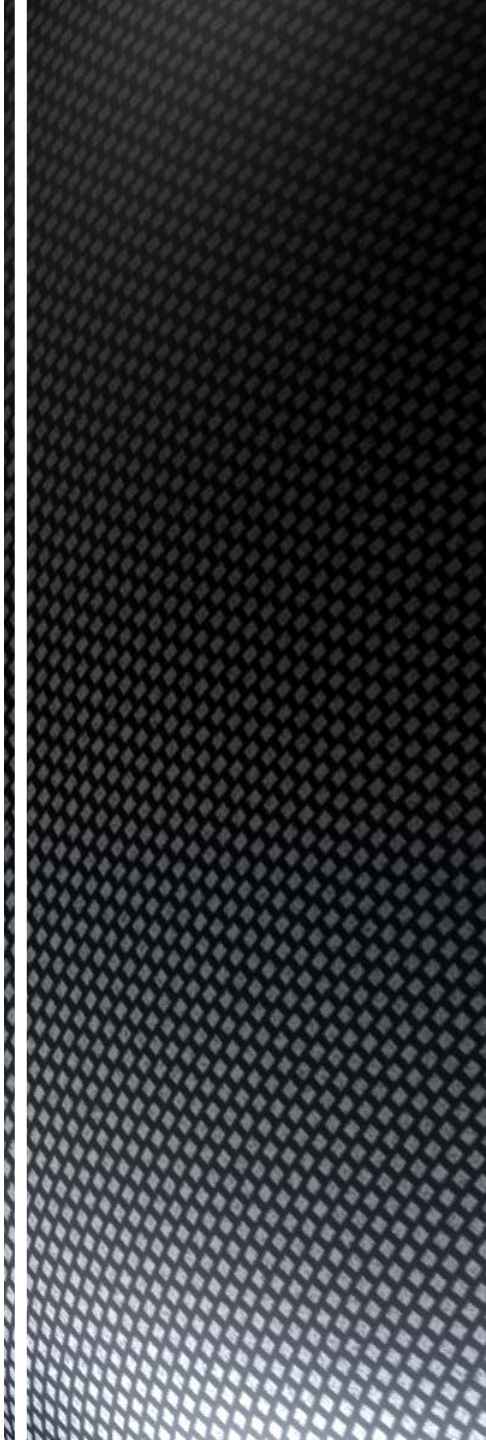
Android

Notificaciones



Notificaciones

Android nos proporciona varios elementos para realizar notificaciones al usuario.

- Dialog (Ya visto)
- Toast (Ya visto)
- Notification Area – Accesible cuando desplazamos hacia abajo la barra superior de la pantalla del movil.

Una notificación es un mensaje que se muestra al usuario fuera del User Interface de la aplicación que lo generó.

Android gestiona las notificaciones.

- Nuestra aplicación le pide a Android que gestione una notificación.
- Android coloca un icono en la Notification Area
- Para ver los detalles de la notificación el usuario abre el Notification Drawer.

Tanto el Notification Area como el Notification Drawer son componentes de Android que gestionan las notificaciones que pedimos que gestione.

Notification area

- El Notificaton area nos proporciona un elemento de interfaz de usuario llamado Drawer
  - El usuario puede deslizar hacia abajo el Notification area para abrir el Drawer y veremos información extra sobre todas las notificaciones ocurridas.
  - Incluso a menudo si pulsamos la notificación nos lleva a la aplicación que generó el aviso.

Notification area

Notificacion básica:

Incluye:

- Titulo,
- Contenido
- Icono pequeño
- Adicionalmente podemos añadir un “ticker text” que se muestra cuando llega la notificacion junto con el icono pequeño en la barra de tareas durante un tiempo

Al abrir el Drawer tendremos acceso a un View que contendrá:

- Titulo
- Contenido
- Icono
- Timestamp
- Acción a realizar cuando el usuario pulse

Notification area

Acciones a realizar sobre la notificación:

- Enviarla
- Actualizarla
- Cancelarla

Estas operaciones las maneja el servicio:

Notification Manager

Para crear una notificación y lanzarla, tendré que:

- Identificar la notificación para posibilitar actualizaciones de dicha notificación.
- Si es una notificación normal tendrá una vista por defecto
- Si es una Custom Notification podré crear una vista (View) personalizada que se mostrará en el Drawer mediante la clase RemoteView.

  Un RemoteView Permite crear una vista que se mostrará en otro proceso.
- Crear Intent explicito que lanzara la activdiad manejadora o subactividad, para gestionar la notificacion
- Crear un PendingIntent. Con esto le damos permiso a la otra aplicación para ejecutar el Intent explícito como si nosotros mismos lo lanzásemos.

  Nota: …. a la otra aplicación porque cuando se reciba la notificación probablemente no estemos en el contexto de nuestra aplicación.
- Insertar el PendingIntent en la notificación

Pasos para crear una Notificación en el Notification Area

- Obtener una referencia al NotificationManager

```
NotificationManager nm =
(NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
```

- Crear un nuevo objeto Notification. (Campos obligatorios):
  - Icono pequeño, Titulo, Texto detalle

```
Notification notificacion = new Notification.Builder(this)
        .setContentTitle("Titulo de la notificacion")
        .setContentText("Texto de la notificación")
        .setSmallIcon(R.drawable.icono)
        .build();
```

- Pasar la notificación al NotificationManager

```
private static final int ID_NOTIFICACION = 1;
nm.notify(ID_NOTIFICACION, notificacion);
```

- Se puede eliminar nuestra notificación llamando al método cancel del NotificationManager pasandole el identificador de la notificación.

```
nm.cancel(ID_NOTIFICACION);
```

Con esto tenemos creada la notificación, y ésta se envía al Notification Área con .notify(…) pero no hace nada, no lleva implicita ninguna acción, sólo se puede ver y cancelar.

Asociar una acción a la Notificación

Un PendingIntent es un token que una actividad o aplicación le entrega a otra (o a la misma) que le permite usar los permisos originales para ejecutar lo indicado en el PendingIntent.

Si le paso el PendingIntend como parte de la notificación (.setContentIntent(pIntent)) al NotificationManager le estoy dando a la notificación la capacidad de lanzar aquello que esté definido en el PendingIntent, aunque no esté la aplicación que generó la notificación ejecutándose o visible, es decir, fuera del contexto de la Application que la creó.

El PendingIntent lo utilizaremos para lanzar una actividad, como un Intent normal,

En el caso de las notificaciones lo vinculamos a la Notificación usando un NotificationBuilder o NotificationCompatBuilder (versiones 4.4 y anteriores)

La notification lleva el PendingIntent de forma que cuando el usuario la seleccione y haga click en la notificaión, el sistema lanzará el Explicit Intent referenciado en el PendingIntent.

Notification area

- Construir la notificación:

```
Notification.Builder notificationBuilder = new Notification.Builder(
getApplicationContext())
    .setTicker(tickerText)
    .setSmallIcon(android.R.drawable.stat_sys_warning)
    .setAutoCancel(true)
    .setContentIntent(mContentIntent)
    .setSound(soundURI)
    .setVibrate(mVibratePattern)
    .setContent(mContentView);
```

- Pasar la notificación al NotificationManager:

```
NotificationManager mNotificationManager = (NotificationManager)
  getSystemService(Context.NOTIFICATION_SERVICE);

mNotificationManager.notify(MY_NOTIFICATION_ID,
notificationBuilder.build());
```

Sonidos de Notificación

Se puede añadir un sonido al llegar la notificación con el método .setSound(Uri) del NotificationBuilder.

Uri es un uniforme resource identifier se puede construir usando la función Uri.parse() desde un fichero de recursos de la siguientes formas:

1- Utilizando el nombre del recurso:

Syntax : android.resource://[package]/[res type]/[res name]

Ejemplos:

```
Uri.parse("android.resource://com.my.package/drawable/icono");
Uri.parse("android.resource://com.my.package/raw/alarm_rooster");
```

2- Usando el Id del recurso:

Syntax : android.resource://[package]/[resource_id]

Ejemplos:

```
Uri.parse("android.resource://com.my.package/" + R.drawable.icono);
Uri.parse("android.resource://com.my.package/" + R.raw.alarm_rooster);
```

Vibración de Notificación

Se puede añadir una vibración al llegar la notificación con el método .setVibrate(pattern) del NotificationBuilder.

pattern es un array de long, es decir long[] pattern

El primer valor es el tiempo que hay que esperar para arrancar la vibración.
El segundo valor es el tiempo que está encendido el vibrador
Los siguientes valores corresponden al tiempo apagado, tiempo encendido, etc…

Asi:

```
private long[] mVibratePattern = { 0, 200, 200, 300 };
```

No espera para arrancar una vibración de 200ms, se para otros 200ms y termina con una vibración de 300ms.

Se necesita el permiso: android.permission.VIBRATE

E0801_NotificationStatusBar