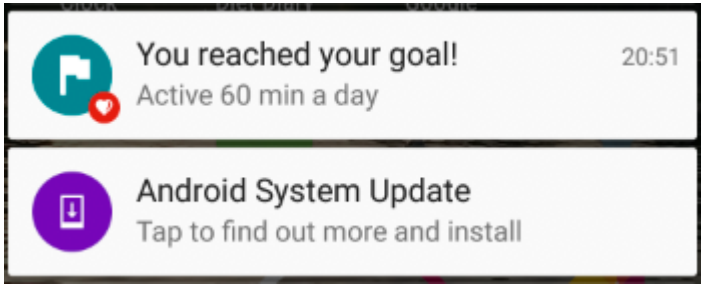
Desarrollo de apps con Android

9. Notificaciones

¿Qué es una notificación?

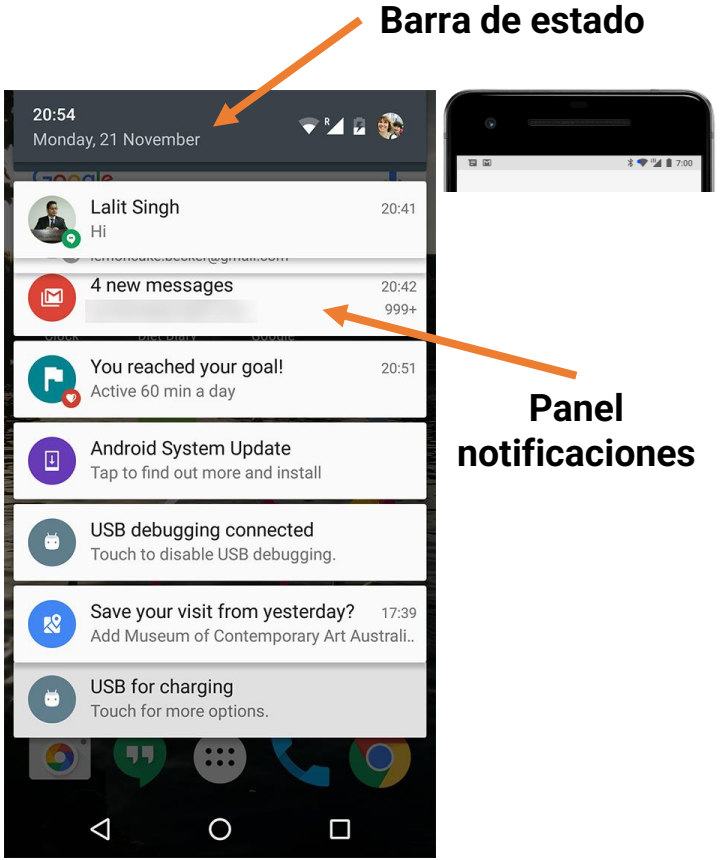
Notificación: Mensaje que se muestra al usuario de manera externa a la IU de la app.



- Icono pequeño
- Título
- Texto de la notificación

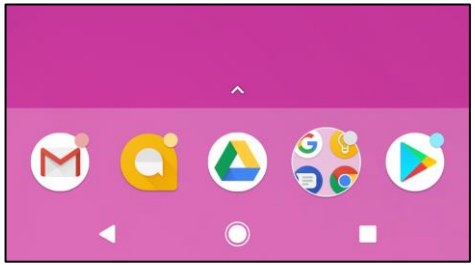
¿Cómo se utilizan las notificaciones?

- Android emite las notificaciones en forma de iconos en la barra de estado.
- Para acceder al detalle de la notificación, hay que abrir el panel de notificaciones.
- El usuario puede visualizar a las notificaciones siempre que lo desee desde el panel de notificaciones.

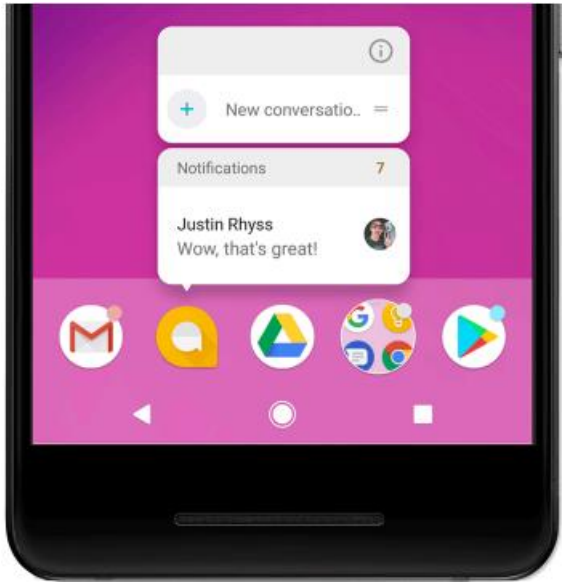


Insignias (*badges*)

Solamente están disponible en dispositivos con Android 8.0 (nivel de API 26) y posteriores.



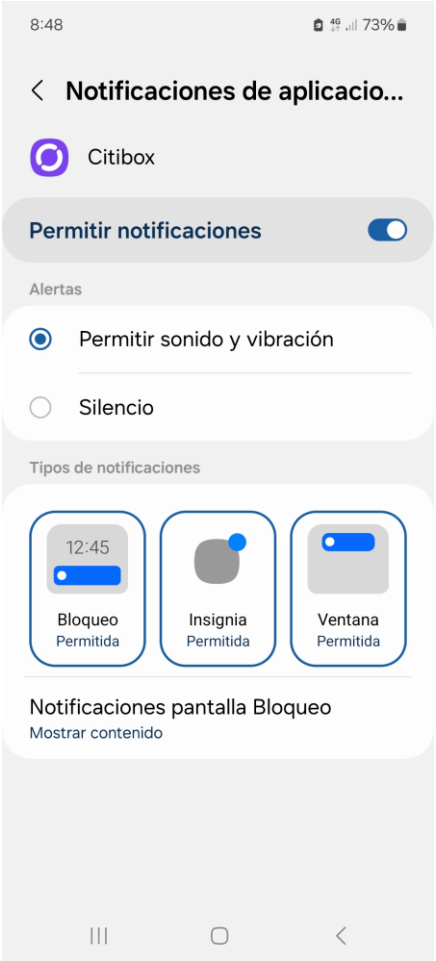
- Las notificaciones nuevas se muestran con una insignia de color sobre el icono de la app en el dispositivo (*“notification dot”*).
- Los usuarios pueden hacer una pulsación larga sobre el icono de la app para ver sus notificaciones. Similar al panel de notificaciones.



Canales de notificación

Canales de notificación

- Los canales de notificación se utilizan para crear un canal personalizable por el usuario para cada tipo de notificación que se desea mostrar.
- Distintas notificaciones, si están relacionadas, se pueden agrupar en un mismo canal.
- El comportamiento asignado a una notificación (ej., sonido y vibración) se aplica a todas las notificaciones del mismo canal.



Obligatoriedad de los canales de notificación

- Los canales de notificación se utilizaron por primera vez en Android 8.0 (nivel de API 26).
- A partir de Android 8.0 todas las notificaciones deben estar asignadas a un canal para que se puedan mostrar.
- Para apps orientadas a funcionar en versiones anteriores no requieren la implementación de los canales de notificación.

Creación de canales de notificación

Crear un canal de notificación (1/2)

- La instancia de un canal de notificación se crea utilizando el constructor NotificationChannel (más detalle).
- Se debe especificar lo siguiente:
  - Un ID que lo identifique de manera unívoca en el paquete de la app.
  - Nombre del canal visible para el usuario.
  - El nivel de importancia del canal.

Crear un canal de notificación (2/2)

```
private void createNotificationChannel() {
    // Crear el canal de notificación, pero solamente en versiones
    // posteriores al nivel de API 26 (es decir, anteriores a Android 8).
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        CharSequence name = getString(R.string.channel_name);
        String description = getString(R.string.channel_description);
        int importance = NotificationManager.IMPORTANCE_DEFAULT;
        NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name,
importance);
        channel.setDescription(description);
        // Registrar el canal en el sistema Android. No se podrá cambiar
        // posteriormente la importancia u otro comportamiento.
        NotificationManager notificationManager =
getSystemService(NotificationManager.class);
        notificationManager.createNotificationChannel(channel);
    }
}
```

Nivel de importancia

- Disponible a partir de Android 8.0 (nivel de API 26).
- Establece el nivel de intrusión de la notificación en el dispositivo (ej., el sonido y la visibilidad establecida para todas las notificaciones pertenecientes a ese canal).
- El rango varía desde `IMPORTANCE_NONE(0)` a `IMPORTANCE_HIGH(4)`.
- `IMPORTANCE_DEFAULT(3)`: Genera sonido, pero no hace intrusión visual.
- Para dar soporte a versiones anteriores de Android (es decir, inferiores al nivel de API 26), establecer la prioridad *(priority).*

Prioridad de la notificación

- Para versiones anteriores a Android 8.0 (nivel de API 26) determina cómo se ha de mostrar la notificación.
- Utilizar el método `setPriority()` para cada notificación.
- El rango varía desde PRIORITY_MIN a PRIORITY_MAX.

```
setPriority(NotificationCompat.PRIORITY_HIGH)
```

Importancia / Prioridad

Nivel de importancia visible para el usuario	Importancia (Android 8.0 y versiones posteriores)	Prioridad (Android 7.1 y versiones anteriores)
Urgente Emite un sonido y aparece como una notificación de atención.	`IMPORTANCE_HIGH`	`PRIORITY_HIGH` o `PRIORITY_MAX`
Alta Emite un sonido.	`IMPORTANCE_DEFAULT`	`PRIORITY_DEFAULT`
Media No emite ningún sonido.	`IMPORTANCE_LOW`	`PRIORITY_LOW`
Baja No emite sonido y no aparece en la barra de estado.	`IMPORTANCE_MIN`	`PRIORITY_MIN`
Ninguno No emite ningún sonido y no aparece en la barra de estado ni en el panel.	`IMPORTANCE_NONE`	`N/A`

Creación de notificaciones

Crear una notificación

- Crear una notification: utilizar la clase `NotificationCompat.Builder`.
- Depende de la librería AndroidX Core (a veces ya viene incluida a través de otras dependencias).
- Pasar al constructor el contexto de la app y el ID del canal de notificación que se le desea asociar.
- El constructor de `NotificationCompat.Builder` ignorará la información del canal si la versión del dispositivo es anterior aAndroid 8.0 (nivel de API 26).

```
NotificationCompat.Builder mBuilder = new
    NotificationCompat.Builder(this, CHANNEL_ID);
```

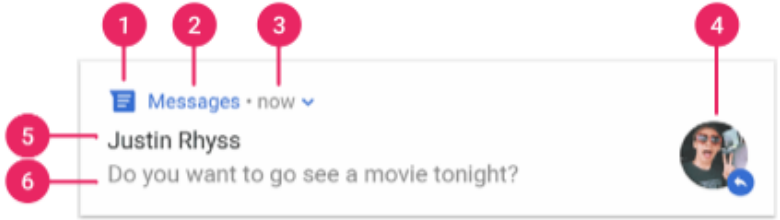
Permiso de envíos de notificaciones

- Desde Android 13 (nivel de API 33) se admite una solicitud de permiso en tiempo de ejecución para enviar notificaciones *no exentas* desde una app (incluyendo los servicios en primer plano, del inglés *Foreground services*, FGS).
- Ayuda a los usuarios a centrarse en las notificaciones que les resultan más importantes.
- Incluir el permiso en el fichero **AndroidManifest**:

```
<manifest ...>
   <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
   <application ...>
       ...
   </application>
</manifest>
```

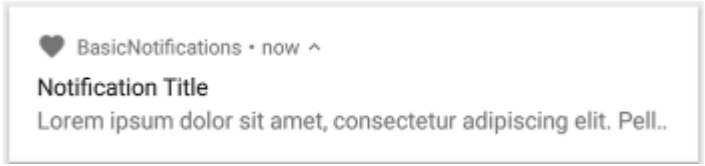
Contenido de una notificación(1/2)

Descripción general de las notificaciones.



1. **Icono pequeño (obligatorio)**: Se establece con `setSmallIcon()`.
2. **Nombre de la app:** Proporcionado por el sistema.
3. **Marca temporal (*timestamp*):** El sistema la proporciona, pero se puede anular con `setWhen()` o bien ocultar con `setShowWhen(false)`.
4. **Icono grande (opcional):** En general, se usa solo para fotos de contacto; no se debe utilizar para el icono de la app y se configura con `setLargeIcon()`.
5. **Título (opcional):** Se establece con `setContentTitle()`.
6. **Texto (opcional):** Se establece con `setContentText()`.

Contenido de una notificación (2/2)



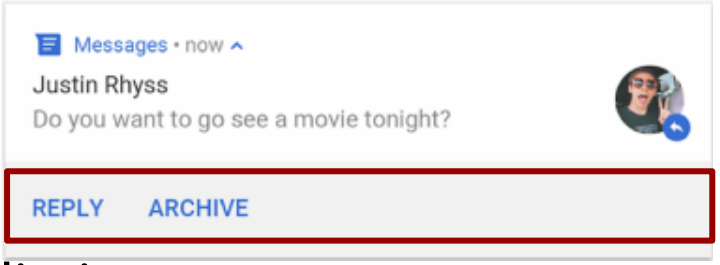

```
NotificationCompat.Builder mBuilder =
    new NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("Notification Title")
        .setContentText("Lorem ipsum dolor sit amet,
                consectetur adipiscing elit. Pell..")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT);
```

Acciones de pulsado y botones de acción

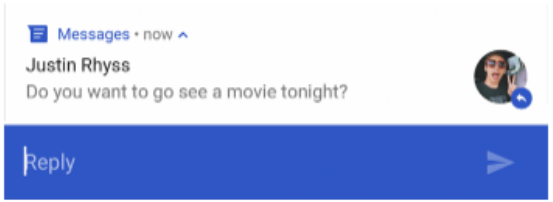
Acciones de pulsado

- Las notificaciones siempre responden cuando se les pulsa, normalmente lanzan una `Activity` en la app.
- Asignar el contenido del `Intent` a través del método `setContentIntent(PendingIntent intent)`.
- Pasar el `Intent` dentro de un objeto `PendingIntent`.

Botones de acción



- Los botones de acción pueden realizar distintas operaciones en la app (ej., iniciar una tarea background o realizar una llamada telefónica).
- A partir de Android 7.0 (nivel de API 24) se pueden responder mensajes directamente desde las notificaciones.



- Para añadir un botón de acción, pasar un `PendingIntent` al método `addAction()`.

PendingIntent

- Un `PendingIntent` representa la descripción de un `Intent` y la acción que se ha de llevar a cabo a través del mismo.
- Permite pasar un `PendingIntent` a otra aplicación para garantizar el permiso a realizar la operación especificada como si fuera la misma app.
  - En este caso, el sistema de notificaciones actúa en nombre de la app.

Métodos para crear PendingIntent

Se debe utilizar uno de los siguientes métodos para crear un `PendingIntent`:

- `PendingIntent.getActivity()`
- `PendingIntent.getBroadcast()`
- `PendingIntent.getService()`

Parámetros PendingIntent

1. Contexto de la aplicación.
2. `RequestCode`: **Constante de tipo entero** (`integer`) que identifica el `PendingIntent` de manera unívoca.
3. `Intent` que se debe entregar.
4. PendingIntent flag determina cómo el sistema ha de manejar los diferentes `PendingIntents` que proceden de la misma app.

Paso 1: Crear Intent

```
Intent notificationIntent =
     new Intent(this, MainActivity.class);

notificationIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK
      | Intent.FLAG_ACTIVITY_CLEAR_TASK);
```

Paso 2: Crear PendingIntent

```
PendingIntent notificationPendingIntent =
    PendingIntent.getActivity(
                this,
                NOTIFICATION_ID,
                notificationIntent,
                PendingIntent.FLAG_UPDATE_CURRENT);
```

Paso 3: Añadir al declarar la notificación

Para asignar la acción de pulsado a la notificación:

```
NotificationCompat.Builder mBuilder =
    new NotificationCompat.Builder(this, CHANNEL_ID)
     .setSmallIcon(R.drawable.notification_icon)
     .setContentTitle("Notification Title")
     .setContentText("Lorem ipsum dolor sit amet,
consectetur adipiscing elit. Pell..")
     .setContentIntent(notificationPendingIntent)
     .setAutoCancel(true);
```

Añadir botones de acción

Para añadir un botón de acción:

```
String ACTION_SNOOZE = "snooze"
Intent snoozeIntent = new Intent(this, MyBroadcastReceiver.class);
snoozeIntent.setAction(ACTION_SNOOZE);
snoozeIntent.putExtra(EXTRA_NOTIFICATION_ID, 0);
PendingIntent snoozePendingIntent =
        PendingIntent.getBroadcast(this, 0, snoozeIntent, 0);
NotificationCompat.Builder builder = new NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("Notification Title")
        .setContentText("Lorem ipsum dolor sit amet, consectetur adipiscing elit. Pell..")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        .setContentIntent(notificationPendingIntent)
        .addAction(R.drawable.ic_snooze, getString(R.string.snooze), snoozePendingIntent);
```

Notificaciones expandibles

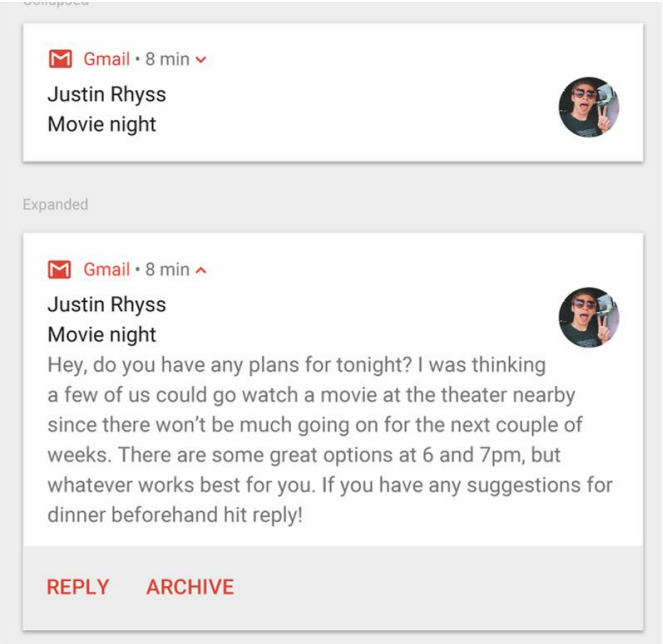
Notificaciones expandibles

- En el panel de notificaciones, las notificaciones se pueden mostrar con una vista normal (por defecto) o expandida.
- Las Notificaciones expandibles aparecieron por primera vez en Android 4.1 (nivel de API 16).
- Hay que utilizarlas con moderación ya que requieren más espacio y atención por parte del usuario.

Notificaciones con textos largos (1/2)

- BigText está orientado a notificaciones que necesitan mostrar más texto de lo normal.
- Establece más texto que una vista estándar.
- Hacer uso de la clase helper:

  `NotificationCompat.BigTextStyle`



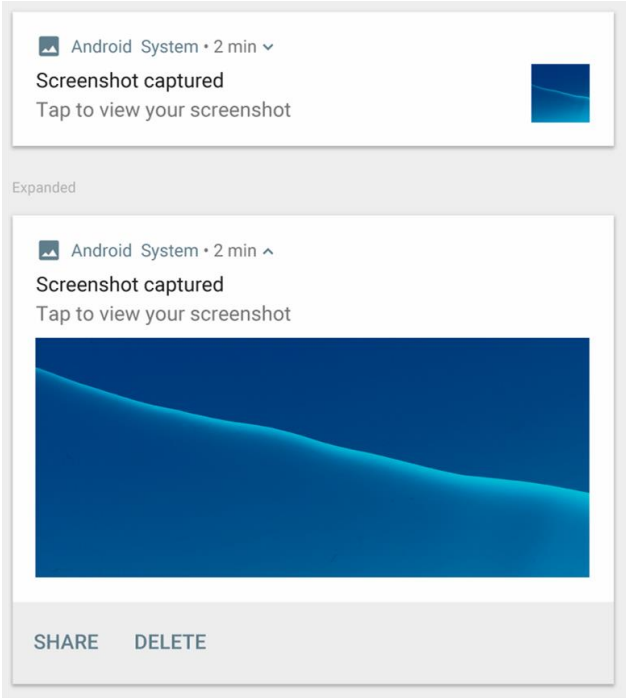
Notificaciones con textos largos (2/2)

```
NotificationCompat.Builder builder = new
NotificationCompat.Builder(this, CHANNEL_ID)
      .setSmallIcon(R.drawable.notification_icon)
      .setContentTitle("Justin Rhyss")
      .setContentText("Movie night")
      .setStyle(new NotificationCompat.BigTextStyle()
            .bigText("Hey, dou you have any plans..."));
```

Notificaciones con imágenes grandes (1/2)

- Para notificaciones extensas que adjuntan una imagen grande.
- Hacer uso de la clase helper:

`NotificationCompa.BigPictureStyle`



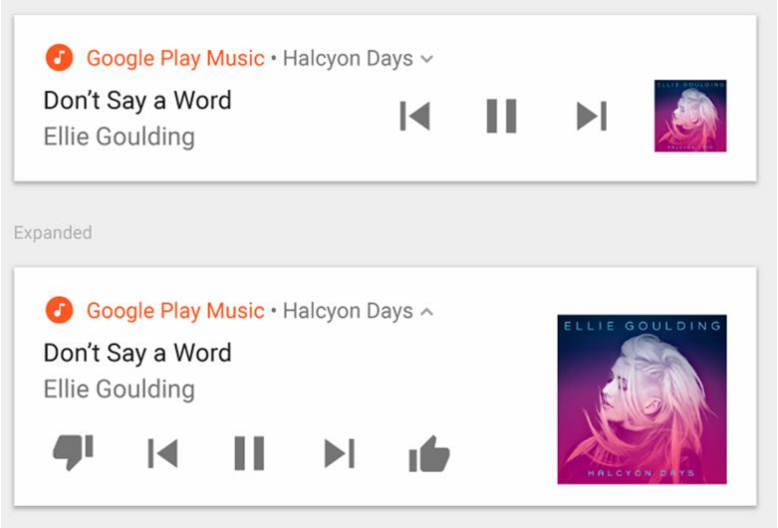
Notificaciones con imágenes grandes (2/2)

```
Notification notification = new NotificationCompat.Builder(context,
                                        CHANNEL_ID)
        .setSmallIcon(R.drawable.new_post)
        .setContentTitle("Screenshot captured")
        .setContentText("Tap to view your screenshot")
        .setLargeIcon(myBitmap)
        .setStyle(new NotificationCompat.BigPictureStyle()
                .bigPicture(myBitmap)
                .bigLargeIcon(null))
        .build();
```

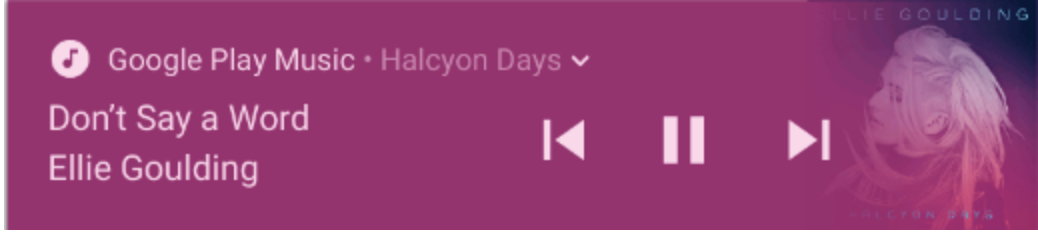
Multimedia (1/2)

- Para notificaciones de multimedia.
- Acciones para controlar multimedia (ej., música).
- Imagen para la portada de un álbum.
- Hacer uso de la clase helper:

  `NotificationCompat.MediaStyle`



Multimedia (2/2)




```
Notification notification = new NotificationCompat.Builder(context, CHANNEL_ID)
        // Mostrar controles en la pantalla de bloqueo incluso cuando el usuario
        // oculta contenido sensible.
        .setVisibility(NotificationCompat.VISIBILITY_PUBLIC)
        .setSmallIcon(R.drawable.ic_stat_player)
        // Añadir botones de control que invocan a intents del servicio multimedia.
        .addAction(R.drawable.ic_prev, "Previous", prevPendingIntent) // #0
        .addAction(R.drawable.ic_pause, "Pause", pausePendingIntent)  // #1
        .addAction(R.drawable.ic_next, "Next", nextPendingIntent)     // #2
        // Aplicar la plantilla del estilo multimedia.
        .setStyle(new MediaStyleNotificationHelper.MediaStyle(mediaSession)
                .setShowActionsInCompactView(1 /* #1: botón pausa*/))
        .setContentTitle("Wonderful music")
        .setContentText("My Awesome Band")
        .setLargeIcon(albumArtBitmap)
        .build();
```

Entrega de notificaciones

Entrega de notificaciones

- Utilizar la clase NotificationManager para entregar notificaciones.
  - Crear una instancia de `NotificationManager`.
  - Llamar al método `notify()` para entregar la notificación mostrándola en la barra de estado.

Instanciar NotificationManager

Invocar al método `getSystemService()` pasándole la constante `NOTIFICATION_SERVICE`.

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

Mostrar notificación (1/2)

- Invocar al método `notify()` para entregar la notificación pasándole dos valores:
  - El ID de la notificación, que se utilizará para actualizar o cancelar la notificación.
  - El objeto que se creó anteriormente con la clase `NotificationCompat.Builder`.

```
Notification myNotification = mBuilder.build();

mNotifyManager.notify(NOTIFICATION_ID, myNotification);
```

Gestionar notificaciones

Actualizar notificaciones

1. Actualizar una notificación cambiando o añadiendo parte de su contenido.
2. Emitir la notificación con los parámetros actualizados haciendo uso del `builder`.
3. Invocar al método `notify()` pasándole el ID de la notificación que se ha actualizado.
   - Si la anterior notificación todavía está visible se actualiza.
   - Si la anterior notificación ya ha sido descartada, se entrega la nueva notificación.

Cancelar notificaciones

Las notificaciones permanecerán visibles hasta las siguientes situaciones:

- El usuario las descarte deslizando el dedo o haciendo uso de la opción "**Borrar**".
- Se invoca al método `setAutoCancel()` cuando se crea la notificación hará que se elimine de la barra de estado cuando el usuario la pulse.
- La app invoca al método `cancel()` o `cancelAll()` a través de `NotificationManager`.

```
mNotifyManager.cancel(NOTIFICATION_ID);
```

Crear servicio de notificaciones

Creación del servicio para notificaciones (1/2)

- Crear una clase `ExampleService` que herede de `Service` (`android.app.Service`).
- Añadir el permiso de envío de notificaciones en el fichero `AndroidManifest.xml`:

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

- En el fichero `AndroidManifest.xml` declarar la clase como un servicio:

```
<application>…<service android:name=".ExampleService"/>…</application>
```

- Para lanzar el servicio, por ejemplo, desde el método `onCreate()` de la actividad principal `MainActivity`:

```
startService(new Intent(MainActivity.this, ExampleService.class));
```

- Se podría parar el servicio de la siguiente manera:

```
stopService(new Intent(MainActivity.this, ExampleService.class));
```

Creación del servicio para notificaciones (2/2)

- Al comienzo de la clase `ExampleService` declarar lo siguiente:

```
private NotificationManager notificationManager;
static final String CHANNEL_ID= "mi_canal";
static final int NOTIFICATION_ID = 1;
```

- En el método `onStartCommand()` del servicio definir el `notificationManager` y el `Builder` (`mBuilder`) según lo explicado. Ahí se invocará al método `notify()` de la siguiente manera:

```
notificationManager.notify(NOTIFICATION_ID, mBuilder.build());
```

- Si se quisiera dejar de notificar si se parara el servicio, declarar la siguiente instrucción en el método `onDestroy()` del servicio (opcional, pues puede interesar que las notificaciones sigan visibles aunque se destruya el servicio):

```
notificationManager.cancel(NOTIFICATION_ID);
```

Buenas prácticas en notificaciones

Si la app envía demasiadas notificaciones, el usuario terminará deshabilitándolas o, en el peor de los casos, desinstalará la app.

- **Relevante:** Si la información es de interés para el usuario.
- **Oportuna:** Las notificaciones aparecen en el momento que son útiles.
- **Corta:** Utilizar el menor número de palabras.
- Proporcionar a los usuarios la capacidad de elegir y utilizar los canales de notificación para categorizar las notificaciones de la app.