Notificaciones

Contenido

Introducción

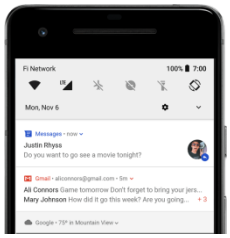
En Android existen varias formas de notificar mensajes al usuario:

- Toast √
- Diálogos √
- SnackBar
- Notificaciones

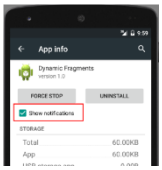
Notificaciones

Cuando se emite una notificación, primero aparece como un icono en la barra de estado.

Al deslizar hacia abajo en la barra de estado para abrir el cajón de notificaciones, pueden verse más detalles y realizar acciones con la notificación.
Además se puede arrastrar hacia abajo una notificación en el cajón para revelar la vista expandida, que muestra contenido adicional y botones de acción, si se proporcionan.



Debido a que las notificaciones pueden ser muy molestas, el usuario puede desactivar las notificaciones de cada aplicación desde la app Ajustes -> Aplicaciones->Seleccionar la aplicación "molesta" y deshabilitar las notificaciones.



Además, a partir de Android 8.1 (API nivel 27), las apps no pueden emitir un sonido de notificación más de una vez por segundo. Si tu app publica varias notificaciones en un segundo, todas aparecerán de la forma esperada, pero solo la primera notificación por segundo emitirá un sonido.
Android también aplica un límite de velocidad cuando actualiza una notificación. Si publicas actualizaciones en una sola notificación con demasiada frecuencia (muchas en menos de un segundo), es posible que el sistema no muestre algunas actualizaciones.
Las notificaciones existen desde la API 1, pero cada nueva API introduce cambios.

Proyecto Notificaciones_1

El proyecto P_65_Notificaciones_1 tiene un botón en la MainActivity que lanza una notificación cada vez que es pulsado (es decir, no sirve nada más que para aprender!).

Construir notificaciones

Para manejar notificaciones se utiliza la clase NotificationManager que puede ser obtenida desde el contexto a través del método getSystemService().

```
notificationManager = (NotificationManager) getSystemService (NOTIFICATION_SERVICE);
```

Las notificaciones en Android están representadas por la clase Notification y se emplea Notification.Builder() para facilitar su construcción.

```
builder = new Notification.Builder(getApplicationContext());
```

Nota 1: Por tanto, en cualquier actividad que use notificaciones estas instancias de objetos normalmente están en el método OnCreate() y declarados como campos para toda la clase.
Nota 2: Si el mínimo SDK es inferior a 21 en vez de las clases Notification y Notification.Builder hay que usar NotificationCompat y NotificationCompat.Builder de la librería com.android.support:support-compat:28.0.0.

Android Developer proporciona la siguiente guía para el buen diseño de las notificaciones en la barra de estado.
Las notificaciones obligatoriamente tienen un diseño básico para la "vista" que aparece al expandir la barra de estado y opcionalmente un diseño para la expansión de la propia notificación.
El diseño básico de la notificación tiene 64dp de altura e incluye como mínimo:

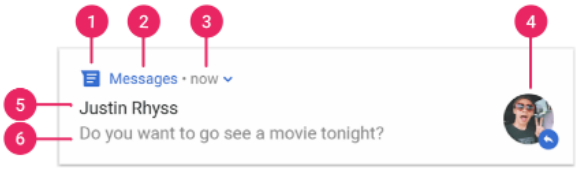


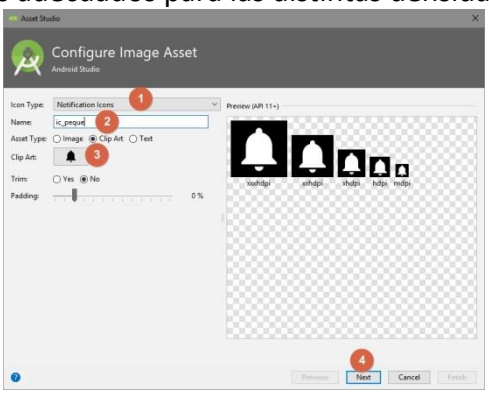
Figura 7. Una notificación con detalles básicos.

Las partes más comunes de una notificación se indican en la figura 7 de la siguiente manera:

1. Icono pequeño: se requiere y se establece con `setSmallIcon()`.
2. Nombre de la aplicación: Esto es proporcionado por el sistema.
3. Marca de tiempo: esto lo proporciona el sistema, pero puede anularlo `setWhen()` u ocultarlo `setShowWhen(false)`.
4. Icono grande: es opcional (generalmente se usa solo para fotos de contacto; no lo use para el icono de su aplicación) y establezca con `setLargeIcon()`.
5. Título: Esto es opcional y se establece con `setContentTitle()`.
6. Texto: esto es opcional y se establece con `setContentText()`.

Icono pequeño: Puede utilizarse:

- alguno de la plataforma (ej:Android.R.drawable.ic_popup_reminder)
- un drawable propio con los siguientes tamaños en px: 24x24 (mdpi), 36x36 (hdpi), 48x48 (xhdpi), 72x72 (xxhdpi) y 96x96 (xxxhdpi) y de color blanco.
- o lo mejor es usar la herramienta de construcción new Image Asset (que fija los colores y tamaños adecuados para las distintas densidades):



Fijar los atributos básicos de la notificación

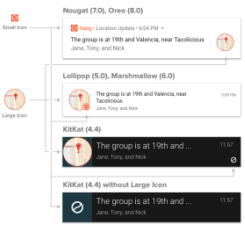
```
builder.setContentTitle("Título")
        .setContentText("Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed ")
        .setSmallIcon(android.R.drawable.ic_popup_reminder);
```

Si el texto es muy largo, solo se ve la primera línea.

Otros atributos

Podrían fijarse otros muchos atributos (enlace).

Por ejemplo, suele añadirse el icono grande que se verá a la derecha hasta Android 6 y a la izquierda a partir de Android 7 y cuyos tamaños en px deben ser 64x64 (mdpi), 96x96 (hdpi), 128x128 (xhdpi), 192x192 (xxhdpi) y 256x2566 (xxxhdpi):




```
Drawable drawable = getApplicationContext().getDrawable(R.drawable.ic_grande);
Bitmap iconoGrande = ((BitmapDrawable)drawable).getBitmap();
builder.setLargeIcon(iconoGrande);
```

Podemos reproducir un sonido al activar la notificación.

```
// Sonido por defecto de notificaciones, podemos usar otro
Uri defaultSound = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
builder.setSound(defaultSound);
```

O mostrar el momento en que se lanzó la notificación:

```
builder.setShowWhen(true);
```

Construir la notificación

```
notification = builder.build();
```

Mostrar la notificación

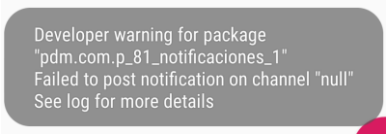
En nuestro ejemplo, será en el método onClick() del setOnClickListener del botón donde llamaremos al método notify(), pasándole un ID único para la notificación y el resultado de la construcción de la notificación:

```
findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        notificationManager.notify(0, notification);
    }
});
```

El primer parámetro del método notify() es el identificador de la notificación (no tiene que ser siempre una constante como en nuestro ejemplo).

Ejecuta para comprobar funcionamiento.

En dispositivos anteriores a Android 8 funciona sin problemas, pero no así en los Android 8 o superiores en los que aprece un "toast" que advierte del error:



Desde Android 8.0 (API 26), se añaden los Canales de notificación.

Los canales de notificación nos permiten a los desarrolladores agrupar nuestras notificaciones en grupos (canales) para que el usuario tenga la posibilidad de modificar los ajustes de notificación para todo el canal entero a la vez. Esta nueva característica ayuda a mejorar enormemente la experiencia de usuario de una aplicación que con solo un click largo en la notificación puede gestionar el canal.

Por ejemplo, para cada canal, los usuarios pueden bloquear completamente todas las notificaciones, cambiar el nivel de importancia o permitir que el distintivo de la notificación se muestre en el icono de la aplicación

Gestión de canales de notificación

Cuando se desarrolla una app para Android 8+ hay que implementar uno o más canales de notificación para poder mostrar las notificaciones a los usuarios de la app, instanciando un nuevo objeto de la clase NotificationChannel. Si la app se usa en Android 7.1- se comportará como si no existieran dichos canales.

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    // Gestión del canal
}
else{
    // Fijar (si interesa) características similares a las del canal
}
```

Cada canal distinto debe tener un identificador y un nombre único. Ambos son String y se diferencian en que como desarrolladores trabajamos con el identificador y al usuario le es visible el nombre:

```
String idCanal="miIdCanal";
String nombreCanal = "miNombreCanal";
```

Hay que fijar también la importancia (el nivel de atención que debe prestar el usuario) de la notificación: Los distintos valores son son:

Importance	Behavior	Usage	Examples
IMPORTANCE_HIGH	Makes a sound and appears on screen	Time-critical information that the user must know, or act on, immediately	Text messages, alarms, phone calls
IMPORTANCE_DEFAULT	Makes a sound	Information that should be seen at the user's earliest convenience, but not interrupt what they're doing	Traffic alerts, task reminders
IMPORTANCE_LOW	No sound	Notification channels that don't meet the requirements of other importance levels	New content the user has subscribed to, social network invitations
IMPORTANCE_MIN	No sound or visual interruption	Non-essential information that can wait or isn't specifically relevant to the user	Nearby places of interest, weather, promotional content

```
int importancia = NotificationManager.IMPORTANCE_DEFAULT;
```

Para instanciar el objeto de la clase:

```
NotificationChannel miCanal= new NotificationChannel(idCanal, nombreCanal, importancia);
```

Podemos manipular más este canal haciendo uso de algunos de los métodos públicos que la clase NotificationChannel nos proporciona:

```
miCanal.setLightColor(Color.GREEN);
miCanal.enableLights(true);
// Patrón de vibración: 2 segundo vibra, 0.5 segundos para (se repite 4 veces)
miCanal.setVibrationPattern(new long[]{2000, 500, 2000, 500, 2000, 500, 2000, 500});
// para que funcione hay que añadir permiso en manifiesto
miCanal.enableVibration(true);
miCanal.setShowBadge(false); //para no mostrar notificación en icono lanzador
```

Una vez gestionado el canal de la notificación, para crearlo se hace uso del método createNotificationChannel() de la clase NotificationManager y se añade al constructor de la notificación:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    String idCanal="miIdCanal";
    String nombreCanal = "miNombreCanal";
    int importancia = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel miCanal= new NotificationChannel(idCanal, nombreCanal,
importancia);
    miCanal.setLightColor(Color.GREEN);
    miCanal.enableLights(true);
    // Patrón de vibración: 2 segundo vibra, 0.5 segundos para (se repite 4 veces)
    miCanal.setVibrationPattern(new long[]{2000, 500, 2000, 500, 2000, 500, 2000, 500});
    // para que funcione hay que añadir persmiso en manifiesto
    miCanal.enableVibration(true);
    miCanal.setShowBadge(false); //para no mostrar notificación en el icono lanzador
    if (notificationManager != null) {
        notificationManager.createNotificationChannel(miCanal);
    }
    builder.setChannelId(idCanal);
}
else{
    //…
}
```

Nota: Si el sdk mínimo es Android 8, entonces

```
builder = new Notification.Builder(getApplicationContext(),idCanal);
```

Fijar características en versiones inferiores a Android 8

En nuestro "else", si queremos podemos fijar características similares

Prioridad

Asignar una prioridad es tan sencillo como llamar al método setPriority() de Notification.Builder.

Las constantes son por orden PRIORITY_MAX, PRIORITY_HIGH, PRIORITY_DEFAULT, PRIORITY_LOW y PRIORITY_MIN.

Comparativa Importancia y Prioridad

Importancia (Android 8.0 y superiores)	Prioridad (Android 7.1 e inferiores)
IMPORTANCE_HIGH	PRIORITY_HIGH o PRIORITY_MAX
IMPORTANCE_DEFAULT	PRIORITY_DEFAULT
IMPORTANCE_LOW	PRIORITY_LOW
IMPORTANCE_MIN	PRIORITY_MIN

Vibración

Llamando al método setVibrate() de Notification.Builder

Color del Led

Llamando al método setLights() de Notification.Builder.

Hay 3 parámetros: el color del led y los parámetros, ledOnMS y ledOffMS, para especificar la frecuencia de parpadeo (aunque la mayoría de dispositivos ignoran estos valores, así que usamos 1 y 0 respectivamente para encenderlo).

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    // Gestión del canal
}
else{
    builder.setPriority(Notification.PRIORITY_DEFAULT);
    builder.setVibrate(new long[]{2000, 500, 2000, 500, 2000, 500, 2000, 500});
    builder.setLights(Notification.COLOR_DEFAULT,1,0);
}
```

Añadir acciones

Cada notificación debe responder a un toque, generalmente para abrir una actividad.
Si queremos añadir esa acción, antes de construir la notificación con el método build() necesitamos crear un Intent que llame a la Activity que queremos abrir.

```
context= getApplicationContext();
intent = new Intent(context, SegundaActivity.class);
```

Si no deseamos que se abra ninguna actividad:

```
intent = new Intent();
```

Pero no llamaremos a startActivity(), crearemos un PendingIntent que será el que contenga la configuración que necesitamos para que cuando pulsemos la notificación se comporte como queremos, abriendo la Activity que hemos indicado en el Intent.
Como el término PendingIntent es la primera vez que aparece por aquí, antes de crearlo vamos a explicar qué es y qué hace. Este objeto es el que permite a componentes externos a nuestra aplicación, generalmente desarrollados por terceros, que ejecuten el código que nosotros indicamos. En nuestro caso su utilidad será la de permitir que NotificationManager sea capaz de lanzar el Intent que nosotros hemos configurado. Para dar una idea más sencilla de lo que hace un PendingIntent, se puede entender como un Intent que se manda al sistema, pendiente de una ejecución futura, pero de la que desconocemos cuándo se producirá.

```
intentPendiente = PendingIntent.getActivity(context, 0, intent,
PendingIntent.FLAG_UPDATE_CURRENT);
```

El método getActivity() de la clase pendingIntent recibe 4 parámetros:

- Context: El contexto en que se crea el PendingIntent.
- RequestCode: En este caso no necesitamos ningún código de respuesta, pero por si es necesario podría fijarse.
- Intent: El Intent que se lanzará cuando se ejecute el PendingIntent.
- int: Banderas que indican el comportamiento del PendingIntent. Pasamos PendingIntent.FLAG_UPDATE_CURRENT que indica que se actualizará la información del PendingIntent en el caso de que se lance uno y ya existiera previamente.

Añadimos al constructor ese intent pendiente justo antes de construir la notificación:

```
builder.setContentIntent(intentPendiente);
```

Ejecuta para comprobar el funcionamiento.
Dada la simplicidad de nuestro ejemplo, la navegación entre actividades (de Segunda a Main) con el PendingIntent funciona sin problemas, pero con aplicaciones más complejas, hay que asegurar la correcta navegación entre actividades.
Hay que construir una pila de tareas artificial porque cuando el usuario llega a la actividad desde la notificación, la tarea debería incluir una pila de actividades completa, que le permita presionar Atrás y navegar hacia arriba en la jerarquía de la aplicación.
Si se quiere iniciar una actividad que incluya una pila de actividades, se necesita crear una instancia de TaskStackBuilder, llamar a addNextIntentWithParentStack() y pasar el Intent de la actividad que se quiere iniciar (que debe tener BIEN definida en el manifiesto cual es su actividad padre!).

```
stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addNextIntentWithParentStack(intent);
```

Siempre y cuando se haya definido la actividad principal para cada actividad en el manifiesto, se puede llamar a getPendingIntent() a fin de recibir un PendingIntent que incluya toda la pila de actividades.

```
intentPendiente = stackBuilder.getPendingIntent(0, PendingIntent.FLAG_UPDATE_CURRENT);
```

Esta explicación sirve para cuando queremos iniciar actividades regulares (están en el flujo normal de la aplicación), pero podría darse el caso de que la actividad que deseamos que se inicie desde la notificación sea especial (solo se ve desde la notificación).

Cancelación

Una notificación permanecerá visible en el cajón de notificaciones hasta que la aplicación o el usuario la cancele.
Por tanto, se permite y recomienda que una notificación se cancele automáticamente cuando pulsamos sobre ella:

```
builder.setAutoCancel(true);
```

También podemos cancelarla desde código (importante para insistentes y en curso o para las que no tienen acción) haciendo

```
notificationManager.cancel(identificador_de_la_notificación);
```

Pueden cancelarse todas las notificaciones con:

```
notificationManager.cancelAll();
```

Una notificación también puede cancelarse si se ha establecido un tiempo de espera al crearla utilizando setTimeoutAfter(long duracion) y el sistema la cancelará una vez transcurrida la duración especificada en milisegundos.

Añadiendo botones

Se pueden agregar hasta 3 botones a la notificación para que realice distintas acciones. Dichos botones se muestran al arrastrar la notificación para ver la vista expandida.
Por simplicidad, en nuestro ejemplo, hay dos botones cuya acción será la misma SegundaActivity; de no ser así, deberíamos construir los Intent, TaskStackBuilder y PendingIntent correspondientes.
El método addAction tiene como primer parámetro el icono del botón, como segundo un texto junto al botón y el último el PendingIntent que interese:

```
builder.addAction(android.R.drawable.ic_menu_view, "Detalles", intentPendiente);
builder.addAction(android.R.drawable.ic_dialog_info, "Información", intentPendiente);
```

Dependiendo de la versión Android se visualiza el icono o el mensaje.
Pero la notificación no se cancela, habrá que hacerlo desde esa nueva actividad usando el método cancel(identificador_de_la_notificacion) de la clase :

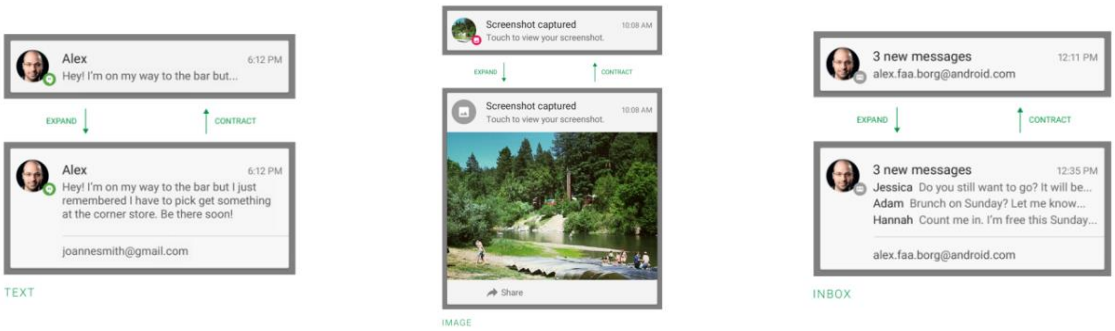
```
public class SegundaActivity extends AppCompatActivity {
    private NotificationManager notificationManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_segunda);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        …
        notificationManager = (NotificationManager) getSystemService (NOTIFICATION_SERVICE);
        notificationManager.cancel(0);
    }

}
```

En nuestro ejemplo, sabemos que el identificador de la notificación es 0, si no fuera una constante conocida, habría que haber añadido el valor al contruir el intent.

Después de probar el buen funcionamiento, quita ese código de SegundaActivity y los botones de MainActivity para continuar con la teoría siguiente.

Diseño expandido de la notificación

El diseño al expander la notificación puede personalizarse, la plataforma ofrece varios estilos de diseños expandidos:



Se hace uso del método setStyle() del constructor y de la clase Notification.Style.

- Comencemos por la de texto:

```
//Notificación expandida con texto
String texto="Bla, bla, bla\n" +"Bla, bla, bla\n" +"Bla, bla, bla\n";
Notification. BigTextStyle style=new Notification.BigTextStyle()
        .bigText(texto)
        .setBigContentTitle("Título Big")
        .setSummaryText("Resumen");
builder.setStyle(style);
```

- Si preferimos imágenes:

La imagen debe tener una relación de aspecto de 2:1, pero su contenido principal debe tener una relación de 43:24 (~ 1.79), ya que algunos dispositivos recortan más allá de este ancho. Android no tiene un límite de tamaño, sin embargo, los tamaños recomendados mínimo, equilibrado y máximo para mdpi son:

Mínimo - 512x256
Equilibrado - 1440x720
Máximo - 2880x1440

```
// Notificación expandida con imagen
Drawable drawable2 = getApplicationContext().getDrawable(R.drawable.simpson);
Bitmap bitmap2 = ((BitmapDrawable)drawable2).getBitmap();
Notification.BigPictureStyle style=new Notification.BigPictureStyle()
        .bigPicture(bitmap2)
        .setBigContentTitle("Título Big")
        .setSummaryText("Resumen");
builder.setStyle(style);
```

- Y el tercer tipo, lo usaremos para varias líneas de texto:

```
Notification.InboxStyle style=new Notification.InboxStyle()
        .setBigContentTitle("Titulo Big" )
        .setSummaryText("Resumen")
        .addLine("línea 1")
        .addLine("línea 2")
        .addLine("n° de líneas recomendadas 5");
builder.setStyle(style);
```

- En cualquiera de los casos, si al extender no deseamos ver el icono grande, añadiremos al estilo

```
.bigLargeIcon((Bitmap) null)
```

- Desde Android 7 hay otros estilos: Notification.MediaStyle para mostrar controles de reproducción multimedia y Notification.MessagingStyle para mostrar una conversación.

Notificaciones insistentes y en curso

También podemos establecer nuestras notificaciones como:

- insistentes, para que repitan indefinidamente su sonido, vibración y luces: son útiles para casos de prioridad máxima como llamadas o alarmas

```
// Insistente
// Son tan intrusivas que han decidido no añadir un método para esto en el Builder
notification.flags = notification.flags | Notification.FLAG_INSISTENT;
```

- o en curso para que se mantengan en la bandeja sin poder quitarse: se pueden usar para indicar al usuario que hay un proceso ejecutándose en segundo plano (como un servicio).

```
// En curso
builder.setOngoing(true);
```

En cualquiera de los dos casos es muy importante que demos la posibilidad de cancelar la notificación o el evento o podemos enfadar a nuestros usuarios.

Actualizaciones.

Para actualizar una notificación sin enviar una nueva sólo tenemos que enviarla usando el mismo número de referencia y se sobreescribirá (lo que hemos hecho hasta ahora).

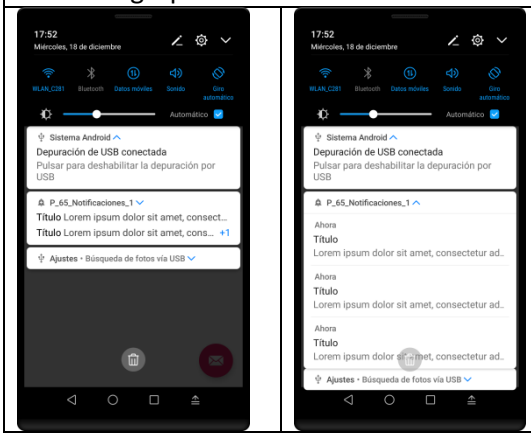
```
findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        notificationManager.notify(0, notification);
    }
});
```

Si en cambios queremos nuevas notificaciones:

```
findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        notificationManager.notify(numNotificacion, notification);
        numNotificacion++;
    }
});
```

En el ejemplo anterior, uso una variable numNotificacion inicializada a 1.

En los dispositivos se muestra:

Android 7+	Android 6-
En Android 7.0 y versiones posteriores, el sistema agrupa automáticamente	No agrupa
	

Si deseamos que se muestren cada una, lo que recomienda la plataforma Android es agrupar (aunque parezca que ya lo hace automáticamente!!!).
¿Cuándo es recomendable que se muestren cada una?

- Son interactivas, con acciones específicas para cada notificación.
- Hay más información en cada notificación que el usuario debería ver.

Si las notificaciones no cumplen con los criterios que se mencionan arriba, procura actualizar una notificación existente con información nueva o crear una notificación con estilo INBOX para mostrar varias actualizaciones en la misma conversación.

Agrupación de repeticiones

Hay que agrupar. En el constructor de las notificaciones que queramos agrupar hay que añadir el grupo al que pertenecen.
Y **antes** de lanzar cualquier notificación que pertenezca a un grupo, hay que lanzar otra notificación que sea la de resumen (que incluirá un fragmento de todas las de su grupo), que pertenezca al mismo grupo y especificar dicho tipo. El identificador de dicha notificación de resumen debe permanecer fijo para que solo se lance una vez.
Por ahorrar instancias de objetos, utilizo el mismo objeto builder (aunque debería ser distinto si quisiéramos que tuviese un título distinto):

```
//notification = builder.build();

findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Fijamos el agrupamiento
        builder.setGroup("com.pdm.miAgrupacion");
        // Fijamos características de la notificación de resumen y lanzamos
        builder.setGroupSummary(true);
        resumen = builder.build();
        notificationManager.notify(0, resumen);
        // Fijamos características de la notificación normal y lanzamos
        builder.setGroupSummary(false);
        notification = builder.build();
        notificationManager.notify(numNotificacion, notification);
        numNotificacion++;
    }
});
```

Tal y como está configurada la cancelación (setAutocancel(true)) en cuanto pulsemos una notificación del grupo (la de resumen o cualquiera de las individuales), se abre la SegundaActividad y se cancelan todas las notificaciones del grupo. Esto puede no ser lógico y que solo deseemos cancelar la notificación pulsada.
Para conseguirlo, deberá ser la actividad SegundaActivity la que cancele la notificación escogida y para ello debe conocer su identificador que le habremos pasado en el intent (y por tanto en PendingIntent):
Código clase SegundaActivity:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_segunda);
    Toolbar toolbar = findViewById(R.id.toolbar);
    setSupportActionBar(toolbar);

    FloatingActionButton fab = findViewById(R.id.fab);
    fab.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Snackbar.make(view, "Replace with your own action", Snackbar.LENGTH_LONG)
                    .setAction("Action", null).show();
        }
    });

    Objects.requireNonNull(getSupportActionBar()).setDisplayHomeAsUpEnabled(true);
    int notificacion=getIntent().getExtras().getInt("notificacionParaCancelar");
    NotificationManager notificationManager = (NotificationManager) getSystemService
(NOTIFICATION_SERVICE);
    notificationManager.cancel(notificacion);
}
```

Código clase MainActivity:

```
builder.setAutoCancel(true);
//notification = builder.build();
findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Fijamos el agrupamiento
        builder.setGroup("com.pdm.miAgrupacion");
        // Fijamos características de la notificación de resumen y lanzamos
        builder.setGroupSummary(true);
        resumen = builder.build();
        notificationManager.notify(0, resumen);
        // Fijamos características de la notificación normal y lanzamos
        builder.setGroupSummary(false);
        intent.putExtra("notificacionParaCancelar",numNotificacion);
        stackBuilder.addNextIntentWithParentStack(intent);
        intentPendiente = stackBuilder.getPendingIntent(0,
PendingIntent.FLAG_UPDATE_CURRENT);
        builder.setContentIntent(intentPendiente);
        builder.setContentText("Notificación "+numNotificacion);
        notification = builder.build();
        notificationManager.notify(numNotificacion, notification);
        numNotificacion++;
    }
});
```

La aplicación debería mejorarse ya que:

- cada vez que se cancela una notificación la variable numNotificacion para las siguientes que se envían vuelve a ponerse a 1 al volverse a crearse la MainActivity.
- si no cancelamos en orden de "modernidad", se pierden notificaciones

Para solucionarlo hay que "jugar" con los métodos del ciclo de vida de una actividad y con la gestión de tareas. Es debido a la "tontería" de notificaciones que enviamos en el ejemplo. ☺

Para versiones anteriores a Android 7 no funciona el agrupamiento y, por tanto, hay dos maneras de conseguir las repeticiones

Opción 1:

Se añade el número de veces que se ha lanzado la notificación:

```
builder.setAutoCancel(true);
//notification = builder.build();
findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            // Fijamos el agrupamiento
            builder.setGroup("com.pdm.miAgrupacion");
            // Fijamos características de la notificación de resumen y lanzamos
            builder.setGroupSummary(true);
            resumen = builder.build();
            notificationManager.notify(0, resumen);
            // Fijamos características de la notificación normal y lanzamos
            builder.setGroupSummary(false);
            intent.putExtra("notificacionParaCancelar", numNotificacion);
            stackBuilder.addNextIntentWithParentStack(intent);
            intentPendiente = stackBuilder.getPendingIntent(0,
PendingIntent.FLAG_UPDATE_CURRENT);
            builder.setContentIntent(intentPendiente);
            notification = builder.build();
            notificationManager.notify(numNotificacion, notification);
            numNotificacion++;
        }
        else{
            builder.setNumber(numNotificacion);
            intent.putExtra("notificacionParaCancelar", 0);
            stackBuilder.addNextIntentWithParentStack(intent);
            intentPendiente = stackBuilder.getPendingIntent(0,
PendingIntent.FLAG_UPDATE_CURRENT);
            builder.setContentIntent(intentPendiente);
            notification = builder.build();
            notificationManager.notify(0, notification);
            numNotificacion++;
        }
    }
});
```

Opción 2:

En la notificación se usa el estilo Inbox para "simular" la de resumen:

```
builder.setAutoCancel(true);
//notification = builder.build();
findViewById(R.id.button).setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
            // Fijamos el agrupamiento
            builder.setGroup("com.pdm.miAgrupacion");
            // Fijamos características de la notificación de resumen y lanzamos
            builder.setGroupSummary(true);
            resumen = builder.build();
            notificationManager.notify(0, resumen);
            // Fijamos características de la notificación normal y lanzamos
            builder.setGroupSummary(false);
            intent.putExtra("notificacionParaCancelar", numNotificacion);
            stackBuilder.addNextIntentWithParentStack(intent);
            intentPendiente = stackBuilder.getPendingIntent(0,
PendingIntent.FLAG_UPDATE_CURRENT);
            builder.setContentIntent(intentPendiente);
            notification = builder.build();
            notificationManager.notify(numNotificacion, notification);
            numNotificacion++;
        }
        else{
            if (numNotificacion >1) {
                builder.setContentTitle("Hay " + numNotificacion + " notificaciones");
                Notification.InboxStyle style = new Notification.InboxStyle();
                for (int i = 1; i <= numNotificacion; i++) {
                    style.addLine("Título num " + i + " - " + "Notificación num " + i);
                }
                builder.setStyle(style);
            }
```

```
            // builder.setNumber(numNotificacion);
            intent.putExtra("notificacionParaCancelar", 0);
            stackBuilder.addNextIntentWithParentStack(intent);
            intentPendiente = stackBuilder.getPendingIntent(0,
PendingIntent.FLAG_UPDATE_CURRENT);
            builder.setContentIntent(intentPendiente);
            notification = builder.build();
            notificationManager.notify(0, notification);
            numNotificacion++;
        }
    }
});
```

Y en Android 7- veremos:

Añadir una acción de respuesta directa

Añadir una barra de progreso

Notificaciones en Modo no molestar

Notificaciones en Pantalla de bloqueo

Practica propuesta

El P_66_Notificaciones_2 es una aplicación que cuando se lanza la vez nº 10 o la vez nº 20 o la vez nº 30 (múltiplos de 10!) lanza una notificación tal que al pulsar sobre ella se abre la web de nuestra supuesta empresa (publicidad!).