

# Apunts

Descarregar aquests apunts [pdf](#) o [html](#)

## Tema 9. Menús Android


### Índex

1. [Introducció](#)
2. [Definició d'un menú com a recurs XML](#)
3. [Overflow menu](#)
4. [Contextual menu](#)
5. [Popup menu](#)
6. [Exposed dropdown menu](#)
7. [Navigation Drawer](#)
8. [Contextual Action Bar](#)
  1. [ActionMode.Callback](#)

# Introducció

Els menús formen part, de forma usual, de la interfície d'una aplicació Android, permetent afegir funcionalitat sense ocupar molt espai en la pantalla. [Información menús Material Design] (<https://material.io/components/menus>).

Hi ha quatre tipus de menús en Android:

- **Overflow Menu**, menú principal que pot ser activat quan es polsa sobre el botó corresponent (sigui físic o programari, depenent de la seua antiguitat) o sobre la icona  de la `AppBar` o `ToolBar`.
- **Contextual Menu**, són menús que aporten accions extra per a un determinat element de la vista. Habitualment s'activen amb una pulsació llarga sobre el mateix. Podem trobar dos tipus de menús contextuals: **Floating Context Menu** (obri un menú contextual que se superposa a l'activitat) i **Contextual Action Mode** (obri una barra d'acció on s'agrupen les actuacions a realitzar sobre els elements seleccionats).
- **Popup Menu**, és un menú emergent semblant al **Overflow Menu**, però que es troba vinculat a un element de la vista, com una forma d'ampliar les accions que es poden executar.
- **Navigation Drawer** o menú lateral lliscant, apareix dins de la guia de disseny com un component específic [Información adicional Material Design](#). Es sol trobar a la pantalla principal de l'app i comptar amb un botó per a desplegar-ho, encara que també pot ser obert lliscant el contingut de la pantalla des de l'extrem esquerre.

## Definició d'un menú com a recurs XML

L'alternativa més senzilla a l'hora de definir un [menú](#), és dins d'un arxiu XML. Podríem definir-ho també programàticament, però és prou més complicat. Els arxius associats a menús han de guardar-se en la carpeta `res/menu` del nostre projecte. L'estructura genèrica d'un xml associat a un menú seria quelcom així:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android">
3      <item android:id="@+id/option_1"
4          android:title="@string/option_1" />
5      <item android:id="@+id/option_2"
6          android:title="@string/option_2" />
7      ...
8      <group android:id="@+id/group1_1">
9          <item android:id="@+id/group_op1"
10             android:title="@string/group_op1" />
11          <item android:id="@+id/group_op2"
12             android:title="@string/group_op2" />
13      </group>
14  </menu>

```

🚩 **Nota:** Línea 2 l'element `<menu>` és l'element arrel del document. Línea 3 `<item>` defineix un element del menú (una opció) , i pot contindre un element `<menu>` per a la definició d'un submenú. Els atributs habituals són: **android:id** , **android:icon** i **android:title** . Amb **android:showAsAction** s'indica on apareixerà esta opció de menú:

Valor	Descripción
<code>ifRoom</code>	Solo coloca este elemento en la barra de la app si hay espacio. Si no hay lugar para todos los elementos marcados como "ifRoom", se muestran como acciones los elementos que tengan los valores <code>orderInCategory</code> más bajos; los restantes aparecerán en el menú ampliado.
<code>withText</code>	Incluye también el texto del título (definido por <code>android:title</code> ) con el elemento de acción. Puedes incluir este valor junto con uno de los otros marcadores separándolos con un canal   .
<code>never</code>	Nunca coloques este elemento en la barra de la app. En su lugar, enumera el elemento en el menú ampliado de la barra de la app.
<code>always</code>	Siempre coloca este elemento en la barra de la app. Evita usar esta opción a menos que sea esencial que el elemento siempre aparezca en la barra de acción. Configurar varios elementos para que siempre aparezcan como elementos de acción puede hacer que se superpongan con otra IU en la barra de la app.
<code>collapseActionView</code>	Es posible contraer la vista de acción asociada a este elemento de acción (declarada por <code>android:actionLayout</code> o <code>android:actionViewClass</code> ). Se introdujo esta opción en la API nivel 14.

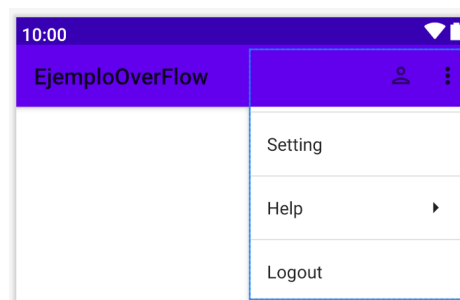
**Línea 8** l'etiqueta `<group>` permet agrupar elements `<item>` , per a poder actuar sobre ells (mostrar, ocultar amb **android:visible** o bé activar o desactivar amb **android:enabled** ) de forma conjunta. L'atribut **android:checkableBehavior** permet que els elements d'este grup es puguin activar amb un botó de selecció, els seus valors són: **single** (només un element del grup es pot seleccionar) **all** (tots es poden seleccionar) i **none** (cap és eleccionable) .

# Overflow menu

La ubicació en la pantalla on apareixen els elements del menú d'opcions depèn de la versió per al que vas desenvolupar l'aplicació: per a versió **Android2.x (nivell API 10)** o **versions anteriors** el menú apareixerà en la part superior de la pantalla quan l'usuari polse el botó **Menú**; per a **Android 3.0 (nivell API 11)** i **versions posteriors** el **Overflow Menu** apareix en la barra de la **App**.



Desenvoltarem un **Overflow menu** amb les opcions següents:"




La primera opció **Edit** apareix amb una icona i ha d'aparèixer fora del **Overflow menu** . Les dos següents opcions volem que estiguen agrupades, sent l'opció **Help** un submenú amb dos opcions més: **App** i **Android** . Finalment una opció **Logout** .

Vegem la definició d'este menú amb ubicció **res/menu/menu\_overflow.xml** :

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto">
4      <item android:id="@+id/opEdit"
5          android:title="@string/opEdit"
6          android:icon="@drawable/outline_person_24"
7          app:showAsAction="ifRoom|withText"/>
8      <group android:id="@+id/group_Config">
9          <item android:id="@+id/opSetting"
10             android:title="@string/opSetting" />
11          <item android:id="@+id/opHelp"
12             android:title="@string/opHelp">
13              <menu>
14                  <item android:id="@+id/opHelpApp"
15                     android:title="@string/opHelpApp" />
16                  <item android:id="@+id/opHelpAndroid"
17                     android:title="@string/opHelpAndroid"/>
18              </menu>
19          </item>
20      </group>
21      <item android:id="@+id/opLogout"
22          android:title="@string/opLogout" />
23  </menu>

```

 **Nota:** Línea 7 amb `android:showAsAction=ifRoom|withText` fem que este element del menú aparega sempre fora del **Overflow menu** i que si hi ha espai suficient es mostre el text i l'ícono. Línea 8 definim un grup amb dos **item**. Línea 13 definim dins de l'opció **Help** un submenú amb dos opcions.

Per a visualitzar el **Overflow menu** anul·lem el mètode `onOptionsItemSelected` en l'activitat o fragment corresponent (recordeu que s'invoca automàticament o polsant el menú depenent de la versió):

```

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


```

Per a controlar les opcions de menú que són polsades implementem el mètode `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
}

```

 **Nota:** Este mètode s'executa quan l'usuari selecciona un element del **Overflow menu** (inclosos els externs al mateix). Ens proporciona un element de tipus **MenuItem** amb informació de l'element d'opció polsat, analitzant el seu **itemId** concretarem quina opció ha sigut polsada.

A diferència de **onCreateOptionsMenu ()**, que només s'anomena la primera vegada que es construeix el menú, el mètode **onPrepareOptionsMenu ()** s'anomena cada vegada que el menú s'obri. Açò ens permet realitzar operacions com afegir o eliminar opcions de manera dinàmica, modificar la visibilitat dels diversos elements o modificar el seu text."

```

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

```

## Contextual menu

La idea és que al realitzar una pulsació llarga sobre un element d'una vista (un **TextView**, **Button**, un element d'un **RecyclerView**,...), s'obriga un menú amb unes opcions que afecten únicament a eixe element.

És necessari registrar que dit element té associat un **Contextual menu**, per a això tenim el mètode **registerForContextMenu (vista)**.

Quan es rel·liça la pulsació llarga sobre un element que té associat un menú contextual s'invoca al mètode `onCreateContextMenu ()`.

Crearem un projecte nou que cridarem **EjemploContextualMenu** on tindrem dos menús contextuais definits, un sobre un `Button` i l'altre sobre un `TextView`. Al `Button` li podrem canviar el color de fons i al `TextView` la grandària de la font a través dels menús contextuais.

En primer lloc definirem dos menús que arrepleguen la vista de cada un dels menús contextuais: Per a `res/menu/menu_contextual_textview`:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/op12"
        android:title="12" />
    <item android:id="@+id/op16"
        android:title="16" />
    <item android:id="@+id/op20"
        android:title="20" />
    <item android:id="@+id/op24"
        android:title="24" />
</menu>
```


Y per a `res/menu/menu_contextual_button`:

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/opRojo"
        android:title="Rojo" />
    <item android:id="@+id/opAzul"
        android:title="Azul" />
    <item android:id="@+id/opVerde"
        android:title="Verde" />
</menu>
```

```

1  class MainActivity : AppCompatActivity() {
2      private lateinit var binding: ActivityMainBinding
3      override fun onCreate(savedInstanceState: Bundle?) {
4          super.onCreate(savedInstanceState)
5          binding = ActivityMainBinding.inflate(layoutInflater)
6          val view = binding.root
7          setContentView(view)
8
9          registerForContextMenu(binding.button)
10         registerForContextMenu(binding.text)
11     }
12
13     override fun onCreateContextMenu(menu: ContextMenu?, v: View?,
14                                     menuInfo: ContextMenu.ContextMenuInfo?) {
15         super.onCreateContextMenu(menu, v, menuInfo)
16         if (v!!.id== id.text){
17             menuInflater.inflate(R.menu.menu_contextual_textview,menu)
18         }
19         if (v!!.id== id.button){
20             menuInflater.inflate(R.menu.menu_contextual_button,menu)
21         }
22     }
23
24     override fun onContextItemSelected(item: MenuItem): Boolean {
25         when (item.itemId){
26             id.op12->binding.text.textSize= 12F
27             id.op16->binding.text.textSize=16F
28             id.op20->binding.text.textSize=20F
29             id.op24->binding.text.textSize=24F
30             id.opAzul->binding.button.setBackgroundColor(BLUE)
31             id.opRojo->binding.button.setBackgroundColor(RED)
32             id.opVerde->binding.button.setBackgroundColor(GREEN)
33         }
34         return super.onContextItemSelected(item)
35     }
36 }

```

 **Nota:** Líneas 8 i 10 registrem les vistes sobre les quals tindrem el menú contextual. Líneas 13, 16 i 19 una vegada polsada una vista hem de determinar quin és, per a visualitzar el menú contextual correspondiente. Líneas 23-34 veiem que opció del menú s'ha polsat i actuem en conseqüència.



# Popup menu

Un **Popup menu** mostra una llista d'opcions de menú associades a la vista que va invocar el menú. És adequat per a proporcionar una ampliació d'accions de l'acció polsada. Suposem que tenim en la nostra aplicació un botó de *Compartir* i que quan polsem sobre ell ens oferix com a opcions *Mail* o *Sms*.

Per a definir este tipus de menú, crearem primer la seua estructura en un arxiu **XML**, tal com vam fer anteriorment.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/opSms"
        android:title="@string/opSms"
        android:icon="@drawable/outline_share_24"/>
    <item android:id="@+id/opMail"
        android:title="@string/opMail"
        android:icon="@drawable/outline_email_24"/>
</menu>
```

El codi que gestiona el menú pot quedar de la manera següent:

```

1  class MainActivity : AppCompatActivity() {
2      private lateinit var binding: ActivityMainBinding
3
4      override fun onCreate(savedInstanceState: Bundle?) {
5          super.onCreate(savedInstanceState)
6          binding = ActivityMainBinding.inflate(layoutInflater)
7          val view = binding.root
8          setContentView(view)
9
10         binding.button.setOnClickListener {
11             showPopup(binding.button)
12         }
13     }
14
15     private fun showPopup(view: View) {
16         val popup = PopupMenu(this, view)
17         popup.inflate(R.menu.poppup_menu)
18         popup.setOnMenuItemClickListener(PopupMenu.
19             OnMenuItemClickListener
20         { item: MenuItem? ->
21             when (item!!.itemId) {
22                 R.id.opMail -> {
23                     Toast.makeText(this@MainActivity, item.title, Toast.LENGTH_SHO
24                         .show()
25                 }
26                 R.id.opSms-> {
27                     Toast.makeText(this@MainActivity, item.title, Toast.LENGTH_SHO
28                         .show()
29                 }
30             }
31             true
32         })
33         popup.show()
34     }
35 }

```

📌 **Nota:** La gestió del menú és diferent del que veu anteriorment. Quan polsem el botó associat al **Popup menu**, executarem una funció que ens visualitzarà el **Popup menu** unflant la vista **líneas 16 i 17** i en eixe moment definim el **Listener** per a gestionar les opcions del menú **línea18**, una vegada tot definit mostrem el menú **línea 33**:

Per a visualitzar la icona del menú anar a la guia de Material d'este mateix enllaç.

# Exposed dropdown menu

"Els menús vistos fins a hora s'engloben en les guies de *Android* com **Dropdown menus** . Es diferencien de **Exposed dropdown menus** perquè estos últims mostren l'última opció seleccionada del mateix. Els vam veure quan expliquem **AutoCompleteTextView**

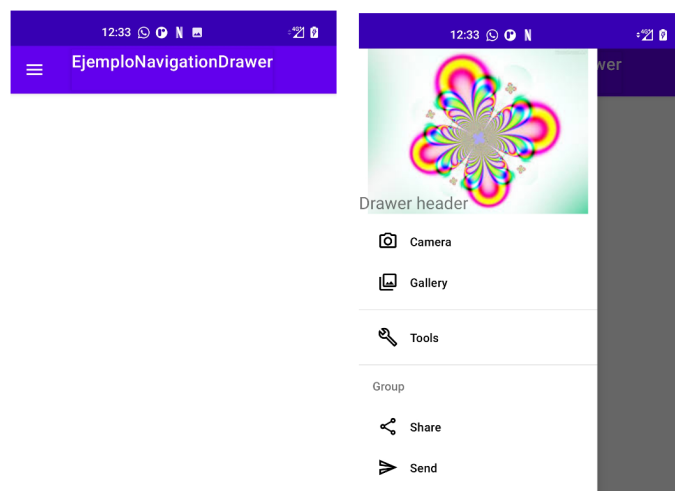
```
<com.google.android.material.textfield.TextInputLayout
    android:id="@+id/menu"
    style="@style/Widget.MaterialComponents.TextInputLayout.FilledBox.ExposedDropdownMenu"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:hint="@string/label">

    <AutoCompleteTextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="none"
    />
</com.google.android.material.textfield.TextInputLayout>
```

## Navigation Drawer

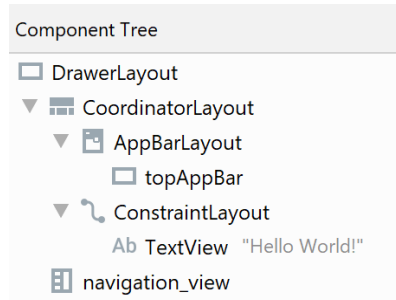
**Navigation Drawer** és un element d'interfície definit per Material Design consistent en el típic menú lateral lliscant des de l'esquerra que sol trobar-se en la pantalla principal de l'app i pot comptar amb un botó per a desplegar-ho.

La idea és tindre la següent interfície gràfica sense i amb el **Drawer menu** desplegat:



Vegem els arxius *XML* necessaris per a implementar l'exemple.

En el disseny de l'activity principal, usarem com a contenidor principal, un **DrawerLayout** el qual contindrà la Toolbar i el component **NavigationView** que és el menú pròpiament dita (a més de qualsevol altre element **View** que vullguem incorporar a la interfície) . Els components de la finestra principal menys el menú aniran dins d'un **CoordinatorLayout** :



```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.drawerlayout.widget.DrawerLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:id="@+id/drawer_layout"
9      tools:context=".MainActivity">
10
11      <androidx.coordinatorlayout.widget.CoordinatorLayout
12          android:layout_width="match_parent"
13          android:layout_height="match_parent">
14
15          <com.google.android.material.appbar.AppBarLayout
16              android:layout_width="match_parent"
17              android:layout_height="wrap_content"
18              android:theme="@style/Widget.MaterialComponents.Toolbar
19                  .Primary"
20              android:fitsSystemWindows="true">
21
22              <com.google.android.material.appbar.MaterialToolbar
23                  android:id="@+id/topAppBar"
24                  android:layout_width="match_parent"
25                  android:layout_height="?attr/actionBarSize"
26                  app:title="@string/app_name"
27                  style="@style/Widget.MaterialComponents.Toolbar.Primary"
28                  app:layout_collapseMode="pin"/>
29          </com.google.android.material.appbar.AppBarLayout>
30
31          <!-- Screen content -->
32
33      </androidx.coordinatorlayout.widget.CoordinatorLayout>
34
35      <com.google.android.material.navigation.NavigationView
36          android:layout_width="wrap_content"
37          android:layout_height="match_parent"
38          android:id="@+id/navigation_view"
39          android:layout_gravity="start"
40          app:headerLayout="@layout/drawer_header"
41          app:menu="@menu/drawer_menu"
42          android:fitsSystemWindows="true"/>
43  </androidx.drawerlayout.widget.DrawerLayout>

```



#### Aclariments:

- Si definim este atribut **tools:openDrawer="start"** en el **DrawerLayout** permetrem que amb el gest de deslizamiento cap a la dreta s'òbriga el panell de navegació.

- Línies 34, definim el **NavigationView**
- Línies 38, `android:layout_gravity=&#34;start&#34;` estableix que el menú isca de l'esquerra de la pantalla.
- Línies 39, `app:headerLayout=&#34;\@layout/drawer_header&#34;` estableix l'arxiu XML que definix la vista de la capçalera del menú, es definix dins de la carteta `res/layout`.
- Línies 40, `app:menu=&#34;\@menu/drawer_menu&#34;` estableix l'arxiu XML que definix la vista de les opcions del menú, es definix dins de la carpeta `res/menu`.
- Línia 41, `android:fitsSystemWindows=&#34;true&#34;` esablece que el **Navigation drawer** aparega per davall de la barra.

XML de la capçalera:"

```
<?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>
```

XML amb les opcions 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" />
        <item
            android:id="@+id/nav_gallery"
            android:icon="@drawable/outline_photo_library_24"
            android:title="Gallery" />
    </group>

    <group
        android:id="@+id/group1"
        android:checkableBehavior="single">
        <item
            android:id="@+id/nav_manage"
            android:icon="@drawable/outline_build_24"
            android:title="Tools" />
    </group>

    <item android:title="Group">
        <menu>
            <group android:checkableBehavior="single">
                <item
                    android:id="@+id/nav_share"
                    android:icon="@drawable/outline_share_24"
                    android:title="Share" />

                <item
                    android:id="@+id/nav_send"
                    android:icon="@drawable/outline_send_24"
                    android:title="Send" />
            </group>
        </menu>
    </item>
</menu>

```

Analitzem ara el codi del nostre *MainActivity.kt*, començan per la inicialització de components en el mètode `onCreate()` de l'activitat:

```

1  class MainActivity : AppCompatActivity(),
2  NavigationView.OnNavigationItemSelectedListener{
3      private lateinit var binding: ActivityMainBinding
4      lateinit var drawer_layout:DrawerLayout
5      lateinit var toggle: ActionBarDrawerToggle
6      lateinit var toolbar: MaterialToolbar
7      lateinit var navigationView: NavigationView
8
9      override fun onCreate(savedInstanceState: Bundle?) {
10         super.onCreate(savedInstanceState)
11         binding = ActivityMainBinding.inflate(layoutInflater)
12         val view = binding.root
13         setContentView(view)
14         navigationView=binding.navigationView
15         navigationView.setNavigationItemSelectedListener(this)
16
17         drawer_layout=binding.drawerLayout
18         toolbar=binding.topAppBar
19         setSupportActionBar(toolbar)
20         toggle = ActionBarDrawerToggle(this, drawer_layout,
21         binding.topAppBar, R.string.navigation_open,
22         R.string.navigation_close)
23         drawer_layout.addDrawerListener(toggle)
24     }
25 }

```

### Aclariments:

- Línia 1 indiquem amb **NavigationView.OnNavigationItemSelectedListener** la interfície que ens permetrà manejar els *clic's* sobre les opcions del menú.
- Línies 14 i 15, definim i registrem l'object **NavigationView** . Posteriorment implementarem el mètode **onNavigationItemSelectedListener()** .
- Línies 17 i 23, inicialitzem les instàncies del nostre **DrawerLayout** i **ActionBarDrawerToggle** , este últim ens permet configurar la icona d'aplicació que s'encarregarà d'obrir i tancar el panell de navegació. Registrem també amb **addDrawerListener()** .

A continuació implementem el mètode **onPostCreate()** i **onConfigurationChanged()** per a poder gestionar els canvis en el **Navigationdrawer** o actualitzar després d'un canvi d'orientació en la *app*. **syncState()** sincronitza l'estat de l'indicador de la barra de navegació amb el **DrawerLayout** vinculat.



```

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)
    }
}

```

Finalment sobreescrivem el mètode `onNavigationItemSelectedListener()` de la interfície `NavigationView.OnNavigationItemSelectedListener` per a gestionar les pulsacions sobre les opcions de menú desplegades. Tingueu en compte que al pulsar en una d'elles es tancarà el panell de navegació (línia 11) :

```

1  override fun onNavigationItemSelectedListener(item: MenuItem): Boolean {
2      val id = item.itemId
3      var s=""
4      when (id) {
5          R.id.nav_camera -> s="CAMARA"
6          R.id.nav_gallery-> s="GALLERY"
7          R.id.nav_manage-> s="TOOLS"
8          R.id.nav_send-> s="SEND"
9          R.id.nav_share-> s="SHARE"
10     }
11     drawer_layout.closeDrawer(GravityCompat.START)
12     Toast.makeText(applicationContext,"Pulsaste la opción "+s,
13                     Toast.LENGTH_SHORT).show()
14     return true
15 }

```

## Contextual Action Bar

La `app bar` de l'aplicació pot transformar-se en una `Contextual Action Bar (CAB)` per a establir accions sobre els elements seleccionats (per exemple d'una llista). Este mode s'activa usualment després d'una pulsació llarga sobre un element i es produeixen els canvis següents:


- Se superposa a la `app bar` una `cab` de color distint.
- La icona de navegació es reemplaça per una icona de tancament.

- Les accions de la **app bar** es reemplacen per accions contextuals.

Al tancar la **CAB** es mostra de nou la **app bar**.

Per a crear un menú d'este tipus en un **recycler** cal combinar una sèrie d'elements:

- Seleccionar diversos elements de la nostra llista, ho farem amb **ItemDetailsLookup** i **SelectionTracker**.
- Definir les opcions del **CAB** dins de la carpeta **res/menu**.
- Implementar **ActionMode.Callback**, que és la interfície encarregada de manejar els esdeveniments generats de la interacció de l'usuari amb el **CAB**.
- Observar l'element **SelectionTracker**, per a interactuar amb el **ActionMode.Callback**

 Com podreu recordar, al final del **tema8** s'explica la selecció de múltiples elements en un recycle. Pel que en este tema reprendrem eixe exemple per a afegir la funcionalitat del **Contextual Action Bar**, sobre els elements seleccionats

## ActionMode.Callback

Per a aconseguir la funcionalitat dita, necessitarem:

1. Realitzar la selecció múltiple dels elements del recycler, implementat en el tema anterior.
2. Definir les opcions del **CAB** dins de la carpeta **res/menu**. Per a això ens crearem un menú amb les opcions desitjades, per exemple el següent amb una icona per a eliminar:

```
<?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/delete"
    android:icon="@android:drawable/ic_menu_delete"
    android:title="Eliminar"
    app:showAsAction="ifRoom" />
</menu>
```

3. Implementarem un objecte de la interfície **ActionMode.Callback**, per a mostrar el **CAB** i gestionar les accions que es necessiten:

```

1 private val actionModeCallback = object : ActionMode.Callback {
2     override fun onCreateActionMode(mode: ActionMode, menu: Menu):
3     Boolean {
4         menuInflater.inflate(R.menu.action_mode_menu, menu)
5         return true
6     }
7     override fun onPrepareActionMode(mode: ActionMode, menu: Menu):
8     Boolean {
9         return false
10    }
11    override fun onActionItemClicked(mode: ActionMode, item: MenuItem):
12    Boolean {
13        return when (item.itemId) {
14            R.id.delete -> {
15                tracker.selection.sorted().reversed().forEach {id->
16                    datos.removeAt(id.toInt())}
17                recyclerView.getRecycledViewPool().clear();
18                adaptador.notifyDataSetChanged();
19                tracker?.clearSelection()
20                actionMode = null
21                true }
22            else -> false
23        }
24    }
25    override fun onDestroyActionMode(mode: ActionMode) {
26        tracker?.clearSelection()
27        actionMode = null
28    }
29 }

```

🚩 **Nota:** a l'implementar la interfície anul·larem els mètodes que s'executaran quan es crega la CAB : **onCreateActionMode** , just en el moment que es mostra a l'usuari **onPrepareActionMode** i el que s'executa quan es clica sobre un element **onActionItemClicked** . En el primer unflarem la vista del menú **Línea 4**. En el tercer codificarem les accions que volem que ocorreguen quan es polsa en cada element del menú **Línea 13 - 22**. Com en este cas el que estem fent és eliminar els elements al polsar sobre la paperera del menú, usem la selecció de tipus **SelectionTracker** , però abans d'eliminar ordenem i invertim la llista, per a evitar errors d'eliminació **Línea 15**. No oblidar netejar la selecció, d'esta manera al realitzar l'acció es desactivaran els elements seleccionats **Línea 19**, notificar a l'adaptador i anul·lar la variable actionMode. Haurem d'anul·lar el mètode **onDestroyActionMode** per a netejar la selecció i anular la CAB .

4. Observar el SelectionTracker per a activar la CAB :

```

1 tracker?.addObserver(
2     object: SelectionTracker.SelectionObserver<Long>() {
3         override fun onSelectionChanged() {
4             if (tracker!!.hasSelection()) {
5                 if (actionMode == null) {
6                     actionMode = this@MainActivity.
7                         startSupportActionMode(actionModeCallback)
8                 }
9                 actionMode?.title = "${tracker!!.selection.size()}"
10            } else {
11                actionMode?.finish()
12            }
13        }
14    })

```

📌 **Nota:** iniciarem la CAB usant `startSupportActionMode` de la següent manera **Línea 6**. Només s'iniciarà si no està iniciada amb anterioritat `actionMode==null` i hi ha elements en la selecció. Si no hi ha cap elements seleccionat, tancarem l'actionMode **Línea 11**. En la **Línea 9** es mostra com a títol de la CAB el nombre d'elements seleccionats.