Android Avanzado

Sesión 9

Servicios de localización (LBS)

- Location Manager
  - Gestor de todos la funcionalidad de localización que nos da Android
- Location Providers
  - Cada uno representa una tecnología de localización diferente, utilizada para determinar la geolocalización

Location Providers más importantes:

- LocationManager.GPS_PROVIDER
  - Se realiza la localización utilizando comunicación con un satélite GPS
- LocationManager.NETWORK_PROVIDER
  - Se realiza la localización utilizando la red celular.

Ver Providers disponibles por equipo

```
LocationManager locationManager;
    TextView tvi;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        tvi = (TextView) findViewById(R.id.tvi);

        //Obtenemos el manager
        locationManager = (LocationManager)this.getSystemService(LOCATION_SERVICE);

        verProviders();
    }

    private void verProviders(){
    StringBuilder sb = new StringBuilder("Providers");
    sb.append("\n");
    List<String> providers = locationManager.getProviders(true);

    for (String provider : providers ){
        sb.append(provider).append("\n");
    }
    tvi.setText(sb);
    }
```

En el AndroidManifest.xml

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

Seleccionando Provider adecuado

Es muy importante seleccionar el provider a utilizar:
Criterios:

- Power utilizado
  - Si se requiere un consumo mayor de batería
- Exactitud (accuracy)
  - Que tanta exactitud se requiere en la localización
- Capacidades

Creamos una Criteria

```
private Criteria dameCriteria(){
        Criteria criteria = new Criteria();
        criteria.setAccuracy(Criteria.ACCURACY_FINE);
        criteria.setPowerRequirement(Criteria.POWER_LOW);
        criteria.setAltitudeRequired(false);
        criteria.setBearingRequired(false);
        criteria.setSpeedRequired(false);
        criteria.setCostAllowed(true);

         return criteria;
     }
```

```
private void verProviders(Criteria criteria){
        StringBuilder sb = new StringBuilder("Providers");
        sb.append("\n");
        List<String> providers = locationManager.getProviders(criteria, true);

        for (String provider : providers ){
            sb.append(provider).append("\n");
        }
        tvi.setText(sb);
    }
```

Localización

Obtenemos la última posición (para una mejor UX)

```
Location location = locationManager.getLastKnownLocation(provider);
```

Para obtener una ubicación más precisa:

```
locationManager.requestSingleUpdate(provider, locationListener, null);
```

Utilizamos los datos de la localización

```
tviLatitud.setText(String.valueOf(location.getLatitude()));
tviLongitud.setText(String.valueOf(location.getLongitude()));
```

Tracking

Necesitamos definir un listener que escuche los cambios de posición

```
final LocationListener  locationListener = new LocationListener() {

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

        @Override
        public void onProviderEnabled(String provider) {}

        @Override
        public void onProviderDisabled(String provider) {}

        @Override
        public void onLocationChanged(Location location) {}
    };
```

onStatusChanged() : Se llama cuando hay un cambio en el status del provider

onProviderEnabled() : Se llama cuando el provider ha sido habilitado por el usuario
onProviderDisabled() : Se llama cuando el provider ha sido deshabilitado por el usuario

onLocationChanged() : Se llama cuando la posición ha cambiado

Seteamos qué hacer cuando haya un cambio de posición

```
@Override
public void onLocationChanged(Location location) {
    updateLocation(location);
}
```

Seteamos la frecuencia de actualización

```
long minTime = 0;//en milisegundos
long minDistance = 100; // en metros
locationManager.requestLocationUpdates(provider, minTime, minDistance, locationListener);
```

Alertas de proximidad

- Es cuando requerimos que el usuario sea alertado cuando se acerca a un punto dado
- Android controla internamente el tipo de provider utilizado según que tan cerca o lejos estés de tu área objetivo.
- Se selecciona un punto (área objetivo), un radio y un timeout para la alerta.

Seteamos un PendingIntent

```
public static final String MYACTIVITY = "com.urp.myactivity";
Intent intent = new Intent(MYACTIVITY);
  PendingIntent proximityIntent = PendingIntent.getBroadcast(this, -1, intent, 0);
```

Seteamos la posición objetivo

```
        double lat = 73.147536;
        double lng = 0.510638;
        float radio = 100f; // metros
        long expiration = -1; // -1 no va a expirar. Está en milisegundos

        locationManager.addProximityAlert(lat, lng, radio, expiration, proximityIntent);
```

Definimos el broadcast receiver

```
public class ProximityAlertReceiver extends BroadcastReceiver{

    @Override
    public void onReceive(Context context, Intent intent) {
        String key = LocationManager.KEY_PROXIMITY_ENTERING; //True si entra en radio, false si sale

        boolean valor = intent.getBooleanExtra(key, false);

        //Hacer algo
    }

}
```

Pero falta ...

Setear el intent filter para que al momento que se de el intent, se ejecute el broadcast receiver que definimos

Vamos a hacerlo por código (en el activity):

```
//Registramos que al darse el intent, se ejecute el broadcast
IntentFilter intentFilter = new IntentFilter(MYACTIVITY);
registerReceiver(new ProximityAlertReceiver(), intentFilter)
```

Geocoding

- Dos tipos de funciones:
  - Forward Geocoding. Encuentra la latitud y longitud dada una dirección (texto)
  - Reverse Geocoding. Dada una latitud y longitud, la transforma en una dirección (texto)

Forward Geocoding

```
Geocoder fwdGeocoder = new Geocoder(this, Locale.getDefault());
        String direccion = "Universitaria 510 Lima, Peru";
        List<Address> locations = null;
        try {
            locations = fwdGeocoder.getFromLocationName(direccion, 10);
        } catch (IOException e) {
            Log.e("sesion4", "Excepcion : " + e.getMessage());
        }
```

Max resultados

Dada una direccion, obtenemos una lista de posible locations, cada una de esta con su latitud y longitud

Mejora :

```
locations = fwdGeocoder.getFromLocationName(direccion, 10, lowerLeftLatitude,
                                lowerLeftLongitude, upperRightLatitude, upperRightLongitude);
```

Reverse Geocoding

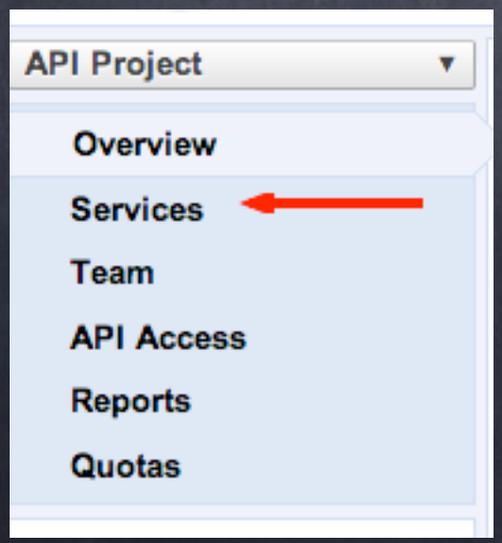
```
location = locationManager.getLastKnownLocation(LocationManager.GPS_PROVIDER);

double latitude = location.getLatitude();
double longitude = location.getLongitude();

Geocoder gc = new Geocoder(this, Locale.getDefault());
List<Address> addresses = null;
try {
    addresses = gc.getFromLocation(latitude, longitude, 10);
} catch (IOException e) {
    Log.e("sesion4", "Excepcion : " + e.getMessage());
}
```

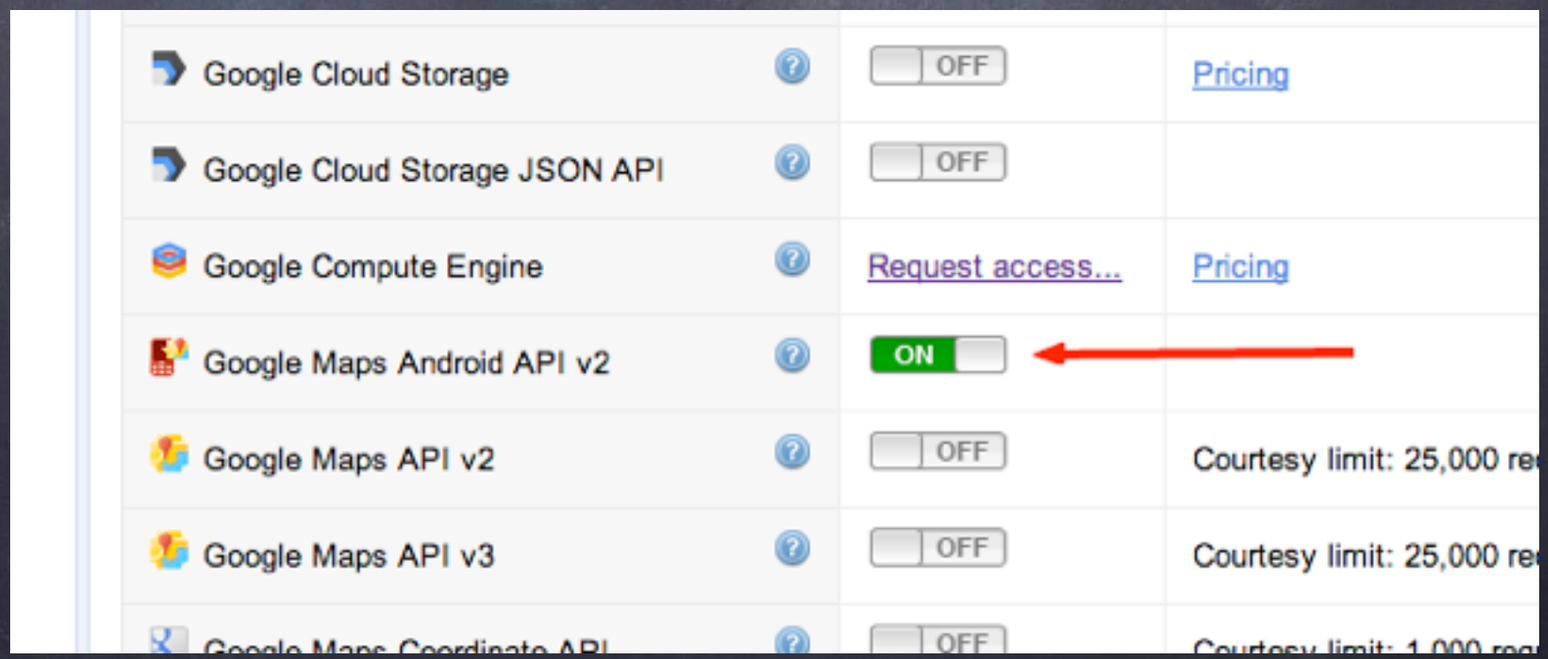
Dada una latitud y longitud, obtenemos una lista de direcciones

Google Maps

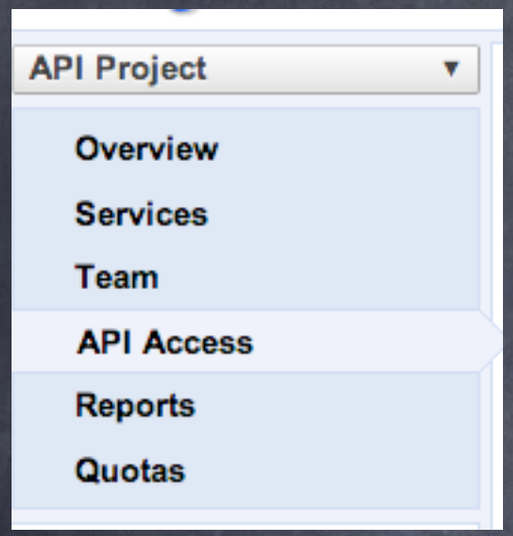
Pasos iniciales

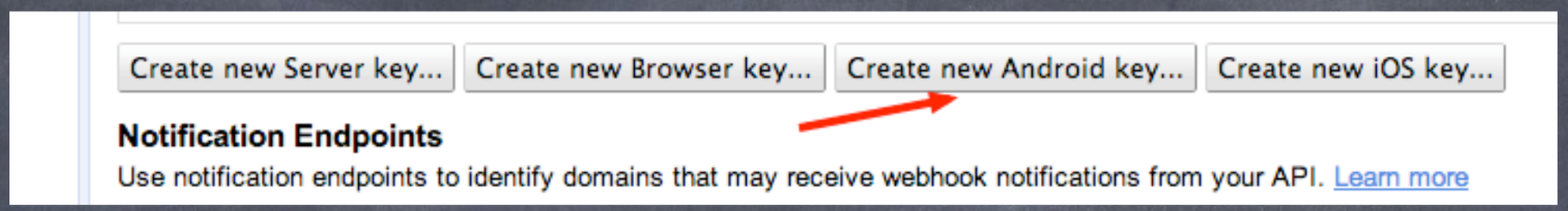
- Configuramos el acceso por el api console
  - Entramos a https://code.google.com/apis/console/
  - Configuramos acceso a Google Maps





Registramos la aplicación





Por la linea de comandos obtenemos SHA1

keytool -list -v -keystore debug.keystore

Registramos el codigo con el paquete de nuestra app

Anotar el API key

Configurar en el AndroidManifest.xml:

```
<uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />


    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>

    <permission
        android:name="com.urp.ejemplo.permission.MAPS_RECEIVE"
        android:protectionLevel="signature"/>
  <uses-permission android:name="com.urp.ejemplo.permission.MAPS_RECEIVE"/>

  <uses-feature
     android:glEsVersion="0x00020000"
     android:required="true"/>

 <application
     android:allowBackup="true"
     android:icon="@drawable/ic_launcher"
     android:label="@string/app_name"
     android:theme="@style/AppTheme" >
    ...
     <meta-data
            android:name="com.google.android.maps.v2.API_KEY"
            android:value="api_key"/>
</application>
```

Versiones Android

Permisos

Verificación de OpenGL2

Api Key

Definimos en el layout, el mapa

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <fragment
        android:id="@+id/fraMapa"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="7"
        android:name="com.google.android.gms.maps.SupportMapFragment"
    />
</LinearLayout>
```

Manejo de mapas

Para animar al centro

```
CameraPosition cameraPosition = new CameraPosition.Builder()
            .target(new LatLng(latitud, longitud))
            .zoom(16)                   // Sets the zoom
            .bearing(0)                // Sets the orientation of the camera to east
            .tilt(30)                   // Sets the tilt of the camera to 30 degrees
             .build();                   // Creates a CameraPosition from the builder
mMap.animateCamera(CameraUpdateFactory.newCameraPosition(cameraPosition));
```

Poner un marcador

```
mMarker = mMap.addMarker(new MarkerOptions()
            .position(new LatLng(mLocation.getLatitude(), mLocation.getLongitude()))
            .title("Mover"));
```

Mas info

- https://developers.google.com/maps/documentation/android/start (versión 2)