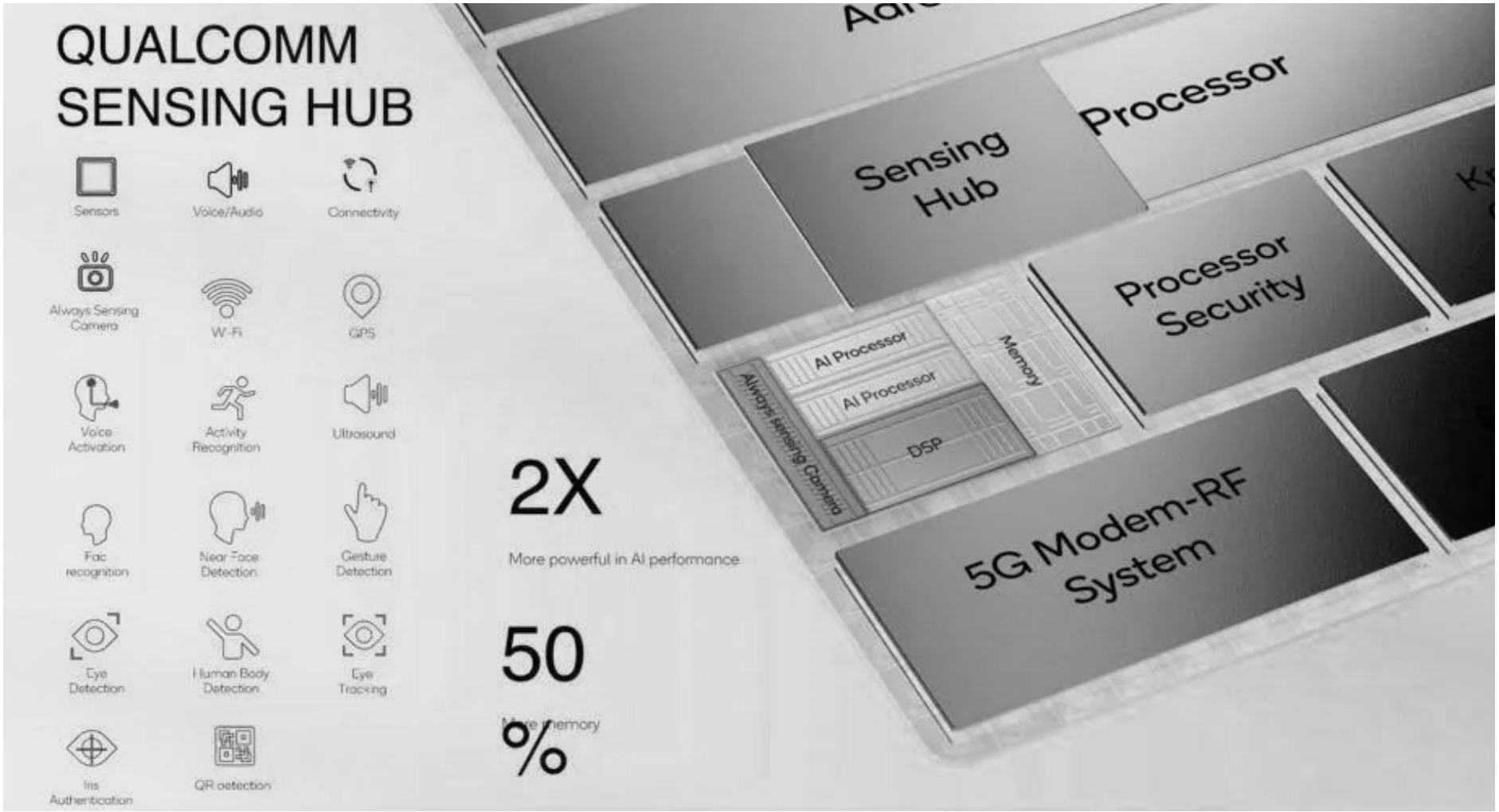
Sensores

1TEL05 - Servicios y Aplicaciones para IoT

Clase 5.2

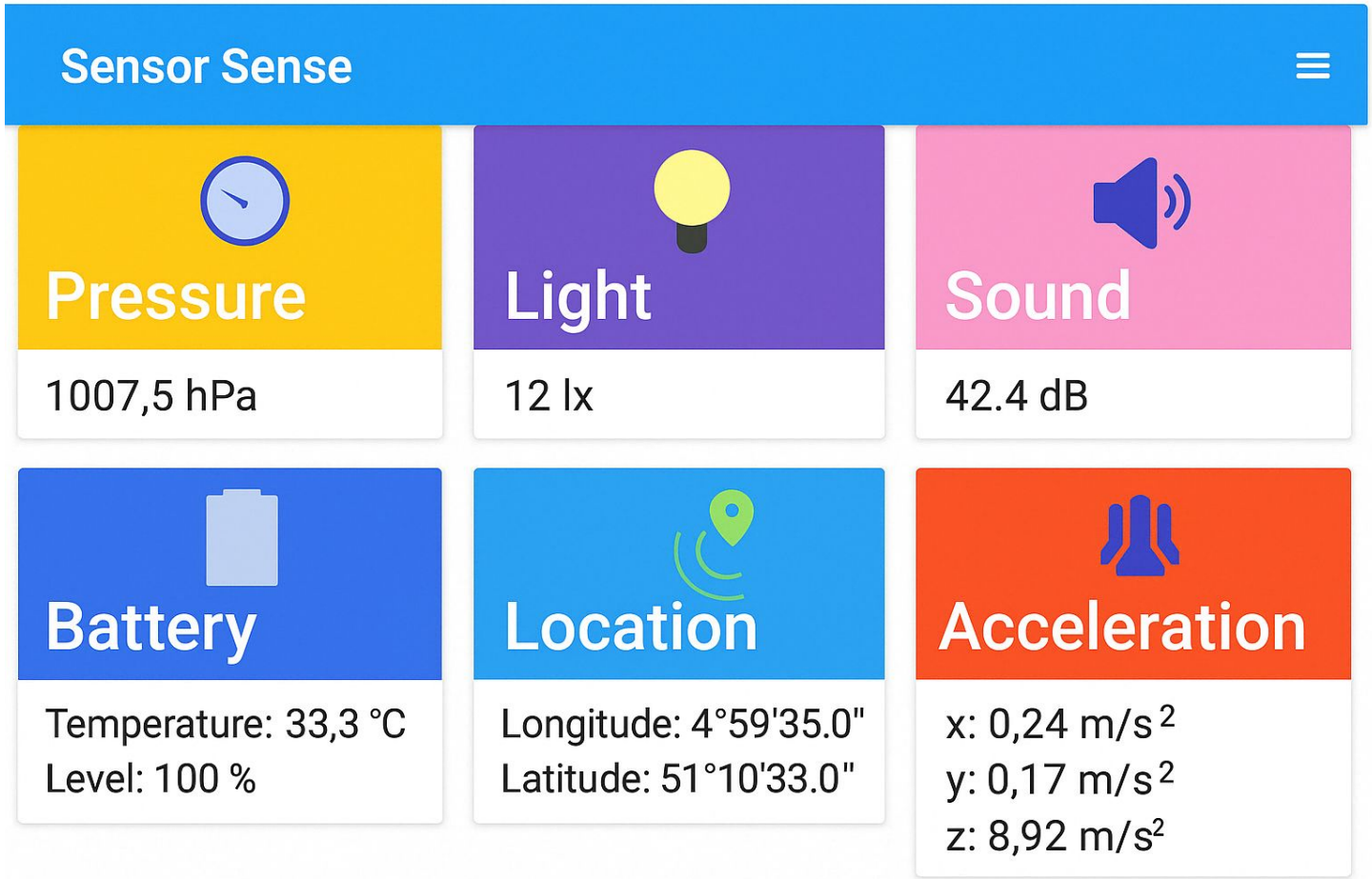
Prof. Oscar Díaz

Sensores



➔ Fuente: https://www.forbes.com/sites/karlfreund/2022/11/16/qualcomm-ups-the-snapgragon-ai-game/

Sensores



➔ Basado en: https://betanews.com/2014/11/09/sensor-sense-monitors-all-your-android-device-sensors/

Sensores

- La mayoría de los dispositivos con Android tienen **sensores integrados** que miden el **movimiento**, la **orientación** y diversas **condiciones ambientales**.
- Estos sensores proporcionan **datos con alta precisión**, son útiles por ejemplo si desea monitorear el posicionamiento tridimensional del dispositivo, o los cambios en el entorno ambiental cerca de un dispositivo.
- Por ejemplo,
    - **Videojuegos**: puede rastrear las lecturas del sensor de gravedad para inferir gestos y movimientos complejos del usuario, como **inclinación**, **agitación** (shake), **rotación** o **balanceo** (swing).
    - **Aplicaciones de brújula:** podría usar el sensor de campo geomagnético y el acelerómetro para mostrar el funcionamiento de una brújula.

Categorías de sensores

Android tiene 3 categorías de sensores:

- **Sensores de movimiento**
- **Sensores ambientales**
- **Sensores de posición**

Todos estos sensores son gestionados por el **“Android Sensor Framework”.**

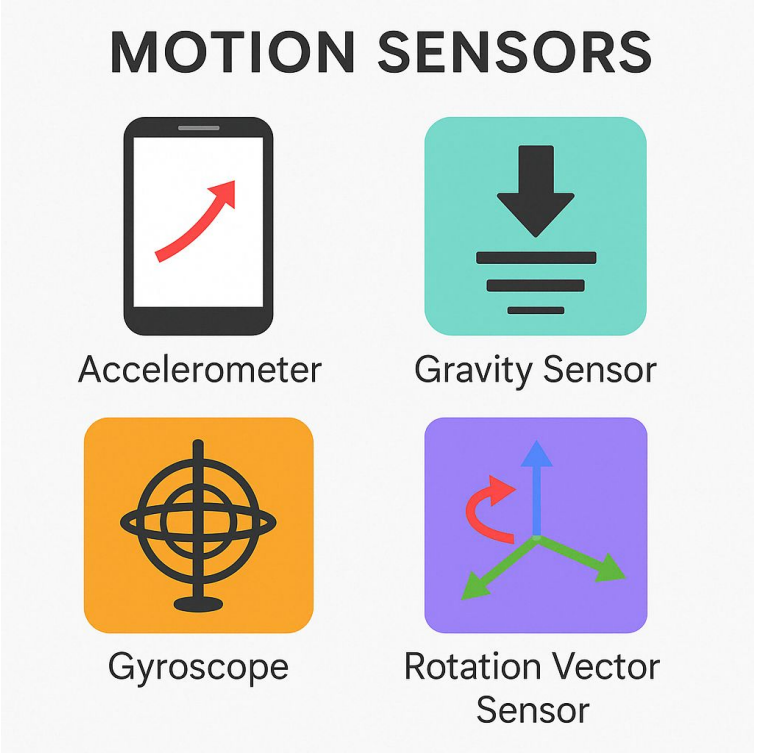
➜ La **cámara** del dispositivo, el **micrófono** y el sensor de **GPS** (ubicación) tienen sus propias API y por ende, no se consideran "sensores".

Sensores de movimiento (Motion Sensor)

Mide el movimiento del dispositivo.

Estos sensores incluyen:

- Acelerómetros
- Sensores de gravedad
- Giroscopios
- Sensores vectoriales de rotación

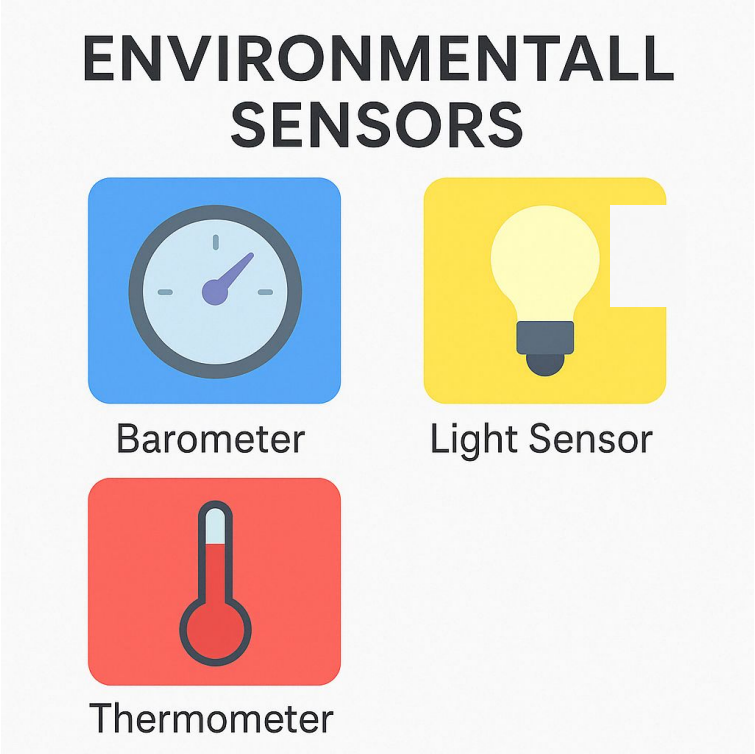


Sensores ambientales (Environmental sensor)

Mide las condiciones ambientales, como temperatura y presión del aire, iluminación y humedad.

Estos sensores incluyen:

- Barómetros
- Fotómetros (sensores de luz)
- Termómetros

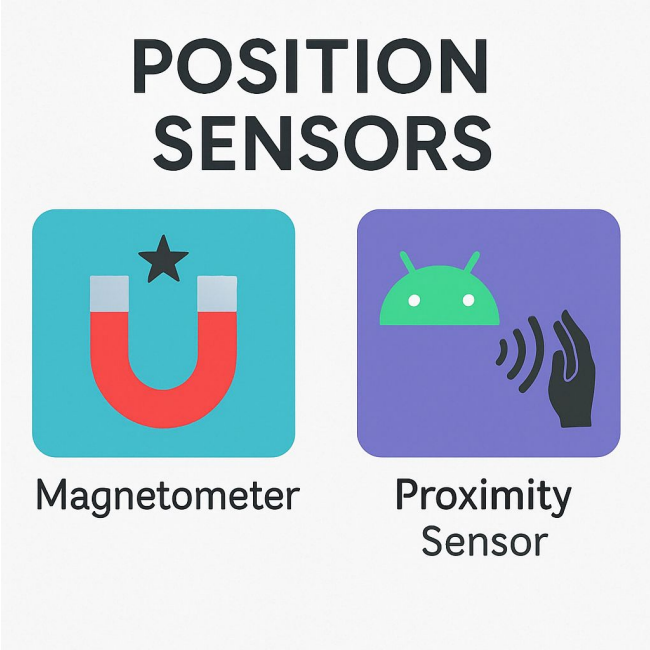


Sensores de posición (Position Sensor)

Mide la posición física de un dispositivo.

Estos sensores incluyen:

- Magnetómetros (sensores de campo geomagnético)
- Sensores de proximidad.



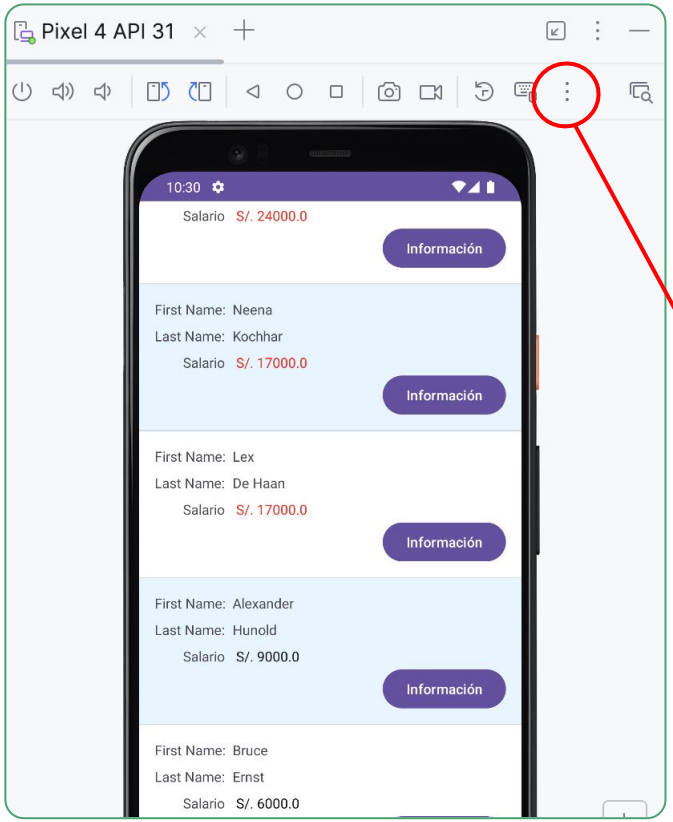
Tipos de sensores

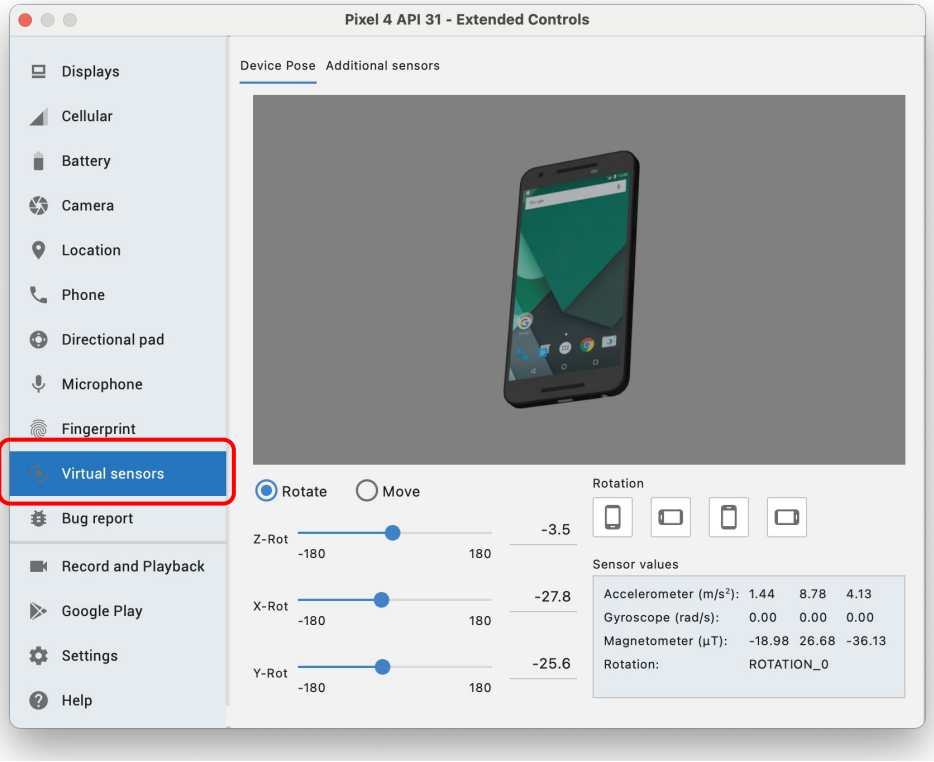
Existen de dos tipos:

- **Basados en hardware**: son **componentes físicos integrados en el teléfono o tableta**. Obtienen sus datos midiendo directamente propiedades ambientales específicas, como: la aceleración, la intensidad del campo geomagnético o el cambio angular.
- **Basados en software**: **no son dispositivos físicos**, aunque imitan sensores basados en hardware. Derivan sus datos de uno o más de los sensores basados en hardware y a veces se los denomina **sensores virtuales** o **sensores compuestos**. El sensor de proximidad y los sensores de contador de pasos son ejemplos de sensores basados en software.

Sensores en el AVD (Android virtual device)

Si no dispone de un dispositivo móvil para probar los sensores que podría utilizar en su aplicación, el emulador de Android permite simular los mismos.

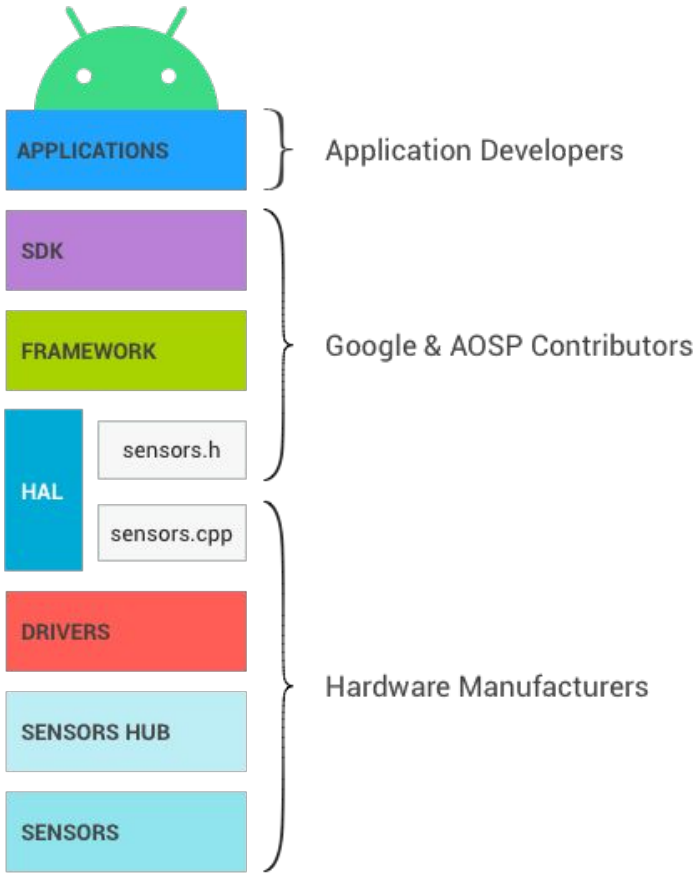




Android Sensor Framework

Android Sensor Framework

- Conjunto de **clases e interfaces** que permiten **obtener** información y gestionar la **data de los sensores del equipo** en nuestra aplicación.
- Son parte del paquete **`android.hardware`**
- Entre las funcionalidades que brinda son:
  - Determinar **qué sensores están disponibles** en el dispositivo
  - Determinar las **capacidades del sensor**, tales como rango máximo, fabricante, energía, resolución, entre otros.
  - Adquirir la **data del sensor** e inclusive (según el sensor) **la velocidad (frecuencia) de adquisición.**
  - Registrar "**listeners**" para supervisar los cambios en los sensores.



➔ Fuente: https://source.android.com/docs/core/interaction/sensors/sensor-stack?hl=es

Clases e interfaces de Android Sensor Framework

`SensorManager`

- Acceso a los sensores
- Permite registrar los “listeners” a los sensores
- Adquirir información de la orientación del dispositivo
- Proporciona las constantes para precisión, frecuencia de adquisición de datos y calibración.

Clases e interfaces de Android Sensor Framework

`Sensor`

- Representa un sensor en particular
- Muestra la capacidades del sensor

`SensorEvent`

- Representa la información de los eventos de un sensor, incluida su data.

`SensorEventListener`

- Recibe notificaciones cuando sucede un evento en el sensor.
  - Cuando hay nueva data
  - Cuando la precisión del sensor cambia.

Clase Sensor - Tipos de sensores

La clase Sensor define los diferentes tipos de sensores disponibles en el dispositivo y que es utilizado para verificar su existencia. Algunos ejemplos:

TYPE_ACCELEROMETER	Detecting motion (shake, tilt, etc.)
TYPE_AMBIENT_TEMPERATURE	Monitoring air temperature
TYPE_GRAVITY	Detecting motion (shake, tilt, etc.)
TYPE_GYROSCOPE	Detecting rotation (spin, turn, etc.)
TYPE_LIGHT	Controlling screen brightness
TYPE_LINEAR_ACCELERATION	Monitoring acceleration along single axis
TYPE_MAGNETIC_FIELD	Creating a compass

➔ La lista completa en: https://developer.android.com/reference/android/hardware/Sensor.html

Usando los sensores

Pasos a seguir

Por lo general, los pasos a seguir para trabajar con el Android Sensor Framework son:

1. Determinar si el sensor está disponible en el dispositivo
2. Registrar “listeners” para los sensores
3. Adquirir data de los sensores
4. Borrar los “listeners” para los sensores

Determinar si el sensor está disponible

Existen dos formas de validar si un sensor está presente en su dispositivo:

- Usando el Android Sensor Framework
- Usando los filtros de Google Play

Determinar si el sensor está disponible
➜ con Android Sensor Framework

Utilice este método si su aplicación no depende del sensor para funcionar, es decir, puede funcionar y el uso del sensor es opcional.

Instanciar la clase **`SensorManager`**.

```
SensorManager mSensorManager = (SensorManager) getSystemService(SENSOR_SERVICE);
```

Utilice **`getSystemService`** para obtener un servicio del sistema y al pasarle **`SENSOR_SERVICE`**, nos brindara información de los sensores.

Determinar si el sensor está disponible
➜ con Android Sensor Framework

Usando el método **`getDefaultSensor()`**, puede obtener información particular sobre un sensor que necesite. Mire la diapositiva 13 para la lista.

```
if(mSensorManager != null){ //validar si tengo sensores

    Sensor acelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);

    if(acelerometer != null){ //validar un sensor en particular
        Toast.makeText( context: this,  text: "Sí tiene acelerómetro", Toast.LENGTH_SHORT).show();
    }else{
        Toast.makeText( context: this,  text: "Su equipo no dispone de acelerómetro",Toast.LENGTH_SHORT).show();
    }
```

Determinar si el sensor está disponible
➜ con Android Sensor Framework

También es posible listar todos los sensores disponibles con **`getSensorList(Sensor.TYPE_ALL)`**:

```
if(mSensorManager != null){ //validar si tengo sensores

    List<Sensor> sensorList = mSensorManager.getSensorList(Sensor.TYPE_ALL);
    for(Sensor sensor : sensorList){
        Log.d( tag: "msg-test-sensorList", msg: "sensorName: " + sensor.getName());
    }
```

```
sensorName: Goldfish 3-axis Accelerometer
sensorName: Goldfish 3-axis Gyroscope
sensorName: Goldfish 3-axis Magnetic field sensor
sensorName: Goldfish Orientation sensor
sensorName: Goldfish Ambient Temperature sensor
sensorName: Goldfish Proximity sensor
sensorName: Goldfish Light sensor
sensorName: Goldfish Pressure sensor
sensorName: Goldfish Humidity sensor
sensorName: Goldfish 3-axis Magnetic field sensor
sensorName: Game Rotation Vector Sensor
```

Determinar si el sensor está disponible
➜ con filtros de Google Play

Si su aplicación depende del sensor para funcionar, entonces, aparte de validar con el Android Sensor Framework, puede restringir que las personas puedan descargar del Play Store su aplicación si no tienen el sensor requerido.

➜ Debe utilizar el tag <uses-feature>

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.clase8">

    <uses-feature android:name="android.hardware.sensor.accelerometer" android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
```

Registrar Listeners

Android genera eventos sobre los sensores, cada vez que éstos tienen nueva data. Para monitorear estos eventos, es necesario:

- Implementar la interfaz **`SensorEventListener`**
- Registrar el **`listener`** en el sensor que se desea monitorear.
- Obtener los valores del **`SensorEvent`** y actualizar la aplicación.
- Desregistrar el **`listener`**.

Implementar la interfaz **`SensorEventListener`**

Debe implementar la interfaz **`SensorEventListener`**.

Tendrá que sobreescribir los dos métodos abstractos:

- **`onSensorChanged()`**: se dispara cuando el sensor tiene nueva data, la cual contiene (en el objeto SensorEvent):
  - Precisión de la data
  - Sensor que generó la data
  - Timestamp
  - La data en sí.
- **`onAccuracyChanged()`**: se dispara cuando la precisión del sensor cambia, con 5 valores:
  - *SENSOR_STATUS_ACCURACY_HIGH*
  - *SENSOR_STATUS_ACCURACY_MEDIUM*
  - *SENSOR_STATUS_ACCURACY_LOW*
  - *SENSOR_STATUS_UNRELIABLE (valores muy bajos o inexactos)*
  - *SENSOR_STATUS_NO_CONTACT (no brinda información)*

Ejemplo

La clase implementa la interfaz **SensorEventListener** y se crea una variable para gestionar el **sensorManager**.

```
public class MainActivity extends AppCompatActivity implements SensorEventListener
{

    ActivityMainBinding binding;
    SensorManager mSensorManager;
```

Registrar listeners

El registro del listener se debe realizar en el método **`onResume()`** y des-registrarlo en **`onStop()`**. No se realiza en **`onCreate()`**, pues la aplicación estaría sensando y obteniendo datos aún cuando no ha terminado de mostrarse en la pantalla.

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

    Sensor mAccelerometer = mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    mSensorManager.registerListener(listener,mAccelerometer,SensorManager.SENSOR_DELAY_UI);
}

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

    mSensorManager.unregisterListener(listener);
}
```

Registrar listeners

Al registrar el listener se le indican 3 parámetros:

1. La clase que implementa la interfaz (la misma actividad)
2. El tipo de sensor que se está midiendo
3. El intervalo en que el sensor le enviará data a la aplicación:
   - **SENSOR_DELAY_UI → 60 ms**
   - **SENSOR_DELAY_NORMAL → 200ms**
   - **SENSOR_DELAY_GAME → 20ms**
   - **SENSOR_DELAY_FASTEST → 0ms**

   → Menos delay, más consumo de batería

   → Desde el API 11, puede especificar el valor en microsegundos.

```
mSensorManager.registerListener(listener,mAccelerometer,SensorManager.SENSOR_DELAY_UI);
```

1 2 3

onSensorChanged()

Aquí se gestionan las notificaciones del sistema sobre el sensor. Para cada sensor Android envía información particular la cual está definida en:

➜ https://developer.android.com/guide/topics/sensors/sensors_motion.html

Sensor	Sensor event data	Description	Units of measure
TYPE_ACCELEROMETER	SensorEvent.values[0]	Acceleration force along the x axis (including gravity).	m/s^2
	SensorEvent.values[1]	Acceleration force along the y axis (including gravity).	
	SensorEvent.values[2]	Acceleration force along the z axis (including gravity).	
TYPE_ACCELEROMETER_UNCALIBRATED	SensorEvent.values[0]	Measured acceleration along the X axis without any bias compensation.	m/s^2
	SensorEvent.values[1]	Measured acceleration along the Y axis without any bias compensation.	
	SensorEvent.values[2]	Measured acceleration along the Z axis without any bias compensation.	
	SensorEvent.values[3]	Measured acceleration along the X axis with estimated bias compensation.	
	SensorEvent.values[4]	Measured acceleration along the Y axis with estimated bias compensation.	
	SensorEvent.values[5]	Measured acceleration along the Z axis with estimated bias compensation.	
TYPE_GRAVITY	SensorEvent	Force of gravity along the x axis	m/s^2

onSensorChanged()

Se debe validar qué sensor está enviando la información pues se puede tener un listener para todos los sensores.

En base a eso, se pueden realizar acciones:

```
@Override
public void onSensorChanged(SensorEvent sensorEvent) {
    int sensorType = sensorEvent.sensor.getType();

    if(sensorType == Sensor.TYPE_ACCELEROMETER){
        float x = sensorEvent.values[0];
        float y = sensorEvent.values[1];
        float z = sensorEvent.values[2];

        Log.d(TAG, msg: "x: " + x + " | y: " + y + " | z: " + z);
    }
}
```

Location

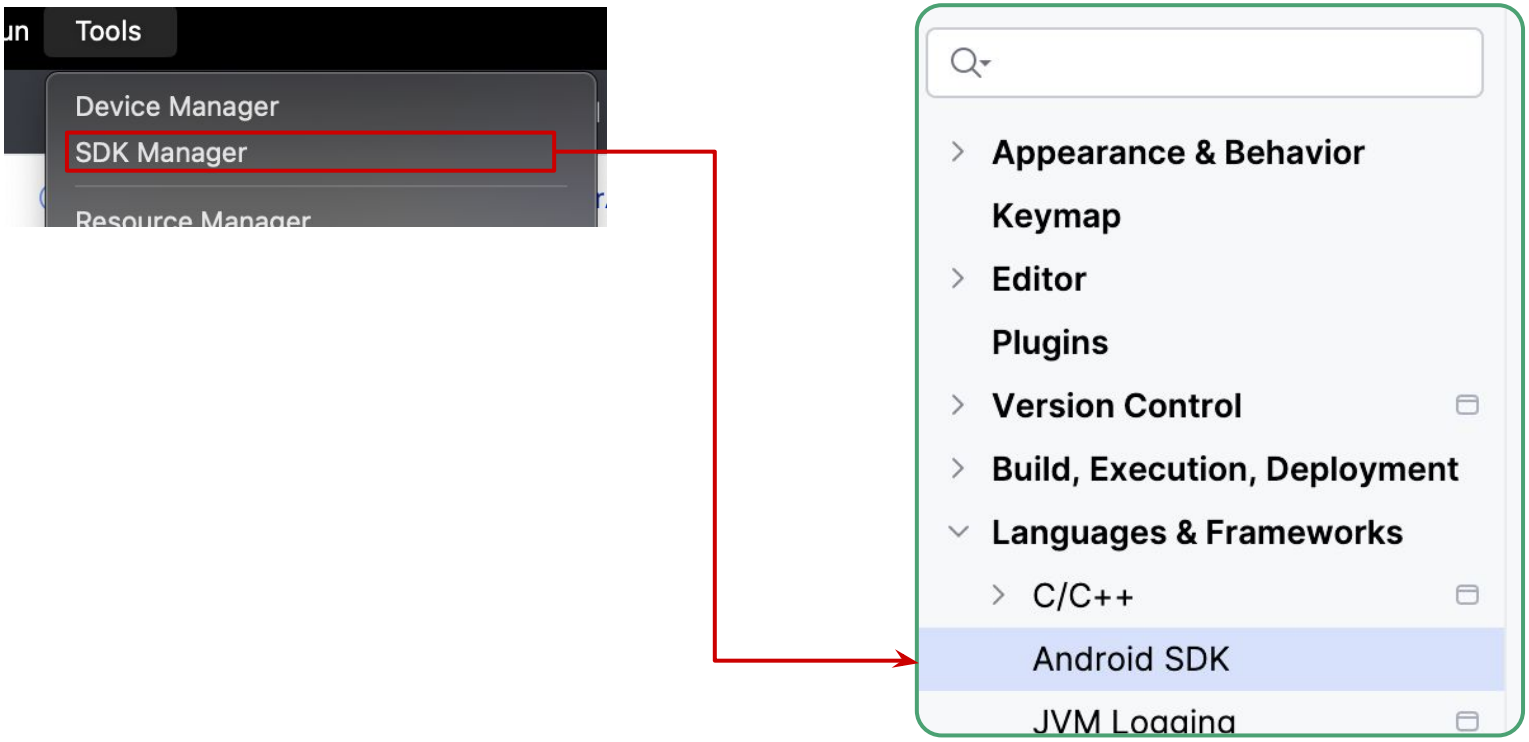
Location

- El sistema Android permite obtener la ubicación del dispositivo combinando la información brindada por el GPS, Wifi y red celular.
- Google Play Services brinda la clase **`FusedLocationProviderClient`**, la cual estima la ubicación del usuario fusionando (“fusing”) todas la data de los sensores de ubicación (Gps, wifi y celular). Esto permite resultados precisos con consumo mínimo de batería.
- Esta clase le brinda mediante un objeto **`Location`** las coordenadas geográficas en forma de latitud y longitud

Configurando el Google Play Services

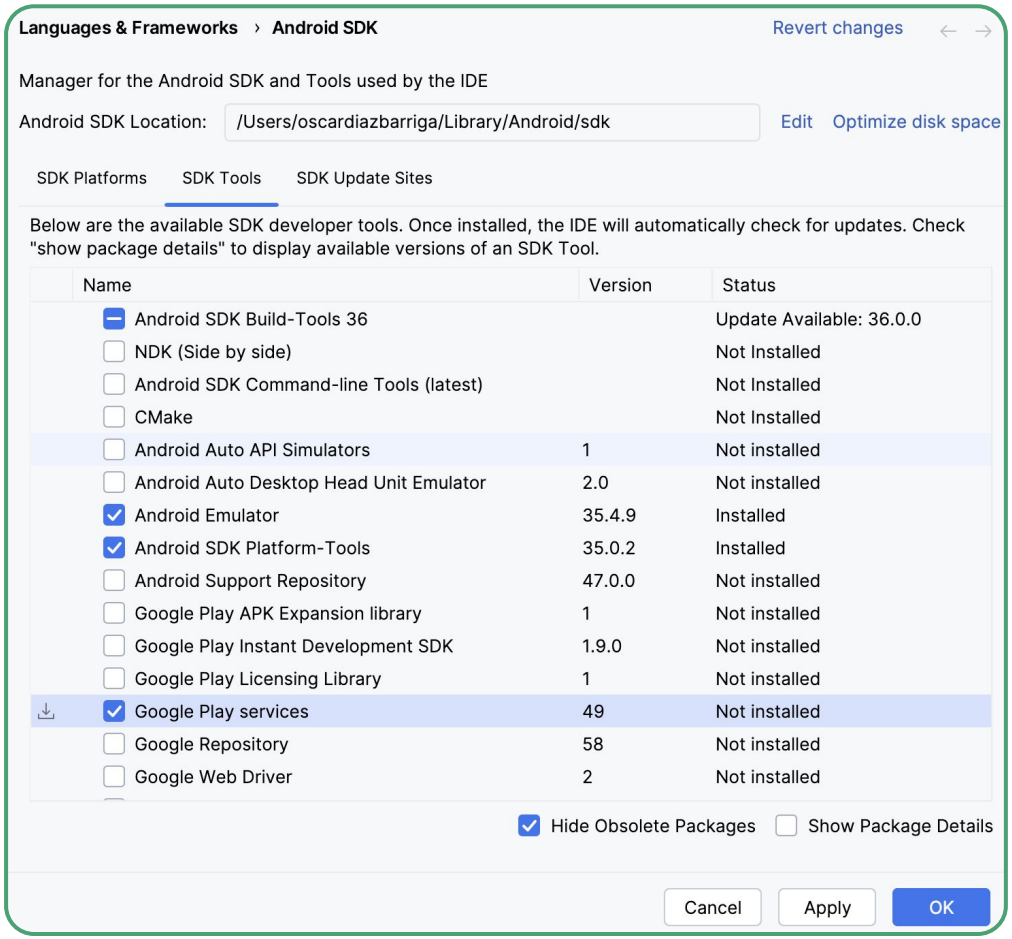
Para poder usar el **`FusedLocation`** es necesario utilizar los **servicios de Google Play**, los cuales primero deben ser adicionados mediante el SDK y luego a Gradle.

➜ Tools ➜ SDK Manager ➜ Android SDK



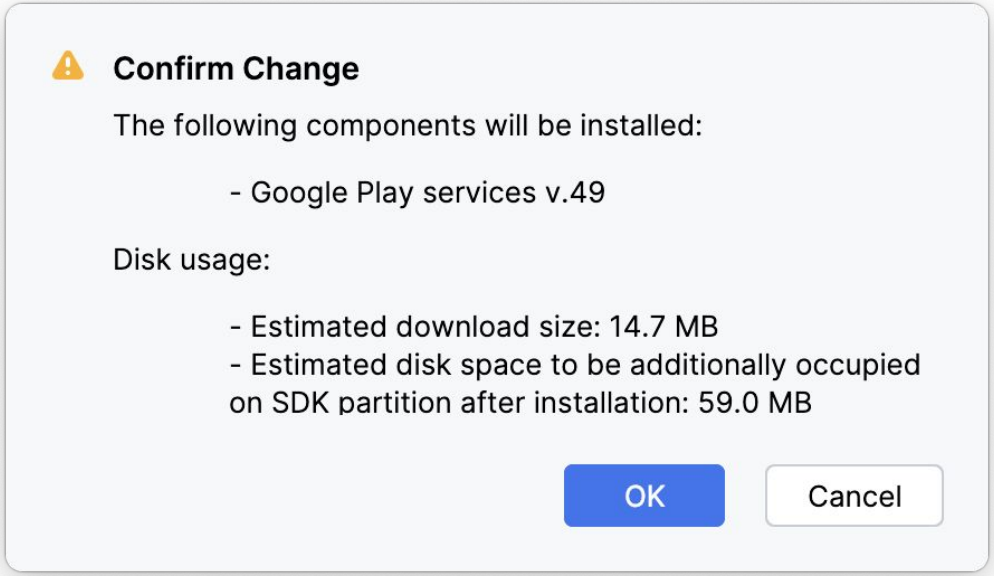
Agregando Google Play Services

En la pestaña **SDK Tools**, seleccione “**Google Play services**” y luego “**Apply**”



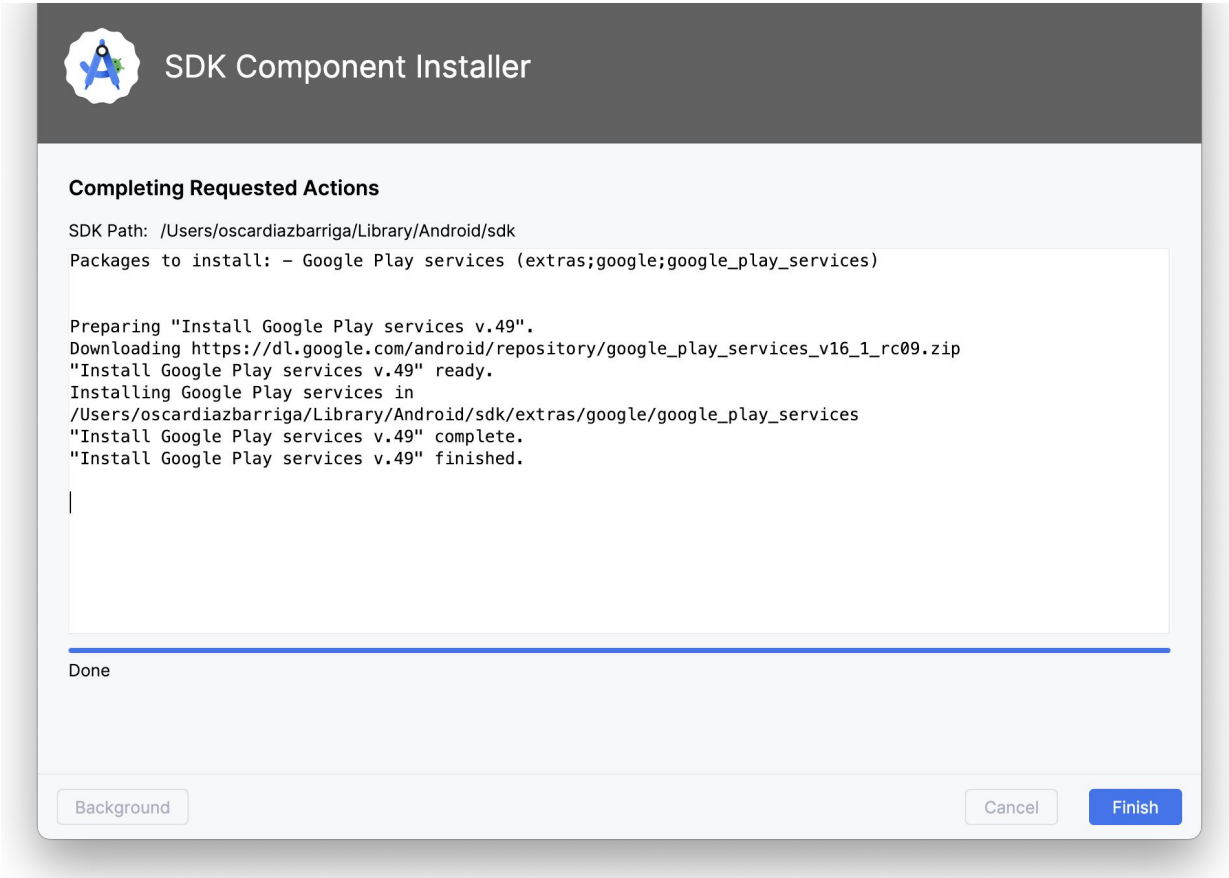
Agregando Google Play Services

Instale los servicios



Agregando Google Play Services

Espere que termine de instalar y presione “**Finish**”



Gradle

Adicionar en Gradle la librería:

- implementation 'com.google.android.gms:play-services-location:21.0.1'

Permisos en Manifest

Para poder acceder a la ubicación, es necesario solicitar permisos en el Manifest.

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

Son necesarios ambos permisos si se desea una precisión alta. Si solo se desea una precisión baja, basta con COARSE_LOCATION.

Estos permisos brindan la opción de obtener su ubicación en un instante de tiempo.

Si lo que desea es enviar su ubicación constantemente o gestionarla en background (como whatsapp), puede utilizar: foreground location o background location.

Mostrar ubicación

Siempre que se desee se debe validar primero que se cuentan con los permisos, esto se logra con **`ActivityCompat.checkSelfPermission()`**

```
public void mostrarUbicacion() {

   int selfPermissionFineLocation = ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION);
   int selfPermissionCoarseLocation = ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_COARSE_LOCATION);

   if (selfPermissionFineLocation == PackageManager.PERMISSION_GRANTED &&
           selfPermissionCoarseLocation == PackageManager.PERMISSION_GRANTED) {

       //tenemos permisos
   } else {
       //no tenemos permisos, se deben solicitar
   }

}
```

Mostrar ubicación

Si se tienen los permisos, se puede mostrar la ubicación utilizando la clase **`FusedLocationProviderClient`**. y registrando un listener.

```
//tenemos permisos
FusedLocationProviderClient providerClient = LocationServices.getFusedLocationProviderClient(this);
providerClient.getLastLocation().addOnSuccessListener(this, location -> {
    if(location != null){
        Log.d("msg-test","latitud: " + location.getLatitude());
        Log.d("msg-test","longitud: " + location.getLongitude());
    }
});
```

Solicitar permisos - creación del callback

En caso no tengamos permisos, estos se deben solicitar usando un callback **`registerForActivityResult`.**

```
ActivityResultLauncher<String[]> locationPermissionLauncher = registerForActivityResult(
        new ActivityResultContracts.RequestMultiplePermissions(),
        result -> {
            Boolean fineLocationGranted = result.get(Manifest.permission.ACCESS_FINE_LOCATION);
            Boolean coarseLocationGranted = result.get(Manifest.permission.ACCESS_COARSE_LOCATION);
            if (fineLocationGranted != null && fineLocationGranted) {
                Log.d("msg", "Permiso de ubicación precisa concedido" );
                mostrarUbicacion();
            } else if (coarseLocationGranted != null && coarseLocationGranted) {
                Log.d("msg", "Permiso de ubicación aproximada concedido" );
            } else {
                Log.d("msg", "Ningún permiso concedido" );
            }
        }
);
```

Si obtenemos el permiso, mostramos la ubicación

Solicitar permisos - llamado al callback

Luego de crear el callback, ya se le puede llamar luego de validar los permisos.

```
public void mostrarUbicacion(View view) {

    int selfPermissionFineLocation = ActivityCompat.checkSelfPermission( context: this,
            android.Manifest.permission.ACCESS_FINE_LOCATION);
    int selfPermissionCoarseLocation = ActivityCompat.checkSelfPermission( context: this,
            android.Manifest.permission.ACCESS_COARSE_LOCATION);

    if (selfPermissionFineLocation == PackageManager.PERMISSION_GRANTED &&
            selfPermissionCoarseLocation == PackageManager.PERMISSION_GRANTED) {

        //tenemos permisos
        FusedLocationProviderClient providerClient =
                LocationServices.getFusedLocationProviderClient( activity: this);
        providerClient.getLastLocation().addOnSuccessListener( activity: this, Location location -> {
            if (location != null) {
                Log.d( tag: "msg-test-location",
                        msg: "latitud: " + location.getLatitude()
                        + " | longitud: " + location.getLongitude());
            }
        });

    } else {
        //no tenemos permisos, se deben solicitar
        locationPermissionLauncher.launch(new String[]{
                android.Manifest.permission.ACCESS_FINE_LOCATION,
                android.Manifest.permission.ACCESS_COARSE_LOCATION
        });

    }
}
```

Observación

Un emulador no tiene el GPS en sí, por lo que es más probable que **`getLastLocation()`** devuelva nulo. Para obligar al emulador a obtener una nueva ubicación, inicie la aplicación **Google Maps** y acepte los términos y condiciones (si aún no lo ha hecho).

Use la pestaña Ubicación de la configuración del emulador para entregar una ubicación al dispositivo y luego vaya **Google Maps** para ver que se actualizo la posición. Esto obliga a almacenar en caché un valor y **`getLastLocation()`** ya no devuelve nulo.

Location + Google Maps

Ahora que conoce cómo geolocalizar en su aplicación, podría integrar con Google maps.

Revise este guía:
https://developers.google.com/maps/documentation/android-sdk/start

Inclusive, podría trackear el movimiento del usuario en su aplicación con un LocationRequest, el cual permite definir intervalos de medición de la ubicación.

¿Preguntas?

Muchas gracias