Servicios Basados en Localización (LBS)

Tema 6
Desarrollo de aplicaciones en Android

Ramón Alcarria
Miguel Ángel Manso

1

Elementos

- Activity
  - Componente con el cual el usuario interactúa
- Intents
  - Mensajes asíncronos que enlazan Activities, Services y Broadcasts
- Manifest
  - Define ciertos aspectos de la aplicación
- Layouts
  - Define componentes para el GUI (Graphical User Interface), es decir, las vistas

Elementos

- Service
  - Componente que se ejecuta en segundo plano, para realizar tareas periódicas o responder a eventos en el sistema
- Broadcasts
  - Evento del sistema: llamada, encendido, etc. Se puede capturar desde app
- Content Provider
  - Componente que administra el acceso a un conjunto de datos estructurado

Activity

- Representa el componente principal de la interfaz gráfica de una aplicación Android. Se puede pensar en una actividad como el elemento análogo a una ventana en cualquier otro lenguaje visual.
- Una aplicación consiste en múltiples actividades consecutivas.
- Cuando una actividad termina su ejecución porque una nueva comienza, se actualiza su estado a través de callbacks.

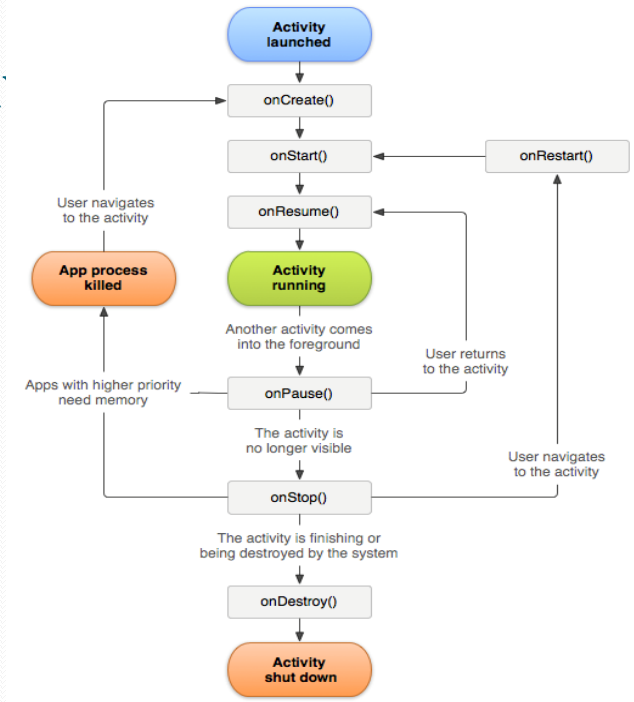
Activity

```
<activity
    android:name=".MainActivity"
    android:label="@string/title_activity_main" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Arranca como actividad inicial

Aparece en el lanzador

Acti



Intents

- Un intent es el elemento básico de comunicación entre los distintos componentes Android descritos anteriormente.
- Son descripciones abstractas de lo que se desea ejecutar
- Mediante un intent se puede mostrar una actividad desde cualquier otra, iniciar un servicio, enviar un mensaje broadcast, iniciar otra aplicación, etc.

Inte

```
public void onClick(View view) {
    int position = spinner.getSelectedItemPosition();
    Intent intent = null;
    switch (position) {
    case 0:
        intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("http://www.marca.com"));
        break;
    case 1:
        intent = new Intent(Intent.ACTION_CALL,
                Uri.parse("tel:(+34)616479034"));
        break;
    case 2:
        intent = new Intent(Intent.ACTION_DIAL,
                Uri.parse("tel:(+34)616479034"));
        startActivity(intent);
        break;
    case 3:
        intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("geo:50.123,7.1434?z=19"));
        break;
    case 4:
        intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("geo:0,0?q=query"));
        break;
    case 5:
        intent = new Intent("android.media.action.IMAGE_CAPTURE");
        break;
    case 6:
        intent = new Intent(Intent.ACTION_VIEW,
                Uri.parse("content://contacts/people/"));
        break;
    }
    if (intent != null) {
        startActivity(intent);
    }
}
```

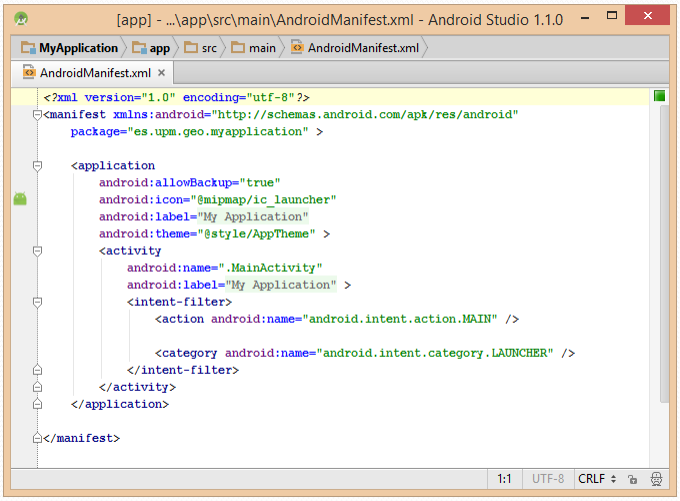
AndroidManifest.xml

- Fichero que describe al SO información esencial sobre la aplicación antes de su ejecución
  - Un nodo por cada uno de los componentes de la app (Activities, Services, Content Providers, and Broadcast Receivers)
  - Intents a los que puede responder la aplicación
  - Metadatos de la aplicación, como por ejemplo el icono
  - Requisitos de seguridad, es decir, qué permisos tiene el usuario que dar a la aplicación (acceso a la agenda, a la red, etc)

AndroidManifest.xml

- Un aspecto sorprendente y relevante de Android es que rompe con el concepto tradicional de aplicación
  - Una aplicación no es más que un conjunto de elementos que si se desea pueden ser utilizados por separado desde otras aplicaciones
- Ver permisos en Android Developers: http://developer.android.com/reference/android/Manifest.permission.html

AndroidManifest.xml



AndroidManifest.xml

- Permisos, ejemplos:
  - <uses-permission android:name="android.permission.INTERNET"></uses-permission>
  - <uses-permission android:name="android.permission.VIBRATE"></uses-permission>
  - <uses-permission android:name="android.permission.CAMERA"></uses-permission>
  - <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
  - <uses-permission android:name="android.permission.NFC" />

Layout

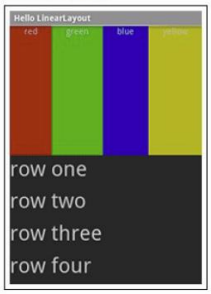
- Elementos
  - TextView: Un texto de sólo lectura. Soporta multilínea, formateo del texto y word wrapping
  - EditText: Cuadro de edición de texto, puede ser multilínea
  - ListView: Un grupo de vistas para mostrar elementos en una lista
  - Spinner: Caja de selección
  - Button
  - CheckBox: Botón de dos estados
  - RadioButton: Botones de dos estados agrupados. Sólo uno activo.

Layout

- Tipos de Layout
  - FrameLayout: El más simple. Cada nueva vista se añade en la esquina superior izquierda. Cada nueva vista tapa la anterior.
  - LinearLayout: Cada nueva vista se añade a continuación, bien verticalmente o bien horizontalmente
  - RelativeLayout: La posición de cada nueva vista se define en relación a las otras y a los límites de la pantalla
  - TableLayout: Las vistas se colocan como una matriz con filas y columnas
  - AbsoluteLayout: La posición de cada vista hija se define en términos de sus coordenadas

Layout

Linear Layout



Relative Layout

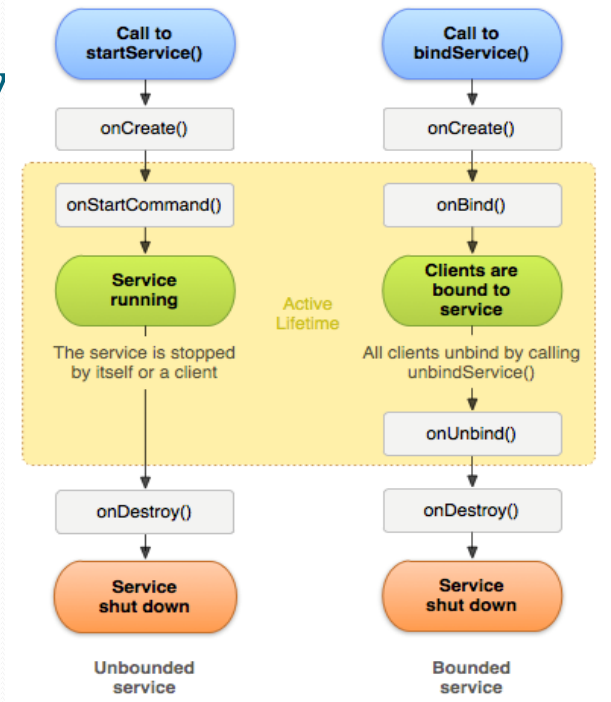
Table Layout

Grid View

Service

- Los servicios son componentes sin interfaz gráfica que se ejecutan en segundo plano. En concepto, son exactamente iguales a los servicios presentes en cualquier otro sistema operativo (de servidores web, impresora, antivirus).
- Los servicios pueden realizar cualquier tipo de acciones, por ejemplo actualizar datos, lanzar notificaciones, o incluso mostrar elementos visuales (activities) si se necesita en algún momento la interacción con el usuario.

Serv



17

Broadcast Receiver

- Un broadcast receiver es un componente destinado a detectar y reaccionar ante determinados mensajes o eventos globales generados por el sistema (por ejemplo: "Batería baja", "SMS recibido", "Tarjeta SD insertada", ...) o por otras aplicaciones.
- Cualquier aplicación puede generar mensajes (intents, en terminología Android) broadcast. Estos mensajes no están dirigidos a una aplicación concreta sino a cualquiera que quiera escucharlos.

Content provider

- Un content provider es el mecanismo que se ha definido en Android para compartir datos entre aplicaciones.
- Mediante estos componentes es posible compartir determinados datos de nuestra aplicación sin mostrar detalles sobre su almacenamiento interno, su estructura, o su implementación.
- De la misma forma, nuestra aplicación podrá acceder a los datos de otra a través de los content provider que se hayan definido.

Fuente: Paco Serradilla

Referencia a recursos

- Referencia a un recurso desde el layout
  - android:text="@string/more" android:src="@drawable/separator"
- Referencia a componentes de un layout desde el controller (el código)
  - En el layout
    - @+id/boton1
    - @+ significa que se crea el id para acceder desde el código
    - @ significa que el recurso se ha definido previamente, o que está en los recursos de la app
- Mas información:
  - http://www.intertech.com/Blog/Post/Android-Layout-and-ID-Attribute.aspx

Acceso desde el código

- Para acceder a un componente definido en un layout, primero debemos identificarlo en el layout con un nombre

```
<org.osmdroid.views.MapView
android:id="@+id/openmapview"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
/>
```

- Después, desde el código, usamos el método findViewById(int resourceID) para obtener una referencia al objeto usando el android:id definido en el XML

```
myOpenMapView = (MapView)findViewById(R.id.openmapview);
```


Acceso desde el código

- Como cambiar el texto de un recurso con una cadena definida en el fichero de strings
  - Obtener el objeto con findViewById

```
TextView texto = (TextView)findViewById(R.id.text1);
```

  - Acceder al string por el ID

```
String mensaje = getResources().getString(R.string.bye_world)
```

  - Cambiar el texto

```
texto.setText(mensaje);
```

Asignar un controlador

- Se puede hacer desde el programa

```
final Button button = (Button) findViewById(R.id.button_id);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Perform action on click
    }
});
```

- O desde el layout

```
<Button
android:id="@+id/button1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_gravity="right"
android:text="@string/more"
android:onClick="doMore" />
```

```
public void doMore(View view) {
Toast.makeText(this, "Hola, controlador", Toast.LENGTH_LONG).show();
}
```


Manejo de Intents

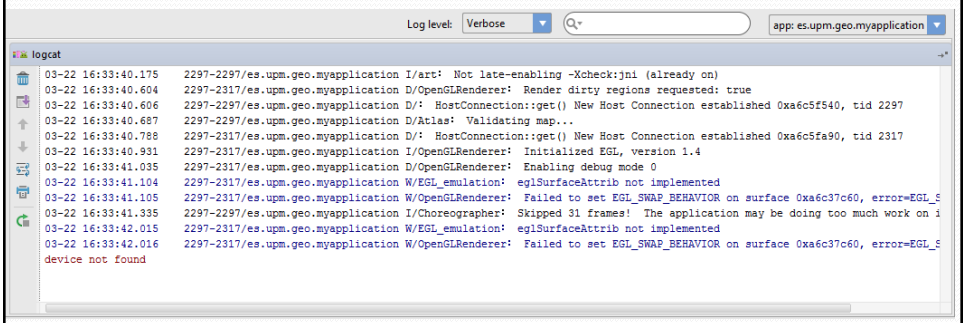
- Los *intents* permiten difundir mensajes en el sistema o hacia una actividad o servicio específico
  - Expresan la "intención" de que se realice una acción
  - Pero no tienen por qué definir quién realizará la acción
    - En estos casos es responsabilidad del sistema decidirlo

```
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

Logging

- Una de las técnicas más útiles a la hora de depurar y/o realizar el seguimiento de aplicaciones sobre cualquier plataforma es la creación de logs de ejecución.
- Android por supuesto no se queda atrás y nos proporciona también su propio servicio y API de logging a través de la clase android.util.Log
- En Android Studio viene abierto por defecto el LogCat, donde se puede ver los logs generados por la aplicación en ejecución

Logging



Logging

- En Android los mensajes de log se van a clasificar por nivel crítico, existiendo así varias categorías (ordenadas de mayor a menor nivel):
  - Error, Warning, Info, Debug, Verbose
- Para cada uno de estos tipos de mensaje existe un método estático independiente que permite añadirlo al log de la aplicación. Así, para cada una de las categorías anteriores tenemos disponibles los métodos e(), w(), i(), d() y v() respectivamente

Logging

```
Log.e(LOGTAG, "Mensaje de error");
Log.w(LOGTAG, "Mensaje de warning");
Log.i(LOGTAG, "Mensaje de información");
Log.d(LOGTAG, "Mensaje de depuración");
Log.v(LOGTAG, "Mensaje de verbose");
```