# Menús y submenús

Un menú se define, como la mayoría de los recursos de Android, mediante un fichero XML, y se colocará en la carpeta */res/menu*. El menú se definirá mediante un elemento raiz `<menu>` y contendrá una serie de elementos `<item>` que representarán cada una de las opciones. Los elementos `<item>` por su parte podrán incluir varios atributos que lo definan, entre los que destacan los siguientes:

- `android:id`. El ID identificativo del elemento, con el que podremos hacer referencia dicha opción.
- `android:title`. El texto que se visualizará para la opción.
- `android:icon`. El icono asociado a la acción.
- `android:showAsAction`. Si se está mostrando una action bar, este atributo indica si la opción de menú se mostrará como botón de acción o como parte del menú de overflow. Puede tomar varios valores:
  - `ifRoom`. Se mostrará como botón de acción sólo si hay espacio disponible.
  - `withText`. Se mostrará el texto de la opción junto al icono en el caso de que éste se esté mostrando como botón de acción.
  - `never`. La opción siempre se mostrará como parte del menú de overflow.
  - `always`. La opción siempre se mostrará como botón de acción. Este valor puede provocar que los elementos se solapen si no hay espacio suficiente para ellos.

Al crear un proyecto nuevo en Android Studio, se crea un menú por defecto para la actividad principal con una única opción llamada "*Settings*". Si abrimos este menú (llamado normalmente*/res/menu/menu_main.xml* si no lo hemos cambiado de nombre durante la creación del proyecto) veremos el siguiente código:

```
<menu 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"
    tools:context=".MainActivity">

    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" />

</menu>
```

Como vemos se define un menú con una única opción, con el texto "*Settings*" y con el atributo `showAsAction="never"` de forma que ésta siempre aparezca en el menú de overflow.

Esta opción por defecto se incluye solo a modo de ejemplo, por lo que podríamos eliminarla sin problemas para incluir las nuestras propias. En mi caso la voy a conservar pero voy a añadir dos más de ejemplo: "Buscar" y "Nuevo", la primera de ellas para que se muestre, si hay espacio, como botón con su icono correspondiente, y la segunda igual pero además acompañada de su título de acción:

```
<menu 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"
tools:context=".MainActivity">

    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" />

    <item android:id="@+id/action_buscar"
        android:title="@string/action_buscar"
        android:icon="@drawable/ic_buscar"
        android:orderInCategory="100"
        app:showAsAction="ifRoom" />

    <item android:id="@+id/action_nuevo"
        android:title="@string/action_nuevo"
        android:icon="@drawable/ic_nuevo"
        android:orderInCategory="100"
        app:showAsAction="ifRoom|withText" />

</menu>
```

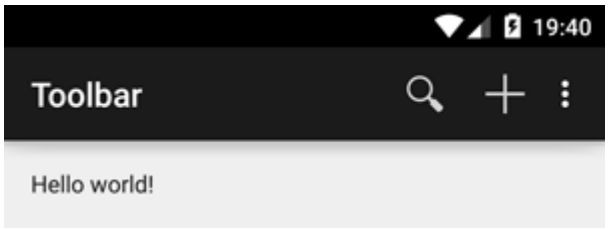
Los iconos `ic_buscar` e `ic_nuevo` los he añadido al proyecto de igual forma que en artículos anteriores, por ejemplo como vimos en el artículo sobre botones.

Como podéis ver además en la segunda opción (`action_nuevo`), se pueden combinar varios valores de `showAsAction` utilizando el caracter "|".

Una vez definido el menú en su fichero XML correspondiente tan sólo queda asociarlo a nuestra actividad principal. Esto se realiza sobrescribiendo el método `OnCreateOptionsMenu()` de la actividad, dentro del cual lo único que tenemos que hacer normalmente es inflar el menú llamando al método `inflate()` pasándole como parámetro el ID del fichero XML donde se ha definido. Este trabajo también suele venir hecho ya al crear un proyecto nuevo desde Android Studio:

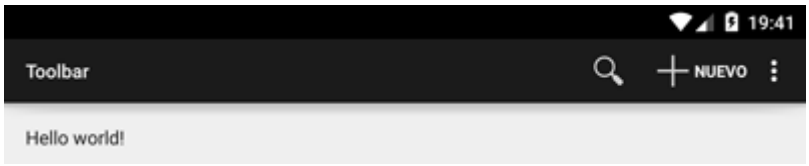
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    // Inflate the menu; this adds items to the action bar if it is
present.
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true;
}
```

Ejecutemos de nuevo la aplicación a ver qué ocurre:




Como podemos observar, la opción "*Settings*" sigue estando dentro del menú de overflow, y ahora aparecen como botones de acción las dos opciones que hemos marcado como `showAsAction="ifRoom"`, pero para la segunda no aparece el texto. ¿Y por qué? Porque no hay espacio disponible suficiente

con la pantalla en vertical. Pero si rotamos el emulador para ver qué ocurre con la pantalla en horizontal (pulsando Ctrl + F12) vemos lo siguiente:




Con la pantalla en horizontal sí se muestra el texto de la segunda opción, tal como habíamos solicitado con el valor `withText` del atributo `showAsAction`.

Este mismo menú también lo podríamos crear directamente mediante código, también desde el evento `onCreateOptionsMenu()`. Para ello, para añadir cada opción del menú podemos utilizar el método `add()` sobre el objeto de tipo `Menu` que nos llega como parámetro del evento.

# Submenús

Un submenú no es más que un menú secundario que se muestra al pulsar una opción determinada de un menú principal.

Como ejemplo, vamos a añadir un submenú a la settings del ejemplo anterior, al que añadiremos dos nuevas opciones secundarias. Para ello, bastará con insertar en el XML de menú un nuevo elemento `<menu>` dentro del item correspondiente a la settings.

```
<menu 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"
tools:context=".MainActivity">

    <item android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" >
            <menu>
                <item android1:id="@+id/settings1.1"
            android1:title="Opcion1.1"></item>
                <item android1:id="@+id/settings1.2"
            android1:title="Opcion1.2"></item>
            </menu>
        </item>
```

```
    <item android:id="@+id/action_buscar"
        android:title="@string/action_buscar"
        android:icon="@drawable/ic_buscar"
        android:orderInCategory="100"
        app:showAsAction="ifRoom" />

    <item android:id="@+id/action_nuevo"
        android:title="@string/action_nuevo"
        android:icon="@drawable/ic_nuevo"
        android:orderInCategory="100"
        app:showAsAction="ifRoom|withText" />

</menu>
```

Construido el menú, la implementación de cada una de las opciones se incluirá en el evento `onOptionsItemSelected()` de la actividad que mostrará el menú. Este evento recibe como parámetro el item de menú que ha sido pulsado por el usuario, cuyo ID podemos recuperar con el método `getItemId`(). Según este ID podremos saber qué opción ha sido pulsada y ejecutar unas acciones u otras. En nuestro caso de ejemplo, lo único que haremos será modificar el texto de una etiqueta (`lblMensaje`) colocada en la pantalla principal de la aplicación.

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.MnuOpc1:
            lblMensaje.setText("Opcion 1 pulsada!");
            return true;
        case R.id.MnuOpc2:
            lblMensaje.setText("Opcion 2 pulsada!");;
            return true;
        case R.id.MnuOpc3:
            lblMensaje.setText("Opcion 3 pulsada!");;
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```