

# Tema 9. Menús Android

[Descargar estos apuntes](#)

## Índice

1. [Introducción](#)
2. [Definición de un menú como recurso XML](#)
3. [Overflow menu](#)
4. [Contextual menu](#)
5. [Popup menu](#)
6. [Exposed dropdown menu](#)
7. [Navigation Drawer](#)

# Introducción

[Información menús Material Design](#).

Hay cuatro tipos de menús en Android:

- **Overflow Menu**, menú principal que puede ser activado cuando se pulsa sobre el botón correspondiente (sea físico o software, dependiendo de su antigüedad) o sobre el icono



de la `AppBar` o `ToolBar`.

- **Contextual Menu**, son menús que aportan acciones extra para un determinado elemento de la vista. Habitualmente se activan con una pulsación larga sobre el mismo. Podemos encontrar a su vez dos tipos de menús contextuales: **Floating Context Menu** (abre un menú contextual que se superpone a la actividad) y **Contextual Action Mode** (abre una barra de acción donde se agrupan las actuaciones a realizar sobre los elementos seleccionados).
- **Popup Menu**, es un menú emergente similar al **Overflow Menu**, pero que se encuentra vinculado a un elemento de la vista, como una forma de ampliar las acciones que se pueden ejecutar.
- **Navigation Drawer** o menú lateral deslizante, aparece dentro de la guía de diseño como un componente específico [Información adicional Material Design](#). Suele encontrarse en la pantalla principal de la app y contar con un botón para desplegarlo, aunque también puede ser abierto deslizando el contenido de la pantalla desde el extremo izquierdo.

## Definición de un menú como recurso XML

### Menú

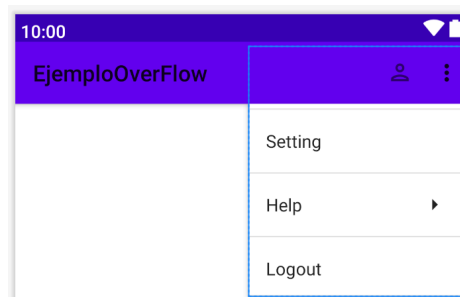
Los archivos asociados a menús deben guardarse en la carpeta `res/menu` de nuestro proyecto.

La estructura genérica de un xml asociado a un menú sería algo así:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/option_1"
        android:title="@string/option_1" />
    <item android:id="@+id/option_2"
        android:title="@string/option_2" />
    ...
    <group android:id="@+id/group1_1">
        <item android:id="@+id/group_op1"
            android:title="@string/group_op1" />
        <item android:id="@+id/group_op2"
            android:title="@string/group_op2" />
    </group>
</menu>
```

## Overflow menu

Para **Android 3.0 (nivel API 11) y versiones posteriores** el **Overflow Menu** aparece en la barra de la **App**.



✎ **android:showAsAction=ifRoom|withText** hacemos que este elemento del menú aparezca siempre fuera del **Overflow menu** y que si hay es

Para visualizar el **Overflow menu** anulamos el método **onCreateOptionsMenu** en la actividad o fragmento correspondiente (recordad que se invoca automáticamente o pulsando el menú dependiendo de la versión):

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    val inflater: MenuInflater = menuInflater
    inflater.inflate(R.menu.menu_overflow, menu)
    return true
}
```

Para controlar las opciones de menú que son pulsadas implementamos el método **onOptionsItemSelected()**.

```

override fun onOptionsItemSelected(item: MenuItem): Boolean {
    var text=""
    when (item.itemId) {
        R.id.opEdit-> text="EDIT"
        R.id.opSetting-> text="SETTING"
        R.id.opHelpApp-> text="APP"
        R.id.opHelp->return true
        R.id.opHelpAndroid-> text="ANDROID"
        R.id.opLogout-> text="LOGOUT"
        else -> super.onOptionsItemSelected(item)
    }
    Toast.makeText(applicationContext,"Pulsaste la opción de menú "+text,
    Toast.LENGTH_SHORT).show()
    return true
}

```

A diferencia de `onOptionsItemSelected()`, que solo se llama la primera vez que se construye el menú, el método `onOptionsItemSelected()` se llama cada vez que el menú se abre. Esto nos permite realizar operaciones como añadir o eliminar opciones de manera dinámica, modificar la visibilidad de los diferentes elementos o modificar su texto.

```

override fun onPrepareOptionsMenu(menu: Menu?): Boolean {
    //add item
    menu.add()
    //add menu
    menu.addSubMenu()
    return super.onPrepareOptionsMenu(menu)
}

```

## Contextual menu

La idea es que al realizar una pulsación larga sobre un elemento de una vista (un `TextView`, `Button`, un elemento de un `RecyclerView`, ...), se abra un menú con unas opciones que afectan únicamente a ese elemento.

Es necesario registrar que dicho elemento tiene asociado un `Contextual menu`, para ello tenemos el método `registerForContextMenu(vista)`.

Cuando se realiza la pulsación larga sobre un elemento que tiene asociado un menú contextual se invoca al método `onCreateContextMenu()`.

```

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)

        registerForContextMenu(binding.button)
        registerForContextMenu(binding.text)
    }

    override fun onCreateContextMenu(menu: ContextMenu?, v: View?,
                                     menuInfo: ContextMenu.ContextMenuInfo?) {
        super.onCreateContextMenu(menu, v, menuInfo)
        if (v!!.id== id.text){
            menuInflater.inflate(R.menu.menu_contextual_textview,menu)
        }
        if (v!!.id== id.button){
            menuInflater.inflate(R.menu.menu_contextual_button,menu)
        }
    }

    override fun onContextItemSelected(item: MenuItem): Boolean {
        when (item.itemId){
            id.op12->binding.text.textSize= 12F
            ...
            id.opAzul->binding.button.setBackgroundColor(BLUE)
        }
        return super.onContextItemSelected(item)
    }
}

```

## Popup menu

Un **Popup menu** muestra una lista de opciones de menú asociadas a la vista que invocó el menú. E

```

class MainActivity : AppCompatActivity() {
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)

        binding.button.setOnClickListener {
            showPopup(binding.button)
        }
    }

    private fun showPopup(view: View) {
        //Creamos el popupMenu asociando la vista
        val popup = PopupMenu(this, view)
        popup.inflate(R.menu.poppup_menu)
        //tratamos el click en cada opción
        popup.setOnMenuItemClickListener(PopupMenu.
            OnMenuItemClickListener
        { item: MenuItem? ->
            when (item!!.itemId) {
                R.id.opMail -> {
                    ...
                }
            }
            true
        })
        //mostramos el menú
        popup.show()
    }
}

```

Para visualizar los icono del menú ir a la guía de Material de este mismo enlace.

## Exposed dropdown menu

Los menús vistos hasta hora se engloban en las guías de *Android* como

**Dropdown menus**. Se diferencian de **Exposed dropdown menus** porque estos últimos muestran la última opción seleccionada del mismo. Los vimos cuando explicamos

**AutoCompleteTextView**

```

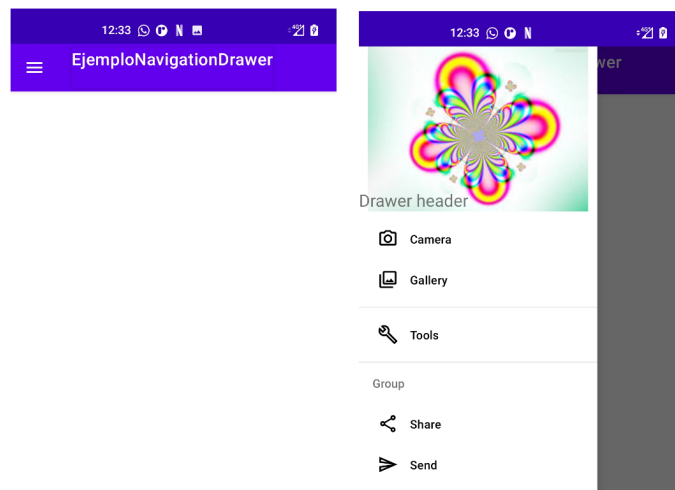
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/menu"
    style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox.ExposedDropdownMenu"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/label">

    <AutoCompleteTextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="none"

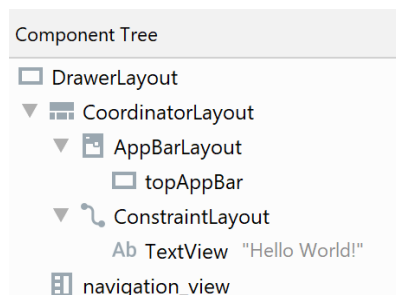
    />
</com.google.android.material.textfield.TextInputLayout>

```

## Navigation Drawer



En el diseño de la activity principal, usaremos como contenedor principal, un **DrawerLayout** el cual contendrá la Toolbar y el componente **NavigationView** que es el menú propiamente dicho (además de cualquier otro elemento **View** que queramos incorporar a la interfaz). Los componentes de la ventana principal menos el menú irán dentro de un **CoordinatorLayout** :



```

<?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"
    android:id="@+id/drawer_layout"
    tools:context=".MainActivity">

    <androidx.coordinatorlayout.widget.CoordinatorLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <com.google.android.material.appbar.AppBarLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:theme="@style/Widget.MaterialComponents.Toolbar
                .Primary"
            android:fitsSystemWindows="true">

            <com.google.android.material.appbar.MaterialToolbar
                android:id="@+id/topAppBar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                app:title="@string/app_name"
                style="@style/Widget.MaterialComponents.Toolbar.Primary"
                app:layout_collapseMode="pin"/>
            </com.google.android.material.appbar.AppBarLayout>

            <!-- Screen content -->

        </androidx.coordinatorlayout.widget.CoordinatorLayout>

        <com.google.android.material.navigation.NavigationView
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:id="@+id/navigation_view"
            android:layout_gravity="start"
            app:headerLayout="@layout/drawer_header"
            app:menu="@menu/drawer_menu"
            android:fitsSystemWindows="true"/>
    </androidx.drawerlayout.widget.DrawerLayout>

```



#### Aclaraciones:

- Si definimos este atributo **tools:openDrawer="start"** en el **DrawerLayout** permitiremos que con el gesto de deslizamiento hacia la derecha se abra el panel de navegación.



- **android:layout\_gravity="start"** establece que el menú salga de la izquierda de la pantalla.
- **app:headerLayout="@layout/drawer\_header"** establece el archivo *XML* que define la vista de la cabecera del menú, se define dentro de la carpeta **res/layout** .
- **app:menu="@menu/drawer\_menu"** establece el archivo *XML* que define la vista de las opciones del menú, se define dentro de la carpeta **res/menu** .
- **android:fitsSystemWindows="true"** establece que el **Navigation drawer** aparezca por debajo de la barra.

*XML* de la cabecera:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <ImageView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_gravity="center_vertical"
        android:scaleType="fitCenter"
        android:src="@drawable/imgfondo"/>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Drawer header"
        android:layout_gravity="bottom"
        android:textSize="20dp"/>
</FrameLayout>
```

Tendremos el *XML* con las opciones del **Navigation drawer** :

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

    <group android:checkableBehavior="single">
        <item
            android:id="@+id/nav_camera"
            android:icon="@drawable/outline_photo_camera_24"
            android:title="Camera" />
        ...
    </group>
    ...

</menu>
```

Analicemos ahora el código de nuestro *MainActivity.kt*, empecemos por la inicialización de componentes en el método `onCreate()` de la actividad:

```

class MainActivity : AppCompatActivity(),
NavigationView.OnNavigationItemSelectedListener{

    private lateinit var binding: ActivityMainBinding
    lateinit var drawer_layout:DrawerLayout
    lateinit var toggle: ActionBarDrawerToggle
    lateinit var toolbar: MaterialToolbar
    lateinit var navigationView: NavigationView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        binding = ActivityMainBinding.inflate(layoutInflater)
        val view = binding.root
        setContentView(view)

        navigationView=binding.navigationView
        navigationView.setNavigationItemSelectedListener(this)

        drawer_layout=binding.drawerLayout

        toolbar=binding.topAppBar
        setSupportActionBar(toolbar)

        toggle = ActionBarDrawerToggle(this, drawer_layout,
            toolbar, R.string.navigation_open,
            R.string.navigation_close)

        drawer_layout.addDrawerListener(toggle)

        //Set whether to include the application home
        //affordance in the action bar.
        // (and put a back mark at icon in ActionBar for "up" navigation)
        supportActionBar?.setDisplayHomeAsUpEnabled(true)

        //Enable or disable the "home" button in
        //the corner of the action bar (clickable or not)
        supportActionBar?.setHomeButtonEnabled(true)
    }
}

```

A continuación implementamos el método `onPostCreate()` y `onConfigurationChanged()` para poder gestionar los cambios en el `Navigationdrawer` o actualizar tras un cambio de orientación en la *app*. `syncState()` sincroniza el estado del indicador de la barra de navegación con el `DrawerLayout` vinculado.

```

class MainActivity : AppCompatActivity() {
    // ...
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        toggle.syncState()
    }

    override fun onConfigurationChanged(newConfig: android.content.
        res.Configuration) {
        if (newConfig != null) {
            super.onConfigurationChanged(newConfig)
        }
        toggle.onConfigurationChanged(newConfig)
    }
}

```

Finalmente debemos sobrescribir el método `onNavigationItemSelectedListener()` de la interfaz `NavigationView.OnNavigationItemSelectedListener` para gestionar las pulsaciones sobre las opciones de menú desplegadas.

```

override fun onNavigationItemSelectedListener(item: MenuItem): Boolean {
    val id = item.itemId
    var s=""
    when (id) {
        R.id.nav_camera -> s="CAMARA"
        ...
    }
    drawer_layout.closeDrawer(GravityCompat.START)
    ...
    return true
}

```