GUIA 22 - Geolocalización en Android

OBJETIVO

En esta guía aprenderemos a utilizar las APIs de reconocimiento de ubicación para que a futuro puedas implementarlo en tus aplicaciones.

Geolocalización en Android

Para implementar proyectos que solicitan la ubicación al usuario final se debe primeramente solicitar los permisos de ubicación (ACCESS_FINE_LOCATION y ACCESS_COARSE_LOCATION). Esta solicitud se lo realiza tanto en el código como en el archivo de manifiesto (Manifest.xml):

```
<manifest ... >
  <!-- To request foreground location access, declare one of these permissions. -->
  <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
  <uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" /><!--
desde el api sdk 29 location -->
</manifest>
```

Donde:

- ACCESS_COARSE_LOCATION, proporciona una precisión de ubicación de hasta una manzana.
- ACCESS_FINE_LOCATION, proporciona una ubicación mas precisa que ACCESS_COARSE_LOCATION.

El acceso a la ubicación en Android se lo puede realizar en primer plano o segundo plano

Ubicación en primer plano Se utiliza generalmente cuando tenemos la necesidad de obtener la ubicación en una sola ocasión o durante un tiempo predefinido.

Ubicación en segundo plano Se utiliza cuando se requiere el acceso a la ubicación de forma continua. Desde la API 29 se debe declarar el permiso ACCESS_BACKGROUND_LOCATION en el manifesto, para solicitar acceso a la ubicación en segundo plano.

Obtener la última ubicación conocida

Google provee y recomienda el uso de *Google Play Location API* para manejar la ubicación en Android; aunque también se puede usar la Clase `LocationManager`. En este ejemplo usaremos la clase recomendada por Google (*FusedLocationProviderClient*). Por las siguientes razones:

1. Los Servicios de Google Play proporcionan una interfaz simple y una superficie de API más limpia.
2. Debes especificar una calidad de servicio deseada y las API administrarán las tecnologías subyacentes por ti.
3. Las API de los Servicios de Google Play tienen un rendimiento y un uso de batería optimizados.

4. Las API de los Servicios de Google Play se mantienen de forma activa. Google mejora los algoritmos y agrega más funciones de manera constante.

Para usar *FuseLocationProviderClient* debemos añadir la librería `play-services-location` en el Build.gradle del módulo:

```
dependencies {
 ...
 implementation 'com.google.android.gms:play-services-location:21.0.1'
...
}
```

Para obtener la última ubicación conocida se usa el método **getLastLocation()** del proveedor de ubicación fusionado (FuseLocationProviderClient), para lo cual primeramente debemos crear una instancia del proveedor en el método `onCreate()` de nuestra actividad:

```
lateinit var fusedLocationClient: FusedLocationProviderClient
private var REQUEST_LOCATION_CODE = 1001

// ..

@Override
override fun onCreate(savedInstanceState: Bundle?) {
    // ...

    fusedLocationClient = LocationServices.getFusedLocationProviderClient(this)

}
```

Una vez que tenemos la instancia la última ubicación conocida se obtiene de la siguiente forma:

```
 @SuppressLint("MissingPermission")
    fun obtenerLastLocation() {

        val addOnSuccessListener = fusedLocationClient.getLastLocation()
            .addOnSuccessListener(OnSuccessListener<Location?> { location: Location?
->
                if (location != null) {
                    txtUbicacion.text = "${location.latitude} - ${location.longitude}"

                }else{
                    txtUbicacion.text = "No se puede obtener la ubicacion"
                }
            })
    }
```

Donde *txtUbicacion* es un TextView que muestra la latitud y longitud cuando se ejecuta el método `obtenerLastLocation`.

Conocer la última ubicación del usuario requiere que se use en conjunto con los permisos en tiempo de ejecución:

```
//en cualquier lugar de nuestro codigo donde deseamos conocer la ubicación
...
if (ActivityCompat.checkSelfPermission(MainActivity.this, ACCESS_FINE_LOCATION)
        == PackageManager.PERMISSION_GRANTED
) {
    obtenerLastLocation();

} else {
    ActivityCompat.requestPermissions(MainActivity.this, new String[]{
            ACCESS_FINE_LOCATION}, REQUEST_LOCATION_CODE);
}

...
```

```
  override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
    ) {
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
        if(requestCode == REQUEST_LOCATION_CODE){
            if(grantResults.isNotEmpty() && grantResults[0] ==
PackageManager.PERMISSION_GRANTED){
                obtenerLastLocation();
            }
        }
    }
```

Obtener actualizaciones de Ubicación

Para obtener actualizaciones de ubicación del usuario se invoca al método `requestLocationUpdates()`:

```
    @SuppressLint("MissingPermission")
    public void obtenerActualizacionesCoordenadas(){
        LocationRequest locationRequest = LocationRequest.create();
        locationRequest.setInterval(5000);
      //  locationRequest.setFastestInterval(2000);
        locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);

        LocationCallback locationCallback = new LocationCallback() {
            @Override
            public void onLocationResult(@NonNull LocationResult locationResult) {
              if(locationResult == null){
                  return;
              }
              String locHist = "";
              for (Location loc : locationResult.getLocations()){
                  locHist = locHist +loc.getLatitude()+"_"+loc.getLongitude()+"\n";
              }
```

```
                txtUbicacion.setText(locHist);
            }
        };

        fusedLocationProviderClient.requestLocationUpdates(locationRequest,
locationCallback, Looper.getMainLooper());
    }
```

Donde:

- locationRequest.setInterval(), define en milésimas de segundos la tasa a la que tu app prefiere recibir actualizaciones de ubicación.
- locationRequest.setFastestInterval(), define en milésimas de segundos la tasa más rápida a la que tu app puede controlar las actualizaciones de ubicación. A menos que a tu app le convenga recibir actualizaciones más rápido que la tasa especificada en setInterval(), no necesitas llamar a este método.
- locationRequest.setPriority(), define la prioridad de la solicitud, lo que les indica a los servicios de ubicación de Servicios de Google Play qué fuentes de ubicación deben usar. Se admiten los siguientes valores:
    - PRIORITY_BALANCED_POWER_ACCURACY: Usa esta configuración para solicitar precisión de ubicación de hasta una manzana, lo que equivale a aproximadamente 100 metros.
    - PRIORITY_HIGH_ACCURACY: Usa esta configuración para solicitar la ubicación más precisa posible. Con esta configuración, es más probable que los servicios de ubicación usen GPS para determinar la ubicación.
    - PRIORITY_LOW_POWER: Usa esta configuración para solicitar precisión a nivel de la ciudad, lo que equivale a una precisión de aproximadamente 10 kilómetros. Este nivel de precisión se considera aproximado y es probable que consuma menos energía.
    - PRIORITY_NO_POWER: Usa esta configuración si necesitas que el impacto en el consumo de energía sea insignificante, pero deseas recibir actualizaciones de ubicación cuando estén disponibles. Con esta configuración, tu app no activa actualizaciones de ubicación, aunque sí recibe ubicaciones activadas por otras apps.

Detener actualizaciones de ubicación

Esta operación se debe ejecutar cuando ya no queremos recibir actualizaciones de ubicación o nuestra app pasa a segundo plano.

```
private void stopLocationUpdates() {
    fusedLocationClient.removeLocationUpdates(locationCallback);
}
```

Práctica Nro. 19

Desarrollar una app que obtiene la ubicación del usuario y visualiza el marcador de ubicación en un Mapa. El presentable del siguiente trabajo es un zip del código fuente y un archivo pdf con las capturas de pantalla de funcionamiento.