Laboratorio 4.4

Apellidos: Moreno Vera
Nombres: Felipe Adrian
Código: 20120354I
Asignatura: Programación en Dispositivos Móviles (CC481)

2016 - I

Indice

Actividad 1

1. Creamos un proyecto nuevo con su Blank Activity. El xml y código Java de nuestra actividad principal no lo vamos a tocar.
2. Una vez abierto vemos a crear nuestra clase. Para ello creamos una nueva clase Java que herede de BroadcastReceiver. Se pide explicar el código expuesto

```
public class OnChargeReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context ctx, Intent i) {
    Log.d("App", "Recibido!"); // Lanza un mensaje al logCat
    Toast.makeText(ctx, "Ha conectado el cargador.",
        Toast.LENGTH_SHORT).show(); // Lanza un Toast en la App.
  }
}
```

3. En AndroidManifest.xml habrá que dejar constancia de nuestro BroadcastReceiver dentro de la etiqueta application. Comentar dicho código.

```
<receiver android:name=".OnChargeReceiver">
  <intent-filter>
    <action android:name =
      "android.intent.action.ACTION_POWER_CONNECTED"/>
  </intent-filter>
</receiver>
```

Se coloca fuera del application

4. El puerto específico del emulador es el 5554. Para ver dicha funcionalidad seguimos los siguientes pasos:

1. Arrancamos el emulador.
2. Una vez arrancado abrimos una consola de sistema y hacemos un telnet.
Manuel:$ telnet localhost 5554

```
jbot@jLap:~$ telnet localhost 5554
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
Android Console: type 'help' for a list of commands
OK
```

3. Una vez conectado vamos a emular quitar o poner el cargador.

1. Para conectar: power ac of

2. Para conectar: power ac on

```
power ac off
OK
power ac on
OK
```

Actividad 2

1. Creamos un proyecto nuevo.
2. Vamos a complicar el ejemplo anterior utilizando Custom Intent, para ello utilizamos el método sendBroadcast(). Este aspecto quiere decir que cuando se mande el Custom Intent la reacción de nuestro sistema será vibrar.

1.

2. En nuestro Layout principal vamos a implementar un Button con nombre “buttonReceive” que al ser pulsado hará que vibre nuestro dispositivo.

3. En la clase Java añadimos las siguientes propiedades. Defina el significado de Custom Intent.

```
private static final String CUSTOM_INTENT
="dispositivosmóviles.broadcastreceiver.vibracion";
Button btnReceive;
```

4. En nuestra Activity Java principal deberemos definir el Custom Intent. Este aspecto lo que realiza es enviar al Intent al BradcastReceiver mediante el método sendBroadcast() al realizar onClick en “buttonReceive”. Por tanto añadimos al método onCreate() el siguiente código.

```
btnReceive = (Button) findViewById(R.id.buttonReceive);
btnReceive.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    sendBroadcast(new Intent(CUSTOM_INTENT),
        Manifest.permission.VIBRATE);
  }
});
```

3. Creamos una nueva clase Java llamada VibrateReceiver que herede de BroadcastReceiver.

1. Implementamos el método onReceive() de manera que crea un objeto Vibrator que vibrará medio segundo.

```
public class VibrateReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent){
    Vibrator mVibrator = (Vibrator)
context.getSystemService(Context.VIBRATOR_SERVICE);
    mVibrator.vibrate(500);
  }
}
```

4. En AndroidManifest.xml introducimos el permiso de VIBRATE.

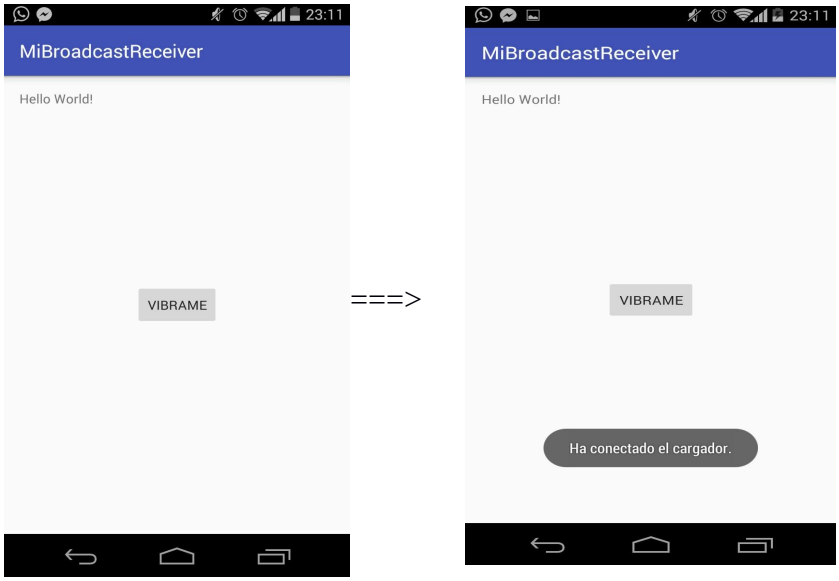
1. Dentro de la etiqueta <uses-permission>

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

2. Añadimos de la etiqueta <receiver>. De ella explique que significa exported

<receiver android:name=”.VibrateReceiver”
android:exported=”false” >
<intent-filter>
<action android:name=
“dispositivosmóviles.broadcastreceiver.vibracion”>
<intent-filter>
</receiver>

La etiqueta exported: “false”, nos dice que el BroadCast puede recibir información de esa aplicación o clase, si es true, recibe info desde cualquier otra aplicación.



Actividad 3

1. Creamos un nuevo proyecto pero no cerramos el anterior ya que vamos a rescatar código de él.

2. En nuestra Main Activity.

1. Creamos nuestra constante CUSTOM_INTENT como realizamos en el ejercicio anterior.

2. Al no declarar en el Manifest el la clase VibrateReceiver y el IntentFilter deberemos declararla aquí. Además declaramos un objeto LocalBroadcastManager, explique brevemente su funcionalidad.

```
private static final String CUSTOM_INTENT
  ="dispositivosmoviles.broadcastreceiver.vibracion";
private final IntentFilter intentFilter = new IntentFilter(CUSTOM_INTENT);
private final VibrateReceiver receiver = new VibrateReceiver();
```

Sirve para ayudar a registrar para y enviar broadcast de intentos a objetos locales con tu proceso. Estos tiene un número de ventajas respecto al envio global de broadcast con senBroadcast(Intent):

- Conoces la data que hace broadcasting no deja la app, entonces no necesitas preocuparte de la data.
- No es posible apra otras aplicaciones enviar broadcast a la tuya, no necesitas preocuparte acerca de temas de seguridad.
- Es más eficiente que evniar broadcasts globales dentro de tu sistema.

3. El siguiente paso es asignar el contexto de nuestra clase a LocalBroadcastManager, con el método getInstance(). Posteriormente, se registra el Receiver pasando el objeto de nuestra clase VibrateReceiver y el IntentFilter.

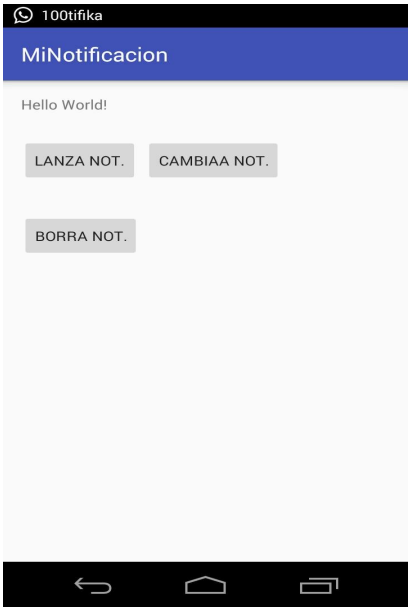
```
MBroadcastMgr =
 LocalBroadcastManager.getInstance(getApplicationContext()),
    mBroadcastMgr.registerReceiver(receiver, intentFilter);
```

4. Al igual que en el ejemplo anterior el Button será el que envíe el Intent al BradcastReceiver con el método sendBroadcast().

Funciona igual que la actividad 2.

Actividad 4

1. Creamos el diseño tal y como se ha expuesto en la imagen.



2. El manifiesto será el código normal de actividad principal que ejecuta, por lo que no deberemos agregar nada.

3. En el xml de Layout en la propiedad onClick le ponemos el nombre a cada Button, por ejemplo, “lanza”, “borra” y “cambia”.

4. En el código Java de mi aplicación principal:

1. Creamos una propiedad de numMessages.

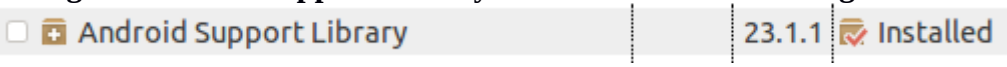
Public int numMessages = 0;

2. Escribimos el siguiente código para el Button “lanza”. Estudie y explique dicho código.

```
public void lanza(View v){      // Funcion Lanza
  NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(this) // Instancia un nuevo objeto
        .setSmallIcon(R.mipmap.ic_launcher).setContentTitle("Mi
notificación")   // le da un icono a mostrarse
        .setContentText("Hola Mundo!"); // Le da texto
  Intent resultIntent = new Intent(this, MainActivity.class); // instancia
intent que redirecciona a la misma main
  PendingIntent resultPendingIntent = PendingIntent.getActivity(this, 0,
resultIntent, PendingIntent.FLAG_UPDATE_CURRENT);
  // este sirve para estar pendiente de un intent
  mBuilder.setContentIntent(resultPendingIntent); // Contiene el intent
```

```
resultante.
  int mNotificationId = 001;
  NotificationManager mNotifyMgr = (NotificationManager)
getSystemService(NOTIFICATION_SERVICE); // gestiona las notificaciones
  mNotifyMgr.notify(mNotificationId, mBuilder.build()); // notifica
}
public void cambia(View v){  // Funcion cambia
  NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
  int notifyID = 1;
  NotificationCompat.Builder mNotifyBuilder =
       new NotificationCompat.Builder(this)
            .setContentTitle("Nuevo Mensaje")
            .setContentText("Tienes mensajes nuevos!")
            .setSmallIcon(R.mipmap.ic_launcher);
  String currentText="Texto";
  mNotifyBuilder.setContentText(currentText).setNumber(+
+numMessages); // Agrega texto al contexto, en este caso el numero de
notificacio que es
  mNotificationManager.notify( notifyID, mNotifyBuilder.build()); // la lanza
}
public void borra(View v){  // funcion borra
  NotificationManager mNotificationManager =
(NotificationManager)getSystemService(Context.NOTIFICATION_SERVICE);
  int notifyID = 1;
  mNotificationManager.cancel(notifyID); // cancela la notificacion
}
```

3. Hay un aspecto muy importante en el código anteriormente visto que es el NotifiationCompact. Para tener la funcionalidad de dicho componente tendremos que tener descargado Android Support Library en Extras del SDK Manager.



Ya la teniamos instalada, el problema era que para agregar el NotificationCompat, la librería es:

```
import android.support.v4.app.NotificationCompat;
```

En vez de v7, eso sucede porque NotificationCompat ha sido introducida después del API nivel 4, y ahora es compatible con superiores.

4. Una vez verificada la versión (columna rev.) abrimos en el código el archivo Build.Gradle. En dependencias deberemos tener:

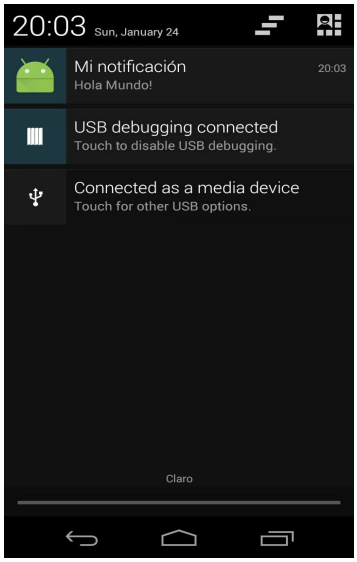
dependencies {
compile fileTree(dir: 'libs', include: ['*.jar'])
compile "com.android.support:appcompat-v7:20.0.+" //añade la versión 20 o superior.
}

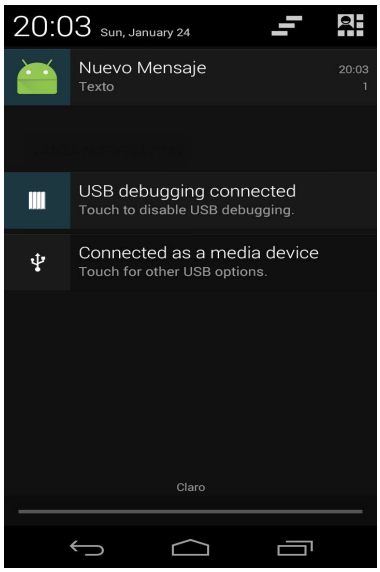
```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'
}
```

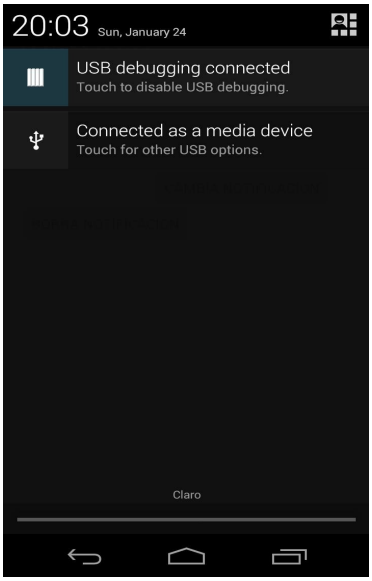
Nuestra dependencia es de 23.1.1 :D

IMPORTANTE:
Tener en cuenta los valores de compileSdkVersion y buildToolsVersión. En mi caso son 20.

5. Comprobamos que funciona correctamente el código. Expón si has tenido algún problema.







6. Si vamos a Home teniendo la notificación activa y pulsamos la notificación desde la barra de tareas vemos que abre nuestra aplicación, ¿cómo se ha producido esto? Identifícalo en el código.

```
Intent resultIntent = new Intent(this, MainActivity.class);
PendingIntent resultPendingIntent = PendingIntent.getActivity(this, 0,
resultIntent, PendingIntent.FLAG_UPDATE_CURRENT);
mBuilder.setContentIntent(resultPendingIntent);
```

Aquí es donde la notificación se encarga de lanzarnos nuestra app, con el intent que es lanzado desde nuestra MainActivitiy(this) y va hacia el MainActivity.class

Link del github con los códigos del laboratorio:

https://github.com/Jenazad/PDM/tree/master/Laboratorio_4

Referencias

http://stackoverflow.com/questions/2169294/how-to-add-manifest-permission-to-android-application
http://developer.android.com/intl/es/guide/topics/manifest/receiver-element.html
http://stackoverflow.com/questions/9425187/broadcastreceiver-declared-in-manifest-is-not-receiving-the-broadcast
http://www.proyectosimio.com/es/programacion-android-broadcastreceiver/
http://stackoverflow.com/questions/13950338/how-to-make-an-android-device-vibrate
http://developer.android.com/intl/es/reference/android/app/PendingIntent.html