



Programación multimedia y dispositivos móviles

UT-5.3 Navigation Drawer

[https://developer.android.com/guide/
https://danielme.com/2018/12/19/disenio-android-menu-lateral-con-navigation-drawer/](https://developer.android.com/guide/https://danielme.com/2018/12/19/disenio-android-menu-lateral-con-navigation-drawer/)



Menú lateral - Navigation drawer

Navigation Drawer es uno de los componentes visuales definidos por Material Design.

Lo conocemos de sobra: el popular menú lateral deslizable desde la izquierda. Es, junto con los Tabs, el principal elemento de navegación de las aplicaciones móviles. Dependiendo del número de elementos de navegación te puede interesar uno u otro (no se recomienda usar tabs con más de 5 elementos ya que es posible que no se vean bien).


El menú Drawer suele encontrarse en la pantalla principal de la aplicación y contar con un botón «hamburguesa» (tres líneas horizontales) para desplegarlo. También es común que se permita su apertura deslizando el contenido de la pantalla desde su borde izquierdo.



Menú lateral - Navigation drawer

Vamos a crear un proyecto nuevo con el que practicar este componente. Para darle “empaque” al menú vamos a crear un menú con 5 opciones, una de ellas será la de salir y las otras 4 serán fragmentos que crearemos (todos bastante sencillos).

Vamos a dotar a nuestro menú de iconos y de una cabecera donde pondremos un logo del instituto y nuestros datos.



Navigation drawer - Paso 1 - Recursos

El primer paso, una vez creado el proyecto, va a ser importar todos como recursos (Resource manager) todos las imágenes e iconos que tenemos en la carpeta que os he dejado en el aula virtual. Ten a mano el fichero de **string.xml** que te paso ya que utilizaremos estas variables.

Trabajaremos con una única actividad. Así que después de incluir los recursos en nuestro proyecto vamos a “dibujar” el layout de nuestra actividad principal.

Usaremos **DrawerLayout** como el layout principal. Es un contenedor (ViewGroup) proporcionado por AndroidX para albergar los elementos relacionados con el menú lateral. Dentro de él colocaremos dos componentes de Material Components:

La barra superior de la pantalla, la ActionBar de Android implementada por el componente **MaterialToolbar**. Está contenida en un **LinearLayout** en el que más adelante incrustaremos los fragmentos.

Navigation drawer - Paso 1 - Recursos

Borra a mano el ConstraintLayout (desde el punto), escribe una “d” minúscula y te debe ofrecer el drawerLayout

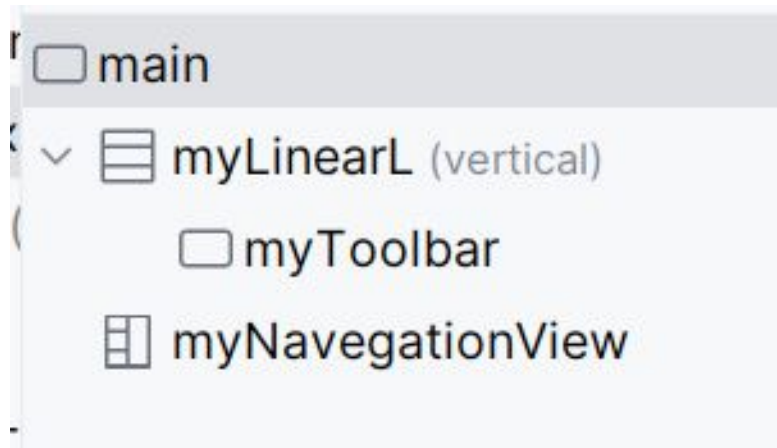
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.drawerlayout.widget.DrawerLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
</androidx.drawerlayout.widget.DrawerLayout>
```

Navigation drawer - Paso 2 - Layout principal

Incluyo dentro del DrawerLayout:

- Un LinearLayout vertical:
 - altura y anchura match_parent
- Un contenedor MaterialToolbar (dentro del anterior):
 - altura wrap_content y anchura match_parent
- Un contenedor NavigationView:
 - anchura wrap_content y altura match_parent
 - Importante: configurar el **Layout_gravity** a start.

Le doy nombre a todo





Navigation drawer - Paso 3 - Integración de Toolbar

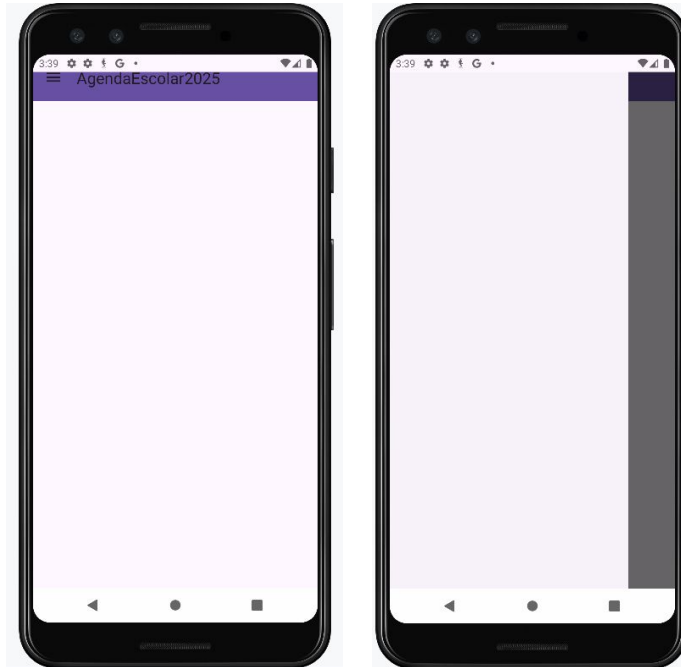
En la actividad, además de la Toolbar, configuramos el DrawerLayout con la clase **ActionBarDrawerToggle**. Su nombre da una pista de su cometido: integrar el menú del Drawer con la ActionBar.

La integración consiste en mostrar el icono "hamburguesa" en la ActionBar de tal modo que su pulsación deslice el menú para hacerlo visible. Si no configuramos este comportamiento, el menú solo se abrirá deslizando el lado izquierdo de la pantalla hacia la derecha.

El código a escribir resulta mecánico, una especie de plantilla reutilizable entre aplicaciones.

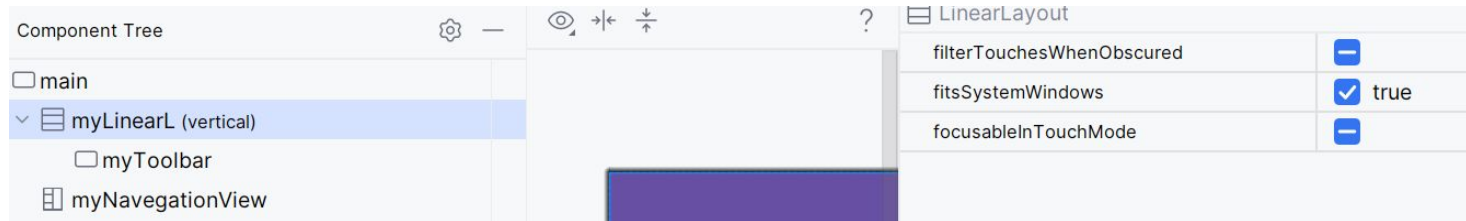
OJO, antes de que se me olvide. Como vamos a poner nuestra propia Toolbar, hay que ir a los temas y coger temas que NO tengan ActionBar.

Navigation drawer - Paso 3 - Integración de Toolbar



Navigation drawer - Paso 3 - Integración de Toolbar

Vamos a mejorar la integración de nuestra app con la barra de arriba que está como “ocupando” espacio de nuestra app. Ponemos a true la siguiente propiedad de la Toolbar.





Navigation drawer - Paso 4 - Botón atrás

Por lo común, otro comportamiento deseable es el cierre del menú con la pulsación del botón Atrás o back de Android.

Capturamos el evento sobrescribiendo el método `onBackPressed`. Su contenido es sencillo. Necesitamos para saber si el menú está abierto (`isDrawerOpen`) y, en ese caso, cerrarlo (`closeDrawer`).

Cuando esté cerrado, simplemente cerraremos la aplicación.



Navigation drawer - Paso 4 - Botón atrás

```
onBackPressedDispatcher.addCallback {  
    if (main.isDrawerOpen(GravityCompat.START)) {  
        main.closeDrawer(GravityCompat.START);  
    } else {  
        finish();  
    }  
}
```



Navigation drawer - Paso 5 - Temas y estilos

Vamos a poner “en blanco” la hamburguesa que despliega el menú. No es tan simple como cambiar el color, sino que hay que definir un estilo (nuevo archivo de styles.xml) y aplicarlo en el tema.

El estilo

```
<style name="AppTheme.DrawerArrowStyle" parent="Widget.AppCompat.DrawerArrowToggle" >
    <item name="color">@android:color/white</item>
</style>
```

Y en el tema

```
<item name="drawerArrowStyle">@style/AppTheme.DrawerArrowStyle</item>
```



Navigation drawer - Paso 5 - Temas y estilos

Ahora en el tema vamos a hacer las siguientes personalizaciones:

1.- Heredar de un tema de MaterialDesign3

2.- Poner en blanco la hamburguesa

2.- Poner en blanco las letras de la toolbar

```
<item name="drawerArrowStyle">@style/AppTheme.DrawerArrowStyle</ item>
```

```
<item name="android:statusBarColor">@android:color/transparent</ item>
```

```
<item  
name="toolbarStyle">@style/Widget.MaterialComponents.Toolbar.PrimarySurface</ item>
```



Navigation drawer - Paso 6 - Menú

Ya tenemos el menú. Literalmente, un lienzo en blanco en el que añadiremos los distintos elementos de navegación.

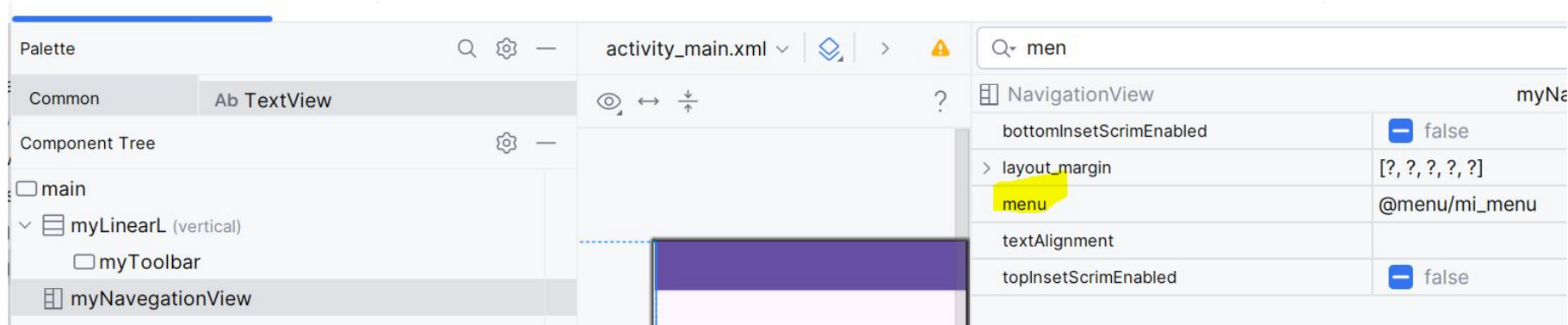
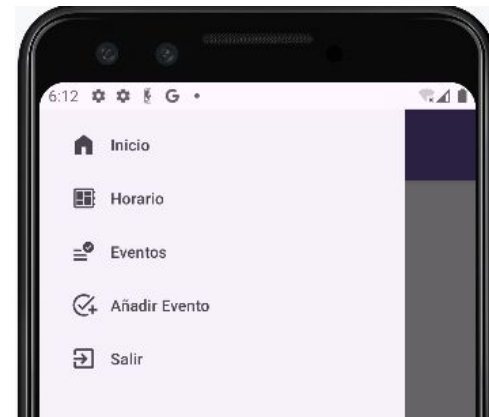
Dar contenido al menú se hace de la misma manera que cuando vimos los menús para la AppBar: en un XML, ubicado en la carpeta /res/menu, con elementos de tipo item que se corresponden con cada entrada del menú. Creamos la carpeta /res/menu. Dentro podéis construir vuestro menú, pero para agilizar la clase tenéis el mío en la carpeta de recursos.

OJO: hay que incluirla propiedad sombreada si hacéis vuestro propio menú

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:tools="http://schemas.android.com/tools"
      tools:showIn="navegacion_view">
```

Navigation drawer - Paso 6 - Menú

Esté menú hay que añadirlo al objeto de navegación, dentro de la propiedad de menú:





Navigation drawer - Paso 6 - Cabecera

En general, los menús laterales disponen de una cabecera o header que muestra información sobre la aplicación y el usuario, si lo hubiera. Este header es un nuevo layout, así que contendrá lo que queramos. En el componente `NavigationView` posteriormente lo podremos insertar.

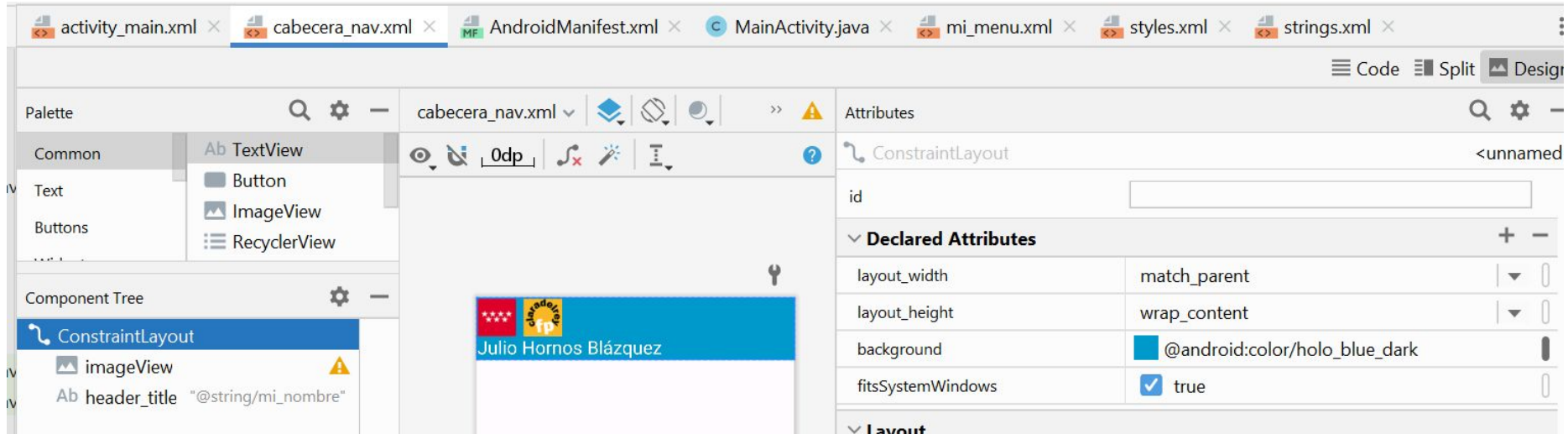
En la cabecera voy a poner un `ImageView` con el logo del centro y un `TextView` que recoge como texto una variable con mi nombre (vosotros poned el vuestro).

He añadido un color al background del `Constraint` que además sus dimensiones son `wrap_content` en altura y anchura.

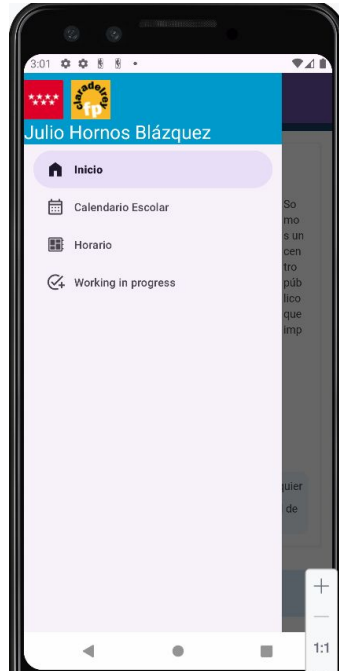
En la propiedad header del `NavigationView`, incluimos el nuevo layout


Vuelvo a tener que marcar la propiedad `"fitsSystemWindows"` del `constraintlayout` de la cabecera si quiero que aparezca.

Navigation drawer - Paso 6 - Cabecera



Navigation drawer - Paso 6 - Cabecera





Navigation drawer - Paso 6 - Implementamos el menú

El siguiente paso es que estas opciones de menú hagan algo cuando las pulsemos.... para conseguir esto vamos a hacer 4 fragmentos (la última acción es simplemente salir de la aplicación).

- Fragmento 1: Vamos a “encastrar” la web del instituto con un nuevo widget WebView
- Fragmento 1: Vamos a “encastrar” la web del instituto con un nuevo widget WebView pero en una sección donde podemos ver el calendario escolar
- Los fragmentos 3 y 4 los vamos a dejar por ahora “under construction” hasta que vemos cómo implementar persistencia

Navigation drawer - Paso 6 - Fragmento 1 y 2

Como siempre creo el fragmento desde el fragmento en blanco (yo lo he llamado WebFragment, me he creado un paquete de “fragments” para meter ahí los fragmentos porque este proyecto acabará teniendo bastantes clases.).

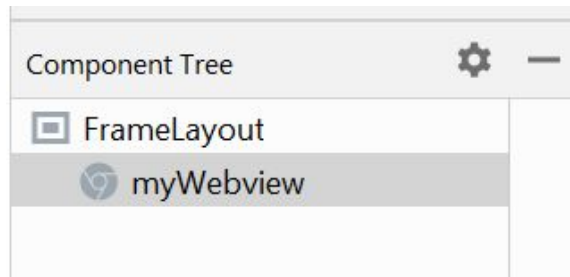
En el Layout añado un webview (match_parent / match_parent) y le doy nombre,.

En la clase kotlin simplifico el código que me se genera para poder pasar 1 argumento (que será la URL) y simplemente tengo que castear el webview y cargar la url.

OJO: hay que darle a la app permisos para acceder a Internet .

Dentro del manifiesto, cuando acaba el tag de aplicación ponemos

```
<uses-permission android:name="android.permission.INTERNET" />
```





Navigation drawer - Paso 6 - Fragmento 1 y 2

Limpio lo relativo al parámetro 2 y al uno lo renombre como url

```
private const val ARG_PARAM1 = "param1"
```

```
class WebFragment : Fragment() {
```

```
    private var url: String? = null
```

```
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        arguments?.let { it: Bundle  
            url = it.getString(ARG_PARAM1)  
        }  
    }  
}
```



Navigation drawer - Paso 6 - Fragmentos 1 y 2

```
override fun onCreateView(  
    inflater: LayoutInflater, container: ViewGroup?,  
    savedInstanceState: Bundle?  
): View? {  
    // Inflate the layout for this fragment  
    val myView = inflater.inflate(R.layout.fragment_web, container, attachToRoot: false)  
    var myWebView = myView.findViewById<WebView>(R.id.myWebView)  
    myWebView.loadUrl(url!!)  
    return myView;  
}
```

Navigation drawer - Paso 6 - Fragmentos 1 y 2

```
@JvmStatic
fun newInstance(param1: String) =
    WebFragment().apply { this: WebFragment
        arguments = Bundle().apply { this: Bundle
            putString(ARG_PARAM1, param1)
            //putString(ARG_PARAM2, param2)
        }
    }
```

Navigation drawer - Paso 6 - Fragmentos 3 y 4



Navigation drawer - Paso 7 - Implementamos el menú

La captura y tratamiento de los elementos pulsados se realiza en un oyente o listener implementado con la interfaz **NavigationView.OnNavigationItemSelectedListener**. Su único método tiene como entrada el elemento que fue pulsado.

```
class MainActivity : AppCompatActivity(), NavigationView.OnNavigationItemSelectedListener {
```

```
    private lateinit var
```

```
    override fun onCreate
```

```
        super.onCreate(s
```

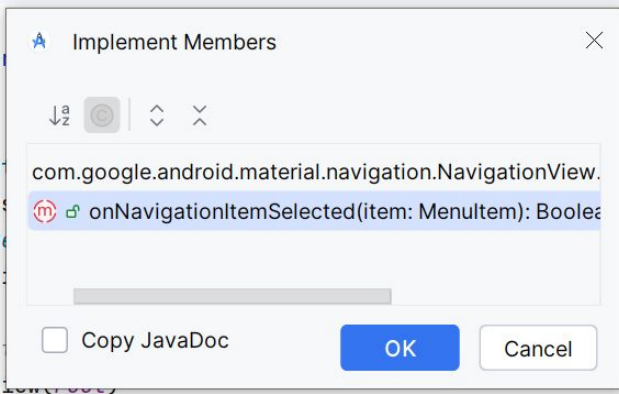
```
        enableEdgeToEdge
```

```
        binding = Activ
```

```
        with(binding){
```

```
            //setContent
```

```
            setContentView
```





Navigation drawer - Paso 7 - Implementamos el menú

Vamos a implementar una función que reciba el item seleccionado y nos “pegue” en el LinearLayout del activity_main el fragment que toque.

El código es básicamente un Switch sobre la opción elegida y en cada caso elegirá el fragmento que toque.

Esta función tendrá unos pasos comunes en lo que respecta a mostrar un fragmento (1, 2 y 4) y sólo el paso 3 dependerá del item seleccionado.

Navigation drawer - Paso 7 - Implementamos el menú

```
private fun showFragment(menuItem: MenuItem) {  
    var titulo: String? = null  
    //Paso 1: Obtener la instancia del administrador de fragmentos  
    val my_fragmentManager: FragmentManager = supportFragmentManager  
    //Paso 2: Crear una nueva transacción  
    val my_transaction: FragmentTransaction = my_fragmentManager.beginTransaction()  
    //Paso 3: Crear un nuevo fragmento y añadirlo  
    if (menuItem.itemId == R.id.id_inicio) {  
        titulo = "Home"  
        var miWebFragment = WebFragment.newInstance( param1: "https://iesclaradelrey.es/portal/index.php/es/")  
        my_transaction.replace(R.id.myLinearL, miWebFragment)  
    } else if (menuItem.itemId == R.id.id_calendario) {  
        titulo = "Calendario Escolar"  
        var miWebFragment = WebFragment.newInstance(  
            param1: "https://iesclaradelrey.es/portal/index.php/es/organizacion/jefatura-de-estudios/calendario-escolar")  
        my_transaction.replace(R.id.myLinearL, miWebFragment)  
    }  
    //Paso 4: Confirmar el cambio  
    my_transaction.commit()  
    title = titulo  
    return  
}
```



Navigation drawer - Paso 7 - Implementamos el menú

Implementamos el método de la interfaz llamando a la función que mostrará el fragmento y a la salida cerrando el menú y retornando true.

```
override fun onNavigationItemSelected(item: MenuItem): Boolean {  
    showFragment(item);  
    binding.main.closeDrawer(GravityCompat.START);  
    return true;  
}
```

Es imprescindible asociar el listener al NavigationView en el onCreate:

```
myNavegationView.setNavigationItemSelectedListener( this@MainActivity );
```

Navigation drawer - Paso 7 - Implementamos el menú





Navigation drawer - Paso 8 - Inicio

Podemos añadir en el onCreate lógica para que la actividad tenga una opción elegida al arrancar. Elegimos que al arrancar se posicionen en la primera opción de menú, la home del instituto.

```
val menuItem: MenuItem = myNavegationView.getMenu().getItem(0)
onNavigationItemSelectedListener(menuItem)
menuItem.setChecked(true)
```



Navigation drawer - Paso 8- Listener Cabecera

Acabamos poniendo un listener en la cabecera de forma que podamos “hacer algo” cuando alguien la pulsa. Dentro del onCreate:

```
//Escuchador de la cabecera
val header: View = myNavigationView.getHeaderView( index: 0)
val myNombre = header.findViewById<TextView>(R.id.header_title)
myNombre.setOnClickListener {
    Toast.makeText(
        applicationContext,
        text: "Si has llegado hasta aquí, eres un monstruo: " + myNombre.text,
        Toast.LENGTH_SHORT
    ).show()
}
```



Navigation drawer - XML / Estilos

En la página web que se incluye en la portada de la presentación explica cómo hacer que la navegación entre fragmentos se configure en un XML.

Y como siempre que queramos personalizar los estilos de un control, tenemos la página oficial de Material Design 3 que nos puede dar ideas:

<https://m3.material.io/components/navigation-drawer/overview>