2025

Programación Multimedia y Dispositivos Móviles (PMDM)

[ANDROID: MENÚ DE LA APLICACIÓN]

Tabla de contenido

1. Introducción

Los menús son un componente común de la interfaz de usuario en muchos tipos de aplicaciones.

Vamos a mostrar cómo crear 3 tipos fundamentales de menús o presentaciones de acciones:

- Menú de opciones y barra de la app
- Menú contextual y modo de acción contextual
- Menú emergente

Menú de opciones y barra de la aplicación
El menú de opciones es la colección principal de elementos de un menú de una actividad. Es donde se ubican las acciones que tienen un impacto global en la app. Ejemplo: “Buscar”, “Redactar correo”, “Configuración”, etc.

Menú contextual y modo de acción contextual
El menú contextual es un menú flotante que aparece cuando el usuario toca/hace click en un elemento. Integra acciones que afectan al contenido o al marco contextual seleccionado.

El modo de acción contextual muestra elementos de acción que afectará el contenido seleccionado en una barra en la parte superior de la pantalla y permite que el usuario seleccione varios elementos.

Menú emergente
Un menú emergente muestra una lista vertical de elementos anclados al que invoca el menú.

Es una buena forma de proporcionar un exceso de acciones que se relacionen con un contenido específico o que brinden opciones para la segunda parte de un comando.

Las acciones de un menú emergente no afectan directamente al contenido correspondiente, para eso están las acciones contextuales. Más bien, es para acciones extendidas relacionadas con partes del contenido en la actividad.

2. Cómo definir un menú en XML

Para todos los tipos de menús, Android proporciona un formato XML estándar para definir los elementos de los menús. En lugar de crear un menú en el código de la actividad se define un menú y todos los elementos en un archivo de **recurso de menú**: `res/menu/filename.xml`.

El uso de un recurso de menú es una práctica recomendada por las siguientes razones:

- Es más fácil visualizar la estructura del menú en XML.

- Separa el contenido del menú del comportamiento de la aplicación.
- Permite crear configuraciones de menú alternativas para diferentes plataformas, tamaños de pantalla y otros parámetros de configuración.

Sintaxis:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@[+][package:]id/resource_name"
      android:title="string"
      android:titleCondensed="string"
      android:icon="@[package:]drawable/drawable_resource_name"
      android:onClick="method name"
      android:showAsAction=["ifRoom" | "never" | "withText" | "always" | "collapseActionView"]
      android:actionLayout="@[package:]layout/layout_resource_name"
      android:actionViewClass="class name"
      android:actionProviderClass="class name"
      android:alphabeticShortcut="string"
      android:alphabeticModifiers=["META" | "CTRL" | "ALT" | "SHIFT" | "SYM" | "FUNCTION"]
      android:numericShortcut="string"
      android:numericModifiers=["META" | "CTRL" | "ALT" | "SHIFT" | "SYM" | "FUNCTION"]
      android:checkable=["true" | "false"]
      android:visible=["true" | "false"]
      android:enabled=["true" | "false"]
      android:menuCategory=["container" | "system" | "secondary" | "alternative"]
      android:orderInCategory="integer" />
  <group android:id="@[+][package:]id/resource name"
       android:checkableBehavior=["none" | "all" | "single"]
       android:visible=["true" | "false"]
       android:enabled=["true" | "false"]
       android:menuCategory=["container" | "system" | "secondary" | "alternative"]
       android:orderInCategory="integer" >
    <item />
  </group>
  <item >
    <menu>
     <item />
    </menu>
  </item>
</menu>
```

Elementos:

Los elementos mínimos que debe tener el fichero XML son:

- **<menu>**

  Define un **Menu**, que es un contenedor para los elementos de menú. Debe ser el **nodo raíz del archivo** y puede contener uno o más **<item>** y **<group>**.

- **<item>**

  Crea un **MenuItem**, que representa un único elemento en un menú. Este elemento puede contener un valor anidado <menú> para crear submenús.

  El elemento <ítem> admite varios atributos utilizados para definir la apariencia y el comportamiento del elemento. Los atributos más habituales son:

- **android:id** -> un ID de recurso que es único para el elemento y que permite a la app reconocer el elemento cuando el usuario lo seleccione.
- **android:icon** -> es la referencia a un elemento de diseño para usar como el icono del elemento.
- **Android:title** -> es la referencia a un string para usarlo como título del elemento.
- **Android:showAsAction** -> es la especificación sobre cuándo y cómo aparece este elemento como un elemento de acción en la barra de la aplicación.

- **<group>**

  Es un contenedor opcional e invisible para <ítem>. Permite categorizar los elementos del menú que comparten ciertas características.

  Con un grupo se puede hacer lo siguiente:
  - Mostrar u ocultar todos los elementos, con ***setGroupVisible()***.
  - Habilitar/inhabilitar todos los elementos, con ***setGroupEnabled()***.
  - Especificar si todos los elementos se pueden marcar, con ***setGroupCheckable()***.

Ejemplo 1: un menú simple

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/new_game"
     android:icon="@drawable/ic_new_game"
     android:title="@string/new_game"
     app:showAsAction="ifRoom"/>
  <item android:id="@+id/help"
     android:icon="@drawable/ic_help"
     android:title="@string/help" />
</menu>
```

Ejemplo 2: un menú que incluye un submenú

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/menu_save"
     android:icon="@drawable/menu_save"
     android:title="@string/menu_save" />
  <!-- menu group -->
  <group android:id="@+id/group_delete">
    <item android:id="@+id/menu_archive"
       android:title="@string/menu_archive" />
    <item android:id="@+id/menu_delete"
       android:title="@string/menu_delete" />
  </group>
</menu>
```

Para usar el menú en la actividad hay que convertir el recurso XML en un objeto programable usando **MenuInflater.inflate()**.

3. Menú de opciones

El menú de opciones es donde se incluyen las acciones y otras opciones relevantes para el contexto de la actividad actual.

Creación de un menú de opciones

Para crear un menú de opciones de opciones hay que realizar las siguientes acciones:

1. Asegurarse de que el tema (***res/values/themes.xml***) de nuestra aplicación permita utilizar menús, es decir, tiene que tener **ActionBar**.
2. Crear el recurso de menú (fichero XML en res/menú).
3. Sobreescribir el método ***onCreateOptionsMenú*** para 'cargar' el menú.
4. Sobreescribir el método ***onOptionsItemSelected*** cuando el usuario selecciona una opción del menú.
    - Si se controla correctamente un elemento de menú, el menú muestra ***true***.
    - Si no se controla el elemento de menú, hay que llamar a la implementación de la superclase, que predeterminadamente devuelve ***false***.

Ejemplo: crear un menú de opciones en la actividad principal

themes.xml

```
<!-- con menú -->
<style name="Base.Theme.LayoutsConMenu" parent="Theme.Material3.DayNight">
</style>

<style name="Theme.Layouts" parent="Base.Theme.LayoutsConMenu" />
```

agenda.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/nuevo_contacto"
      android:title="Añadir conctacto"/>
   <item android:id="@+id/borrar_contacto"
      android:title="Borrar contacto" />
</menu>
```

MainActivity.kt

```
// Crear menú de opciones
override fun onCreateOptionsMenu(menu: Menu): Boolean {
      super.onCreateOptionsMenu(menu)
      val inflater: MenuInflater = menuInflater
      inflater.inflate(R.menu.agenda, menu)
      return true
}
```

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    // Handle item selection.
    return when (item.itemId) {
      R.id.nuevo_contacto -> {
        Toast.makeText(this, "Opción 'Nuevo contacto'",Toast.LENGTH_SHORT).show()
        true
      }
      R.id.borrar_contacto -> {
        Toast.makeText(this, "Opción 'Borrar contacto'",Toast.LENGTH_SHORT).show()
        true
      }
      else -> super.onOptionsItemSelected(item)
    }
  }
```

4. Menú contextual

Un menú contextual ofrece acciones que afectan a un elemento o contexto específico. Se puede proporcionar un menú contextual para cualquier vista, pero a menudo se utilizan para elementos en un **RecyclerView** u otras colecciones de vistas en las que el usuario puede realizar acciones directas sobre cada elemento.

Existen dos maneras de proporcionar acciones contextuales:

1. En un **menú contextual flotante**: el menú aparece como una lista flotante de elementos de menú, similar a un diálogo, cuando el usuario realiza un toque y retiene una vista (click prolongado). Los usuarios pueden realizar una acción contextual en un elemento por vez.
2. En **modo de acción contextual**: es un sistema de implementación de ***ActionMode*** que muestre una barra de acciones contextuales, o CAB, en la parte superior de la pantalla con elementos de acción que afectan a los elementos seleccionados. Cuando este modo está activo, los usuarios pueden realizar una acción en varios elementos a la vez, siempre y cuando la aplicación sea compatible.

Nosotros vamos a ver la más habitual que es la primera de las opciones, el menú contextual flotante.

Creación de un menú contextual flotante

Para crear un menú contextual flotante hay que realizar las siguientes acciones:

1. Crear el recurso de menú (fichero XML en res/menú).
2. Registrar la vista a la que asociar el menú contextual con el método **registerForContextMenu(View)**.
3. Cuando la vista asociada recibe del usuario un click prolongado se llama al método **onCreateContextMenu**, por tanto hay que sobreescribirlo. Este método debe cargar el menú que se desee mostrar.

4. Cuando el usuario selecciona uno de los elementos del menú se llama al método **onContextItemSelected**, por tanto, hay que sobreescribirlo. Al igual que en el caso de creación del menú de opciones:
    a. Si se controla correctamente un elemento de menú, el menú muestra ***true***.
    b. Si no se controla el elemento de menú, hay que llamar a la implementación de la superclase, que predeterminadamente devuelve ***false***.

Ejemplo: crear un menú contextual sobre una imagen

menu_imagen.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/cambiar_imagen"
      android:title="@string/cambiar_imagen"/>
   <item android:id="@+id/borrar_imagen"
      android:title="@string/borrar_imagen" />
</menu>
```

MainActivity.kt

```
// Menú contextual flotante
val imagen = findViewById<ImageView>(R.id.imagen_layouts)
registerForContextMenu(imagen);


// --- Inicio Menú Contextual Flotante
override fun onCreateContextMenu(menu: ContextMenu, v: View, menuInfo:
ContextMenu.ContextMenuInfo?) {
   super.onCreateContextMenu(menu, v, menuInfo)
   val inflater: MenuInflater = menuInflater
   inflater.inflate(R.menu.menu_imagen, menu)
}

override fun onContextItemSelected(item: MenuItem): Boolean {
   return when (item.itemId) {
      R.id.cambiar_imagen -> {
         Toast.makeText(this, "Menú 'Cambiar imagen'", Toast.LENGTH_SHORT).show()
         true
      }
      R.id.borrar_imagen -> {
         Toast.makeText(this, "Menú 'Borrar imagen'", Toast.LENGTH_SHORT).show()
         true
      }
      else -> super.onContextItemSelected(item)
   }
}
//------------ FIN ---------------
```

5. Menú emergente

Un **PopupMenu** es un menú modal anclado a una vista. Es útil para:

- Proporcionar un menú de estilo ampliado para las acciones que se relacionan con contenido específico, como por ejemplo, los encabezados de correo electrónico de Gmail.
- Proporcionar la segunda parte de una lista de comandos, como un botón marcado **Add**, que genera un menú emergente con diferentes opciones de estado.
- Ofrecer un menú similar a un **Spinner**, que no retiene una selección persistente.

Creación de un menú emergente

Para crear un menú emergente hay que realizar las siguientes acciones:

1. Crear el recurso de menú (fichero XML en res/menú).
2. Al igual que el menú contextual flotante, este menú se lanza cuando se hace click sobre una vista, por lo tanto habrá que crear el código que genera/muestra el menú emergente dentro del método **setOnClickListener** de la vista.
3. Crear una instancia de **PopupMenu** con su constructor, que toma la app actual como c*ontext* y la vista a la que se ancla el menú.
4. Usar **MenuInflater** para cargar el menú.
5. Mostrar el menú llamando a **PopupMenu.show()**.
6. Cuando el usuario selecciona uno de los elementos del menú se llama al método **setOnMenuItemClickListener**:
    a. Si se controla correctamente un elemento de menú, el menú muestra ***true***.
    b. Si no se controla el elemento de menú, el menú muestra false.
7.

Ejemplo: se añade un menú emergente en el botón ‘Añadir contacto’

boton_new_contact.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
   <item android:id="@+id/nuevo"
      android:title="@string/aniadir"/>
   <item android:id="@+id/editar"
      android:title="@string/edit"/>
</menu>
```

MainActivity.kt

```
val boton = findViewById<Button>(R.id.boton_crear_contacto)
boton.setOnClickListener {
        // --- Inicio Menú Emergente en un botón
        val popup = PopupMenu(this, boton)
        val inflater: MenuInflater = popup.menuInflater
        inflater.inflate(R.menu.boton_new_contact, popup.menu)
```

```
        popup.setOnMenuItemClickListener(
            fun (menuItem: MenuItem): Boolean {
                Toast.makeText(ctx,"Opción '" + menuItem.getTitle() + "'",
                Toast.LENGTH_SHORT).show()

                return when (menuItem.itemId) {
                        R.id.nuevo -> {
                                val intent = Intent(this, NuevoActivity::class.java)
                                startActivity(intent)
                                true

                        }
                        R.id.editar -> {
                                val intent = Intent(this, EditarActivity::class.java)
                                startActivity(intent)
                                true
                        }
                        else -> false
                }
            }
        )
        popup.show()
        //------------ FIN ---------------
}
```