104. Notificaciones

Una notificación es un mensaje que muestra Android fuera de la IU de la app para proporcionar al usuario recordatorios, mensajes de otras personas y otra información puntual de la app. Los usuarios pueden presionar la notificación para abrir la app o realizar una acción directamente desde la notificación

104.1 Presentación en el dispositivo

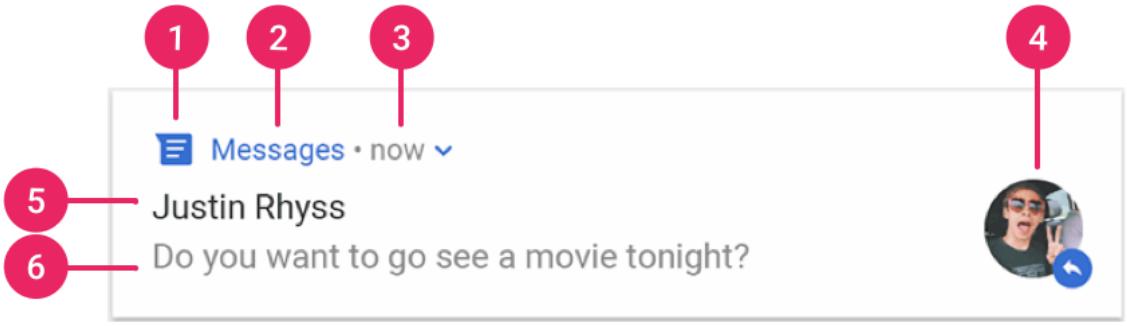
Ver en AD

Podemos tener notificaciones en:

- Barra de estados y panel de notificaciones
- Notificaciones emergentes
- Pantalla de bloqueo
- Distintivo en el icono de la aplicación (normalmente un punto de color)
-

104.2 Anatomia de las notificaciones

Ver en AD



Las partes más comunes de una notificación se indican en la Figura 7:

1. Ícono pequeño: Es obligatorio y se establece con setSmallIcon().
2. Nombre de la app: Proporcionado por el sistema.
3. Marca de tiempo: El sistema la proporciona, pero puedes anularla con setWhen() o bien ocultarla con setShowWhen(false).
4. Ícono grande: Es opcional (en general, se usa solo para fotos de contacto; no lo uses para el ícono de la app) y se configura con setLargeIcon().
5. Título: Es opcional y se establece con setContentTitle().
6. Texto: Es opcional y se establece con setContentText().

104.3 Como crear una notificación básica

Ver en AD

1. Incluir dependencias:

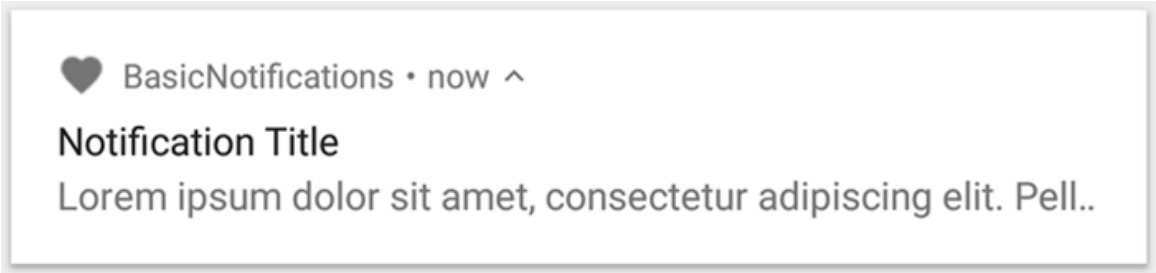
```
dependencies {
        implementation()"com.android.support:support-compat:28.0.0")
    }
```

104.3.1 Notificaciones básicas

En AD
Muestra un notificación compacta

104.3.1.1 Definiir contenido de la notificación

Para comenzar, debes configurar el contenido y el canal de la notificación con un objeto NotificationCompat.Builder. En el ejemplo de abajo, se muestra cómo crear una notificación con los siguientes elementos:



Para comenzar, debes configurar el contenido y el **canal de la notificación** con un objeto `NotificationCompat.Builder`. En el ejemplo de abajo, se muestra cómo crear una notificación con los siguientes elementos:

- un ícono pequeño, establecido por setSmallIcon(), que es el único contenido necesario visible para el usuario
- un título, establecido por setContentTitle()
- el texto del cuerpo, establecido por setContentText()

la prioridad de notificación, establecida por setPriority(). La prioridad determina cuán intrusiva debería ser la notificación en Android 7.1 y versiones anteriores. (Para Android 8.0 y posteriores, debes establecer la importancia del canal, lo que se muestra en esta sección).

```
    var builder = NotificationCompat.Builder(this, CHANNEL_ID)
            .setSmallIcon(R.drawable.notification_icon)
            .setContentTitle(textTitle)
            .setContentText(textContent)
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
```

104.3.1.2 Crear un canal de notificaciones y su importancia

```
private fun createNotificationChannel() {
    // Create the NotificationChannel, but only on API 26+ because
    // the NotificationChannel class is new and not in the support library
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        val name = getString(R.string.channel_name)
        val descriptionText = getString(R.string.channel_description)
        val importance = NotificationManager.IMPORTANCE_DEFAULT
        val channel = NotificationChannel(CHANNEL_ID, name, importance).apply {
            description = descriptionText
        }
        // Register the channel with the system
        val notificationManager: NotificationManager =
            getSystemService(Context.NOTIFICATION_SERVICE) as NotificationManager
        notificationManager.createNotificationChannel(channel)
    }
}
```

104.3.1.3 Establecer la acción del toque de notificación

```
// Create an explicit intent for an Activity in your app
val intent = Intent(this, AlertDetails::class.java).apply {
    flags = Intent.FLAG_ACTIVITY_NEW_TASK or Intent.FLAG_ACTIVITY_CLEAR_TASK
}
val pendingIntent: PendingIntent = PendingIntent.getActivity(this, 0, intent, 0)

val builder = NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
        .setContentTitle("My notification")
        .setContentText("Hello World!")
        .setPriority(NotificationCompat.PRIORITY_DEFAULT)
        // Set the intent that will fire when the user taps the notification
        .setContentIntent(pendingIntent)
        .setAutoCancel(true)
```

104.3.1.4 Mostrar la notificación

```
with(NotificationManagerCompat.from(this)) {
    // notificationId is a unique int for each notification that you must define
    notify(notificationId, builder.build())
}
```

104.3.1.5 Agregar botones de acción

Para agregar un botón de acción, pasa un elemento PendingIntent al método addAction(). Esto es similar a configurar la acción de toque predeterminada de la notificación, excepto que en lugar de iniciar una actividad, puedes llevar a cabo una variedad de otras funciones, como iniciar un BroadcastReceiver que realice un trabajo en segundo plano, de modo que la acción no interrumpa a la app que ya está abierta.

Por ejemplo, en el siguiente código se muestra cómo enviar una transmisión a un receptor específico:

```
val snoozeIntent = Intent(this, MyBroadcastReceiver::class.java).apply {
    action = ACTION_SNOOZE
    putExtra(EXTRA_NOTIFICATION_ID, 0)
}
val snoozePendingIntent: PendingIntent =
    PendingIntent.getBroadcast(this, 0, snoozeIntent, 0)
val builder = NotificationCompat.Builder(this, CHANNEL_ID)
        .setSmallIcon(R.drawable.notification_icon)
```

```
            .setContentTitle("My notification")
            .setContentText("Hello World!")
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)
            .setContentIntent(pendingIntent)
            .addAction(R.drawable.ic_snooze, getString(R.string.snooze),
                    snoozePendingIntent)
```

Más opciones para las notificaciones aqui

104.4 Notificaciones y Broadcast Receiver

Las notificaciones son mensajes que se presentn al usuario en la barra de estado o la pantalla de bloqueo, proporcionando retroalimentación o información. Pueden ser disparadas por un BroadcastReceiver o cualquier otro componente de la aplicación (como un servicio o una actividad)

Relación y Ejemplo de Interacción

Un escenario común de interacción es cuando un BroadcastReceiver detecta un evento específico y luego desencadena una notificación para informar al usuario. Por ejemplo, podrías tener un BroadcastReceiver que escucha un cambio en la conectividad de red y muestra una notificación cuando el dispositivo se desconecta de una red Wi-Fi.

```
class NetworkChangeReceiver : BroadcastReceiver() {

    override fun onReceive(context: Context, intent: Intent) {
        if (ConnectivityManager.CONNECTIVITY_ACTION == intent.action) {
            val noConnectivity =
intent.getBooleanExtra(ConnectivityManager.EXTRA_NO_CONNECTIVITY, false)

            if (noConnectivity) {
                showNotification(context, "Conectividad Cambiada", "Te has desconectado
de Wi-Fi.")
            }
        }
    }

    private fun showNotification(context: Context, title: String, text: String) {
        val notificationBuilder = NotificationCompat.Builder(context, CHANNEL_ID)
            .setSmallIcon(R.drawable.ic_notification)
            .setContentTitle(title)
            .setContentText(text)
            .setPriority(NotificationCompat.PRIORITY_DEFAULT)

        with(NotificationManagerCompat.from(context)) {
            notify(NOTIFICATION_ID, notificationBuilder.build())
        }
    }
}
```

104.4.1 API de NotificationListener

Como alternativa a `Broadcast Receiver` Android ofrece una API específica para escuchar las notificaciones del sistema: `NotificationListenerService`. Esta API permite a las aplicaciones acceder a las notificaciones publicadas por otras aplicaciones, siempre y cuando el usuario haya concedido explícitamente el permiso necesario.

Ejemplo:

```
class MyNotificationListener : NotificationListenerService() {

    override fun onNotificationPosted(sbn: StatusBarNotification) {
        // Aquí manejas cuando una notificación es publicada. Podrías enviar esta
información a tu UI.
        Log.d("NotificationListener", "Notificación recibida:
${sbn.notification.tickerText}")
    }

    override fun onNotificationRemoved(sbn: StatusBarNotification) {
        // Maneja cuando una notificación es removida.
    }
}
```

y registrar el servicio:

```
<service android:name=".MyNotificationListener"
         android:label="Notification Listener"
         android:permission="android.permission.BIND_NOTIFICATION_LISTENER_SERVICE">
    <intent-filter>
        <action android:name="android.service.notification.NotificationListenerService"
/>
    </intent-filter>
</service>
```

Solicitar permisos al usuario. Los usuarios deben habilitarlo manualmente en la configuración de su dispositivo (Configuración -> Notificaciones -> Acceso a notificaciones). Podrías dirigir a los usuarios a esta configuración mediante un intent:

```
if (!Settings.Secure.getString(contentResolver,
"enabled_notification_listeners").contains(applicationContext.packageName)) {
    // Si no tenemos permiso
    startActivity(Intent("android.settings.ACTION_NOTIFICATION_LISTENER_SETTINGS"))
}
````

Para conectar con el IU usamos un viewMoel

```kotlin

class NotificationViewModel : ViewModel() {
    private val _notifications = MutableStateFlow<String>()
    val notifications: StateFlow<String> = _notifications

    fun postNotification(notification: String) {
        _notifications.value = notification
        // equivalente  _notifications.update { _ -> notification }
    }
}
```

Actualizar el ViewModel desde tu Service:
Necesitarás alguna forma de comunicación entre tu NotificationListenerService y tu ViewModel. Esto podría ser a través de LiveData, usando un BroadcastReceiver, o cualquier otro mecanismo de comunicación intercomponentes en Android.

Un ejemplo muy básico sería registrar un BroadcastReceiver en tu ViewModel y enviar un broadcast desde tu NotificationListenerService:

```
override fun onNotificationPosted(sbn: StatusBarNotification) {
    val intent = Intent("NOTIFICATION_RECEIVED")
    intent.putExtra("notification", "Notificación recibida:
${sbn.notification.tickerText}")
    sendBroadcast(intent)
}
```

Observar en el composable

Observar en Compose UI: En tu UI Compose, podrías observar este LiveData o estado y reaccionar a los cambios mostrando la información de la notificación:

```
@Composable
fun NotificationDisplay(viewModel: NotificationViewModel = viewModel()) {
    val context = LocalContext.current
    val notifications by viewModel.notifications.collectAsState()

    LaunchedEffect(Unit) {
        viewModel.registerReceiver(context)
    }

    Text(text = notifications ?: "No notifications")
}
```

- **Permisos**: Para usar NotificationListenerService, los usuarios deben conceder explícitamente el permiso para acceder a sus notificaciones a través de la configuración del sistema. Esto se hace por razones de privacidad, ya que las notificaciones pueden contener información sensible.
- **Políticas de la Tienda de Aplicaciones**: Si tu aplicación está destinada a ser publicada en Google Play, asegúrate de que el uso de estos servicios cumpla con las políticas de Google Play, especialmente en lo que respecta a la privacidad del usuario.

104.5 Notificaciones PUSH

Las notificaciones push en Android son mensajes que se pueden enviar desde un **servidor** directamente a los dispositivos Android de los usuarios, incluso cuando la aplicación no está abierta o en uso. Estas notificaciones sirven para informar a los usuarios sobre nuevos mensajes, actualizaciones, o para realizar otras acciones específicas que pueden requerir su atención o respuesta.

Cómo funcionan

Para enviar y recibir notificaciones push, tu aplicación Android debe utilizar un servicio de mensajería en la nube, como **Firebase Cloud Messaging (FCM)**, que es el servicio recomendado y proporcionado por Google. El proceso general para enviar una notificación push es el siguiente:

1. Registro del Dispositivo: Primero, tu aplicación en el dispositivo del usuario se registra en FCM (o cualquier otro servicio de mensajería en la nube que estés utilizando) para obtener un token único de registro. Este token identifica la aplicación y el dispositivo para poder recibir mensajes.

2. Enviar el Token al Servidor: Después de obtener el token, tu aplicación lo envía a tu servidor de backend, donde se almacena de manera segura.
3. Enviar Notificaciones desde el Servidor: Cuando quieras enviar una notificación push a los usuarios, tu servidor backend envía un mensaje a FCM, especificando el token del dispositivo (o tokens, si deseas enviar a múltiples dispositivos) junto con el contenido de la notificación.
4. FCM envía la Notificación: FCM recibe el mensaje de tu servidor y lo envía al dispositivo del usuario. Aunque la aplicación no esté en ejecución, el sistema operativo Android recibe el mensaje y muestra la notificación.
5. Interacción del Usuario: Cuando el usuario interactúa con la notificación (por ejemplo, tocándola), puede abrir la aplicación o realizar una acción específica dentro de la aplicación, dependiendo de cómo hayas configurado la notificación

Otros servidores de notificaciones PUSH:

- **OneSignal**: Es una plataforma de notificaciones push muy popular que ofrece una amplia gama de características, incluyendo segmentación avanzada, programación de notificaciones, y análisis en tiempo real. Soporta Android, iOS, y aplicaciones web.
- **Pusher Beams**: Ofrece soluciones de mensajería en tiempo real y es conocido por su facilidad de uso. Pusher Beams permite enviar notificaciones push a iOS, Android, y aplicaciones web, ofreciendo también análisis y herramientas de depuración.
- **Airship** (antes Urban Airship): Es una plataforma de engagement para móviles que proporciona mensajería push, notificaciones in-app, y más. Airship permite una segmentación detallada y personalización de mensajes, ideal para campañas de marketing móvil.
- **Amazon Simple Notification Service (SNS)**: Este es un servicio gestionado por Amazon Web Services que proporciona mensajería masiva para la entrega de mensajes a una gran cantidad de suscriptores, incluyendo soporte para notificaciones push móviles. Es muy escalable y se integra bien con otras soluciones de AWS.

etc...

104.6 Apendice

Enlaces:

- Descripción general en AD
- Ejemplos de

Versión: 0.5 12-12-23