APP GPS INTERMEDIO

En este ejemplo, igualmente vamos a obtener la latitud, longitud, altitud, precisión, y dirección postal, pero una vez hice el proyecto sencillo, se me ocurrió la posibilidad de poder actualizar los datos a tiempo real, es decir, si nos movemos los datos van a ir cambiando sin tener que salir de la app, a diferencia del proyecto sencillo donde para ver la nueva posición tendremos que salir y entrar de nuevo en la app, sin embargo con este ejemplo, nos va a ir diciendo la posición en cada momento sin salir de la app.

Luego se me ocurrió la posibilidad de poder enviar un sms a un numero de tlf y enviarle nuestras coordenadas y posición, por lo que también se lo he incorporado.

También he añadido una opción indispensable en una app GPS y es la de aparte de conocer nuestras coordenadas, poder ver en un mapa donde nos encontramos a tiempo real,

y por último, nada que ver con el GPS, pero llevaba tiempo queriendo saber cómo se hace un loading o intro, el cual uso en las Web que he realizado a clientes, (la típica pantalla, ya sea de una app o de una web, donde se visualiza una imagen o un texto y el típico icono de cargando.., y después de unos segundos (valor modificable), se abre la página web o en este caso la activity asignada.

Por tanto, para realizar este proyecto, lo primero es irnos al archivo **AndroidManifest.xml,** y añadirle los permisos tanto de localización, internet, y envío de sms, y añadimos estas líneas de permisos antes de <application….

```
<!-- Permisos -->
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission
android:name="com.luisgomez.gps_avanzado_con_mapas.permission.MAPS_REC
EIVE" />
<uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
android:name="com.google.android.providers.gsf.permission.READ_GSERVIC
ES" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" /> <!--
Internet Permissions -->
<uses-permission android:name="android.permission.INTERNET" />
```

Además, como vamos a usar mapas de google, necesitamos una clave API de Google Map ya que Google lo implantó como norma para poder hacer uso de sus mapas.

Para obtener una API de Google Map, se puede obtener desde aquí:

https://console.cloud.google.com/apis/library

Para este proyecto he usado una API mía que ya tenía para otros proyectos web y wordpress para pruebas, por lo que la pueden usar si quieren los compañeros para probar la app en el móvil.

Por tanto, añadimos estas líneas también en el archivo **AndroidManifest.xml :**

```
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version" />
<meta-data
    android:name="com.google.android.geo.API_KEY"
    android:value="AIzaSyAFR76CPobEiv0JVwLangVodqjt_BSw2aY" />
```

Una vez preparado el archivo **AndroidManifest.xml,** nos vamos a Gradle Script y dentro de él a **build.gradle** y tenemos que añadir lo siguiente para poder usar los servicios de mapas de Google:

```
implementation 'com.google.android.gms:play-services-location:16.0.0'
implementation 'com.google.android.gms:play-services-maps:16.1.0'
implementation 'com.google.maps.android:android-maps-utils:0.5'
```

implementation 'com.google.maps.android:android-maps-utils:0.5'

Como he comentado, como he añadido un loading, este loading lo añado en el Main Activity, donde pasados unos segundos nos llevará a otra actividad, la cual ya va a contener los datos obtenidos por el GPS, asi que en el MainActiviy y dentro del método onCreate ponemos estas líneas:

```
// Para Abrir una pagina de inicio (loading) y despues del tiempo
fijado, ira a la siguiente activity.
Thread welcomeThread = new Thread() {

    @Override
    public void run() {
        try {
            super.run();
            sleep(2000);  //Delay of 10 seconds
        } catch (Exception e) {

        } finally {

            Intent i = new Intent(MainActivity.this,
                    GPS.class);
            startActivity(i);
            finish();
        }
    }
};

welcomeThread.start();
```

Donde le decimos el tiempo de espera para pasar a la actividad que le indicamos en el Intent.

Ahora nos vamos al layout activity_main, y ponemos lo que queramos que muestre, ya sea texto, una imagen etc, pero en este caso he querido poner el típico circulito dando vueltas, y para hacerlo añadimos lo siguiente:

```
<ProgressBar
    android:id="@+id/progressBar"
    android:layout_width="40dp"
    android:layout_height="40dp"
    android:progressDrawable="@drawable/loading_personalizado"
    android:layout_marginTop="40dp"
    android:layout_gravity="center"
    android:indeterminateTint="#000"
    />
```

y creamos un nuevo archivo resource en Drawable llamado **loading_personalizado** ,donde le escribimos estas líneas:

```
<rotate
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="90"
    android:pivotX="50%"
    android:pivotY="50%"
    android:toDegrees="360">

    <shape
        android:innerRadiusRatio="3"
        android:shape="ring"
        android:thicknessRatio="7.0">

        <gradient
            android:centerColor="#007DD6"
            android:endColor="#007DD6"
            android:startColor="#007DD6"
            android:angle="0"
            android:type="sweep"
            android:useLevel="false" />
    </shape>

</rotate>
```

y ya tendemos el loading listo.

Ahora, creamos otra actividad vacía llamada GPS, y en su layout llamado **activity_gps**, añadimos los TextView para mostrar la latitud, longitud, altitud, precisión, dirección, un botón para enviar un sms y otro botón para ir a otra activity donde podemos visualizar un mapa y donde nos encontramos en él.

Después en el **archivo GPS.java,** primeramente le decimos que extienda de Activity el cual lo importará y seguidamente declaramos las variables de altitud y precisión, los String para poder recibir los datos a tiempo real de la latitud, longitud y direccion, declaramos también los botones para enviar el sms y para ver el mapa, y declaramos la class Location, que se encarga de estar siempre atenta a cualquier cambio de posicion recibido en el GPS del dispositivo.

Por tanto, también hay que importar el paquete “android.location.LocationListener”,

Y por último con un if y else, le decimos que dependiendo si se aceptan o no, hará los cálculos o mostrará que no se han aceptado.

Ahora tenemos que poner el método locationStar el cual va a activar el GPS dependiendo de si hemos aceptado los servicios o no.

En cuanto a cómo obtener la **dirección,** lo hacemos con el método **setLocation**, haciendo uso de la clase **Geocoder,** que es una clase para manejar **geocodificación** y **geocodificación inversa**.

La **geocodificación** es el proceso de transformar una dirección de calle u otra descripción de una ubicación en una coordenada (latitud, longitud).

La **geocodificación inversa** es el proceso de transformar una coordenada (latitud, longitud) en una dirección (parcial)”.

Por tanto, en este caso usaremos la geocodificacion inversa y de las coordenadas obtenidas la transformaremos a nuestra dirección.

Mas info: https://developer.android.com/reference/android/location/Location.html
Más info: ttps://developer.android.com/reference/android/location/Geocoder.html

Ahora nos queda recibir los datos obtenidos para imprimirlos por un lado y también para enviarlos por SMS, y lo hacemos método ***onLocationChanged*** donde le pasamos como parámetro a la clase **Location**,la cual nos permitirá utilizar sus dos atributos que son **getLatitude** y **getLongitude** y con ella podremos imprimir los valores y además con un botón, le digo que envie un sms al teléfono que le asignemos, ya sea uno o los que queramos (he probado a varios números, y lo envia a varios teléfonos simultáneamente), donde podremos personalizar el texto que enviamos asi como la información que queremos enviar.

También incluye un Toast para decirnos si se ha enviado correctamente el SMS o si ha surgido un fallo en el envío.

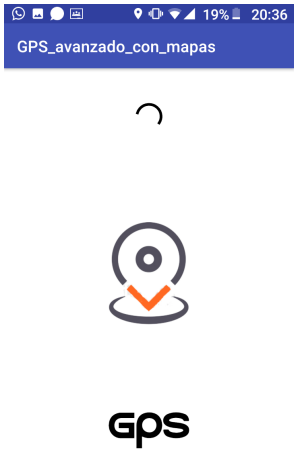
Por último, luego tenemos 2 métodos para verificar si el GPS del móvil está activado o no

Además, como he comentado antes, como queremos ver en un mapa nuestra posición, tenemos que crear un botón que nos lleve a otra actividad llamada MapasActivity.java, la cual va a contener una serie de métodos genéricos que he usado poder obtener nuestra posición según los valores recibidos por el GPS del dispositivo, y también he declarado un botón para volver al activity GPS de nuevo.

Y para acabar, creamos un Fragment en activity_mapas.xml, para mostrar el mapa y datos enviados por la activity MapasActivity, y añadimos el botón declarado en MapasActivity para volver atrás.

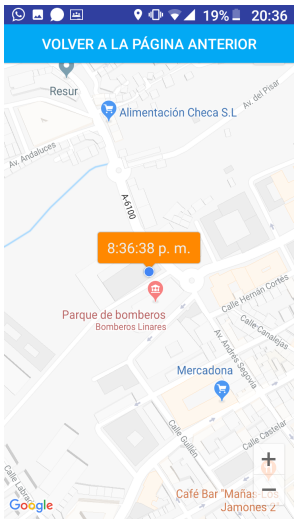
Probar la app. Capturas funcionando en mi móvil:

Activity - MainActivity:



Activity - GPS:

Activity - MapasActivity:



Mensaje SMS que llega al numero de telefono definido: