity – Componentes clave

- **Fragmentos:** Representan pantallas **parciales** de la aplicación
 - Se instancian **dentro de una Activity** o dentro de otro Fragment
 - Tienen su propio ciclo de vida, eventos....
- **Navegación:** Representa todos los posibles destinos
 - Se define en un fichero **XML**
 - Navegar de una pantalla a otra implica **cambiar el fragmento mostrado**

Single Activity – Idea fundamental



Navegación – Partes fundamentales

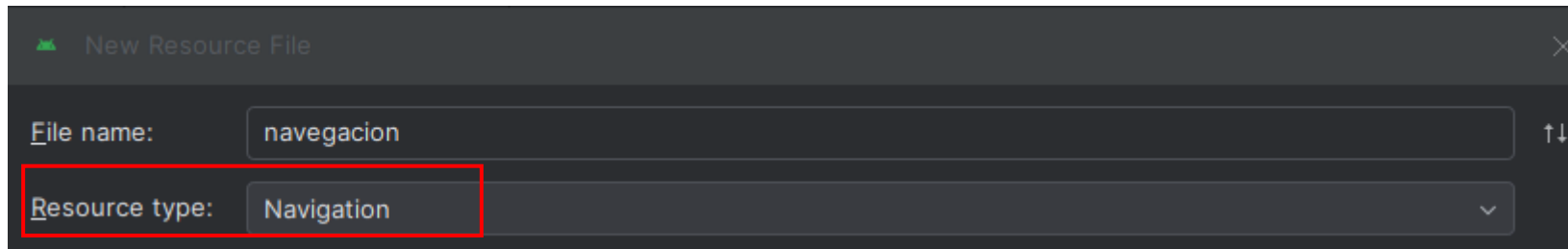
- Un **componente de navegación con un menú XML** asociado
 - **BottomNavigationView** o **Navigation Drawer**
- Fichero **XML** representando un **grafo de navegación**
 - Define posibles *rutas de navegación* entre fragmentos
- **NavHostFragment**: Componente que almacena el Fragment actual
- **NavController**: Se encarga de gestionar el proceso de navegación

Navegación – Componente de navegación

- Crea un proyecto en blanco (empty views activity)
- En el layout de la Activity, añade un **BottomNavigationView**
 - Haz que ocupe todo el ancho y se ajuste al contenido en la vertical
 - Debes situarlo en la parte inferior
 - Asígnale una **id**
- **Más adelante** le asociarás un menú XML

Navegación – Crear un XML de navegación

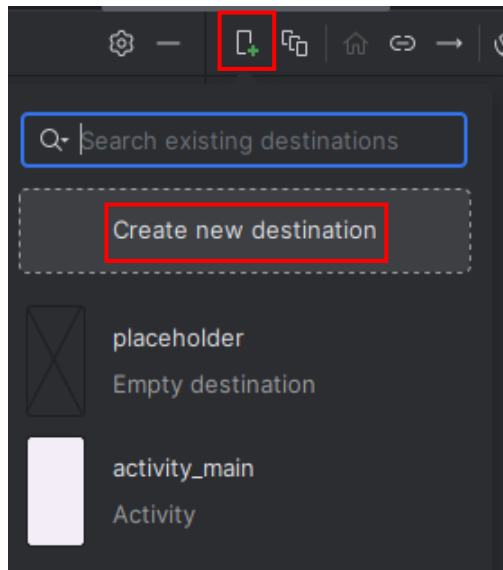
- Clic derecho en la carpeta res → New.. → Android Resource File



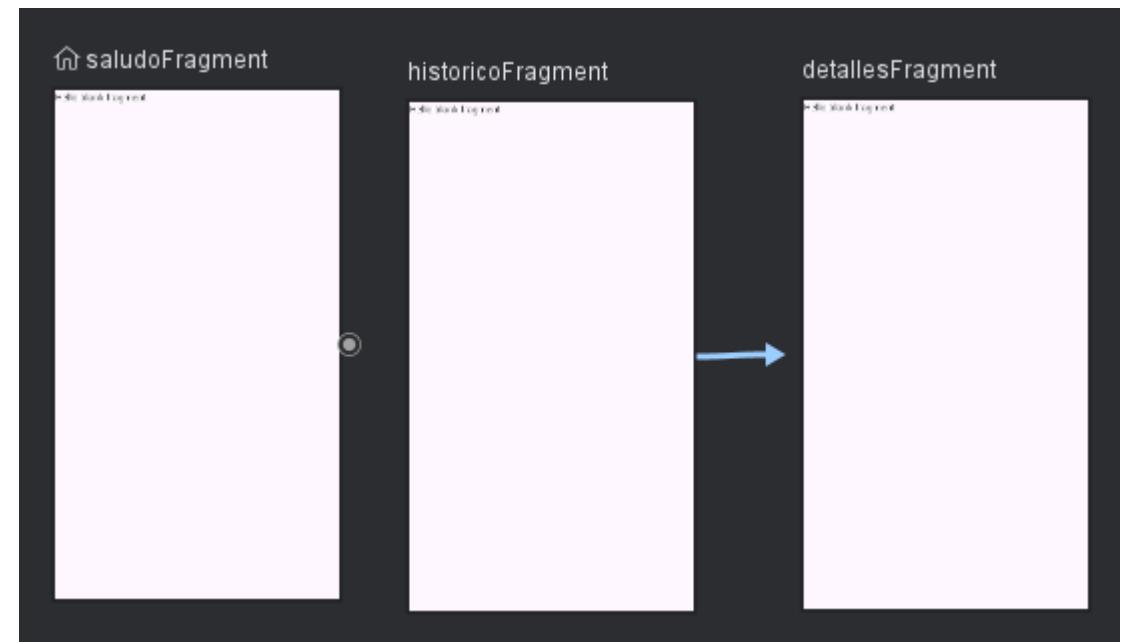
- Acepta el posible problema/error de compatibilidad
- Esto habrá generado el **fichero en la carpeta res/navigation**
 - **Ábrelo**

Navegación – Grafo y Fragmentos

- Añade destinos mediante fragmentos (Fragment Blank)
 - SaludoFragment
 - HistoricoFragment
 - DetallesFragment



- **Replica el siguiente grafo:**



Navegación – Grafo en XML

- Analiza el XML e **identifica**:
 - Qué propiedades son necesarias para definir un fragmento en la navegación
 - Cómo se indica el fragmento inicial o home
 - Cómo se define la navegación entre dos fragmentos

Navegación – NavHostFragment I

- Componente en el que se **cargará el fragmento actual**
- En el **layout de la activity**, búscalos en la **paleta de componentes y añádelos**
 - Selecciona el fichero XML de navegación creado en el paso anterior
- Ajusta el componente para que **ocupe el espacio restante**
- **Cambia `<androidx.fragment.app.FragmentContainerView` por `<fragment`**

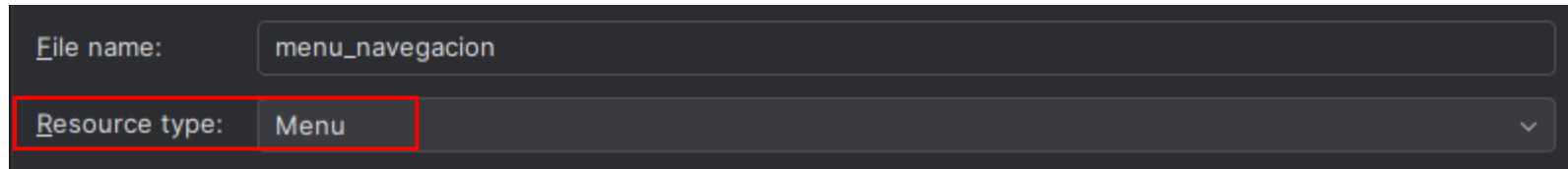
Navegación – NavHostFragment y II

- Deberías tener algo similar a la imagen
- Fíjate en el **defaultNavHost** y en el **navGraph**

```
<fragment
    android:id="@+id/fragmentContainerView"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    app:defaultNavHost="true"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toTopOf="@id/bottomNav"
    app:navGraph="@navigation/navegacion" />
```

Navegación – Crear el menú

- Clic derecho en la carpeta res → New.. → Android Resource File



The screenshot shows a dialog box with two fields. The first field, labeled 'File name:', contains the text 'menu_navegacion'. The second field, labeled 'Resource type:', contains the text 'Menu' and has a dropdown arrow on its right side. A red rectangle highlights the 'Resource type:' field.

- Añade al menú **dos ítems**: 

- En la vista XML hay que añadir **una id a cada ítem** ¿De dónde salen las ids? **Completa la id que falta**

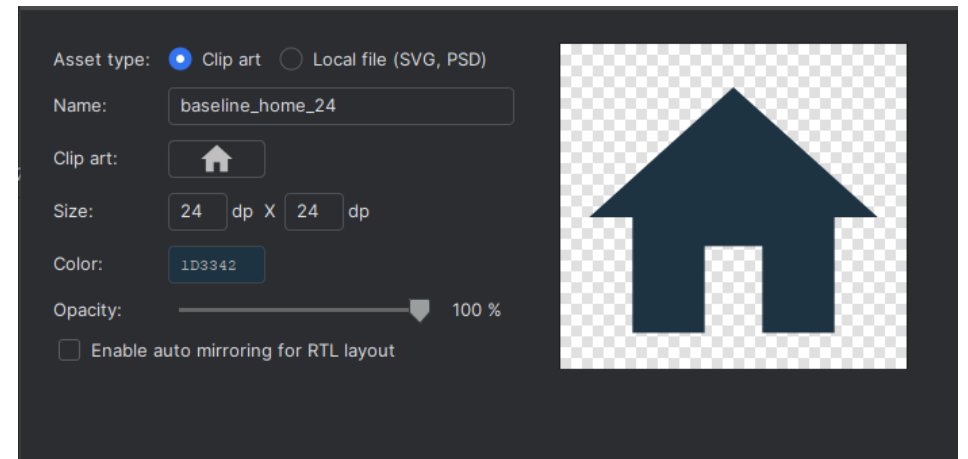
```
<item
    android:title="Inicio"
    android:id="@+id/saludoFragment"
/>
<item
    android:title="Histórico"
    android:id=""
/>
```

Navegación – Añadir iconos al menú

- En el **editor visual del menú**, selecciona un ítem y en los atributos



- Clic en **+** y selecciona **vector asset**
- Haciendo clic en **Clip art** puedes buscar otro icono
- Fíjate cómo se define en el XML



Navegación – Vincular menú

- Para **vincular** al BottomNavigationView un **menú**:

```
app:menu="@menu/nombre del fichero del menu"
```

- Al ejecutar la aplicación deberías **ver** la navegación
- Lo único que falta es añadir el **NavController** para que funcione

Navegación – Menú funcional

- Tienes un <fragment> para mostrar los fragmentos
- Tienes un BottomNavigationView que representa el menú
- En el **onCreate** de la **Activity** hay que **vincularlos**

```
//Buscamos el BottomNavigation
val bottomNavView = findViewById<BottomNavigationView>(R.id.bottomNav)
//El contenedor de fragments (la etiqueta <fragment>)
val navHostFragment = findNavController(R.id.fragmentContainerView)
//Se vinculan
bottomNavView.setupWithNavController(navHostFragment)
```

Navegación – Barra de acción con título

- En **values/themes.xml** puedes eliminar el **NoActionBar**

```
parent="Theme.Material3.DayNight";
```

- Habrás **activado la barra de acción**. Si ejecutas, verás que el título nunca cambia.
 - Hay que vincular la navegación y la barra de acción en el onCreate

¿De dónde salen esas ids?

```
val appBarConfig = AppBarConfiguration(  
    listOf(R.id.saludoFragment, R.id.historicoFragment, R.id.detallesFragment)  
)  
setupActionBarWithNavController(navHostFragment, appBarConfig)
```

Fragmentos – Recibir/Enviar parámetros

- Vamos a hacer que desde el Fragment Histórico se envíe un texto al Fragment Detalles
- Añade en el **layout de Histórico** un **botón** y un **editText**
- Redefine el **onStart()** del fragment e **inicializa** los anteriores
 - **requireview().findViewById(...)**

Fragmentos – Habilitar SafeArgs

- En el **build.gradle.kts** más externo / top level (**menos texto**):

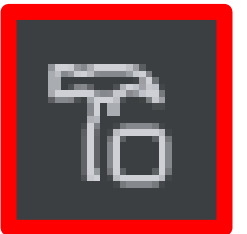
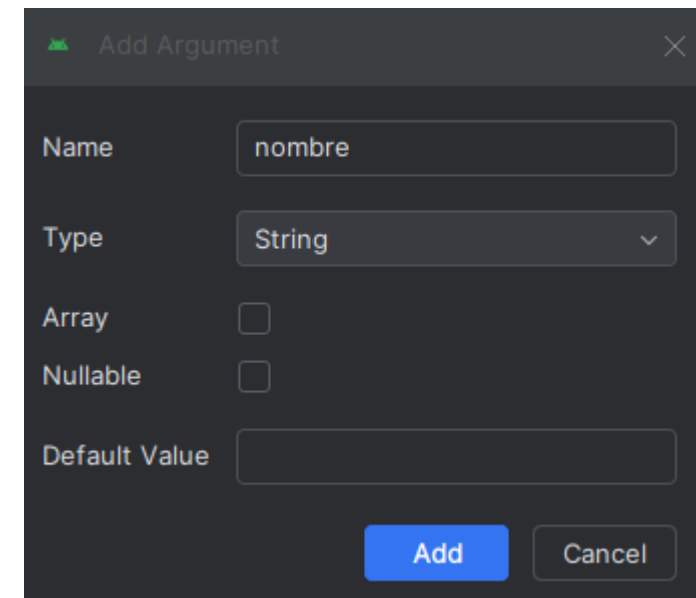
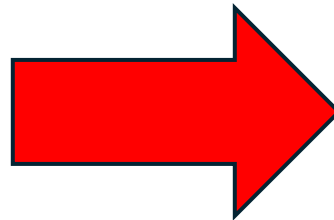
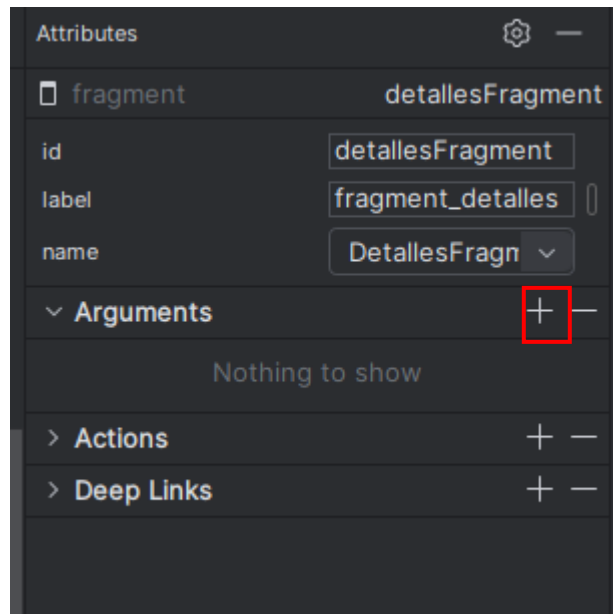
```
buildscript{  
    dependencies {  
        val nav_version = "2.8.2"  
        classpath("androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version")  
    }  
}
```

androidx.navigation:navigation-safe-args-gradle-plugin:\$nav_version

- Y en el otro **build.gradle.kts** (nivel módulo) añade en **plugins**
id("androidx.navigation.safeargs")
- Genera clases automáticamente para el paso seguro de datos

Fragmentos – Recibir parámetros

- Abre el **fichero XML de navegación** con la vista de diseño
 - Selecciona el fragmento de destino: **detallesFragment**
- Despliega **Arguments** y añade un **string**, llámalo **nombre**



Fragmentos – Histórico hacia Detalles

- Añade el listener al botón con el código equivalente

```
val destino = HistoricoFragmentDirections  
    .actionHistoricoFragmentToDetallesFragment2( nombre: "PRUEBA")  
findNavController().navigate(destino)
```

- HistoricoFragmentDirections → Clase automática gracias a SafeArg
 - El nombre coincide con el del fragment
- El nombre del método coincide con el nombre de la **id** del action
 - Es decir, con la id de la **flecha de navegación del XML**

Fragmentos – Recibir datos

- En el fragmento de destino (DetallesFragment) se ha generado :

```
private val argumentos : DetallesFragmentArgs by navArgs()
```

- Fíjate que DetallesFragmentArgs es una clase automática
- En la variable están todos los argumentos que has indicado en el XML
 - argumentos.nombre

Fragmentos – Botón “volver”

- DetallesFragment no está en el menú inferior
- Puedes implementar un botón “Atrás” con el siguiente código en el listener

```
findNavController().popBackStack()
```

- Recuerda que **antes finalizábamos la otra activity**
 - Pero aquí **ya no hay “otra activity”**