
	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2019/2020
	UNIDAD COMPETENCIA					

Tema 6

Hardware

Índice

1.	Permisos	1
2.	Vibrador	2
3.	Sensores	4
3.1.	Lista de sensores soportados por un dispositivo	4
3.2.	Acceder a los datos de un sensor	5
3.3.	Sensores comunes en dispositivos en Android	8
4.	Localización	11
4.1.	Android Location API	11
4.2.	Ejemplo de uso de localizaciones	12
5.	Mapas	22
5.1.	Osmdroid	22
5.1.1.	Visualización del mapa	22
5.1.2.	Visualización de la posición actual y su actualización	25
6.	Cámara	31
7.	Ejercicios	35

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2019/2020
	UNIDAD	COMPETENCIA				

1. Permisos

Un punto importante que no hemos visto hasta ahora es el de los permisos. Cuando una aplicación se instala, puede pedir una serie de permisos al usuario: si puede acceder a internet, si puede hacer llamadas, si puede usar el *gps*, ...

Esto es algo obligatorio ya que una aplicación en un dispositivo no puede disponer, sin que el usuario lo acepte, de los elementos protegidos del mismo. Especialmente aquellos que tienen que ver con la privacidad del usuario.

Por tanto para realizar ciertas tareas como hacer una llamada o acceder a internet es necesario indicarlo en el fichero *AndroidManifest.xml* mediante el elemento `<uses-permission>`.

Veamos un ejemplo: Toma una aplicación cualquiera y accede al manifiesto insertando la siguiente línea que permite abrir un socket en la aplicación para conectarse por internet.

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

Su colocación, dentro del fichero *AndroidManifest.xml* es la siguiente:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.permisos">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
    ....
```

Para ver la lista de permisos disponibles se puede consultar el siguiente enlace:

<https://developer.android.com/reference/android/Manifest.permission>

A partir de la versión 6 de Android se requiere que el usuario acepte, en tiempo de ejecución, una serie de permisos que Android considera peligrosos.

La solicitud de permisos provoca la aparición de un dialogo en la que el usuario puede aceptar o rechazar la concesión de los permisos. Cuando se escoge una opción se ejecuta de forma automática el método *onRequestPermissionsResult*. Su funcionamiento es muy similar al método *onActivityResult* que se ejecuta de forma automática tras lanzar una actividad con *startActivityForResult*.



¿Quieres permitir que **Mapa_con_Location** acceda a la ubicación de este dispositivo?

DENEGAR PERMITIR

Veamos un ejemplo de su funcionamiento solicitando los permisos que se suelen utilizar para obtener las localizaciones además del de escritura en disco.

- El primer paso sería añadir estos permisos en el manifiesto.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

- Este paso es opcional. Al igual que cuando lanzamos actividades con *startActivityForResult* la solicitud de permisos usa un *requestCode* para que *onRequestPermissionsResult*, ya que al igual que *onActivityResult*, este método es único y se invoca tras la solicitud de cada uno de los permisos que solicitamos.


El *requestCode* le permite identificar cual es el permiso que estamos solicitando.

Crearemos dos constantes para los *requestCode*:

```
final static int requestCode_Localizacion = 1;
final static int requestCode_Escritura = 2;
```

- Solicitamos los permisos:

```
// Se comprueban si se tiene los permisos necesarios
//ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
if (    checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
    && checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
    // En caso de no tenerlos se solicita al usuario su concesión
    String[] permisos = new String[]{
        Manifest.permission.ACCESS_FINE_LOCATION,
```

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2019/2020
	UNIDAD	COMPETENCIA				

```

Manifest.permission.ACCESS_COARSE_LOCATION
};
ActivityCompat.requestPermissions(MainActivity.this, permisos, requestCode_Localizacion);
} else { // Si ya se tienen los permisos se ejecuta de forma normal
    Toast.makeText(this, "Ya tengo los permisos de localización", Toast.LENGTH_SHORT).show();
}
if (checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
    // En caso de no tenerlos se solicita al usuario su concesión
    String[] permisos = new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE};
    ActivityCompat.requestPermissions(MainActivity.this, permisos, requestCode_Escritura );
} else { // Si ya se tienen los permisos se ejecuta de forma normal
    Toast.makeText(this, "Ya tengo los permisos de escritura", Toast.LENGTH_SHORT).show();
}

```

- Implementar el método `onRequestPermissionsResult` que se ejecuta tras aceptar o rechazar la solicitud de un permiso.

```

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults){
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if (requestCode == requestCode_Localizacion) { // Procede de la solicitud de permisos de localización
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) { // permiso concedido
            Toast.makeText(getApplicationContext(), "Permiso de localización concedido", Toast.LENGTH_LONG).show();
        } else { // Se ha rechazado el permiso
            Toast.makeText(getApplicationContext(), "Permiso localización denegado", Toast.LENGTH_LONG).show();
        }
    } else if (requestCode == requestCode_Escritura) { // Procede de la solicitud de permisos de localización
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) { // permiso concedido
            Toast.makeText(getApplicationContext(), "Permiso de escritura concedido", Toast.LENGTH_LONG).show();
        } else { // Se ha rechazado el permiso
            Toast.makeText(getApplicationContext(), "Permiso escritura denegado", Toast.LENGTH_LONG).show();
        }
    }
}

```

2. Vibrador

Este es un elemento *hardware* muy sencillo de usar y que es muy práctico de cara a realizar realimentación al usuario sobre todo en situaciones en donde el teléfono esté silenciado. También, por supuesto, es muy usado en juegos a la hora de mejorar la experiencia del usuario.

En el siguiente ejemplo veremos un ejemplo sencillo de uso del vibrador. Lo que es necesario resaltar es que se necesitan permisos para usarlo. Para ello se establecerá el siguiente permiso en el fichero *AndroidManifest.xml*.

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


El objeto base para gestionar la vibración de un dispositivo es *Vibrator*¹. Usaremos el siguiente código para instanciarlo:

```
Vibrator vibrator = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE);
```

Para comprobar si el dispositivo disponen de vibrador se dispone del método `mVibrator.hasVibrator()`.

Para explicar el uso del vibrador se verán tres ejemplos: vibrar una vez cierto tiempo, vibrar según cierto patrón y repetir la vibración hasta su cancelación.

¹ <https://developer.android.com/reference/android/os/Vibrator.html>

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2019/2020
	UNIDAD	COMPETENCIA				

En una nueva aplicación añade el permiso anterior y luego asocia al atributo *onClick* (o en un *listener*) de un botón el siguiente código:

```
public void onClickVibracion(View v){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
        vibrator.vibrate(VibrationEffect.createOneShot(300, VibrationEffect.DEFAULT_AMPLITUDE));
    } else {
        vibrator.vibrate(300);
    }
}
```

El código es sencillo de comprender. Tras obtener el acceso al servicio de vibración el método *onClickVibracion* ejecuta una vibración de 300ms. A partir de la api 26 (Oreo) el método *vibrate(tiempoMs)* está deprecado teniendo como atributo, en vez de una duración, un efecto de vibración: *VibrationEffect*². El método *VibrationEffect.createOneShot(long vibrateTimeInMills, int amplitude)* es equivalente al anterior. Para la amplitud usaremos el valor por defecto de Android: *VibrationEffect.DEFAULT_AMPLITUDE*.

Además de reproducir un sonido se puede crear un patrón de vibraciones representado como un *Array* de *longs*.

Donde:

- El primer valor indica el tiempo de espera antes de empezar a sonar.
- En el resto de valores impares indican tiempos de vibración mientras que los valores pares determinan tiempos de silencio.

Por ejemplo:

```
long[] patron = { 10, // Tiempo de espera para comenzar la vibración, un valor 0 indica un comienzo inmediato
                  1000, 500, 40, 15, 100, 500 }; // Alterna tiempo de vibración, tiempo de pausa
```

Al igual que en ejemplo anterior, dependiendo de la versión de la *Api*, usaremos uno de los siguientes métodos:

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    vibrator.vibrate(VibrationEffect.createWaveform(patron, -1)); // -1=No repetir; 0=Repetir desde el principio, ...
} else {
    vibrator.vibrate(patron, -1); // -1=No repetir; 0=Repetir desde el principio 1= repetir desde el primer elemento, 2= ...
}
```

El segundo parámetro indica la repetición del patrón. Ejemplos de valores son:

- -1: el patrón solo se reproduce una vez.
- 0: el patrón se repite de forma indefinida desde el principio.
- 1: el patrón se reproduce una vez y luego se repite de forma indefinida desde el valor 1: en el ejemplo 1000.
- 2: el patrón se reproduce una vez y luego se repite de forma indefinida desde el valor 2: en el ejemplo 500.
- 3: el patrón se reproduce una vez y luego se repite de forma indefinida desde el valor 3: en el ejemplo 40.
-

El código anterior reproduce el patrón una vez: se espera 10ms antes de comenzar a vibrar luego vibra durante un segundo y para medio segundo, luego vibra durante 40ms y para 15ms, vibra 100ms y para otro medio segundo.

Además a la nueva versión de *vibrate* se le puede añadir un segundo parámetro un objeto de tipo *AudioAttributes*³ que permite encapsular información sobre la pista de audio.

Por ejemplo *AudioAttributes.USAGE_ALARM* puede usarse con vibraciones asociadas con alarmas o *AudioAttributes.USAGE_NOTIFICATION_RINGTONE* asociado con vibraciones asociadas a llamadas entrantes.

Un ejemplo de su funcionamiento sería:


```
vibrator.vibrate (
    VibrationEffect.createOneShot(200,VibrationEffect.DEFAULT_AMPLITUDE), new AudioAttributes.Builder().
    setUsage(AudioAttributes.USAGE_MEDIA). setContentType(AudioAttributes.CONTENT_TYPE_MUSIC).build() );
```

En cualquier momento se puede usar el método *cancel* para detener la vibración, por ejemplo desde un botón.

```
vibrator.cancel();
```

² <https://developer.android.com/reference/android/os/VibrationEffect>

³ <https://developer.android.com/reference/android/media/AudioAttributes.html>

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2019/2020
	UNIDAD	COMPETENCIA				

3. Sensores⁴

Los dispositivos móviles vienen hoy en día con múltiples sensores: acelerómetro, giróscopos, brújulas, luz, temperatura, proximidad, ...

Android los clasifica en tres categorías:

- Sensores de movimiento (*Motion Sensors*): miden fuerzas de aceleración y rotación en tres ejes. Entre otros tenemos: acelerómetros, sensores de gravedad, giroscopios, ...
- Sensores ambientales (*Environmental sensors*): miden valores ambientales como pueden ser temperatura, humedad, iluminación, presión del aire, ... Los sensores implicados son: barómetros, fotómetros, termómetros, ...
- Sensores de posición (*Position sensors*): permiten establecer la posición física del dispositivo. Esta categoría incluye sensores de orientación y magnetómetros.

Se debe tener en cuenta que no todos los dispositivos tienen todos los sensores y que incluso algunas marcas tienen sus propias *APIs* para sus propios sensores, esto provoca que en ocasiones sea más complejo de programar este *hardware* debido a la fragmentación (o directamente que no funcione en algunos terminales).

Además, de acuerdo a su naturaleza, podemos encontrar dos tipos de sensores:

- *Hardware*: son componentes físicos integrados en un dispositivo. Sus datos se obtienen directamente desde un sensor físico.
- *Software*: no son componentes *hardware* específicos sino que usan uno o más sensores *hardware*. Dos ejemplos de estos sensores son el sensor de aceleración lineal y el sensor de gravedad.

Para ver los tipos de sensores soportados en Android ver el apéndice 1:

Todo el *framework* de los sensores en *Android* se apoya en las siguientes cuatro clases:

- *SensorManager*⁵: se usa para crear una instancia del servicio del sensor, registrar y anular el registro de los sensores.
- *Sensor*⁶: representa un sensor específico.
- *SensorEvent*⁷: proporciona información sobre un evento de un sensor: datos del sensor en bruto, tipo de sensor que generó el evento, la precisión de los datos y la marca de tiempo del evento.
- *SensorEventListener*⁸: interfaz para recibir notificaciones, eventos del sensor, cuando cambian los valores del sensor o cuando cambia la precisión del sensor.

3.1. Lista de sensores soportados por un dispositivo

Para empezar es interesante saber cuáles son los sensores de los que dispone un dispositivo. Mediante el siguiente ejemplo obtenemos un listado de los sensores disponibles. Para ello se usará la clase *SensorManager*.

En una nueva aplicación, añade un *NestedScrollView* (por si la lista es muy larga) y dentro un *TextView*.

El *onCreate* de la *Activity* principal se puede introducir un código similar al siguiente:

```
SensorManager sensorManager;
Sensor sensor;

@Override
protected void onCreate(Bundle savedInstanceState) {
```

Tipo:1: android.sensor.accelerometer
Nombre: BMI120 Accelerometer
Máximo. rango: 156.9064
Mínimo tiempo entre lecturas(ms): 2500
Resolución: 0.0023956299

Tipo:35: android.sensor.accelerometer_uncalibrated
Nombre: BMI120 Accelerometer Uncalibrated
Máximo. rango: 156.9064
Mínimo tiempo entre lecturas(ms): 2500
Resolución: 0.0023956299

Tipo:2: android.sensor.magnetic_field
Nombre: AK09918 Magnetometer
Máximo. rango: 4911.9995
Mínimo tiempo entre lecturas(ms): 20000
Resolución: 0.14953613

Tipo:14: android.sensor.magnetic_field_uncalibrated
Nombre: AK09918 Magnetometer Uncalibrated
Máximo. rango: 4911.9995
Mínimo tiempo entre lecturas(ms): 20000
Resolución: 0.14953613

Tipo:4: android.sensor.gyroscope
Nombre: BMI120 Gyroscope
Máximo. rango: 34.906586
Mínimo tiempo entre lecturas(ms): 2500
Resolución: 0.0010681152


⁴ https://developer.android.com/guide/topics/sensors/sensors_overview.html

⁵ <https://developer.android.com/reference/android/hardware/SensorManager.html>

⁶ <https://developer.android.com/reference/android/hardware/Sensor.html>

⁷ <https://developer.android.com/reference/android/hardware/SensorEvent.html>

⁸ <https://developer.android.com/reference/android/hardware/SensorEventListener.html>

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2019/2020
	UNIDAD	COMPETENCIA				

```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
TextView tv=findViewById(R.id.textView);

sensorManager=(SensorManager) getSystemService(Context.SENSOR_SERVICE);
List<Sensor> sensores = sensorManager.getSensorList(Sensor.TYPE_ALL);
for (Sensor s: sensores){
    tv.append("Tipo:" + s.getType() + ": " + s.getStringType()
        + "\nNombre: " + s.getName()
        + "\nMáximo. rango: " + s.getMaximumRange()
        + "\nMínimo tiempo entre lecturas(ms): " + s.getMinDelay()
        + "\nResolución: " + s.getResolution() + "\n\n");
}
}

```

Un ejemplo de ejecución del programa anterior se puede ver en la imagen de la derecha. En ella se muestran los tipos de sensores soportados por el dispositivo en concreto. Si solo se quiere mostrar un tipo de sensor en concreto se debe cambiar el parámetro `Sensor.TYPE_ALL` por el tipo que se desea mostrar: `TYPE_LINEAR_ACCELERATION`, `TYPE_GRAVITY`, ...

3.2. Acceder a los datos de un sensor

Para obtener los datos proporcionado por un sensor necesitaremos usar la interfaz `SensorEventListener` e implementar los dos siguientes métodos:

- `onSensorChanged`: indica que un sensor a generado un nuevo valor como un objeto `SensorEvent`.
- `onAccuracyChange`: en el sensor se ha producido un cambio de precisión, no lo usaremos.

El siguiente ejemplo mostrar cómo usar el sensor `TYPE_LIGHT` que controla la claridad ambiental. Sus valores se insertarán un `textView` que hemos definido en el `Layout`.

```

public class MainActivity extends AppCompatActivity implements SensorEventListener {
    SensorManager sensorManager;
    Sensor sensorLuz;
    TextView tvLuz;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tvLuz=findViewById(R.id.textView);
        sensorManager = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        sensorLuz = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
    }

    @Override
    public final void onSensorChanged(SensorEvent sensorEvent) {
        // El sensor de luz devuelve un único valor, otros sensores pueden devolver más valores.
        float luz = sensorEvent.values[0];
        tvLuz.setText(String.valueOf(luz));
    }

    @Override
    public final void onAccuracyChanged(Sensor sensor, int accuracy) { // No lo usamos
    }

    @Override
    protected void onResume() { // Antes de usar el sensor hay que registrarlo
        super.onResume();
        if (sensorLuz!=null) {
            sensorManager.registerListener(this, sensorLuz, SensorManager.SENSOR_DELAY_NORMAL);
        }
    }
}

```


	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2019/2020
	UNIDAD	COMPETENCIA				

```

@Override
protected void onPause() { // Cuando se deja de usar el sensor hay que desregistrarlo
    super.onPause();
    if (sensorLuz!=null) sensorManager.unregisterListener(this);
}
}

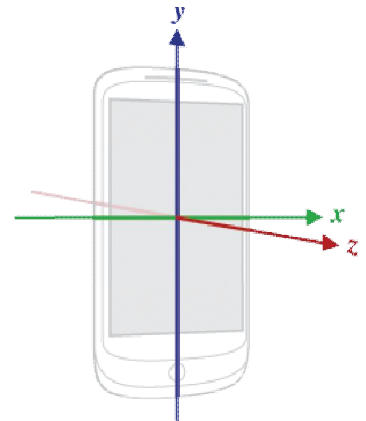
```

Donde las líneas:

- `sensorLuz = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);`
Se obtiene el sensor por defecto para un sensor en específico. En este caso un sensor de tipo *Light*.
- `float luz = event.values[0];`
Se obtiene el valor que devuelve el sensor (en este caso es solo un valor) a través del *array* de *floats* *values*.
- `sensorManager.registerListener(this, sensorLuz, SensorManager.SENSOR_DELAY_NORMAL);`
Se registra el sensor para su uso. Sus parámetros son:
 - 1: Clase que implementa la interfaz *SensorEventListener*.
 - 2: Sensor que queremos registrar.
 - 3: Periodo de muestreo. A periodos más cortos más uso de batería. Tiene que ser uno de los siguientes valores:
 - ❖ `SENSOR_DELAY_NORMAL`: 200 microsegundos
 - ❖ `SENSOR_DELAY_UI`: 60 microsegundos
 - ❖ `SENSOR_DELAY_GAME`: 20 microsegundos
 - ❖ `SENSOR_DELAY_FASTEST`: 0 microsegundos
- `sensorManager.unregisterListener(this);`
Se libera el sensor cuando no se usa para liberar recursos y evitar un gasto innecesario de batería, ya que los sensores consumen bastante.

El sistema de coordenadas es relativo a la pantalla del dispositivo en su orientación por defecto (en los móviles suele ser vertical y en las *tablets* suele ser horizontal) y aunque el dispositivo se gire estos nos cambian. Si se quiere modificar el sistema de coordenadas se debe usar [getRotation](#) para determinar la rotación de la pantalla y luego [remapCoordinateSystem](#) para mapear el sistema de coordenadas a las coordenadas de la pantalla.

- El eje X es horizontal y apunta de izquierda a derecha.
- El eje Y es vertical y apunta de abajo hacia arriba.
- El eje Z apunta desde la parte de atrás de la pantalla hacia la parte frontal atravesándola. Las coordenadas en la parte posterior de la pantalla tienen valores negativos de Z.



Dependiendo del tipo de sensor la información que nos proporciona *SensorEvent*, a través de *values*, difiere.


Vamos a modificar el ejemplo anterior para, además de visualizar la información del sensor de claridad, ver también la información del sensor *Gravity*.

```

public class MainActivity extends AppCompatActivity implements SensorEventListener{
    SensorManager sensorManager;
    Sensor sensorLuz, sensorGravedad;
    TextView tvLuz, tvGravedad;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tvLuz=findViewById(R.id.textView);
        tvGravedad=findViewById(R.id.textView2);
    }
}

```

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2019/2020
	UNIDAD	COMPETENCIA				

```

        sensorManager=(SensorManager) getSystemService(Context.SENSOR_SERVICE);
        sensorLuz = sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);
        sensorGravedad = sensorManager.getDefaultSensor(Sensor.TYPE_GRAVITY);
    }

    @Override
    public final void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    @Override
    public final void onSensorChanged(SensorEvent sensorEvent) {
        synchronized (this) {
            if (sensorEvent.sensor==sensorLuz){
                float luz = sensorEvent.values[0];
                tvLuz.setText(String.valueOf(luz));
            }

            if (sensorEvent.sensor==sensorGravedad){
                tvGravedad.setText(String.format("x: %f\t y: %f\t z: %f", sensorEvent.values[0], sensorEvent.values[1], sensorEvent.values[2]));
            }
        }
    }

    @Override
    protected void onResume() { // Antes de usar el sensor hay que registrarlo
        super.onResume();
        if (sensorLuz!=null) { // si el sensor existe lo registro
            sensorManager.registerListener(this, sensorLuz, SensorManager.SENSOR_DELAY_NORMAL);
        }
        if (sensorGravedad!=null) { // si el sensor existe lo registro
            sensorManager.registerListener(this, sensorGravedad, SensorManager.SENSOR_DELAY_NORMAL);
        }
    }

    @Override
    protected void onPause() { // Cuando se deja de usar el sensor hay que desregistrarlo
        super.onPause();
        if (sensorLuz!=null) sensorManager.unregisterListener(this, sensorLuz); // si el sensor existe lo libero
        if (sensorGravedad!=null) sensorManager.unregisterListener(this, sensorGravedad); // si el sensor existe lo libero
    }
}

```

Cabe destacar el uso del comando *synchronized* cuyo objetivo es que si hay varios *threads* (de servicios u otras aplicaciones) que se ejecutan en paralelo con el nuestro no haya conflictos. Es equivalente al *lock* de *.Net*.

Para ver información detallada de todos los tipos de sensores se puede consultar las siguientes enlaces:

- Sensores de movimiento:
https://developer.android.com/guide/topics/sensors/sensors_motion.html
- Sensores de posición:
https://developer.android.com/guide/topics/sensors/sensors_position.html
- Sensores ambientales:
https://developer.android.com/guide/topics/sensors/sensors_environment.html
- Los valores devueltos por cada sensor se pueden consultar en (columna Índices devueltos en la tabla del punto siguiente):
<https://developer.android.com/reference/android/hardware/SensorEvent#values>

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles			CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD COMPETENCIA					

3.3. Sensores comunes en dispositivos en Android


Id Tipo	Tipo	Nombre tipo: Const. Su valor es: "STRING_" + Tipo	Sensor de: HW/SF	Que Mide	Usos	Índices devueltos en: event.values
-1	TYPE_ALL			Constante para todos los sensores		
1	TYPE_ACCELEROMETER	android.sensor.accelerometer	Movimien. Hardware	Fuerza de aceleración en m/s^2 que se aplica a un dispositivo en los tres ejes físicos: x, y, z incluida la fuerza de la gravedad	Detección de movimiento: sacudida (<i>shake</i>), inclinación (<i>tilt</i>), etc.	Fuerza de aceleración, en los ejes: x, y, z \rightarrow 0, 1, 2 Para trabajar con este sensor se deben filtrar valores que afectan a su precisión. Ver enlace .
2	TYPE_MAGNETIC_FIELD	android.sensor.magnetic_field	Posicional Hardware	Campo magnético terrestre para los tres ejes físicos: x, y, z en μT (micro-Tesla).	Brújula y uso con <i>smart covers</i> (fundas de cierre magnético)	Fuerza del campo magnético terrestre en los ejes x, y, z \rightarrow 0, 1, 2
3	TYPE_ORIENTATION	android.sensor.orientation	Posicional Software	Grados de rotación que hace un dispositivo alrededor de los tres ejes físicos (x, y, z). Fue deprecado en el API Level 8. En ángulos. Usar en vez de este sensor el método de SensorManager getOrientation (ver enlace) que devuelve los mismos valores.	Determina la posición del dispositivo	Acimut: áng. alrededor del eje Z \rightarrow 0 Inclinación: áng. alrededor del eje X \rightarrow 1 Rotación: áng. alrededor del eje y \rightarrow 2
4	TYPE_GYROSCOPE	android.sensor.gyroscope	Movimien. Hardware	Tasa de rotación de un dispositivo en radianes/segundo alrededor de cada uno de los tres ejes físicos: x, y, z.	Detección de rotación (<i>spin</i> , <i>turn</i> , etc.).	Tasa de rotación, en los ejes: x, y, z \rightarrow 0, 1, 2 Para trabajar con este sensor se deben filtrar valores que afectan a su precisión. Ver enlace .
5	TYPE_LIGHT	android.sensor.light	Ambiental Hardware	Nivel de luz ambiental en lux (wikipedia).	Controlar el brillo de la pantalla.	Iluminación \rightarrow 0
6	TYPE_PRESSURE	android.sensor.pressure	Ambiental Hardware	Presión del aire en hPa (hectopascal) o mbar (milibar).	Monitoreo de cambios de presión de aire	Presión del aire \rightarrow 0
7	TYPE_TEMPERATURE	android.sensor.temperature	Ambiental Hardware	Temperatura del dispositivo en $^{\circ}C$. Fue sustituido, en API Level 14, por el sensor TYPE_AMBIENT_TEMPERATURE.	Monitoreo de temperatura	Temperatura \rightarrow 0
8	TYPE_PROXIMITY	android.sensor.proximity	Posicional Hardware	Proximidad de un objeto en cm en relación con la pantalla de visualización de un dispositivo. Algunos sensores solo	Distancia desde el objeto.	Distancia \rightarrow 0

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desenvolvimiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles			CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD COMPETENCIA					

				devuelven dos valores: para cerca y lejos.		
9	TYPE_GRAVITY	android.sensor.gravity	Movimien. Software o Hardware	Fuerza y dirección de la gravedad en m/s^2 que se aplica a un dispositivo en los tres ejes físicos: x, y, z.	Detección de sacudidas (<i>shake</i>), inclinación (<i>tilt</i>), etc.	Fuerza de la gravedad, en los ejes: x, y, z \rightarrow 0, 1, 2
10	TYPE_LINEAR_ACCELERATION	android.sensor.linear_acceleration	Movimien. Software o Hardware	Fuerza de aceleración en m/s^2 que se aplica a un dispositivo en los tres ejes físicos: x, y, z, excluyendo la fuerza de la gravedad.	Monitorear la aceleración a lo largo de un solo eje.	Fuerza de la aceleración, en los ejes sin incluir la gravedad: x, y, z \rightarrow 0, 1, 2
11	TYPE_ROTATION_VECTOR	android.sensor.rotation_vector	Movimien. Software o Hardware	Orientación de un dispositivo al proporcionar los tres elementos del vector de rotación del dispositivo más un ángulo.	Detección rotación y de movimiento.	Componente del vector de rotación en el los ejes x, y, z \rightarrow 0, 1, 2 C. escalar vector de rot. \rightarrow 3 (Api +18) Precisión estimada (rad) \rightarrow 4 (Api +18)
12	TYPE_RELATIVE_HUMIDITY	android.sensor.relative_humidity	Ambiental Hardware	Humedad ambiental relativa en porcentaje (%).	Monitoreo del punto de rocío y de humedad.	Humedad \rightarrow 0
13	TYPE_AMBIENT_TEMPERATURE	android.sensor.ambient_temperature	Ambiental Hardware	Temperatura ambiente del aire. (°C)	Monitoreo de la temperatura del aire	Temperatura \rightarrow 0
14	TYPE_MAGNETIC_FIELD_UNCALIBRATED	android.sensor.magnetic_field_uncalibrated	Posicional	Campo magnético terrestre para los tres ejes físicos: x, y, z sin tener en cuenta las distorsiones producidas por metales cercanos. En μT .	Brújula y uso con <i>smart covers</i> (fundas de cierre magnético)	Fuerza del campo magnético con distorsiones en los ejes x, y, z \rightarrow 0, 1, 2 Fuerza del campo magnético con eliminación de distorsiones estimadas en los ejes x, y, z \rightarrow 3, 4, 5
15	TYPE_GAME_ROTATION_VECTOR	android.sensor.game_rotation_vector	Posicional	Igual que TYPE_ROTATION_VECTOR pero no usa el campo magnético. El eje Y apunta a su propia referencia y no al norte.	Rotaciones relativas más precisas que con TYPE_ROTATION_VECTOR	Componente del vector de rotación en el los ejes x, y, z \rightarrow 0, 1, 2
16	TYPE_GYROSCOPE_UNCALIBRATED	android.sensor.gyroscope_uncalibrated	Movimien.	Igual que TYPE_GYROSCOPE pero sin calibrar.	Igual que TYPE_GYROSCOPE	Tasa de rotación, sin deriva, en los ejes: x, y, z \rightarrow 0, 1, 2 Tasa de rotación, con deriva estimada, en los ejes: x, y, z \rightarrow 3, 4, 5
17	TYPE_SIGNIFICANT_MOTION	android.sensor.significant_motion	Movimien.	Permite despertar el dispositivo y ejecutar código cuando se detecta un evento significativo.	Ejecutar código ante un evento.	No devuelve nada
18	TYPE_STEP_DETECTOR	android.sensor.step_detector	Movimien.	Lanza un evento cada vez que se detecta un paso con una latencia inferior a 2s.		Solo devuelve 1 \rightarrow 0,
19	TYPE_STEP_COUNTER	android.sensor.step_counter	Movimien. Hardware	Número de pasos desde el último reinicio. Tiene más latencia y más precisión que STEP_DETECTOR. No se	Pasos	Número de pasos \rightarrow 0

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles			CURSO:	2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD COMPETENCIA					

				debe desregistrar.		
20	TYPE_GEOMAGNETIC_ROTATION_VECTOR	android.sensor.geomagnetic_rotation_vector	Posicional Software o Hardware	Igual que TYPE_ROTATION_VECTOR pero consume menos batería porque es menos preciso.	Detección rotación y de movimiento.	Componente del vector de rotación en el los ejes x, y, z → 0, 1, 2
21	TYPE_HEART_RATE	android.sensor.heart_rate		Latidos del corazón de la persona tocando el dispositivo.	Obtener los latidos del corazón por minuto	PPM → 0
22	TYPE_TILT_DETECTOR	android.sensor.tilt_detector		Sin documentar. Puede que devuelva 1 cuando se detecta una inclinación.		Valor → 0
28	TYPE_POSE_6DOF	android.sensor.pose_6dof		Similar a TYPE_ROTATION_VECTOR pero con 6 grados de libertad.		Ver enlace para sus valores
29	TYPE_STATIONARY_DETECT	android.sensor.stationary_detect		Detecta si el dispositivo ha estado en reposo durante al menos 5 segundos y hasta un máximo de 10.	Ver si ha cumplido con el tiempo de reposo	Solo devuelve 1 → 0
30	TYPE_MOTION_DETECT	android.sensor.motion_detect		Detecta si el dispositivo ha estado en movimiento durante al menos 5 segundos y hasta un máximo de 10.	Ver si ha cumplido con el tiempo de movimiento	Solo devuelve 1 → 0
31	TYPE_HEART_BEAT	android.sensor.heart_beat		Lanza un evento cada vez que detecta un latido.	Latido detectado con una probabilidad	Probabilidad de latido entre 0 y 1 → 0 0: más improbable ... 1: seguro
34	TYPE_LOW_LATENCY_OFFBODY_DETECT	android.sensor.low_latency_offbody_detect		Detecta cuando el dispositivo se mueve del cuerpo hacia fuera de este y viceversa. Por ejemplo sale o se introduce en un bolsillo.	El usuario guarda o coge el dispositivo de su cuerpo	Estado → 0 Valor 0: dispositivo fuera de cuerpo Valor 1: dispositivo guardado en el cuerpo.
35	TYPE_ACCELEROMETER_UNCALIBRATED	android.sensor.accelerometer_uncalibrated	Movimiento Hardware	Igual que TYPE_ACCELEROMETER pero sin calibrar.	Igual que TYPE_ACCELEROMETER	Fuerza de aceleración, sin inclinación, en los ejes: x, y, z → 0, 1, 2 Fuerza de aceleración, con inclinación estimada, en los ejes: x, y, z → 3, 4, 5

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

4. Localización

En este apartado aprenderemos a usar un importante elemento hardware que traen hoy en día la mayoría de dispositivos móviles: el *GPS*.

Como sabrás, este sistema permite al dispositivo recibir datos de satélites *GPS* para hacer una estimación de su coordenada mediante su latitud y su longitud. Además recibiremos la hora, *timestamp*, en que se hizo la medida y opcionalmente otros datos como altitud, velocidad, precisión, etc...

El problema que presenta esto es que *GPS* no está siempre presente (debe haber visión directa hacia los satélites) y además consume mucha batería, por tanto, los dispositivos también permiten obtener el su posición mediante la conexión *wifi* a la que está conectada o por las torretas de telefonía móvil a las que se conectan.

Estos métodos son menos precisos pero consumen menos batería y suelen estar más disponibles, por ejemplo en interiores. Además como el *GPS* suele ser un método más lento para realizar una primera localización se suele usar para afinar la posición que se obtiene por uno de los otros método.

Android dispone además de otro método de obtener la posición: el método pasivo. Consiste en que una aplicación recibe cambios en el *GPS* cuando otras aplicaciones piden el posicionamiento del dispositivo. Apenas se usa pues tiene el riesgo de no recibir ningún posicionamiento si no hay otras aplicaciones que lo soliciten.

También hay que tener en cuenta que la localización es independiente de los mapas, aunque en muchas ocasiones se usen juntos. Así por un lado tenemos los servicios de localización, que nos dan datos de posición y por otro los de presentación en mapas que además pueden ser muy diversos: *Google Maps*, *OpenStreetMaps*, *Bing maps*, ...

Se puede ver más variedad de mapas, con sus *APIs*, en el siguiente:

<https://stackoverflow.com/questions/4151593/alternatives-to-google-maps-api#4233638>

Esto es importante porque implica que la aplicación en desarrollo, aunque sea para *Android*, no es imprescindible realizarla sobre la *API* de *Google maps*. Comenzaremos viendo cómo se usa el localizador del dispositivo para obtener las coordenadas de su localización.

4.1. Android Location API

Las clases que vamos a utilizar para los servicios de localización son las siguientes (estos datos también se podrían obtener usando para ello los servicios de *Google Play*: <https://developer.android.com/training/location>):

- *LocationManager*⁹: Gestor general de posicionamiento. Indica que proveedores hay disponibles: *gps*, *wifi*, red móvil, última posición guardada, ...
- *LocationProvider*¹⁰: Abstrae las distintas fuentes de datos a una única clase.
- *Location*¹¹: representa una ubicación geográfica. Una ubicación puede consistir en una latitud, longitud, marca de tiempo y otra información como rumbo, altitud y velocidad.
- *Criteria*¹²: Clase que permite especificar características deseadas por nuestra aplicación a la hora de realizar geoposicionamiento. Se pueden establecer o conocer elementos como la precisión, si disponemos de lectura de altitud, requisitos de potencia para consumo de batería, ... Los requerimientos de posicionamiento disponibles son:
 - *ACCURACY_LOW*: precisión baja.
 - *ACCURACY_COARSE*: precisión aproximada.
 - *ACCURACY_MEDIUM*: precisión media.
 - *ACCURACY_HIGH*: precisión alta.
 - *ACCURACY_FINE*: precisión más fina.


Dependiendo de esos criterios se pueden usar distintos proveedores o se configura el dispositivo de distinta forma a la hora de hacer la lectura.

⁹ <https://developer.android.com/reference/android/location/LocationManager>

¹⁰ <https://developer.android.com/reference/android/location/LocationProvider>

¹¹ <https://developer.android.com/reference/android/location/Location.html>

¹² <https://developer.android.com/reference/android/location/Criteria.html>

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

Y además usaremos el interfaz *LocationListener*¹³ el cual dispone de una serie de funciones *callback* que son llamadas ante determinados cambios en el sistema de localización.

4.2. Ejemplo de uso de localizaciones

Veamos el uso de estas clases mediante un ejemplo que nos dé en tiempo real nuestra localización.

- Crea una nueva aplicación denominada Posicionamiento.
- Añadir, en el manifiesto, los permisos para acceder a los ajustes de posicionamiento:

- Si queremos que solo use para posicionarse los *GPS*:

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

- Si queremos que use para posicionarse tanto los *GPS* como con la red móvil:

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


- Añadir las siguientes cadenas en el fichero *Strings.xml*.

```
<string name="app_name">GPS</string>
<string name="latitud">Latitud:</string>
<string name="longitud">Longitud:</string>
<string name="altitud">Altitud:</string>
<string name="proveedor">Proveedor:</string>
<string name="precision">Precisión:</string>
<string name="proveedores_activos">P. activos:</string>
<string name="hora">Hora:</string>
<string name="velocidad">Velocidad:</string>
<string name="configuracion_proveedor">Configurar\nproveedor</string>
<string name="nueva_petition">Nueva\npetición</string>
<string name="vermapa">Ver en\nmapa</string>
<string name="get_location">Obteniendo localización</string>
<string name="permiso_si">Permiso concedido</string>
<string name="permiso_no">Permiso denegado</string>
```

- Añadimos los siguientes *TextViews* en el *layout* para mostrar, en tiempo real, información sobre diversos datos del posicionamiento del dispositivo. Además añadimos una barrera para una mejor distribución de estos elementos.

```
<TextView
    android:id="@+id/textPrecision"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textAltitud"
    android:layout_alignStart="@+id/textAltitud"
    android:layout_marginStart="4dp"
    android:layout_marginTop="4dp"
    android:text="@string/precision"
    android:textAppearance="?android:attr/textAppearanceMedium"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
<TextView
    android:id="@+id/precision"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    android:background="#B5F8CCCC"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#3F51B5"
```

¹³ <https://developer.android.com/reference/android/location/LocationListener>

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

app:layout_constraintBottom_toBottomOf="@+id/textPrecision"
app:layout_constraintStart_toEndOf="@+id/textProveedor"
app:layout_constraintTop_toTopOf="@+id/textPrecision" />

```

```
<TextView
```

```

    android:id="@+id/textAltitud"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textLongitud"
    android:layout_alignStart="@+id/textLongitud"
    android:layout_marginStart="4dp"
    android:layout_marginTop="4dp"
    android:text="@string/altitud"
    android:textAppearance="?android:attr/textAppearanceMedium"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textPrecision" />

```

```
<TextView
```

```

    android:id="@+id/altitud"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="#B5F8CCCC"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#3F51B5"
    app:layout_constraintBottom_toBottomOf="@+id/textAltitud"
    app:layout_constraintStart_toStartOf="@+id/proveedor"
    app:layout_constraintTop_toTopOf="@+id/textAltitud" />

```

```
<TextView
```

```

    android:id="@+id/textProveedor"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textPrecision"
    android:layout_alignStart="@+id/textPrecision"
    android:layout_marginStart="4dp"
    android:layout_marginTop="4dp"
    android:text="@string/proveedor"
    android:textAppearance="?android:attr/textAppearanceMedium"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textAltitud" />

```

```
<TextView
```

```

    android:id="@+id/proveedor"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    android:background="#B5F8CCCC"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#3F51B5"
    app:layout_constraintBottom_toBottomOf="@+id/textProveedor"
    app:layout_constraintStart_toEndOf="@+id/textProveedor"
    app:layout_constraintTop_toTopOf="@+id/textProveedor" />


```

```
<TextView
```

```

    android:id="@+id/textSpeed"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textProveedor"
    android:layout_alignStart="@+id/textProveedor"
    android:layout_marginStart="4dp"
    android:layout_marginTop="4dp"


```


	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

    android:text="@string/velocidad"
    android:textAppearance="?android:attr/textAppearanceMedium"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textProveedor" />
<TextView
    android:id="@+id/speed"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    android:background="#B5F8CCCC"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#3F51B5"
    app:layout_constraintBottom_toBottomOf="@+id/textSpeed"
    app:layout_constraintStart_toEndOf="@+id/textProveedor"
    app:layout_constraintTop_toTopOf="@+id/textSpeed" />
<TextView
    android:id="@+id/textLatitud"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="14dp"
    android:layout_marginTop="4dp"
    android:text="@string/latitud"
    android:textAppearance="?android:attr/textAppearanceMedium"
    app:layout_constraintStart_toEndOf="@+id/barrier"
    app:layout_constraintTop_toTopOf="parent" />
<TextView
    android:id="@+id/latitud"
    android:layout_width="wrap_content"
    android:layout_height="24dp"
    android:layout_marginStart="4dp"
    android:background="#B5F8CCCC"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#3F51B5"
    app:layout_constraintBottom_toBottomOf="@+id/textLatitud"
    app:layout_constraintStart_toEndOf="@+id/textProveedoresActivos"
    app:layout_constraintTop_toTopOf="@+id/textLatitud" />
<TextView
    android:id="@+id/textLongitud"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="14dp"
    android:layout_marginTop="4dp"
    android:text="@string/longitud"
    android:textAppearance="?android:attr/textAppearanceMedium"
    app:layout_constraintStart_toEndOf="@+id/barrier"
    app:layout_constraintTop_toBottomOf="@+id/textLatitud" />
<TextView
    android:id="@+id/longitud"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    android:background="#B5F8CCCC"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#3F51B5"
    app:layout_constraintBottom_toBottomOf="@+id/textLongitud"


```

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

app:layout_constraintStart_toEndOf="@+id/textProveedoresActivos"
app:layout_constraintTop_toTopOf="@+id/textLongitud" />
<TextView
    android:id="@+id/textProveedoresActivos"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textProveedor"
    android:layout_alignStart="@+id/textProveedor"
    android:layout_marginStart="14dp"
    android:layout_marginTop="4dp"
    android:text="@string/proveedores_activos"
    android:textAppearance="?android:attr/textAppearanceMedium"
    app:layout_constraintStart_toEndOf="@+id/barrier"
    app:layout_constraintTop_toBottomOf="@+id/textLongitud" />
<TextView
    android:id="@+id/proveedoresActivos"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    android:background="#B5F8CCCC"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#3F51B5"
    app:layout_constraintBottom_toBottomOf="@+id/textProveedoresActivos"
    app:layout_constraintStart_toEndOf="@+id/textProveedoresActivos"
    app:layout_constraintTop_toTopOf="@+id/textProveedoresActivos" />
<TextView
    android:id="@+id/textHora"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/textProveedor"
    android:layout_alignStart="@+id/textProveedor"
    android:layout_marginStart="14dp"
    android:layout_marginTop="4dp"
    android:text="@string/hora"
    android:textAppearance="?android:attr/textAppearanceMedium"
    app:layout_constraintStart_toEndOf="@+id/barrier"
    app:layout_constraintTop_toBottomOf="@+id/textProveedoresActivos" />
<TextView
    android:id="@+id/hora"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    android:background="#B5F8CCCC"
    android:text=""
    android:textAppearance="?android:attr/textAppearanceMedium"
    android:textColor="#3F51B5"
    app:layout_constraintBottom_toBottomOf="@+id/textHora"
    app:layout_constraintStart_toEndOf="@+id/textProveedoresActivos"
    app:layout_constraintTop_toTopOf="@+id/textHora" />
<androidx.constraintlayout.widget.Barrier
    android:id="@+id/barrier"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    app:barrierDirection="end"
    app:constraint_referenced_ids="precision,altitud,proveedor,speed"
    tools:layout_editor_absoluteX="108dp" />

```

	RAMA:	Informática	CICLO:	Desenvolvemto de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

- Justo debajo de los *TextViews* insertamos los siguientes tres botones que nos permiten respectivamente:
 - Configurar el proveedor de localización.
 - Solicitar una nueva localización.
 - Ver la localización actual en un mapa.

```

<Button
    android:id="@+id/btnConfigurar"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginEnd="8dp"
    android:enabled="false"
    android:text="@string/configuracion_provider"
    app:layout_constraintEnd_toStartOf="@+id/btnAcceso"
    app:layout_constraintTop_toTopOf="@+id/btnAcceso" />

<Button
    android:id="@+id/btnAcceso"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_above="@+id/btnConfigurar"
    android:layout_marginTop="4dp"
    android:enabled="false"
    android:text="@string/nueva_peticion"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textSpeed" />

<Button
    android:id="@+id/bmapa"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:enabled="false"
    android:text="@string/vermapa"
    app:layout_constraintStart_toEndOf="@+id/btnAcceso"
    app:layout_constraintTop_toTopOf="@+id/btnConfigurar" />

```


- Los siguientes elementos se utilizan para decoración de la interfaz:

```

<TextView
    android:id="@+id/textInfo"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:background="#D5EEC0C0"
    android:gravity="center"
    android:text="@string/get_location"
    android:textSize="24sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<ProgressBar
    android:id="@+id/progressBar"
    style="?android:attr/progressBarStyle"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="4dp"
    android:layout_marginTop="4dp"
    android:layout_marginEnd="4dp"
    android:layout_marginBottom="4dp"
    app:layout_constraintBottom_toBottomOf="parent"

```

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```
app:layout_constraintEnd_toEndOf="parent"
app:layout_constraintHorizontal_bias="0.5"
app:layout_constraintStart_toStartOf="parent"
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintVertical_bias="0.58000004" />
```

- **MainActivity** debe implementar el interfaz *LocationListener* y sobrescribir los siguientes métodos de la interfaz:
 - **onLocationChanged**: Método llamado cuando una nueva posición está disponible en *LocationManager*. En el parámetro *location* está la información de dicha nueva posición. Es la más usada. En muchos casos los demás métodos quedarán en blanco.
 - **onStatusChanged**: Indica el estado de un proveedor. En el parámetro de tipo *Bundle* se indica otra información. Estados posibles son: *LocationProvider.OUT_OF_SERVICE*, *TEMPORARILY_UNAVAILABLE*, *AVAILABLE*. Esta deprecado desde la versión de *API 29*. No lo usaremos.
 - **onProviderEnabled**: Informa cuando el usuario activa un proveedor de posicionamiento. En el parámetro nos da el nombre del *LocationProvider* activado.
 - **onProviderDisabled**. Informa cuando el usuario desactiva un proveedor de posicionamiento. En el parámetro nos da el nombre del *LocationProvider* desactivado. Será uno de los existentes en *LocationManager.getProviders()*.
- Se informa, en *onCreate* de *MainActivity*, que se va a usar el sistema de posicionamiento:


```
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
```

- Se inicializan los componentes quedando de la siguiente forma:

```
final static int requestCode_SOLICITO_PERMISOS = 1; // requestCode usado para la solicitud de permisos
TextView longitud, latitud, proveedor, precision, proveedoresActivos, altitud, velocidad, hora; // textviews de datos
Button bOpc, bNueva, bMapa; // botones de acciones

TextView textInfo; // capa roja sobre la interfaz cuando se está buscando la primera localización
ProgressBar pb; // progressBar animado cuando se está buscando la primera localización

boolean perderPermisos=true; // impide que tras negar permisos se sigan solicitando de forma automática

Location pos; // Posición geográfica auxiliar para su visualización en un intent de mapa
LocationManager locationManager; // utilizado para acceder a los servicios de localización


@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // textview que contendrán los datos
    longitud = findViewById(R.id.longitud);
    latitud = findViewById(R.id.latitud);
    altitud = findViewById(R.id.altitud);
    proveedor = findViewById(R.id.proveedor);
    precision = findViewById(R.id.precision);
    proveedoresActivos = findViewById(R.id.proveedoresActivos);
    velocidad = findViewById(R.id.speed);
    hora = findViewById(R.id.hora);

    // Botones de acción
    bOpc = findViewById(R.id.btnConfigurar);
    bNueva = findViewById(R.id.btnAcceso);
    bMapa = findViewById(R.id.bmapa);

    // Elementos decorativos
    textInfo = findViewById(R.id.textInfo);
    pb = findViewById(R.id.progressBar);
    pb.animate();

    // Se informa, en onCreate de MainActivity, que se va a usar el sistema de posicionamiento:
    locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
```

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

    Toast.makeText(this, "Gps: " + locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) +
        "\n" + "Network: " + locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER),
        Toast.LENGTH_SHORT).show();
}

```

- Se puede observar que en el *onCreate* se obtiene el *locationManager*, pero en ningún momento se inicializa ni se le dice qué *listener* usar. Esto es porque suele realizarse en el *onResume* del *activity*, ya que si viene de una pausa permite realizar dicha inicialización. Además, en este caso, no basta con que se pongan los permisos en el manifiesto sino que se ha de solicitar al usuario que los conceda mediante el método *requestPermissions* que visualizara un dialogo de aceptación der permisos . Por lo que tendríamos:

```

@Override
protected void onResume() {
    super.onResume();


    // Se comprueban si se tiene los permisos necesarios
    if (perdirPermisos
        //ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
        && checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
        && checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {

        // En caso de no tenerlos se solicita al usuario su concesión
        String[] permisos = new String[]{
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION
        };
        ActivityCompat.requestPermissions(MainActivity.this, permisos, requestCode_SOLICITO_PERMISOS);
    } else { // Si ya se tienen los permisos se ejecuta de forma normal

        Criteria criteria = new Criteria(); // Se establece el criterio de precisión
        criteria.setAccuracy(Criteria.ACCURACY_COARSE); // gps + network
        //criteria.setAccuracy(Criteria.ACCURACY_FINE); // solo gps

        List<String> proveedores = locationManager.getProviders(criteria, true);
        if (proveedores.isEmpty()) { // Si no se encuentran proveedores de localización
            proveedoresActivos.setText("0");
            pb.setVisibility(View.GONE); // oculto el spinner
            textInfo.setVisibility(View.GONE); // oculto la capa superior roja
            bOpc.setEnabled(true); // habilito los botones
            bNueva.setEnabled(true); // habilito los botones
            bMapa.setEnabled(true); // habilito los botones
        } else {
            // Se realiza una actualización de estado para cada proveedor. El parámetro looper (tercer parámetro
            // de requestSingleUpdate) a null indica los métodos callback están en el hilo que ejecuta la llamada
            proveedoresActivos.setText("");
            for (String proveedorActivo : proveedores) {
                // Se solicita una nueva ubicación
                // locationManager.requestSingleUpdate(proveedorActivo, this, null);
                // Parámetros -> 1: String proveedor 2: tiempo Mínimo Actu 3: Distancia Mínima Actu. 4: listener 5: looper
                locationManager.requestLocationUpdates(proveedorActivo, 0, 0, this, null);
                proveedoresActivos.append(proveedorActivo + " ");
            }
        }
    }
}

```

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

- Se debe implementar el método `onRequestPermissionsResult` que se ejecuta de forma automática cuando se selecciona una opción en el dialogo de concesión de permisos del punto anterior.

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    // Se comprueba que se viene del dialogo que solicita permisos de localización
    if (requestCode == requestCode_SOLICITO_PERMISOS) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) { // permiso concedido
            Toast.makeText(getApplicationContext(), R.string.permiso_si, Toast.LENGTH_LONG).show();
        } else { // Se ha rechazado el permiso
            pedirPermisos=false; // No solicito más los permisos
        }
    }
}
```

- Creamos un método que, a partir de un objeto de tipo `Location`, actualiza los `TextViews` con sus valores:

```
public void actualizarDatos(Location location){
    longitud.setText(location.getLongitude() + "");
    latitud.setText(location.getLatitude() + "");
    altitud.setText((Math.floor(location.getAltitude()*100)/100)+"");
    proveedor.setText(location.getProvider() + "");
    precision.setText(location.getAccuracy() + "");
    velocidad.setText(location.getSpeed()+"");
    Calendar cal=Calendar.getInstance();
    cal.setTime(new Date(location.getTime()));
    hora.setText(new SimpleDateFormat("hh:mm:ss").format(cal.getTime()));
}
```


- Hay que tener en cuenta que, por defecto, no es la aplicación quién pide la posición cuando quiere, si no que mediante el `listener` es `Android` quién avisa de cambios de posición a nuestra aplicación ejecutando el método `onLocationChange`. Implementamos todos los métodos `callback`.

```
@Override
public void onLocationChanged(Location location) {
    // Si tengo una localización oculto la capa superior y el spinner y habilito los botones
    if (pb.getVisibility() == View.VISIBLE) {
        pb.setVisibility(View.GONE); // oculto el spinner
        textInfo.setVisibility(View.GONE); // oculto la capa superior roja
        bOpc.setEnabled(true); // habilito los botones
        bNueva.setEnabled(true); // habilito los botones
        bMapa.setEnabled(true); // habilito los botones
        this.pos=location; // actualizo el objeto posición para usarse con el botón de lanzar el mapa
    }
    actualizarDatos(location);
}

@Override
public void onStatusChanged(String s, int i, Bundle bundle) {
}

@Override
public void onProviderEnabled(String s) {
    Toast.makeText(this, "Proveedor habilitado " + s, Toast.LENGTH_SHORT).show();
}

@Override
public void onProviderDisabled(String s) {
    Toast.makeText(this, "Proveedor deshabilitado " + s, Toast.LENGTH_SHORT).show();
}
```


	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

Finalmente implementaremos, en *onCreate*, el comportamiento de los tres botones:

- Con el botón *configurar proveedor* se puede configurar el sistema de posicionamiento:

```
bOpc.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
    }
});
```

- Con el botón *ver en mapa* se lanza un *intent* implícito de mapas que visualiza y marca la posición actual.

```
bMapa.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (pos!=null){
            Intent intent = new Intent(Intent.ACTION_VIEW);
            double longitud= pos.getLongitude();
            double latitud= pos.getLatitude ();
            String direccion=String.format(Locale.US,"geo:%f,%f?q=%f,%f," ,latitud, longitud, latitud, longitud);
            intent.setData(Uri.parse(direccion));
            startActivity(intent);
        }
    }
});
```


Donde:

- Se usa el *Locale* para que la conversión del float al string se utilice con punto en lugar de con coma.
- En la *Uri* geo, primero se pasan las coordenadas y con la *q=* indicamos que queremos un pin en ese punto.
- Con el botón nueva petición se solicita una nueva ubicación usando para ello la *API* de ubicación de los Servicios de *Google Play*. *Android Studio* añadirá, de forma automática, la dependencia en el archivo *build.gradle* de la aplicación:

```
implementation 'com.google.android.gms:play-services-location:17.0.0'
```

Se pueden, entre otros, establecer los siguientes valores:

- Periodicidad de actualizaciones. Establece la preferencia de tiempo, en milisegundos, para realizar las actualizaciones de posición usando el método *setInterval*.
- Periodicidad máxima de actualizaciones. Indica, en milisegundos, la mayor frecuencia de actualización de posición que nuestra aplicación es capaz de procesar. Se establece con el método *setFastestInterval*.
- Número de actualizaciones: por defecto las localizaciones se están actualizando de forma continua pero con el método *setNumUpdates* se puede limitar a un número concreto. En nuestro ejemplo lo limitaremos que cada vez que se pulse el botón se produzca una única actualización.
- Precisión. Establece es tipo de precisión que recibiremos. Se establece con el método *setPriority* u sus valores puede ser:
 - ❖ *PRIORITY_HIGH_ACCURACY*. Indica la máxima precisión.
 - ❖ *PRIORITY_BALANCED_POWER_ACCURACY*. Tiene una precisión aproximada de 100 metros. El consumo de energía será bajo.
 - ❖ *PRIORITY_LOW_POWER*. Indica una precisión cercana a los 10 kilómetros con un consumo de energía muy bajo.
 - ❖ *PRIORITY_NO_POWER*. En este modo nuestra aplicación solo recibirá datos si éstos están disponibles porque alguna otra aplicación los haya solicitado. Es decir, nuestra aplicación no tendrá un impacto directo en el consumo de energía solicitando nuevas ubicaciones, pero si éstas están disponibles las utilizará.

	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

bNueva.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        LocationRequest mLocationRequest = new LocationRequest();
        mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        mLocationRequest.setInterval(0);
        mLocationRequest.setFastestInterval(0);
        mLocationRequest.setNumUpdates(1);

        FusedLocationProviderClient mFusedLocationClient = LocationServices.getFusedLocationProviderClient(MainActivity.this);
        mFusedLocationClient.requestLocationUpdates(
            mLocationRequest, new LocationCallback() {
                @Override
                public void onLocationResult(LocationResult locationResult) {
                    Toast.makeText(MainActivity.this, "Nueva petición", Toast.LENGTH_SHORT).show();
                    Location mLastLocation = locationResult.getLastLocation();
                    pos=mLastLocation;
                    actualizarDatos(mLastLocation);
                }
            },
            Looper.myLooper()
        );
    }
});

```

- Se puede simular, en el emulador, distintas posiciones tanto mediante la interfaz gráfica destinada a ello como mediante telnet. Como primer paso nos autenticarnos mediante el *token* que se crea en el fichero (marcado en la imagen) y el comando:


```
auth wnbddraULNpGX (token del fichero)
```

El comando para simular las posiciones es *geo*, al cual se le especifica la nueva longitud y latitud:

```
geo fix 42.2 -8.8
```

Se puede probar también a configurar *COARSE* en lugar de *FINE* tanto en el Manifiesto como en el *Criterio* para ver como varía el comportamiento.

Para probar una localización en concreto podemos obtener su longitud y latitud de múltiples páginas. Una de ellas es: <https://www.latlong.net/>.

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

5. Mapas

La localización en una gran cantidad de ocasiones se realizará sobre un mapa para mostrar cierto o ciertos puntos al usuario.

Para ello existen una gran variedad de mapas, cada uno de ellos con sus propias librerías. Aunque en *Android* los mapas de *Google* suelen ser los más usados nosotros no los usaremos ya que requieren registrarse y obtener una *API Key*.

Además de los de Google, existen, entre otras, las siguientes alternativas¹⁴:

- Osmroid: <http://osmdroid.github.io/osmdroid/>
<https://github.com/osmdroid/osmdroid/wiki/How-to-use-the-osmdroid-library>
- Mapsforge: <https://github.com/mapsforge>
- Mapquest: <http://developer.mapquest.com/web/products/featured/android-maps-api>
- Skobbler: <https://www.skobbler.com/>
- Bing: <https://docs.microsoft.com/en-us/bingmaps/sdk-native/>
- ...

Los anteriores son algunos ejemplos de proyectos bastante extendidos. Los tres primeros están basados en *OpenStreetMaps*, lo que da mucha versatilidad al desarrollador a la hora de realizar aplicaciones, ya que no está sometido a las decisiones de uso de ninguna empresa. En cualquier caso cada uno tiene sus ventajas e inconvenientes.

Para el siguiente ejemplo usaremos *Osmroid*.

5.1. Osmroid

La clase *MapView* de *Osmroid* es básicamente un reemplazo de la clase *MapView* de *Google* que usa *OpenStreetMaps*.

Podemos separar la aplicación en dos partes:

- La visualización del mapa y el marcado en el de un punto o puntos.
- Visualización de la posición actual y actualización de la posición del según nos movemos.

5.1.1. Visualización del mapa

Los pasos que tenemos que realizar para visualizar el mapa son los siguientes:

- Se añade la dependencia necesaria en el fichero *build.gradle* dentro de la sección *dependencies*.

```
implementation 'org.osmdroid:osmdroid-android:(INSERT_VERSION_HERE)'
```

Para ver cuál es la última versión se puede consultar en *github*:

<https://github.com/osmdroid/osmdroid/releases>

En nuestro caso en la 6.1.5 por lo que la dependencia quedaría:


```
implementation 'org.osmdroid:osmdroid-android:6.1.5'
```

Tras añadir la línea anterior sincronizamos *Gradle* para que baje, de forma automática, todas las librerías necesarias.

- Añadir en el manifiesto el permiso para poder acceder a internet.

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

¹⁴ <https://stackoverflow.com/questions/4151593/alternatives-to-google-maps-api#4233638>

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

- Escribimos el siguiente código. Todos el código está explicado mediante los correspondientes comentarios.

```
public class MainActivity extends AppCompatActivity {
    ArrayList<OverlayItem> puntos = new ArrayList<>(); // Array que contiene los puntos que se dibujaran en el mapa
    MapView mMapView = null; // Objeto que mapa a representar

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Si se necesitan se solicitan, en este punto, los permisos al usuario

        // configuración inicial de osmdroid. Se recomienda realizarlo antes del setContentView
        Configuration.getInstance().load(getApplicationContext(),
            PreferenceManager.getDefaultSharedPreferences(getApplicationContext()));

        setContentView(R.layout.activity_main);

        // obtenemos el MapView
        mMapView = findViewById(R.id.openmapview);
        mMapView.setTileSource(TileSourceFactory.MAPNIK);

        // Se añaden los botones de zoom
        mMapView.getZoomController().setVisibility(CustomZoomButtonsController.Visibility.SHOW_AND_FADEOUT);

        // Se añade la posibilidad de hacer zoom con dos dedos: multi-touch
        mMapView.setMultiTouchControls(true);

        // Creamos tres puntos geográficos en el mapa
        GeoPoint torreEiffel = new GeoPoint(48.8583, 2.2944); // La torre Eiffel
        GeoPoint trocadero = new GeoPoint(48.861253, 2.289382); // Jardines de Trocadero
        GeoPoint campoDeMarte = new GeoPoint(48.856256, 2.297575); // Jardines del Campo de Marte

        // Centramos el mapa en un punto y establecemos el zoom por defecto
        MapController mapController = (MapController)mMapView.getController();
        mapController.setCenter(torreEiffel); // Se centra el mapa en el punto creado torreEiffel
        mapController.setZoom(16.5); // Se estable el zoom por defecto del mapa


        // Permite dibujar los tiles del mapa. Se necesitan los permisos necesarios
        MyLocationNewOverlay mLocationOverlay = new MyLocationNewOverlay(new GpsMyLocationProvider(this), mMapView);
        mLocationOverlay.enableMyLocation();
        mMapView.getOverlays().add(mLocationOverlay);

        // Se añaden marcadores, a la lista de marcadores, para los puntos a visualizar
        // Punto 1: con el marcador por defecto
        puntos.add(new OverlayItem("Torre Eiffel", "En Paris", torreEiffel));

        // Punto 2: con una cruceta como marcador
        OverlayItem marcador = new OverlayItem("Trocadero", "En Paris", trocadero);
        marcador.setMarker(ResourcesCompat.getDrawable(getResources(), R.drawable.center, null));
        puntos.add(marcador);

        // Punto 2: con la imagen de una persona como marcador
        OverlayItem marcador2 = new OverlayItem("Jardines", "Campo de Marte", campoDeMarte);
        marcador2.setMarker(ResourcesCompat.getDrawable(getResources(), R.drawable.osm_ic_follow_me_on, null));
        puntos.add(marcador2);

        // Se dibujan los puntos anteriores
        ItemizedOverlayWithFocus<OverlayItem> capas = new ItemizedOverlayWithFocus<OverlayItem>(puntos,
            new ItemizedIconOverlay.OnItemGestureListener<OverlayItem>() {
                @Override
                public boolean onItemSingleTapUp(final int index, final OverlayItem item) {
                    //do something
                    return true;
                }
            }
        );
        @Override
        public boolean onItemLongPress(final int index, final OverlayItem item) {
```

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

        return false;
    }
}, this);
capas.setFocusItemsOnTap(true);

mMapView.getOverlays().add(capas);

// Se visualiza una brújula en la zona superior izquierda
CompassOverlay mCompassOverlay = new CompassOverlay(this, new InternalCompassOrientationProvider(this), mMapView);
mCompassOverlay.enableCompass();
mMapView.getOverlays().add(mCompassOverlay);

// Visualiza cuadrícula con coordenadas
LatLonGridlineOverlay2 overlay = new LatLonGridlineOverlay2();
mMapView.getOverlays().add(overlay);

// Habilita gesto de rotación con los dedos
RotationGestureOverlay rotationGestureOverlay = new RotationGestureOverlay(this, mMapView);
rotationGestureOverlay.setEnabled(true);
mMapView.setMultiTouchControls(true);
mMapView.getOverlays().add(rotationGestureOverlay);


// Muestra en la parte superior un marcador que indica la escala de mapa
final DisplayMetrics dm = getResources().getDisplayMetrics();
ScaleBarOverlay mScaleBarOverlay = new ScaleBarOverlay(mMapView);
mScaleBarOverlay.setCentred(true);
// Cambiando los valores siguientes se puede ajustar su posicionamiento en pantalla
mScaleBarOverlay.setScaleBarOffset(dm.widthPixels / 2, 10);
mMapView.getOverlays().add(mScaleBarOverlay);

// Se añade un minimapa
MinimapOverlay mMinimapOverlay = new MinimapOverlay(this, mMapView.getTileRequestCompleteHandler());
mMinimapOverlay.setWidth(dm.widthPixels / 5);
mMinimapOverlay.setHeight(dm.heightPixels / 5);
mMapView.getOverlays().add(mMinimapOverlay);
}

@Override
protected void onResume() {
    super.onResume();
    //this will refresh the osmdroid configuration on resuming. if you make changes to the configuration, use
    //SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
    //Configuration.getInstance().load(this, PreferenceManager.getDefaultSharedPreferences(this));
    mMapView.onResume(); //needed for compass, my location overlays, v6.0.0 and up
}

@Override
protected void onPause() {
    super.onPause();
    //this will refresh the osmdroid configuration on resuming. if you make changes to the configuration, use
    //SharedPreferences prefs = PreferenceManager.getDefaultSharedPreferences(this);
    //Configuration.getInstance().save(this, prefs);
    mMapView.onPause(); //needed for compass, my location overlays, v6.0.0 and up
}
}

```

	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

5.1.2. Visualización de la posición actual y su actualización

Para la visualización de la posición actual tendremos que realizar una mezcla de las dos aplicaciones anteriores. Por un lado tendremos que conseguir la posición actual mediante un *LocationListener* para luego dibujarlo en el mapa.

Vamos a adaptar la aplicación del punto 4.2 para que visualice la posición actual en un mapa.

- Comenzamos añadiéndole la dependencia necesaria en el fichero *build.gradle* dentro de la sección *dependencies*.

```
implementation 'org.osmdroid:osmdroid-android:6.1.5'
```

- Se añade, en el manifiesto, el permiso para acceder a Internet.

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

Puede ser que también se necesite el permiso de escritura para guardar los mapas en la memoria interna (aunque puede no hacer falta).

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

- Se añade el siguiente componente justo por encima del *TextView textInfo*.

```
<org.osmdroid.views.MapView
    android:id="@+id/openmapview"
    android:layout_width="0dp"
    android:layout_height="0dp"
    android:layout_marginStart="4dp"
    android:layout_marginTop="4dp"
    android:layout_marginEnd="4dp"
    android:layout_marginBottom="4dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/btnAcceso" />
```

- MainActivity quedaría de la siguiente manera. El código está documentado para ayudar a su comprensión.

```
public class MainActivity extends AppCompatActivity implements LocationListener {
    final static int requestCode_SOLICITO_PERMISOS = 1; // requestCode usado para la solicitud de permisos
    TextView longitud, latitud, proveedor, precision, proveedoresActivos, altitud, velocidad, hora; // textviews de datos
    Button bOpc, bNueva, bMapa; // botones de acciones

    TextView textInfo; // capa roja sobre la interfaz cuando se está buscando la primera localización
    ProgressBar pb; // progressBar animado cuando se está buscando la primera localización

    boolean perderPermisos=true; // impide que tras negar permisos se sigan solicitando de forma automática


    Location pos; // Posición geográfica auxiliar para su visualización en un intent de mapa
    LocationManager locationManager; // utilizado para acceder a los servicios de localización

    MapView mMapView = null; // Objeto que representa el mapa a dibujar
    MapController mapController; // Permite modificar aspectos del mapa como puede ser el zoom

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // configuración inicial de osmdroid. Se recomienda realizarlo antes del setContentView
        Configuration.getInstance().load(getApplicationContext(),
        PreferenceManager.getDefaultSharedPreferences(getApplicationContext()));

        setContentView(R.layout.activity_main);
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON); // Mantiene la pantalla encendida
        //getWindow().clearFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON); // Permite apagar la pantalla
    }
}
```


	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

// textview que contendrán los datos
longitud = findViewById(R.id.longitud);
latitud = findViewById(R.id.latitud);
altitud = findViewById(R.id.altitud);
proveedor = findViewById(R.id.proveedor);
precision = findViewById(R.id.precision);
proveedoresActivos = findViewById(R.id.proveedoresActivos);
velocidad = findViewById(R.id.speed);
hora = findViewById(R.id.hora);

// Botones de acción
bOpc = findViewById(R.id.btnConfigurar);
bNueva = findViewById(R.id.btnAcceso);
bMapa = findViewById(R.id.bmapa);

// Elementos decorativos
textInfo = findViewById(R.id.textInfo);
pb = findViewById(R.id.progressBar);
pb.animate();

// Se informa, en onCreate de MainActivity, que se va a usar el sistema de posicionamiento:
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);
Toast.makeText(this, "Gps: " + locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER) +
    "\n" + "Network: " + locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER),
    Toast.LENGTH_SHORT).show();


// boton modificar opciones
bOpc.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(android.provider.Settings.ACTION_LOCATION_SOURCE_SETTINGS));
    }
});

// boton lazar intent del mapa
bMapa.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (pos != null) {
            Intent intent = new Intent(Intent.ACTION_VIEW);
            double longitud = pos.getLongitude();
            double latitud = pos.getLatitude();
            String direccion = String.format(Locale.US, "geo:%f,%f?q=%f,%f," , latitud, longitud, latitud, longitud);
            intent.setData(Uri.parse(direccion));
            startActivity(intent);
        }
    }
});

// boton para solicitar una nueva ubicación usando para ello la API de ubicación de los Servicios de Google Play
bNueva.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        LocationRequest mLocationRequest = new LocationRequest();
        mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        mLocationRequest.setInterval(0);
        mLocationRequest.setFastestInterval(0);
        mLocationRequest.setNumUpdates(1);

        FusedLocationProviderClient mFusedLocationClient = LocationServices.getFusedLocationProviderClient(MainActivity.this);
        mFusedLocationClient.requestLocationUpdates(
            mLocationRequest, new LocationCallback() {
                @Override

```

	RAMA:	Informática	CICLO:	Desenvolvemto de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

        public void onLocationResult(LocationResult locationResult) {
            Toast.makeText(MainActivity.this, "Nueva petición", Toast.LENGTH_SHORT).show();
            Location mLastLocation = locationResult.getLastLocation();
            pos=mLastLocation;
            actualizarDatos(mLastLocation);
            actualizaPosicion();
        }
    },
    Looper.myLooper()
);
}
});

// obtenemos el MapView
mMapView = findViewById(R.id.openmapview);
mMapView.setTileSource(TileSourceFactory.MAPNIK);

// Se añaden los botones de zoom
mMapView.getZoomController().setVisibility(CustomZoomButtonsController.Visibility.SHOW_AND_FADEOUT);

// Se añade la posibilidad de hacer zoom con dos dedos(multi-touch)
mMapView.setMultiTouchControls(true);

// Centramos el mapa en un punto y establecemos el zoom por defecto
mapController =(MapController)mMapView.getController();
mapController.setZoom(19); // Se estable el zoom por defecto del mapa

// Permite dibujar los tiles del mapa. Se necesitan los permisos necesarios
MyLocationNewOverlay mLocationOverlay = new MyLocationNewOverlay(new GpsMyLocationProvider(this), mMapView);
mLocationOverlay.enableMyLocation();
mMapView.getOverlays().add(mLocationOverlay);

// Se actualiza el mapa centrandose la posición actual
actualizaPosicion();
}

@Override
protected void onResume() {
    super.onResume();


    // Se comprueban si se tiene los permisos necesarios
    if (perdirPermisos
//ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) !=PackageManager.PERMISSION_GRANTED
        && checkSelfPermission(Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
        && checkSelfPermission(Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {

        // En caso de no tenerlos se solicita al usuario su concesión
        String[] permisos = new String[]{
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION
        };
        ActivityCompat.requestPermissions(MainActivity.this, permisos, requestCode_SOLICITO_PERMISOS);
    } else { // Si ya se tienen los permisos se ejecuta de forma normal

        Criteria criteria = new Criteria(); // Se establece el criterio de precisión
        criteria.setAccuracy(Criteria.ACCURACY_COARSE); // gps + network
        //criteria.setAccuracy(Criteria.ACCURACY_FINE); // solo gps

        List<String> proveedores = locationManager.getProviders(criteria, true);
        if (proveedores.isEmpty()) { // Si no se encuentran proveedores de localización
            proveedoresActivos.setText("0");
            pb.setVisibility(View.GONE); // oculto el spinner
            textInfo.setVisibility(View.GONE); // oculto la capa superior roja
            bOpc.setEnabled(true); // habilito los botones
            bNueva.setEnabled(true); // habilito los botones
            bMapa.setEnabled(true); // habilito los botones
        }
    }
}

```

	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

    } else {
        // Se realiza una actualización de estado para cada proveedor.
        // El parámetro looper a null indica los métodos callback están en el hilo que ejecuta la llamada
        proveedoresActivos.setText("");
        for (String proveedorActivo : proveedores) {
            // Se solicita una nueva ubicación
            // locationManager.requestSingleUpdate(proveedorActivo, this, null);
            // Parámetros -> 1: String proveedor 2: tiempo Mínimo Actu 3: Distancia Mínima Actu. 4: listener 5: looper
            locationManager.requestLocationUpdates(proveedorActivo, 0, 0, this, null);
            proveedoresActivos.append(proveedorActivo + " ");
        }
    }
}

// Método que se ejecuta de forma automática cuando se selecciona una opción en el diálogo de concesión de permisos
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults){
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    // Se comprueba que se viene del dialogo que solicita permisos de localización
    if (requestCode == requestCode_SOLICITO_PERMISOS) {
        if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) { // permiso concedido
            Toast.makeText(getApplicationContext(), R.string.permiso_si, Toast.LENGTH_LONG).show();
        } else { // Se ha rechazado el permiso
            perdirPermisos=false; // No solicito más los permisos
        }
    }
}


// Funciones callback de la interfaz LocationListener
@Override
public void onLocationChanged(Location location) {
    Toast.makeText(this, "actualizo posicion", Toast.LENGTH_SHORT).show();
    // Si tengo una localización oculto la capa superior y el spinner y habilito los botones
    if (pb.getVisibility() == View.VISIBLE) {
        pb.setVisibility(View.GONE); // oculto el spinner
        txtInfo.setVisibility(View.GONE); // oculto la capa superior roja
        bOpc.setEnabled(true); // habilito los botones
        bNueva.setEnabled(true); // habilito los botones
        bMapa.setEnabled(true); // habilito los botones
    }
    this.pos=location; // actualizo el objeto posición para usarse con el botón de lanzar el mapa
    actualizarDatos(location); // Actualiza el contenido de los TextEdits con los datos de location
    actualizaPosicion(); // Actualiza la posición en el mapa y dibuja elementos auxiliares del mismo
}

@Override
public void onStatusChanged(String s, int i, Bundle bundle) {
}

@Override
public void onProviderEnabled(String s) {
    Toast.makeText(this, "Proveedor habilitado " + s, Toast.LENGTH_SHORT).show();
}

@Override
public void onProviderDisabled(String s) {
    Toast.makeText(this, "Proveedor deshabilitado " + s, Toast.LENGTH_SHORT).show();
}

```

	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

// Actualiza el contenido de los TextViews con los datos de location
public void actualizarDatos(Location location){
    longitud.setText(location.getLongitude() + "");
    latitud.setText(location.getLatitude() + "");
    altitud.setText((Math.floor(location.getAltitude()*100)/100)+"");
    proveedor.setText(location.getProvider() + "");
    precision.setText(location.getAccuracy() + "");
    velocidad.setText(location.getSpeed()+"");
    Calendar cal=Calendar.getInstance();
    cal.setTime(new Date(location.getTime()));
    hora.setText(new SimpleDateFormat("hh:mm:ss").format(cal.getTime()));
}

// Actualiza la posición en el mapa y dibuja elementos auxiliares del mismo
public void actualizaPosicion(){
    if (pos!=null) {
        GeoPoint aux = new GeoPoint(pos.getLatitude(), pos.getLongitude()); // Nuevo punto usando la posición actual
        mapController.setCenter(aux); // Centro el mapa en esta posición

        mMapView.getOverlays().clear(); // Se eliminan los marcadores previos para que no se visualicen
        addElementosMapa(); // Se añaden elementos auxiliares al mapa: compas, escala, minimap, ....

        ArrayList<OverlayItem> puntos = new ArrayList<>(); // Array que contiene los puntos que se dibujaran en el mapa

        // Añado la posición actual
        puntos.add(new OverlayItem("Mi posición", pos.getLatitude()+", "+ pos.getLongitude(), aux));

        ItemizedOverlayWithFocus<OverlayItem> capas = new ItemizedOverlayWithFocus<OverlayItem>( puntos,
            new ItemizedIconOverlay.OnItemGestureListener<OverlayItem>() {
                @Override
                public boolean onItemSingleTapUp(final int index, final OverlayItem item) {
                    //do something
                    return true;
                }


                @Override
                public boolean onItemLongPress(final int index, final OverlayItem item) {
                    return false;
                }
            }, this);
        capas.setFocusItemsOnTap(true);
        mMapView.getOverlays().add(capas); // añado la capa con los puntos
    }
}

// Se añaden elementos auxiliares al mapa: compas, escala, minimap, ....
public void addElementosMapa(){
    // Visualiza una brújula en la zona superior izquierda
    CompassOverlay mCompassOverlay =
        new CompassOverlay(this, new InternalCompassOrientationProvider(this), mMapView);
    mCompassOverlay.enableCompass();
    mMapView.getOverlays().add(mCompassOverlay);

    // Visualiza cuadrícula con coordenadas
    LatLonGridlineOverlay2 overlay = new LatLonGridlineOverlay2();
    mMapView.getOverlays().add(overlay);

    // Habilita gesto de rotación con los dedos
    RotationGestureOverlay rotationGestureOverlay = new RotationGestureOverlay(this, mMapView);
    rotationGestureOverlay.setEnabled(true);
    mMapView.setMultiTouchControls(true);
    mMapView.getOverlays().add(rotationGestureOverlay);
}

```


	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

```

// Muestra en la parte superior un marcador que indica la escala de mapa
final DisplayMetrics dm = getResources().getDisplayMetrics();
ScaleBarOverlay mScaleBarOverlay = new ScaleBarOverlay(mMapView);
mScaleBarOverlay.setCentred(true);
//play around with these values to get the location on screen in the right place for your application
mScaleBarOverlay.setScaleBarOffset(dm.widthPixels / 2, 10);
mMapView.getOverlays().add(mScaleBarOverlay);

//How to add the built-in Minimap
MinimapOverlay mMinimapOverlay = new MinimapOverlay(this, mMapView.getTileRequestCompleteHandler());
mMinimapOverlay.setWidth(dm.widthPixels / 5);
mMinimapOverlay.setHeight(dm.heightPixels / 5);
//optionally, you can set the minimap to a different tile source mMinimapOverlay.setTileSource(...);
mMapView.getOverlays().add(mMinimapOverlay);
}
}

```

	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

6. Cámara

Como en otros casos vistos, se puede usar la cámara directamente mediante *intents* implícitos que llaman a los programas correspondientes (en este caso Camera App) para realizar fotos o videos.

Cierto es que la cámara es un elemento que dispone de sus propias librerías y se puede explotar a fondo, pero para los requisitos de este curso nos llega con saber tomar una foto aprovechando la funcionalidad de los *intents* implícitos y luego recuperar dicha foto en nuestra *activity*.

También hay que tener en cuenta que desde la versión 21 de la API (Android 5), las clases han cambiado y pasa de usarse *android.hardware.Camera* a *android.hardware.Camera2*. Esto debe ser tenido en cuenta a la hora de realizar nuevas aplicaciones que exploten a fondo el uso de la cámara.

Veamos con un ejemplo como podemos llevar a cabo lo indicado al principio tal cual lo explica en la documentación de *Google Developers*.

Lo primero es indicar que la aplicación va a requerir el uso de la cámara. No es exactamente un permiso (que sería necesario si se usan las clases directamente), pero si indica los recursos necesarios para una aplicación. Así en el manifiesto se pondría:

```
<uses-feature android:name="android.hardware.camera" android:required="true" />
```

El parámetro booleano indica que la existencia de cámara es esencial para la aplicación. De esta forma a dispositivos sin cámara no se les permite instalar la aplicación. Si la aplicación puede usar la cámara, pero no es esencial, se colocará a false. Pero es luego labor del programador comprobar su existencia.

Vamos a crear una aplicación que contenga dos botones y un *imageView*.

- Captura de *thumbnail*


Sobre la propiedad *onClick* del primer programaremos la llamada al *intent* donde indicaremos el uso de la cámara.

```
public void getThumbnail (View v){
    Intent fotoIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    // Comprobamos que existe algún programa asociado al intent implícito, ya que si no saltaría una excepción.
    if (fotoIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(fotoIntent, 1); // requestCode 1 para Thumbnail
    }
}
```

La foto de tamaño reducido, el *thumbnail*, viene con el valor "data" del intent

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    ImageView iv = (ImageView) findViewById(R.id.imageView);
    if (requestCode == 1 && resultCode == RESULT_OK) { // Thumbnail
        Bitmap imagen = (Bitmap) data.getExtras().get("data");
        iv.setImageBitmap(imagen);
        try { // Guardo la imagen
            FileOutputStream fout = openFileOutput("imagen.jpg", MODE_PRIVATE);
            imagen.compress(Bitmap.CompressFormat.JPEG, 80, fout);
            fout.close();
            String[] files = getApplicationContext().fileList(); // Visualizo el nombre de las imagenes guardadas
            for (String file : files) {
                Log.i("imagen -> ", file);
            }
        } catch (IOException e) {
            Log.e("Error imagen", e.getMessage());
        }
    }
}
```


	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

- Foto a tamaño real

Para obtener la foto completa hay que realizarlo a través de algún archivo, y en el *intent* se le pasaría simplemente el nombre del archivo.

Normalmente al hacer una foto esta debe ser guardada en un directorio especial accesible por todas las aplicaciones. Para llegar a él se puede usar el comando:

```
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)
```

También existe la posibilidad de usar uno privado, que se obtiene mediante:

```
getExternalFilesDir(Environment.DIRECTORY_PICTURES)
```

En nuestro ejemplo usaremos el primero. Como es una escritura de datos en una zona de memoria externa (SD o un directorio en la interna que emula la SD), hay insertar el siguiente permiso en el manifiesto (incluye la lectura):

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


Y solicitar, antes de obtener la foto, al usuario que nos otorgue el permiso.

A continuación hay que crear un archivo con un nombre único, para ello usaremos el siguiente método que crea un nombre a partir de la fecha y hora del momento de la foto (ojo, no estaría preparado para fotos realizadas dentro del mismo segundo) en el archivo público:

```
String pathActualFotos=null;
private File creaNombreUnicoArchivomagen() throws IOException {
    String formatoFecha=new SimpleDateFormat("yyyyMMdd_HH:mm:ss").format(new Date());
    String nombremagen="jpg_" + formatoFecha + "_";
    File directorio= Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES);
    File imagen=File.createTempFile(nombremagen, ".jpg", directorio);
    pathActualFotos = imagen.getAbsolutePath(); // Path para el recoger luego el archivo
    Log.i("Path fotos",pathActualFotos);
    return imagen;
}
```

Ahora, en el segundo botón establecemos el nombre del método en la propiedad *onClick*. En este caso el método será el siguiente.

```
public void getFoto(View v){
    if (checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
        // En caso de no tenerlos se solicita al usuario su concesión
        String[] permisos = new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE};
        ActivityCompat.requestPermissions(MainActivity.this, permisos, 3);
    } else { // Si ya se tienen los permisos se ejecuta de forma normal
        Intent fotoIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        // Aseguramos que se puede lanzar el intent implícito
        if (fotoIntent.resolveActivity(getPackageManager()) != null) {
            File archivomagen = null; // Creamos el fichero donde se guardará la foto
            try {
                archivomagen = creaNombreUnicoArchivomagen();
                if (archivomagen != null) { // Si se ha creado el archivo preparamos el intent con el archivo
                    Uri photoURI = FileProvider.getUriForFile(this, "com.example.camara.fileprovider", archivomagen);
                    fotoIntent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
                    startActivityForResult(fotoIntent, 2); // requestCode 2 para foto normal
                } else {
                    Log.e("ERROR", "No creó el archivo: ");
                }
            } catch (Exception ex) {
                Log.e("ERROR", ex.getMessage());
                ex.printStackTrace();
            }
        }
    }
}
```

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

Finalmente modificamos `onActivityResult` para que al recoger el resultado, procedente del `requestCode` 2, se decodifica la imagen mediante la clase `BitmapFactory` que dispone de diversas funciones estáticas para trabajo con `bitmaps`:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    ImageView iv = (ImageView) findViewById(R.id.imageView);
    if (requestCode == 1 && resultCode == RESULT_OK) { // Thumbnail
        Bitmap imagen = (Bitmap) data.getExtras().get("data");
        iv.setImageBitmap(imagen);
        try { // Guardo la imagen
            FileOutputStream fout = openFileOutput("imagen.jpg", MODE_PRIVATE);
            imagen.compress(Bitmap.CompressFormat.JPEG, 80, fout);
            fout.close();
            // Visualizo las imagenes guardadas
            String[] files = getApplicationContext().fileList();
            for (String file : files) {
                Log.i("imagen -> ", file);
            }
        } catch (IOException e) {
            Log.e("Error imagen", e.getMessage());
        }
    }
    else if (requestCode == 2 && resultCode == RESULT_OK) { // Foto normal
        if (pathActualFotos!=null) {
            Bitmap imagen = BitmapFactory.decodeFile(pathActualFotos);
            Log.i("tamaño", imagen.getWidth()+"."+imagen.getHeight());
            iv.setImageBitmap(imagen);
        } else Log.e("Error", "No hay ruta a la imagen");
    }
}
```

Se debe añadir, además, las siguientes líneas en el fichero de manifiesto dentro de la sección `application`:

```
<provider
    android:name="androidx.core.content.FileProvider"
    android:authorities="com.example.camara.fileprovider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_paths"></meta-data>
</provider>
```

Donde el valor de la línea:

```
android:authorities="com.example.camara.fileprovider "
```

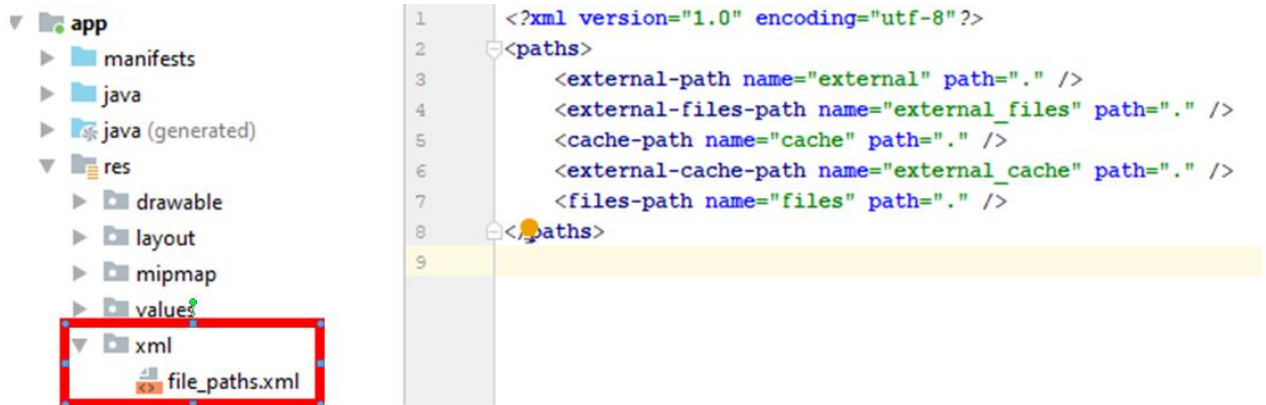
debe coincidir presente en la siguiente línea de `getFoto`:

```
Uri photoURI = FileProvider.getUriForFile(this,"com.example.camara.fileprovider", archivImagen);
```

Por último deberemos crear en fichero de recurso `file_paths.xml` en el directorio `res/xml` con el contenido siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<paths>
    <external-path name="external" path="." />
    <external-files-path name="external_files" path="." />
    <cache-path name="cache" path="." />
    <external-cache-path name="external_cache" path="." />
    <files-path name="files" path="." />
</paths>
```

COLEXIO VIVAS S.L.	RAMA:	Informática	CICLO:	Desenvolvemiento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				




Se puede realizar un escalado de la imagen como indica *Google* en su ejemplo del final del enlace:

<https://developer.android.com/training/camera/photobasics>

O usar otros *layouts* que incluyan un *Scroll*.

Para un uso más exhaustivo se recomienda el enlace:

<https://developer.android.com/training/camera/cameradirect.html?hl=es>

	RAMA:	Informática	CICLO:	Desenvolvimento de Aplicacions Multiplataforma		
	MÓDULO	Programación Multimedia y Dispositivos Móviles				CURSO: 2º
	PROTOCOLO:	Apuntes clases	AVAL:	2	DATA:	2018/2019
	UNIDAD	COMPETENCIA				

7. Ejercicios

Ejercicios

- 1 Realiza una aplicación para trabajar con código *morse* usando el vibrador. Debe tener los siguientes elementos:

- Un botón de punto que al pulsarlo vibre 50ms
- Un botón de raya que al pulsarlo vibre 200ms
- Un botón de *SOS* que debe mandar esas tres letras en *morse* (busca como se escribe)
- Un *Toggle Button* que si está a *On* el *SOS* lo manda de forma continua, y si está a *Off* lo envía solo una vez. Si estuviera enviando y se pasa a off también cancela el envío.

Opcional: Haz que funcione con la luz del flash al mismo tiempo. Debes controlar la duración del flash mediante hilos.

<http://stackoverflow.com/questions/6068803/how-to-turn-on-camera-flash-light-programmatically-in-android>

- 2 Realiza un medidor de luz de forma que por debajo de cierto umbral (decide tu cual) encienda la luz del flash, y por encima del mismo la apague. Además debe realizar las siguientes acciones:

- Al moverlo de lado lanza la cámara de fotos, permite realizar la foto y la pone de fondo en la *activity* principal ocupando la mitad de la misma en ancho y en alto.
- Al moverlo hacia adelante lanza un Mapa indicando la posición actual con un pin.
- Si el movimiento es confuso (diagonal) no hace nada.

- 3 Realiza un juego de llevar una bola (puede ser un icono simple) que está en un extremo de la pantalla a un hueco que está en el extremo opuesto simplemente inclinando en diversos ejes el móvil. Si se sale de los bordes se pierde. Debe guardar el tiempo que le lleva y el nombre del jugador en un archivo de récords. Para contar el tiempo lanza un hilo que cuente segundos y décimas de segundo y pare al llegar a la meta. Debe haber un menú para jugar, ver los records o borrar el archivo de records. Debe vibrar al conseguirlo.