

# Tema 12. Notificacions

## Índice

1. [Introducció](#)
2. [Elements d'una notificació](#)
3. [Canales de notificació](#)
4. [Crear una notificació](#)
5. [Estils de notifiacions](#)
6. [Botons](#)
7. [Alguns extres de notifiacions](#)

# Introducció

Les **notificacions** són missatges que es mostren en el sistema per a proporcionar a l'usuari informació addicional de l'app (missatges, avisos, etc). Els usuaris poden pressionar la notificació per a obrir l'app o realitzar una acció directament des de la notificació.

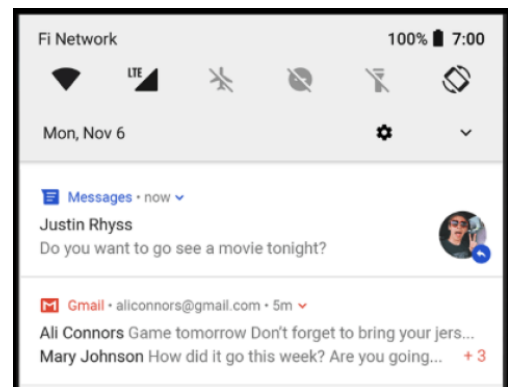
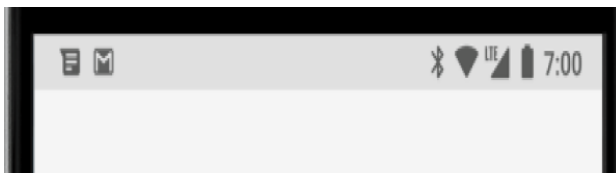
Les notificacions es mostren de diferents maneres:

- Icones en la barra d'estat.
- Entrada més detallada en el panell desplegable de notificacions.
- Notificacions emergents (normalment per a notificacions importants).
- Notificacions de pantalla de bloqueig.
- Distintiu en la icona de l'app, en versions més recents.

## Notificaciones en la Barra de Estado

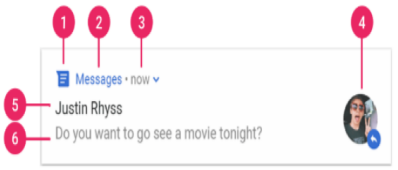
La barra d'estat d'Android està situada en la part superior de la pantalla. La part esquerra d'aquesta barra està reservada per a visualitzar notificacions. Quan es crea una nova notificació, apareix un text desplaçant-se en la barra, i a continuació, una xicoteta icona romandrà en la barra per a recordar a l'usuari la notificació.

L'usuari pot arrossegar la barra de notificacions cap avall, per a mostrar el llistat de les notificacions per llegir.



## Elements d'una notificació


Les notificacions podem veure-les de dues formes diferents, la vista normal abans que l'usuari la desplegue i la vista ampliada.

	<ol style="list-style-type: none"> <li>1. Icona xicoteta: s'estableix amb <code>setSmallIcon()</code>.</li> <li>2. Nom de l'aplicació: proporcionat pel sistema.</li> <li>3. Hora a la qual es va emetre la notificació. Es pot establir un valor explícit amb <code>setWhen()</code> o ocultar-lo <code>setShowWhen(false)</code>; si no es fa es mostrarà per defecte l'hora del sistema en el moment de recepció de la notificació.</li> <li>4. Icona gran: és opcional (generalment s'usa només per a fotos de contacte; no ho use per a la icona de la seua aplicació) s'estableix amb <code>setLargeIcon()</code>.</li> <li>5. Títol: Això és opcional i s'estableix amb <code>setContentTitle()</code>.</li> <li>6. Text: això és opcional i s'estableix amb <code>setContentText()</code>.</li> </ol>
---	---

La vista ampliada només apareix quan s'expandeix la notificació, la qual cosa succeeix quan la notificació està en la part superior de la bústia de notificacions, o quan l'usuari amplia la notificació amb un gest.

## Canales de notificació

A partir d'Android 8.0 (nivell de API 26), totes les notificacions han d'assignar-se a un canal o directament no seran visibles. Els canals de notificació permeten als desenvolupadors agrupar les notificacions en categories (canals). Això permetrà a l'usuari l'habilitat de modificar els ajustos de notificació per al canal sencer alhora. Per exemple, per a cada canal, els usuaris poden bloquejar completament totes les notificacions, anul·lar el nivell d'importància, o permetre que la insígnia de la notificació es mostre. Els usuaris també poden pressionar una notificació per a canviar els comportaments del canal associat. Totes les notificacions publicades en el mateix canal de notificació tenen el mateix comportament.

 la interfície d'usuari es refereix als canals de notificació com a "categories".

En els dispositius que executen Android 7.1 (nivell de API 25) i inferior, els usuaris poden administrar les notificacions només per aplicació (de fet, cada aplicació només té un canal). Per tant abans crear la notificació, haurem de crear el canal o els canals als quals volem afegir les nostres notificacions.

## Crear canals de notificacions

Abans de publicar una notificació s'ha de crear el canal, per la qual cosa és important que en iniciar-se l'aplicació o abans de crear la notificació s'execute el codi de creació. Vegem un exemple per a crear dos canals de notificació:

```
0 private fun crearCanal(
    idCanal: String,
    nombreCanal: String,
    descripcion: String,
    importancia: Int
): NotificationChannel? {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val canal = NotificationChannel(idCanal, nombreCanal, importancia)
        canal.description = descripcion
        return canal
    }
    return null
}

private fun crearCanalesNotificacion() {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val notificationManager = getSystemService(
            NotificationManager::class.java
        )
        var canal = crearCanal(
            CHANNELAVISOS_ID,
            "Avisos",
            "Avisos importantes",
            NotificationManager.IMPORTANCE_HIGH
        )
        canal!!.vibrationPattern = longArrayOf(400, 600, 100, 300, 100)
        // Registrando el canal en el sistema. Después de esto no se
        //podrá cambiar las características del canal
        // (importancia u otras propiedadesdel)
        notificationManager.createNotificationChannel(canal)
        canal = crearCanal(
            CHANNELMENSAJES_ID,
            "Mensajes",
            "Mensajes",
            NotificationManager.IMPORTANCE_LOW
        )
        notificationManager.createNotificationChannel(canal!!)
    }
}
```

 Els passos a seguir són:

1. Construir un objecte del tipus **NotificationChannel** , amb un id de canal únic, un nom visible per a l'usuari, una descripció i un nivell d'importància.

2. Si en el constructor no especifiquem la descripció, es pot especificar després amb `setDescription()`.
3. Podem aplicar altres comportaments visuals i sonors al nostre canal, en l'exemple hem activat vibració per al canal d'Avisos.
4. Finalment registrar el canal de notificació passant-lo al sistema amb el mètode `createNotificationChannel()`.

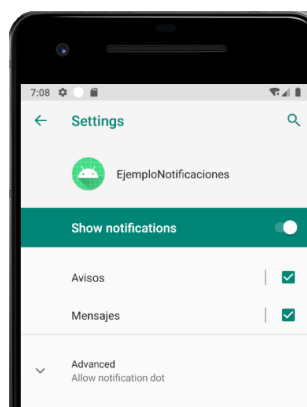
En el nostre exemple definim un mètode que crea un canal i que és cridat pel mètode que assigna els canals creats a l'aplicació.

La prioritat de les nostres notificacions s'ajusta a les següents opcions:

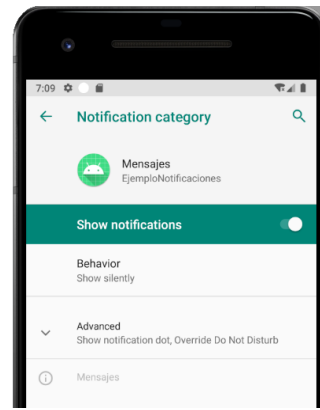
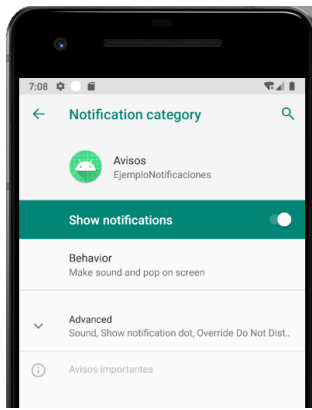
User-visible importance level	Importance (Android 8.0 and higher)	Priority (Android 7.1 and lower)
<b>Urgent</b> Makes a sound and appears as a heads-up notification	<code>IMPORTANCE_HIGH</code>	<code>PRIORITY_HIGH</code> or <code>PRIORITY_MAX</code>
<b>High</b> Makes a sound	<code>IMPORTANCE_DEFAULT</code>	<code>PRIORITY_DEFAULT</code>
<b>Medium</b> No sound	<code>IMPORTANCE_LOW</code>	<code>PRIORITY_LOW</code>
<b>Low</b> No sound and does not appear in the status bar	<code>IMPORTANCE_MIN</code>	<code>PRIORITY_MIN</code>

Evidentment, a totes les notificacions per a un canal se'ls donarà el mateix nivell d'importància.

Les categories creades apareixeran en les opcions de configuració del nostre dispositiu. *Anar a Ajustos->Notificaciones->Seleccionem Aplicació i ens apareixerà la categoria creada. També es mostrarà si es prem llargament sobre la notificació desplegada i després sobre la icona d'informació.*



Si entrem dins de la categoria veuríem la seua definició detallada:



És en aquest panell on l'usuari activa o no les notificacions d'una determinada categoria, també pot canviar l'acció que ocorrerà en llançar-se la notificació associada a la categoria.

## Crear una notificació

Per a crear notificacions s'utilitza la classe `NotificationCompat.Builder`. Per a crear una notificació solament crearem un objecte d'aquest tipus i li assignarem el context de l'aplicació.

```
val notificacion: NotificationCompat.Builder
notificacion = NotificationCompat.Builder(applicationContext, CHANNELMENSAJES_ID)
```

se usa como en el següent exemple:

```
val notificacion: NotificationCompat.Builder
notificacion = NotificationCompat.Builder(applicationContext, CHANNELMENSAJES_ID)
notificacion.setContentText("Este es el texto de mi notificación")
notificacion.setTitle("Mi notificación")
notificacion.setSmallIcon(R.mipmap.ic_launcher)
notificacion.setLargeIcon(
    ContextCompat.getDrawable(
        this,
        android.R.drawable.sym_action_email
    ) as BitmapDrawable?!!).bitmap
)
notificacion.setTicker("Optional ticker")
notificacion.setWhen(System.currentTimeMillis())
notificacion.setAutoCancel(true) //elimina notificación una vez visualizada
```

📌 Els mètodes `setTitle()` i `setContentText()`, permeten col·locar el títol i el text de la notificació. Per a mostrar les icones s'utilitzaran els mètodes `setSmallIcon()` i `setLargeIcon()` que es corresponen amb les icones mostrades a la dreta i a l'esquerra del contingut de la notificació en versions

recents d'Android, en versions més antigues tan sols es mostrarà la icona xicoteta a l'esquerra de la notificació. A més, la icona xicoteta també es mostrarà en la barra d'estat superior.

El *\*ticker* (text que apareix per uns segons en la barra d'estat en generar-se una nova notificació) ho afegirem mitjançant `setTicker()` i el text auxiliar (opcional) que apareixerà a l'esquerra de la icona xicoteta de la notificació mitjançant `setContentInfo()`.

La data/hora associada a la nostra notificació es prendrà automàticament de la data/hora actual si no s'estableix res, o bé pot utilitzar-se el mètode `setWhen()` per a indicar una altra marca de temps.

El següent pas, una vegada tenim completament configurades les opcions de la nostra notificació, serà el de generar-la cridant al mètode `notify()`. Per a això ens crearem un objecte de tipus `NotificationManagerCompat` al qual li passarem el context, a través d'aquest cridarem al mètode `notify()`. `notify` necessitarà un enter que identifique la nostra notificació i el resultat del builder que hem construït abans.

```
val idNotificacion:Int=0
val notificationManager = NotificationManagerCompat.from(this)
notificationManager.notify(idNotificacion, notificacion.build())
```

L'últim pas serà establir l'activitat a la qual hem de dirigir a l'usuari automàticament si aquest prem sobre la notificació. Per a això hem de construir un objecte `PendingIntent`, que serà el que continga la informació de l'activitat associada a la notificació i que serà llançat en prémer sobre ella. Per a això definirem en primer lloc un objecte `Intent`, indicant la classe de l'activitat a llançar, que en el nostre cas serà una cerca per `URL`. Aquest intent ho utilitzarem per a construir el `PendingIntent` final mitjançant el mètode `PendingIntent.getActivity()`. Finalment associarem aquest objecte a la notificació mitjançant el mètode `setContentIntent()` de la notificació creada.

Vegem com quedaria aquesta última part comentada:

```
val intent = Intent(Intent.ACTION_VIEW)
intent.data = Uri.parse("http://www.ua.es")
val intent2 = Intent(this@MainActivity, MainActivity::class.java)
notificacion.setContentIntent(PendingIntent.getActivity(this@MainActivity,0,intent2))
notificacion.setDeleteIntent(PendingIntent.getActivity(this@MainActivity,0,intent2))
```

⚠ En aquest cas, a més, hem afegit més funcionalitat en llançar un altre `PendingIntent` una vegada esborrada la notificació. Aquest `PendingIntent` ho afegirem

amb el mètode `setDeleteIntent()` sobre la notificació, i s'executarà en lliscar la notificació per a eliminar-la.

## Estils de notifiacions

Les notifiacions ens permeten [personalitzar-les](#) creant layouts propis, però nosaltres ens centrarem en els 3 estils que venen de sèrie. Recordeu que aquests estils s'expandeixen i contrauen, per la qual cosa hem de definir unes dades per a la notificació normal i altres per a l'expandida. A més, per defecte només apareixerà expandida la primera notificació de la safata, la resta estaran contraetes i l'usuari podrà obrir-les manualment amb un gest. Els estils possibles són:

### Estil BigText

S'usa a partir d'un objecte de tipus Builder, passant-ho com a paràmetre a la classe

`NotificationCompat.BigTextStyle` per a poder aplicar-li alguns mètodes opcionals.

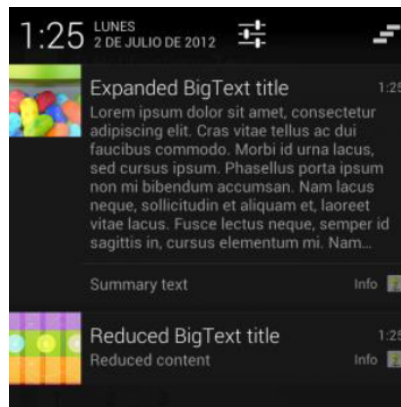
Queda més clar amb un exemple:

```
NotificationCompat.BigTextStyle(notificacion)
    .bigText("Lorem ipsum dolor sit amet, consectetur adipiscing
    Donec id ex hendrerit, ullamcorper lacus quis, auctor odio.
    Curabitur in feugiat elit, ut laoreet ex. Nunc in tristique
    at posuere magna. Curabitur augue lectus, tristique a consec
    vitae, luctus a nisi. Quisque nec nulla neque. Proin gravida
    lorem in tempus. Interdum et malesuada fames ac ante ipsum
    in faucibus.")
    .setBigContentTitle("Expanded BigText title")
    .setSummaryText("Summary text")
notificationManager.notify(2, notificacion.build())
```

📌 Com es pot suposar en crear l'estil, el farem sobre una notificació base creada amb anterioritat i que la passarem a l'objecte `BigTextStyle`.

En crear el nou estil establirem el text llarg, com pot ser el contingut d'un correu electrònic o d'un SMS, el títol que portarà en la seua forma expandida si volem que siga diferent, i un resum opcional que es mostrarà en 1 línia al final del text. En la captura podeu veure un exemple de com quedaria la mateixa notificació expandida i contraeta (la imatge no coincideix exacta amb el codi).





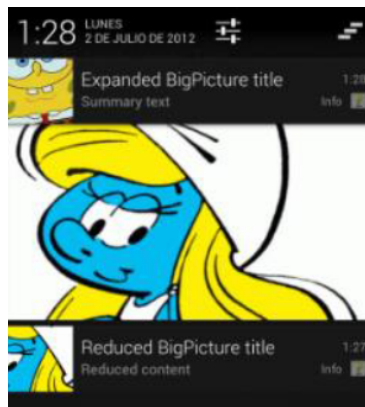
## Estil BigPicture

Si la notificació és relativa a una imatge també podem usar una notificació expandida per a mostrar aquesta imatge en la safata gràcies a

`NotificationCompat.BigPictureStyle`. El seu ús és quasi idèntic a l'anterior, canviant la classe i mètodes de l'estil.

```
NotificationCompat.BigPictureStyle(notificacion)
    .bigPicture(
        ContextCompat.getDrawable(
            this,
            R.drawable.ic_launcher_foreground
       )?.toBitmap()
    )
    .bigLargeIcon(
        ContextCompat.getDrawable(
            this,
            R.drawable.ic_launcher_foreground
       )?.toBitmap()
    )
    .setBigContentTitle("Expanded BigPicture title")
    .setSummaryText("Summary text")
notificationManager.notify(7, notificacion.build())
```

📌 En aquest cas en comptes d'un text, s'afegirà imatge gran que volem mostrar amb `bigPicture()` i si volem que la imatge "xicoteta" a l'esquerra de la notificació siga diferent en la forma expandida també la canviem amb `bigLargeIcon()`.

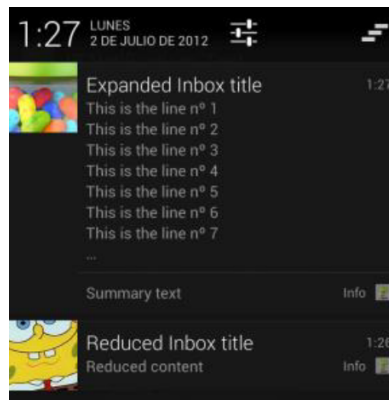


## Estil Inbox, línia a línia

L'últim estil que proporciona Android és el `Notification.InboxStyle`, que mostra una llista de text. Com diversos correus rebuts, o el que ens escriu algú per missatgeria instantània. Molt similar també als anteriors.

```
val notificacionInboxStyle = NotificationCompat.InboxStyle(notificacion)
    .setBigContentTitle("Expanded Inbox title")
    .setSummaryText("Summary text")
notificacionInboxStyle.addLine("Mensaje 1")
notificacionInboxStyle.addLine("Mensaje 2")
notificationManager.notify(3, notificacionInboxStyle.build())
```

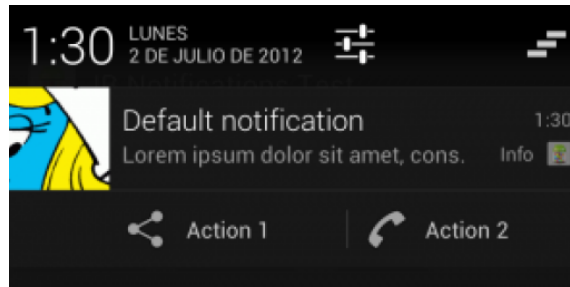
En aquesta ocasió afegim les línies una a una amb `addLine()`, encara que ens aconsellen un màxim de 5. Ens quedaria una cosa així (codi i imatge no coincideixen).



## Botons

Una altra novetat introduïda en Jelly Bean és la possibilitat d'afegir fins a 3 botons a les notificacions amb les seues accions associades. Es fa sobre l'objecte de tipus `Notification.Builder` amb el mètode `addAction()`, així que és compatible amb qualsevol dels 4 tipus de notificacions anteriors. En la notificació normal els botons es

consideraran com a part "expandida", per la qual cosa tindran el mateix comportament que aquestes.



En l'exemple veiem un mètode que afig 2 botons a qualsevol builder que passem com a paràmetre.

```
val action = NotificationCompat.Action.Builder(  
    android.R.drawable.ic_menu_delete,  
    "Delete", PendingIntent.getActivity(this@MainActivity, 0, i,  
        .build()  
notification.addAction(action)  
notification.addAction(  
    android.R.drawable.ic_menu_share,  
    "Share",  
    PendingIntent.getActivity(this@MainActivity, 0, i, 0))  
notificationManager.notify(4, notificacion.build())
```



Li passem com a paràmetres un drawable de la icona que portarà el botó a l'esquerra del text, el text que portarà, i un PendingIntent que s'activarà quan premem el botó.

## Alguns extres de notificaciones

Ara com ara tot el que hem vist ha sigut per a crear la notificació, però encara hem de mostrar-la i potser fer que el mòbil sone i/o vibre. També podem fer aquestes coses amb els mètodes de la classe `NotificationCompat.Builder`. Vegem uns exemples:

### So

Podem reproduir un so en activar la notificació. El la imatge es reproduueix el so per defecte, pot ser un altre.

```
val uri=RingtoneManager.getDefaultUri(RingtoneManager.TYPE_ALARM)  
notificacion.setSound(uri)
```

## Vibració

També podem fer que el telèfon vibre. Per a això el primer que necessitem és declarar el permís de vibració

```
<uses-permission android:name="android.permission.VIBRATE" />
```

Després creem un patró de vibració. Consisteix en un array de long alternant el temps de vibració i pausa en mil·lisegons. I li ho assignem a la notificació.

```
val pattern = longArrayOf(1000, 500, 1000)
notificacion.setVibrate(pattern)
```

La vibració per defecte la podem aconseguir de la següent manera:

```
notificacion.setDefaults(Notification.DEFAULT_VIBRATE)
```

## getActiveNotifications

A partir del Api 23 s'ha agregat a la classe `NotificationManager` el mètode `getActiveNotifications()`. Molt útil per a consultar l'estat de les notificacions, aquest mètode retorna un \*array amb l'estat de totes les notificacions actives en el moment.

## StatusBarNotification [] getActiveNotifications ()

Recupera una llista de notificacions actives: aquelles publicades per l'aplicació que encara no han sigut descartades per l'usuari o cancel·lades per l'aplicació. Es pot identificar la notificació a partir del tag o del id subministrat al `notify()`, així com una còpia de l'original Notification objecte a través `getNotification()`.

 **Exercici Resolt Notificaciones Expandibles**

 **Exercici Propost Temporizador Huevo**