

Tema 8. Menús y preferencias de usuario

8.1 Definición de un menú XML

8.2 Menú de opciones y barra de app

8.2.1 SearchView

8.3 Bottom navigation menu

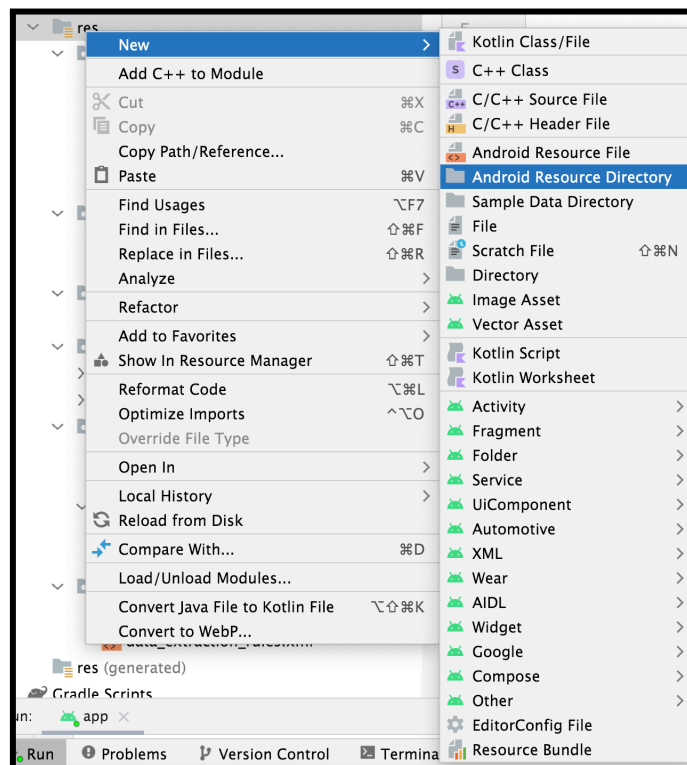
8.4 Navigation drawer menu

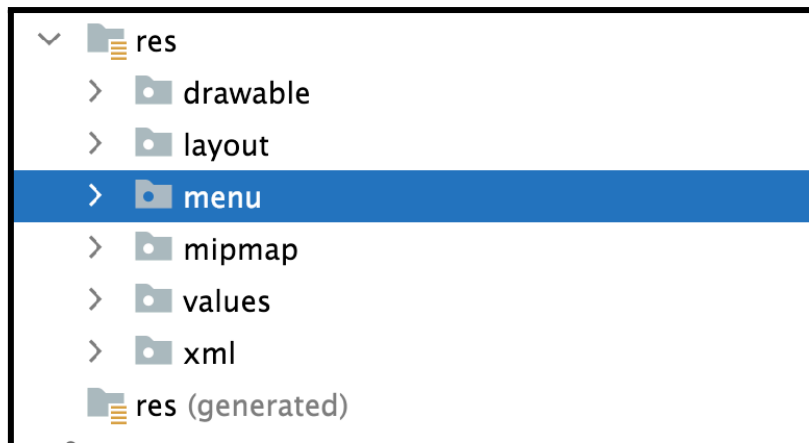
8.5 SharedPreferences (preferencias de usuario)

8.1 Definición de un menú XML

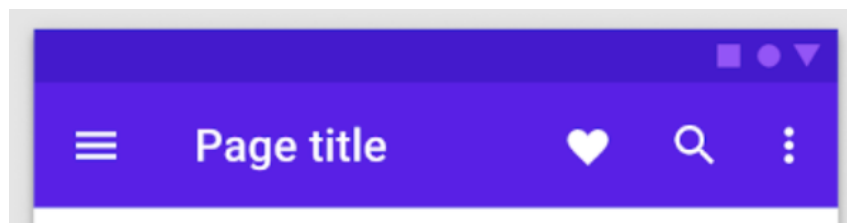
Los menús son un componente común de la interfaz de usuario en muchos tipos de aplicaciones. Para proporcionar una experiencia de usuario conocida y uniforme, debes usar las API de **Menu** a fin de presentar al usuario acciones y otras opciones en las actividades.

Los menús se incluyen en la carpeta de recursos de Android Studio **/res/menu**. Esta carpeta no está creada de inicio, por lo que deberás añadirla.



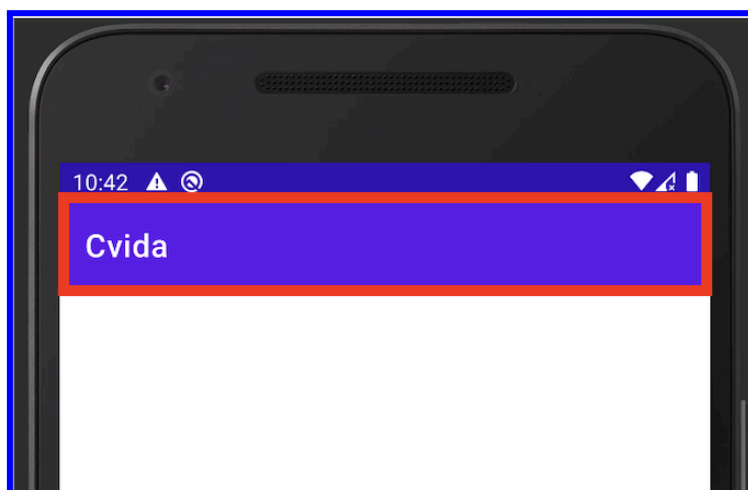


8.2 Menú de opciones y barra de app



La primera parte de la configuración de la **ToolBar** consiste en anular la **ActionBar**, y posteriormente insertar la **ToolBar** como un objeto del *Activity*. Puedes ver los pasos en la [documentación de material design](#).

- **Paso 1. Anular la ActionBar**



La **ActionBar** fue introducida en versiones más antiguas de Android (antes de la versión 3.0, Honeycomb). Es parte integral de la actividad y se obtiene mediante **actionBar** en la actividad.

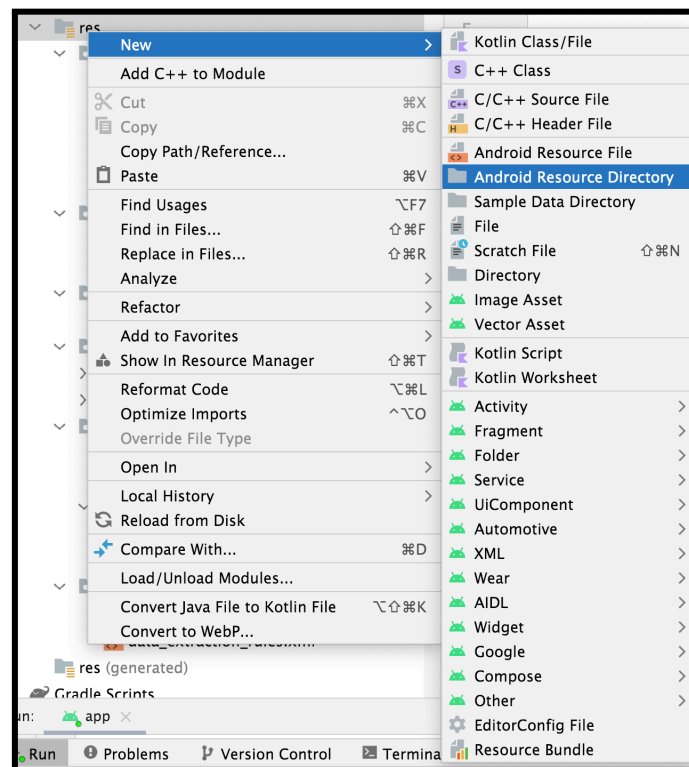
Tiene funcionalidades predeterminadas, como la capacidad de mostrar el título de la actividad, un icono de aplicación y acciones contextuales.

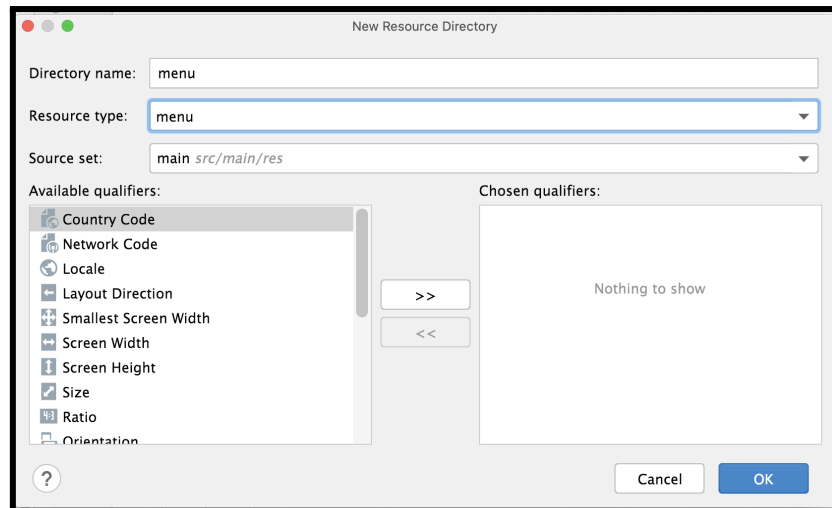
La personalización de **ActionBar** es posible, pero puede ser limitada en comparación con Toolbar.

Para anular esta barra, debemos cambiar el tema en el fichero **AndroidManifest.xml**, por un tema con el sufijo **NoActionBar**.

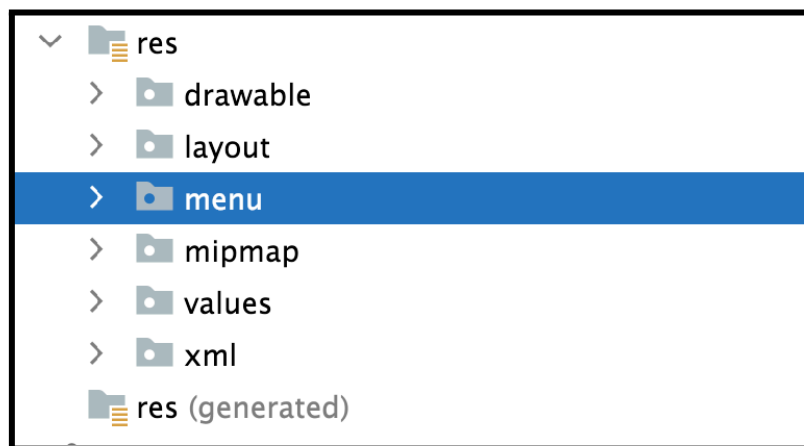
```
android:theme="@style/Theme.MaterialComponents.DayNight.NoActionBar"
```

- **Paso 2. Añadimos la carpeta de recursos MENU**

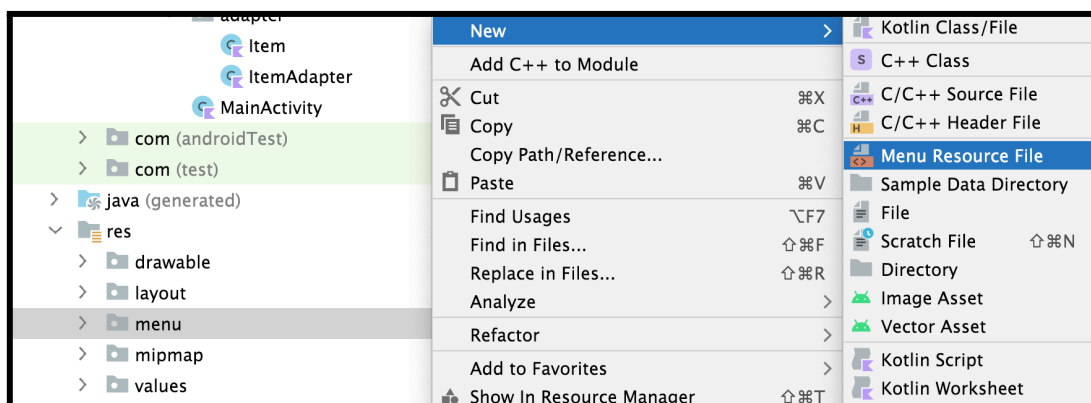




Nuestra carpeta de recursos ahora incluye la subcarpeta */menu*.



- **Paso 3. Añadimos un fichero de menú a la carpeta MENU**



- **Paso 4. Insertamos el siguiente código en el fichero XML “top_app_bar.xml”**

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

    <item
        android:id="@+id/action_overflow"
        android:icon="@drawable/ic_more"
        android:title="more"
        app:showAsAction="always">

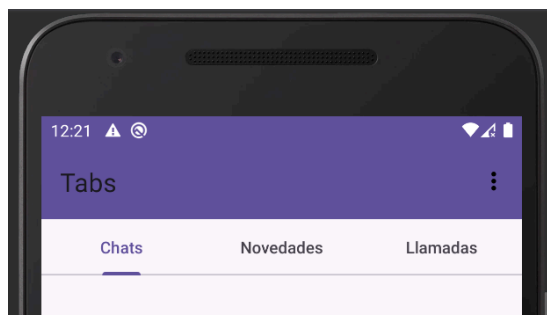
        <menu>
            <!-- Opciones dentro del menú de desbordamiento -->
            <item
                android:id="@+id/menu_option1"
                android:title="Opción A" />

            <item
                android:id="@+id/menu_option2"
                android:title="Opción B" />
            </menu>
        </item>
    </menu>
```

- **Paso 5. Y en el fichero *activity_main.xml* se añade:**

```
<androidx.appcompat.widget.Toolbar
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    android:elevation="4dp"
    android:theme="@attr/actionBarTheme"/>
```

Ahora faltaría programar el comportamiento de la **ToolBar**.



- **Paso 6. Programar el comportamiento de la *ToolBar* desde el fichero *MainActivity.kt*:**

En primer lugar, debemos añadir la siguiente línea en el método ***onCreate()***:

```
class MainActivity : AppCompatActivity()
{
    override fun onCreate(savedInstanceState: Bundle?)
    {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //Establecemos nuestra ToolBar en lugar de la ActionBar
        setSupportActionBar(findViewById(R.id.toolbar))
    }
}
```

Además, se deben redefinir los métodos ***onCreateOptionsMenu()*** y ***onOptionsItemSelected()***.

El primer método, infla la vista de la ***ToolBar***, tal como se haría con un ***fragment***.

```
//Infla la ToolBar
override fun onCreateOptionsMenu(menu: Menu?): Boolean
{
    menuInflater.inflate(R.menu.top_app_bar, menu)
    return true
}
```

El segundo método, gestiona el evento ***onClick()*** sobre cada uno de los elementos de la ***ToolBar***.

```
//Gestionar los eventos onClick sobre los elementos de la ToolBar
override fun onOptionsItemSelected(item: MenuItem): Boolean
{
    when (item.itemId)
    {
        R.id.action_overflow->{
            // Acción cuando se selecciona el primer elemento del menú
            return true
        }

        R.id.menu_option1 ->{
            Toast.makeText(context: this, text: "Has pulsado la opción A",
                Toast.LENGTH_LONG).show()

            return true
        }

        R.id.menu_option2 ->{
            Toast.makeText(context: this, text: "Has pulsado la opción B",
                Toast.LENGTH_LONG).show()

            return true
        }

        // Agrega más casos según tus necesidades
        else -> return super.onOptionsItemSelected(item)
    }
}
```

8.2.1 SearchView

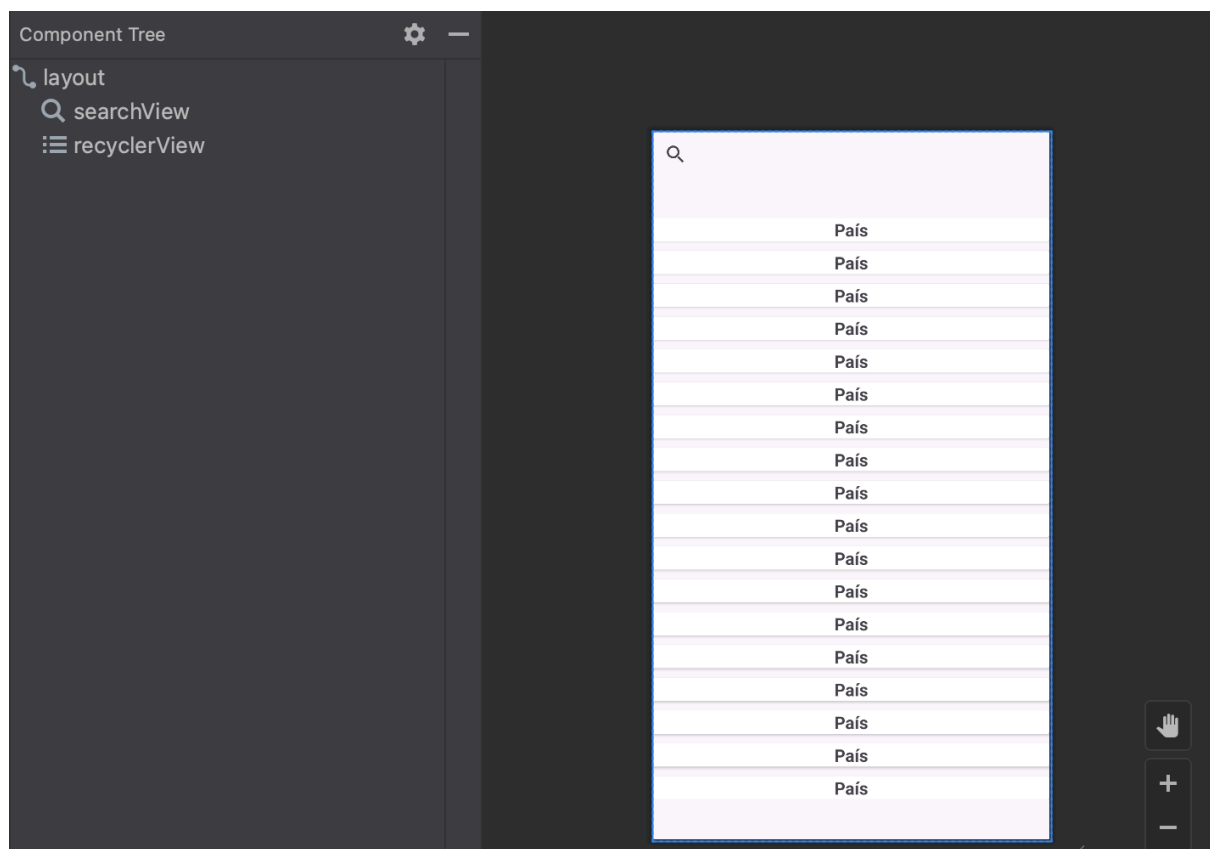
Es un widget que proporciona una interfaz de usuario para que los usuarios ingresen y realicen búsquedas en una aplicación. Este widget generalmente se coloca en la barra de acción (ActionBar) o en la barra de herramientas (**Toolbar**) de una actividad.

Al usar **SearchView**, los usuarios pueden ingresar consultas de búsqueda y ver los resultados de la búsqueda directamente en la interfaz de usuario de la aplicación.

El widget **SearchView** es bastante flexible y ofrece varias opciones de personalización para adaptarse al diseño y estilo de la aplicación.

Vamos a ver un ejemplo a continuación.

En primer lugar, se ha diseñado la vista del activity, donde se tiene un **ConstraintLayout**, que contiene una **SearchView** y un **RecyclerView**, sobre el que haremos el filtrado de elementos.



Cada uno de los ítems del **RecyclerView**, es simplemente un elemento de tipo **TextView**, para hacerlo lo más sencillo posible.

El código necesario para resolver el funcionamiento de la **SearchView** es:

```
class MainActivity : AppCompatActivity() {

    private lateinit var searchView: SearchView           Rescatamos los elementos de la vista
    private lateinit var recyclerView: RecyclerView

    private val allCountries = listOf(
        Country( name: "Argentina"),
        Country( name: "Brazil"),
        Country( name: "Canada"),
        Country( name: "Germany"),      Añadimos varios elementos al RecyclerView
        Country( name: "India"),
        Country( name: "Japan"),
        Country( name: "United States"),
        // Add more countries as needed
    )

    private var filteredCountries = allCountries.toMutableList()  Esta lista se usará para almacenar los elementos
                                                                    que cumplen los criterios de filtrado

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

        searchView = findViewById(R.id.searchView)           Rescatamos los elementos de la vista
        recyclerView = findViewById(R.id.recyclerView)

        recyclerView.layoutManager = LinearLayoutManager( context: this)
        val adapter = CountryAdapter(filteredCountries)      Configuración del RecyclerView
        recyclerView.adapter = adapter
    }
}
```

```
searchView.setOnQueryTextListener(object : SearchView.OnQueryTextListener {           Este es el Listener a
                                                                                          implementar,
                                                                                          para las búsquedas por texto

    override fun onQueryTextSubmit(query: String?): Boolean
    {
        return false
    }

    override fun onQueryTextChange(newText: String?): Boolean
    {
        filterCountries(newText)                Se captura este evento,
        return true                               cada vez que se añade o elimina un carácter en la SearchView
    }
})
}
```

Por último, nos faltaría implementar el método **filterCountries**, que selecciona aquellos países que cumplen con el patrón de texto introducido en la **SearchView**.


```

private fun filterCountries(query: String?)
{
    filteredCountries.clear()

    if (query.isNullOrEmpty()) {
        filteredCountries.addAll(allCountries)
    } else {
        val lowercaseQuery = query.lowercase()
        filteredCountries.addAll(
            allCountries.filter { country ->
                country.name.lowercase().contains(lowercaseQuery)
            }
        )
    }

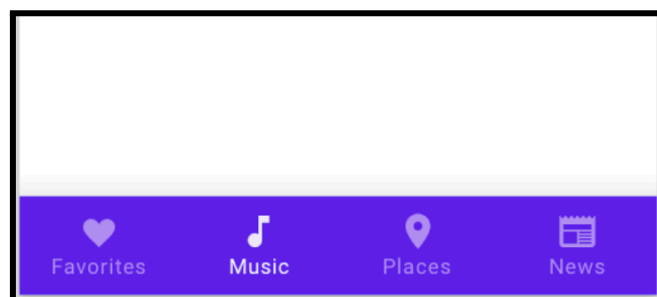
    recyclerView.adapter?.notifyDataSetChanged()
}

```

El método ***notifyDataSetChanged()*** actualiza la vista del **RecyclerView**, para que solo se muestren los elementos de la lista que cumplen con el texto filtrado.

8.3 Bottom navigation menu

Las barras de navegación inferiores permiten el movimiento entre los destinos principales de una aplicación. Cada uno de los items del menú, nos llevará a un fragment distinto.

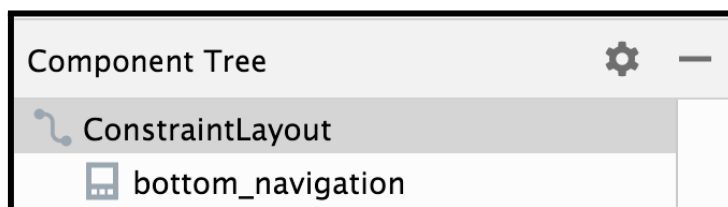


Para ver cómo implementar esta barra de navegación inferior, vamos a seguir el manual de la web <https://m2.material.io/components/bottom-navigation>

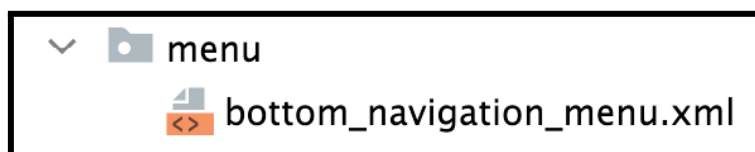
- En el fichero **activity_main.xml** se añade un objeto de tipo **BottomNavigationView**:

```
<com.google.android.material.bottomnavigation.BottomNavigationView  
    android:id="@+id/bottom_navigation"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    app:layout_constraintBottom_toBottomOf="parent"  
    app:layout_constraintEnd_toEndOf="parent"  
    app:layout_constraintStart_toStartOf="parent"  
    app:menu="@menu/bottom_navigation_menu" />
```

Quedando así nuestro árbol de componentes:



- Como consecuencia de lo añadido en el fichero **activity_main.xml**, ahora debemos crear un nuevo fichero de menú que tenga el nombre especificado anteriormente **"bottom_navigation_menu"** :



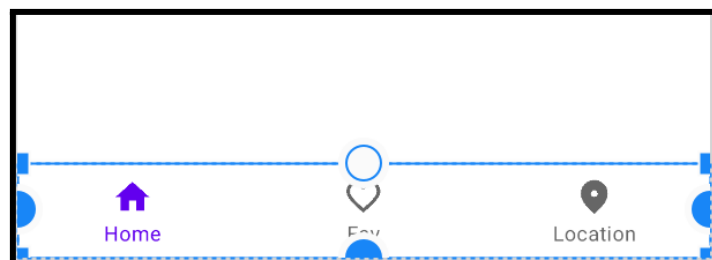
- Este fichero va a tener el siguiente código:

```

<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/page_1"
        android:enabled="true"
        android:icon="@drawable/ic_baseline_home_24"
        android:title="Home"/>
    <item
        android:id="@+id/page_2"
        android:enabled="true"
        android:icon="@drawable/ic_baseline_favorite_border_24"
        android:title="Fav"/>
    <item
        android:id="@+id/page_3"
        android:enabled="true"
        android:icon="@drawable/ic_baseline_fmd_good_24"
        android:title="@string/icono3"/>
</menu>

```

Se han insertado tres items, el primero llamado “Home”, el segundo “Fav” y el tercero “Location”. El menú se ve ahora así:



- ***Ya solo falta indicar en el activity qué fragment se debe cargar en cada caso:***

```

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

    cargarFragment(FragmentUno())

    val bnv : NavigationBarView = findViewById(R.id.bottom_navigation)
    bnv.setOnItemSelectedListener(
        { item ->
            when(item.itemId) {
                R.id.page_1 -> {
                    cargarFragment(FragmentUno())
                    true ^lambda
                }
                R.id.page_2 -> {
                    cargarFragment(FragmentDos())
                    true ^lambda
                }
                R.id.page_3 -> {
                    cargarFragment(FragmentTres())
                    true ^lambda
                }
                else -> false ^lambda
            }
        }
    )
}

```

Al arrancar la app se mostrará el primer fragment

Asignamos la NavigationBarView a una variable

Se carga el fragment asociado al icono del menú seleccionado

8.4 Navigation drawer menu