Manejo de GPS (Ubicación) en Android Studio con Java

1. Configuración de Dependencias y Permisos

Para usar el servicio de ubicación de Google, necesitas la biblioteca de **Play Services Location**.

A. Dependencia en app/build.gradle

Asegúrate de tener la dependencia para el servicio de ubicación (y la de Maps, si aún no la tienes, ya que suelen ir de la mano):

Groovy

```
dependencies {
    // ... otras dependencias
    // Dependencia clave para la ubicación y GPS
    implementation 'com.google.android.gms:play-services-location:21.0.1'
}
```

B. Permisos en AndroidManifest.xml

Necesitas permisos de ubicación peligrosos. Es mejor solicitar el permiso de alta precisión (FINE_LOCATION).

XML

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

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

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

    <application ...>
        </application>
</manifest>
```

---

2. Manejo de Permisos en Tiempo de Ejecución (Runtime)

Como **ACCESS_FINE_LOCATION** es un permiso peligroso, debes solicitarlo al usuario antes de acceder al GPS.

Java

```java
// MainActivity.java
private static final int LOCATION_PERMISSION_REQUEST_CODE = 1000;

private void checkLocationPermission() {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
            != PackageManager.PERMISSION_GRANTED) {

        ActivityCompat.requestPermissions(this,
                new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                LOCATION_PERMISSION_REQUEST_CODE);
    } else {
        // Permiso concedido, puedes iniciar la solicitud de ubicación
        requestLocationUpdates();
    }
}

// Sobrescribir onRequestPermissionsResult para manejar la respuesta del usuario.
// En caso de concesión, llamar a requestLocationUpdates().
```

---

3. Uso del Fused Location Provider (Obtener Ubicación)

El **Fused Location Provider** gestiona de forma inteligente el uso del GPS, Wi-Fi, y otras redes para devolver la mejor ubicación posible, optimizando el uso de batería.

A. Componentes Clave

1. **FusedLocationProviderClient**: El punto de entrada para el API.
2. **LocationRequest**: Define la calidad del servicio (precisión, frecuencia, consumo de batería).
3. **LocationCallback**: Maneja los resultados de las actualizaciones de ubicación.

B. Código Java para Iniciar Solicitudes

Java

```
// MainActivity.java
private FusedLocationProviderClient fusedLocationClient;
private LocationRequest locationRequest;
private LocationCallback locationCallback;

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

    // 1. Inicializar el cliente
    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this);

    // 2. Definir la solicitud de ubicación (LocationRequest)
    locationRequest = LocationRequest.create();
    locationRequest.setInterval(10000); // Intervalo deseado de 10 segundos
    locationRequest.setFastestInterval(5000); // Intervalo más rápido de 5 segundos
    locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY); // Usar GPS
```

```
    // 3. Configurar el callback para manejar los resultados
    locationCallback = new LocationCallback() {
        @Override
        public void onLocationResult(LocationResult locationResult) {
            if (locationResult == null) {
                return;
            }
            // Itera sobre todas las ubicaciones recibidas
            for (Location location : locationResult.getLocations()) {
                // Aquí se maneja la nueva ubicación (latitud y longitud)
                String latLng = location.getLatitude() + ", " + location.getLongitude();
                Log.d("GPS", "Ubicación actual: " + latLng);
                // Ejemplo: actualizar un TextView con la ubicación
                // tvLocation.setText(latLng);
            }
        }
    };

    checkLocationPermission(); // Iniciar el proceso de permisos
}

// Método para iniciar las actualizaciones de ubicación
private void requestLocationUpdates() {
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED) {

        fusedLocationClient.requestLocationUpdates(locationRequest,
                locationCallback,
                Looper.getMainLooper()); // Ejecutar el callback en el hilo principal
    }
```

```
}
```

---

4. Finalizar las Solicitudes (Liberación de Recursos)

Para ahorrar batería, es crucial **detener la escucha de ubicación** cuando la aplicación ya no la necesite (ej. cuando la actividad pasa a segundo plano).

Java

```
@Override
protected void onPause() {
    super.onPause();
    // Detener las actualizaciones de ubicación al salir de la actividad
    stopLocationUpdates();
}

@Override
protected void onResume() {
    super.onResume();
    // Reanudar las actualizaciones al volver a la actividad (si el permiso está OK)
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED) {
        requestLocationUpdates();
    }
}

private void stopLocationUpdates() {
    fusedLocationClient.removeLocationUpdates(locationCallback);
    Log.d("GPS", "Actualizaciones de ubicación detenidas.");
}
```

---

5. Consideración Adicional: Verificar Configuración del GPS

A veces, el usuario tiene el GPS desactivado en el teléfono. Puedes usar un objeto **LocationSettingsRequest** y la clase **SettingsClient** para verificar la configuración del dispositivo e invitar al usuario a activarlo con un diálogo.

Java

```
// Ejemplo de verificación de configuración de ubicación
LocationSettingsRequest.Builder builder = new LocationSettingsRequest.Builder()
        .addLocationRequest(locationRequest);

SettingsClient client = LocationServices.getSettingsClient(this);
Task<LocationSettingsResponse> task = client.checkLocationSettings(builder.build());

task.addOnSuccessListener(this, locationSettingsResponse -> {
    // La configuración de ubicación está OK. Podemos iniciar la solicitud de ubicación.
    requestLocationUpdates();
});

task.addOnFailureListener(this, e -> {
    if (e instanceof ResolvableApiException) {
        // La configuración de ubicación no es adecuada, pero puede ser resuelta.
        try {
            // Muestra el diálogo para que el usuario active el GPS.
            ResolvableApiException resolvable = (ResolvableApiException) e;
            resolvable.startResolutionForResult(MainActivity.this,
                    REQUEST_CHECK_SETTINGS);
        } catch (IntentSender.SendIntentException sendEx) {
            // Ignorar el error.
        }
```

```
    }
});
```

Este proceso robusto garantiza que tu aplicación use el GPS de manera eficiente, respetando la configuración y los permisos del usuario.