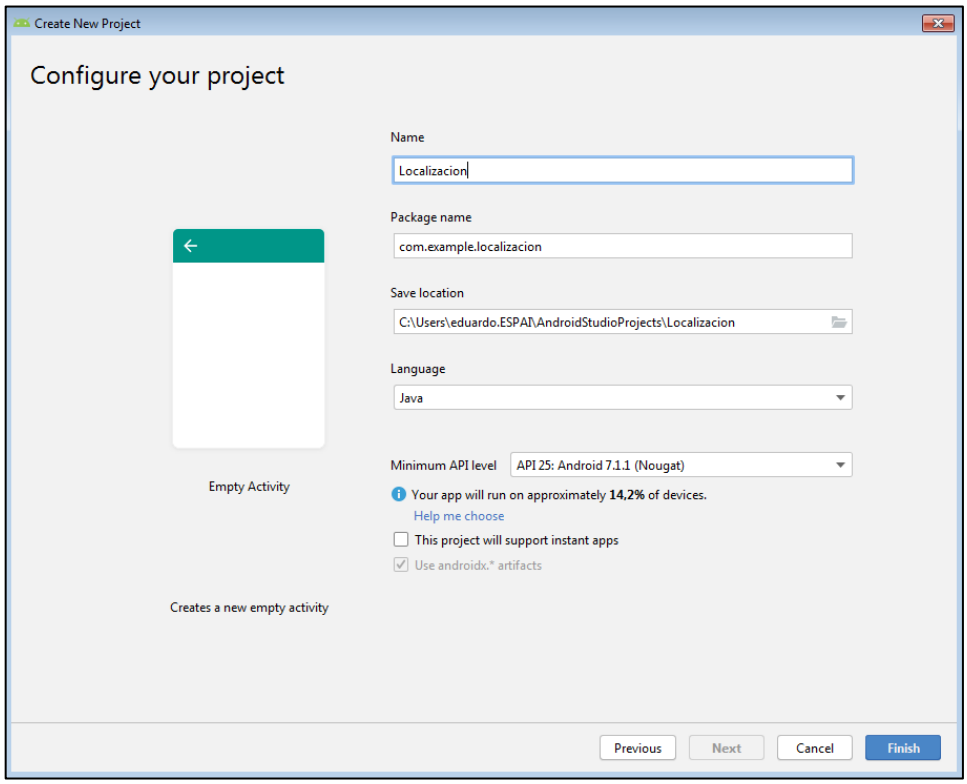
PRACTICA 13: LOCALIZACION

El API de localización de Android

En este ejercicio crearemos una aplicación que es capaz de leer información de localización del dispositivo y actualizarla cada vez que se produce un cambio.

Paso 1. Crea un nuevo proyecto de nombre Localización:



Paso 2. Por razones de privacidad acceder a la información de localización está en principio prohibido a las aplicaciones. Si estas desean hacer uso de este servicio han de solicitar el permiso adecuado. En concreto hay que solicitar ACCESS_FINE_LOCATION para acceder a cualquier tipo de sistema de localización o ACCESS_COARSE_LOCATION para acceder al sistema de localización basado en redes. Añade la siguiente línea en el fichero AndroidManifest.xml dentro de la etiqueta <manifest>:

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

Por lo tanto en este ejemplo vamos a utilizar, tanto la localización fina, que nos proporciona el GPS, como una menos precisa, que nos proporcionada las torres de telefonía celular y las redes WiFi.

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.localizacion">

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

Paso 3. Sustituye el fichero res/layout/activity_main.xml por:

activity_main.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:id="@+id/salida"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
</ScrollView>
```

En este ejemplo nos limitaremos a mostrar en modo de texto la información obtenida desde el API de localización. Como puede observarse nos limitaremos a mostrar un TextView con *scroll*, tal y como se muestra en la siguiente pantalla:

Paso 4. Ve ahora al fichero MainActivity.java y copia el siguiente código:

MainActivity.java

```
public class MainActivity extends AppCompatActivity implements LocationListener {
    static final long TIEMPO_MIN = 10 * 1000; // 10 segundos
    static final long DISTANCIA_MIN = 5; // 5 metros
    static final String[] A = {"n/d", "preciso", "impreciso"};
    static final String[] P = {"n/d", "bajo", "medio", "alto"};
    static final String[] E = {"fuera de servicio",
            "temporalmente no disponible ", "disponible"};
    LocationManager manejador;
    String proveedor;
    TextView salida;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        salida = findViewById(R.id.salida);
        manejador = (LocationManager) getSystemService(LOCATION_SERVICE);
        log("Proveedores de localización: \n ");
        muestraProveedores();

        Criteria criterio = new Criteria();
        criterio.setCostAllowed(false);
        criterio.setAltitudeRequired(false);
        criterio.setAccuracy(Criteria.ACCURACY_FINE);
        proveedor = manejador.getBestProvider(criterio, enabledOnly: true);
        log("Mejor proveedor: " + proveedor + "\n");
        log("Comenzamos con la última localización conocida:");
        Location localizacion = manejador.getLastKnownLocation(proveedor);
        muestraLocaliz(localizacion);
    }

    @Override
    public void onLocationChanged(Location location) {
    }

    @Override
    public void onStatusChanged(String provider, int status, Bundle extras) {
    }

    @Override
    public void onProviderEnabled(String provider) {
    }

    @Override
    public void onProviderDisabled(String provider) {
    }
}
```

La primera línea que nos interesa es la llamada a getSystemService(LOCATION_SERVICE) que crea el objeto manejador de tipo LocationManager.

La siguiente línea hace una llamada al método log() que será definido más adelante. Simplemente saca por el TextView el texto indicado.

La siguiente llamada a muestraProveedores() también es un método definido por nosotros, que listará todos los proveedores de localización disponibles.

En las tres siguientes líneas vamos a seleccionar uno de estos proveedores de localización. Comenzamos creando un objeto de la clase Criteria, donde se podrá indicar las características que ha de tener el proveedor buscado. En este ejemplo indicamosque no ha de tener coste económico, ha de poder obtener la altura y ha de tener precisión fina. Para consultar otras restricciones, consultar documentación de la clase Criteria[1]. Con estas restricciones parece que estamos interesados en el proveedor basado en GPS, aunque de no estar disponible, se seleccionará otro que cumpla el mayor número de restricciones. Para seleccionar el proveedor usaremos el método getBestProvider(). En este método hay que indicar el criterio de selección y un valor booleano, donde indicamos si solo nos interesa los sistemas que el usuario tenga actualmente habilitados. Nos devolverá un Stringcon el nombre del proveedor seleccionado.

Algunos proveedores, como el GPS, puedentardar un cierto tiempo en darnos una primera posición. No obstante, Android recuerda la última posición que fue devuelta por este proveedor. Es lo que nos devuelve la llamada a getLastKnownLocation(). El método muestraLocaliz() será definido más tarde y muestra en pantalla una determinada localización.

Paso 5. Copia a continuación el resto del código:

```
// Métodos del ciclo de vida de la actividad
@Override
protected void onResume() {
    super.onResume();
    manejador.requestLocationUpdates(proveedor, TIEMPO_MIN, DISTANCIA_MIN, listener: this);
}

@Override
protected void onPause() {
    super.onPause();
    manejador.removeUpdates( listener: this);
}

// Métodos de la interfaz LocationListener
@Override
public void onLocationChanged(Location location) {
    log( cadena: "Nueva localización: ");
    muestraLocaliz(location);
}

@Override
public void onProviderDisabled(String proveedor) {
    log( cadena: "Proveedor deshabilitado: " + proveedor + "\n");
}

@Override
public void onProviderEnabled(String proveedor) {
    log( cadena: "Proveedor habilitado: " + proveedor + "\n");
}

@Override
public void onStatusChanged(String proveedor, int estado, Bundle extras) {
    log( cadena: "Cambia estado proveedor: " + proveedor + ", estado="
            + E[Math.max(0, estado)] + ", extras=" + extras + "\n");
}
```

```
    // Métodos para mostrar información
    private void log(String cadena) {
        salida.append(cadena + "\n");
    }

    private void muestraLocaliz(Location localizacion) {
        if (localizacion == null)
            log( cadena: "Localización desconocida\n");
        else
            log( cadena: localizacion.toString() + "\n");
    }

    private void muestraProveedores() {
        log( cadena: "Proveedores de localización: \n ");
        List<String> proveedores = manejador.getAllProviders();
        for (String proveedor : proveedores) {
            muestraProveedor(proveedor);
        }
    }

    private void muestraProveedor(String proveedor) {
        LocationProvider info = manejador.getProvider(proveedor);
        log( cadena: "LocationProvider[ " + "getName=" + info.getName()
                + ", isProviderEnabled="
                + manejador.isProviderEnabled(proveedor) + ", getAccuracy="
                + A[Math.max(0, info.getAccuracy())] + ", getPowerRequirement="
                + P[Math.max(0, info.getPowerRequirement())]
                + ", hasMonetaryCost=" + info.hasMonetaryCost()
                + ", requiresCell=" + info.requiresCell()
                + ", requiresNetwork=" + info.requiresNetwork()
                + ", requiresSatellite=" + info.requiresSatellite()
                + ", supportsAltitude=" + info.supportsAltitude()
                + ", supportsBearing=" + info.supportsBearing()
                + ", supportsSpeed=" + info.supportsSpeed() + " ]\n");
    }

}
```

Para conseguir que se notifiquen cambios de posición hay que llamar al método requestLocationUpdates() y para indicar que se dejen de hacer las notificaciones hay que llamar a removeUpdates(). Dado que queremos ahorrar batería nos interesa que se reporten notificaciones solo cuando la aplicación esté activa. Por lo tanto tenemos que reescribir los métodos onResume() y onPause().

El método requestLocationUpdates() dispone de 4 parámetros: el nombre del proveedor, el tiempo entre actualizaciones en ms (se recomienda valores mayores de 60.000 ms), la distancia mínima (de manera que si es menor, no se notifica) y un objeto LocationListener.

A continuación implementamos los métodos de un LocationListener:onLocationChanged se activará cada vez que cambie la posición. Los otros tres métodos pueden ser usados para cambiar de proveedor en caso de que se active uno mejor o deje de funcionar el actual. Sería buena idea llamar de nuevo aquí al método getBestProvider(). El resto del código resulta fácil de interpretar.

Paso 6. Observa cómo aparecen algunos errores. El sistema nos advierte de que estamos actuando de forma no correcta:

```
// Métodos del ciclo de vida de la actividad
@Override
protected void onResume() {
    super.onResume();
    manejador.requestLocationUpdates(proveedor, TIEMPO_MIN, DISTANCIA_MIN, listener: this);
}
protected void onPause() {
    super.onPause();
    manejador.removeUpdates( listener: this);
}
```

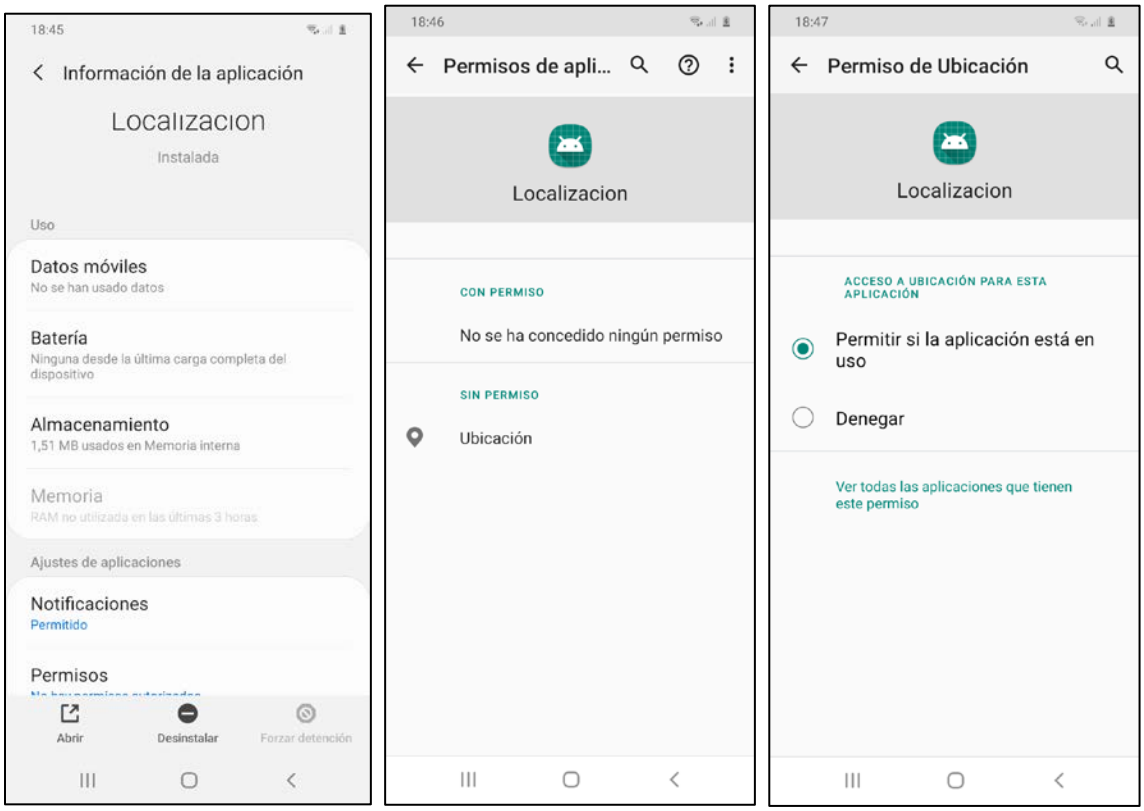
Call requires permission which may be rejected by user: code should explicitly check to see if permission is available (with `checkPermission`) or explicitly handle a potential `SecurityException` more... (Ctrl+F1)

Paso 7. Sobre ese codigo subrayado en rojo, clica en Show Content Actions y añade el checkeo de permisos.

```
protected void onResume() {
    super.onResume();
    manejador.requestLocationUpdates(proveedor, TIEMPO_MIN, DISTANCIA_MIN, listener: this);
}
protected void onPause(){
    super.onPause();
    manejador.removeUpdates( listener: this);
}
```

Add permission check
Suppress MissingPermission with an annotation
Put arguments on separate lines
Press Ctrl+Q to toggle preview

Paso 8. Si ejecutas el proyecto en un dispositivo con una versión 6.0 o superior, podrás verificar que se produce el error. Para evitar que se produzca, en el dispositivo accede a Ajustes / Aplicaciones / Localización / Permisos: y activa el permiso de Ubicación.



Paso 9. Vuelve a ejecutar la aplicación y verificar que ya no se produce el error.

Paso 10. Para aligerar el código del ejercicio no hemos incluido el código necesario para solicitar permiso a partir de la versión 6.0. Si vas a distribuir la aplicación, resulta imprescindible realizar los pasos descritos en la sección Permisos en Android 6 Marshmallow.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    salida = findViewById(R.id.salida);

    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
        ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
            ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.ACCESS_FINE_LOCATION}, requestCode: 42);
    } else {
        resto();
    }

}

public void resto() {
    manejador = (LocationManager) getSystemService(LOCATION_SERVICE);
    log( cadena: "Proveedores de localización: \n ");
    muestraProveedores();
    Criteria criterio = new Criteria();
    criterio.setCostAllowed(false);
    criterio.setAltitudeRequired(false);
    criterio.setAccuracy(Criteria.ACCURACY_FINE);
    proveedor = manejador.getBestProvider(criterio, enabledOnly: true);
    log( cadena: "Mejor proveedor: " + proveedor + "\n");
    log( cadena: "Comenzamos con la última localización conocida:");

    Location localizacion = manejador.getLastKnownLocation(proveedor);
    muestraLocaliz(localizacion);
}
```

```java
@Override
public void onRequestPermissionsResult(int requestCode, String permissions[], int[] grantResults) {
    switch (requestCode) {
        case 42:
            if (grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED) {
                // Permiso garantizado.
                resto();

            } else {
                //Permiso denegado. Desactiva la funcionalidad que depende de este permiso
                Toast.makeText( context: this, text: "Sin el permiso, no puedo realizar la acción",
                        Toast.LENGTH_SHORT).show();
            }
            break;
    }
}
```

```java
// Métodos del ciclo de vida de la actividad
@Override
protected void onResume() {
    super.onResume();
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {

        return;
    }
    manejador.requestLocationUpdates(proveedor, TIEMPO_MIN, DISTANCIA_MIN, listener: this);
}

@Override
protected void onPause() {
    super.onPause();
    if (ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
            ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {

        return;
    }
    manejador.removeUpdates( listener: this);
}
```

Paso 11. Verifica el funcionamiento del programa, si es posible con un dispositivo real con el GPS activado.