

Computación Móvil

Localización y GPS

Pontificia Universidad Javeriana
Ingeniería de Sistemas
Profesor: Carlos Andrés Parra
E-mail: ca.parraa@javeriana.edu.co



Agenda

1. Localización
 1. Localización – Play Services
 2. Dependencias en Gradle
2. Recibir actualizaciones de ubicación
3. Distancia entre distintos puntos
4. Guardar datos en almacenamiento externo

Localización - Permisos

- La localización es una de las características propias de los dispositivos móviles.
- Hacer una aplicación sensible a la ubicación permite dar servicios únicos a dispositivos en movimiento.
- La localización es un **recurso con riesgo** pues involucra información privada del usuario.
- Es necesario solicitar permiso y validar si el permiso ha sido aceptado, de la misma forma que se hizo con los contactos, la cámara o el acceso a la memoria interna.

Permisos

- Localización fina (GPS)

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

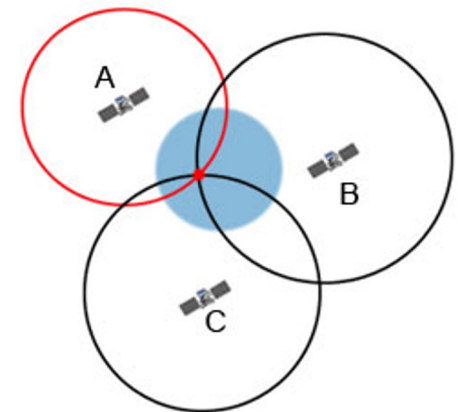
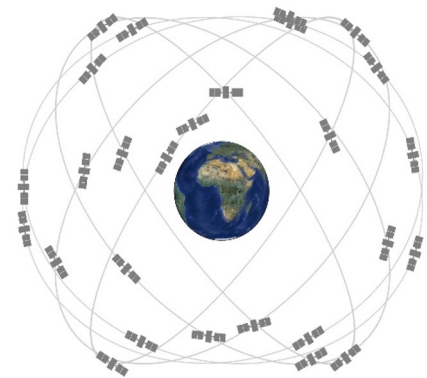
- Localización Gruesa (triangulación WIFI)

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

Nota. Se deben pedir ambos permisos en el manifest!

¿GPS - Cómo funciona?

- Alrededor de 30 satélites orbitan alrededor de la tierra a aproximadamente 20.000 kilómetros. (*Circle the earth twice a day*)
- Cada satélite envía en intervalos regulares su posición.
- En cualquier punto de la tierra se deben ver al menos 4 satélites.
- Con la ubicación de 3 satélites se puede hacer el cálculo de la posición exacta (alrededor de 1 metro de precisión)

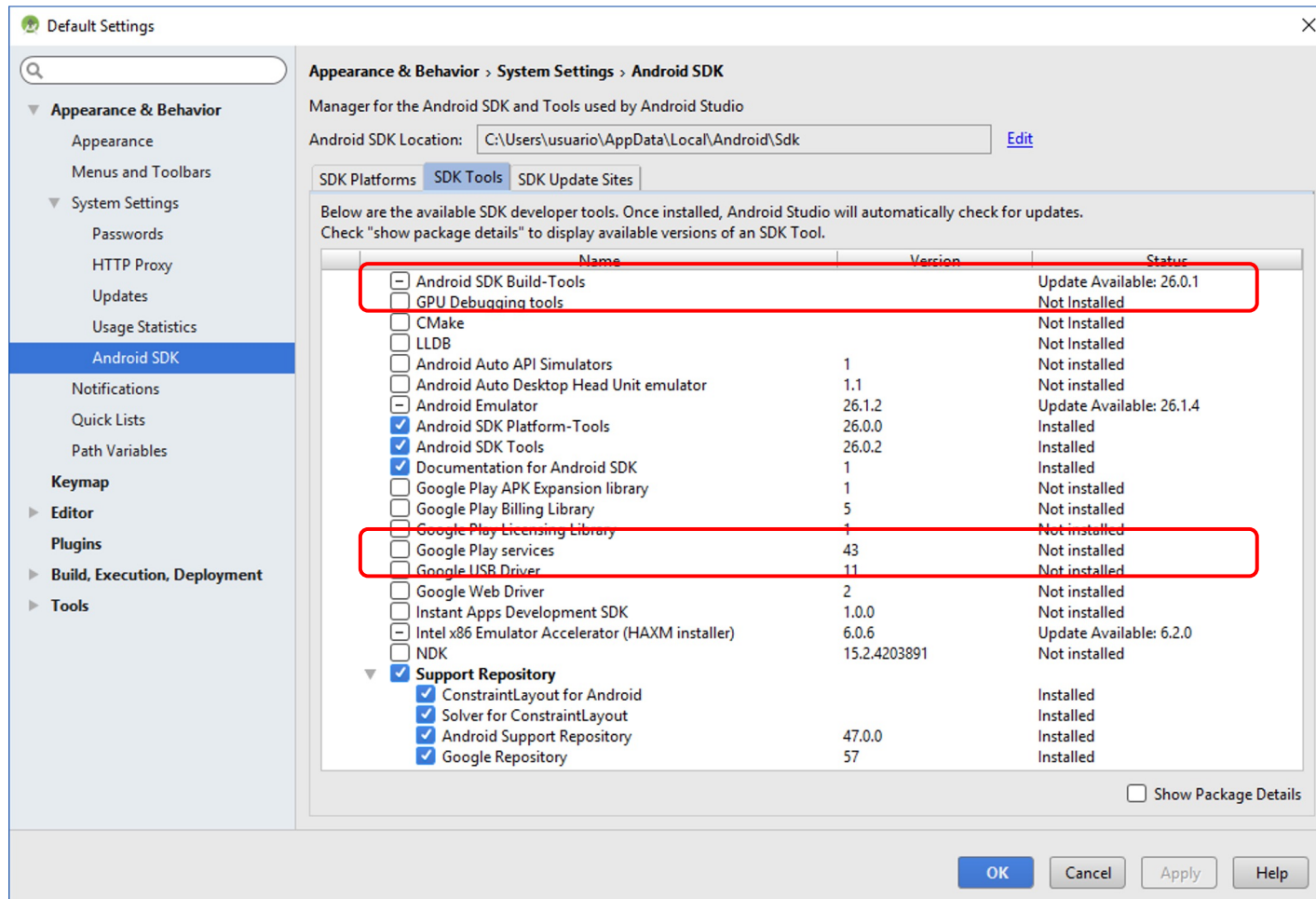


<http://www.physics.org/article-questions.asp?id=55>
<https://www.gps.gov/systems/gps/space/>

Location

- La aplicación puede consultar la ubicación una sólo vez
- Sin embargo, para aplicaciones que ayudan al usuario a encontrar una ruta mientras se desplazan, la ubicación debe actualizarse en intervalos regulares.
- Además de la ubicación geográfica ([latitud](#) y [longitud](#)), también se puede obtener mas información como [altitud](#) y [velocidad de desplazamiento](#).
- Esta información está disponible en la clase [Location](#) que se obtiene del [fused location provider](#).
- Esta funcionalidad es parte de los servicios de Google Play.

Activar Google Play Services Tools->Android->SDK Manager



Agregar las dependencias

- En el archivo gradle del **módulo de la aplicación** agregar la dependencia a todos los servicios de Google:

Última versión **19.0.1**

```
implementation 'com.google.android.gms:play-services:17.0.0'
```

O sólo a los servicios que se necesitan de forma individual:

```
implementation 'com.google.android.gms:play-services-location:19.0.1'
```

<https://developers.google.com/android/guides/setup>

Consultar la localización

- Se utiliza el servicio de localización de Google
 - Atributo de la clase

```
private FusedLocationProviderClient mFusedLocationClient;
```

- En onCreate se inicializa

```
mFusedLocationClient = LocationServices.getFusedLocationProviderClient(this);
```

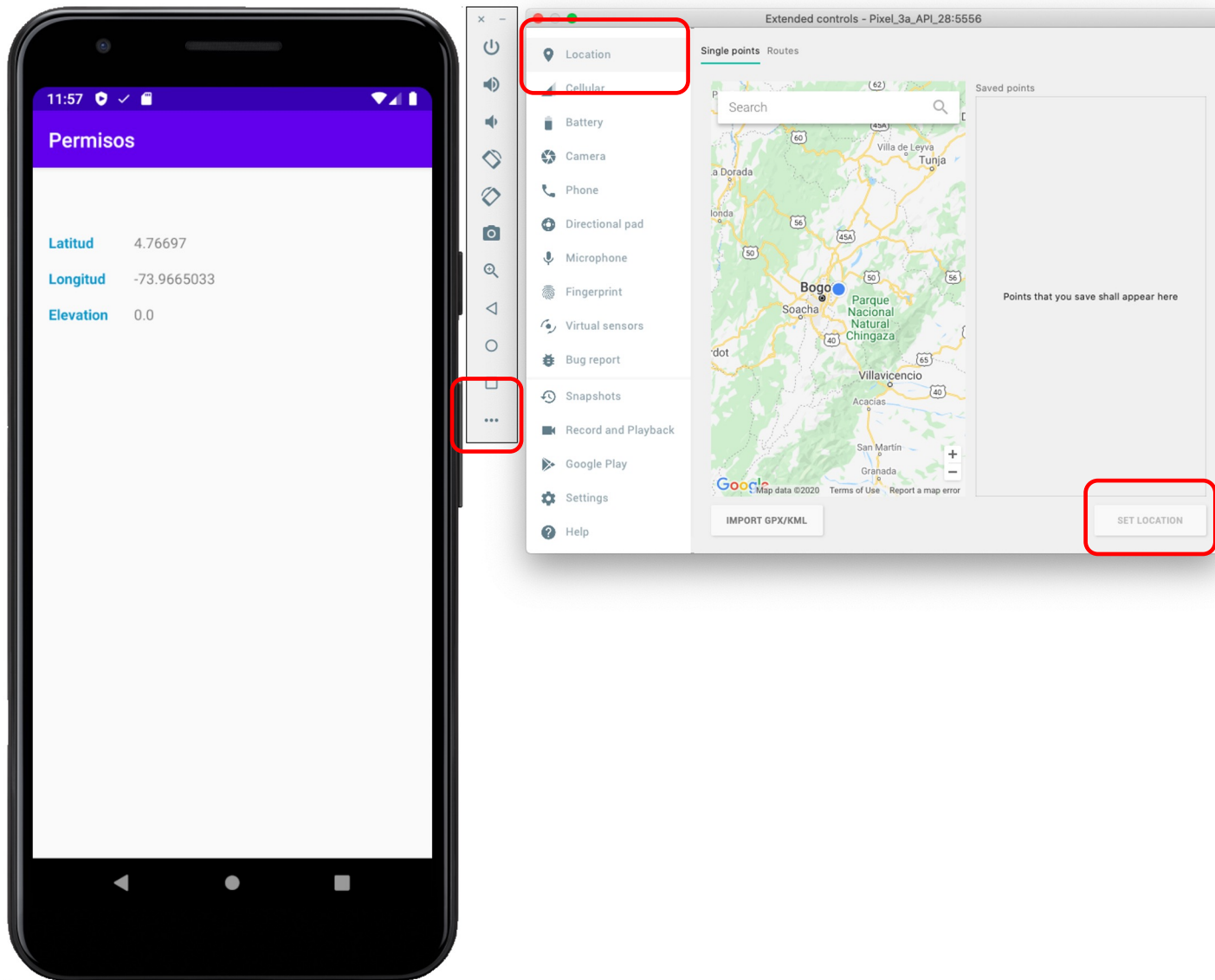
Consultar la localización

- **Cuando se tengan permisos**, se puede acceder a la localización:

```
mFusedLocationClient.getLastLocation().addOnSuccessListener(this,  
new OnSuccessListener<Location>() {  
    @Override  
    public void onSuccess(Location location) {  
        Log.i("LOCATION", "onSuccess location");  
        if (location != null) {  
            Log.i(" LOCATION ", "Longitud: " + location.getLongitude());  
            Log.i(" LOCATION ", "Latitud: " + location.getLatitude());  
        }  
    }  
});
```

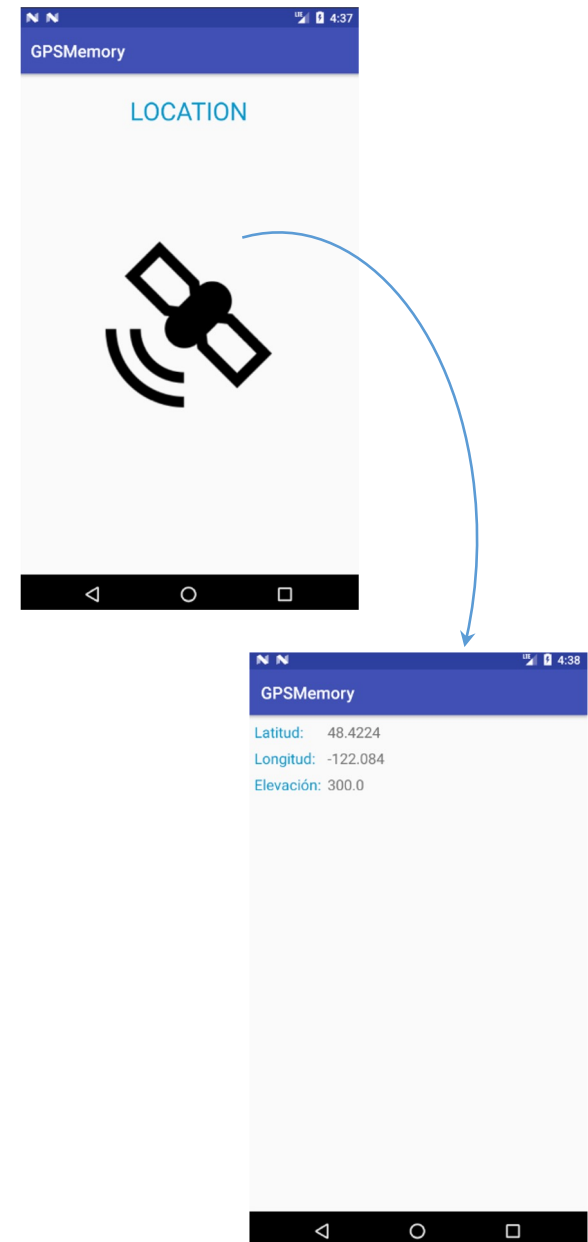
Nota 1: Al probar es necesario encender la localización en el teléfono. Si se usa el emulador se debe contar con los servicios de google play. En el emulador se pueden simular los datos o cargar datos de un archivo gpx.

Localización en el emulador



Ejercicio 1

- Construya una aplicación como la de la imagen con dos actividades. Puede usar la imagen de BrightSpace para el nuevo botón de la actividad principal.
- Programe el botón para que lance una nueva actividad que gestione los permisos para utilizar la localización fina (GPS)
- Cree elementos TextView como los de la figura, y muestre en pantalla la información de latitud, longitud, y altitud de la ubicación actual del dispositivo.



Location-aware Apps

Notificación de cambios de posición

- Para especificar mejor las condiciones que se requieren de localización y suscribirse a actualizaciones periódicas es necesario crear un objeto de tipo `RequestLocation`

```
import com.google.android.gms.location.LocationRequest;
```

```
//Atributo
```

```
private LocationRequest mLocationRequest;
```

```
//en onCreate
```

```
mLocationRequest = createLocationRequest();
```

```
private LocationRequest createLocationRequest(){  
    LocationRequest locationRequest = LocationRequest.create()  
        .setInterval(10000)  
        .setFastestInterval(5000)  
        .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);  
    return locationRequest;  
}
```

Notificación de cambios de posición

Prioridad de la solicitud

- [PRIORITY BALANCED POWER ACCURACY](#) - Use this setting to request location precision to within a city block, which is an accuracy of approximately **100 meters**. This is considered a coarse level of accuracy and is likely to consume less power. With this setting, the location services are likely to use WiFi and cell tower positioning. Note, however, that the choice of location provider depends on many other factors, such as which sources are available.
- [PRIORITY HIGH ACCURACY](#) - Use this setting to request **the most precise location** possible. With this setting, the location services are more likely to use GPS to determine the location.
- [PRIORITY LOW POWER](#) - Use this setting to request city-level precision, which is an accuracy of approximately **10 kilometers**. This is considered a coarse level of accuracy and is likely to consume less power.
- [PRIORITY NO POWER](#) - Use this setting if you need negligible impact on power consumption, but want to receive location updates **when available**. With this setting, your app does not trigger any location updates, but receives locations triggered by other apps.

Notificación de cambios de posición

Suscripción a la localización

- Para suscribirse a la localización, se debe definir un objeto `callback()` cuyos métodos son invocados de acuerdo a la tasa de refresco antes establecida.

//Atributo

```
private LocationCallback mLocationCallback;
```

- Este atributo se asigna al `fusedLocationClient` para recibir las actualizaciones.

```
private void startLocationUpdates() {  
//Verificación de permiso!!  
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==  
        PackageManager.PERMISSION_GRANTED) {  
        mFusedLocationClient.requestLocationUpdates(mLocationRequest, mLocationCallback, null);  
    }  
}
```

Petición de Localización

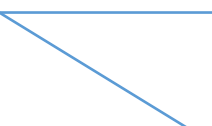
Objeto Callback!

Notificación de cambios de posición

Objeto Callback

- Finalmente se programa el objeto callback para reaccionar a la actualización de la ubicación:

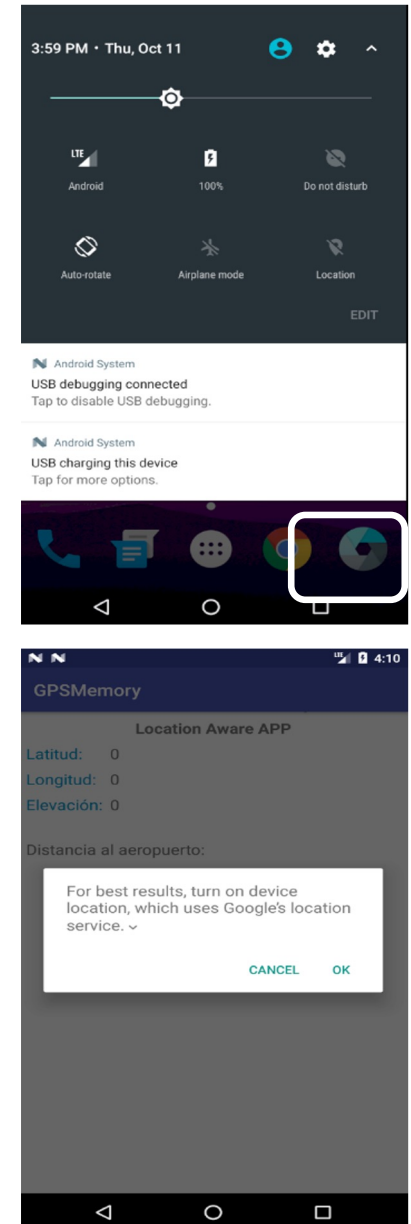
```
//En onCreate()
mLocationCallback = new LocationCallback() {
    @Override
    public void onLocationResult(LocationResult locationResult) {
        Location location = locationResult.getLastLocation();
        Log.i("LOCATION", "Location update in the callback: " + location);
        if (location != null) {
            latitude.setText("Latitude: " + String.valueOf(location.getLatitude()));
            longitude.setText("Longitude: " + String.valueOf(location.getLongitude()));
            altitude.setText("Altitude: " + String.valueOf(location.getAltitude()));
        }
    }
};
```



Se asignan los valores de la localización a los TextViews
Esto va a suceder cada 5 segundos aproximadamente

Encender la localización de forma programática

- Si el usuario no tiene encendida la localización, la aplicación puede encenderla mostrando un diálogo al usuario. Para eso se debe:
 1. Acceder a la configuración del usuario
 2. Si la tarea es exitosa, la localización está encendida y se puede seguir adelante.
 3. Sino, pero si existe el hardware en el dispositivo se debe enviar un diálogo para encender la localización de forma programática.
 4. Finalmente si no se cuenta con el hardware se debe deshabilitar la funcionalidad asociada a la localización.



Encender la localización de forma programática

```
private void checkLocationSettings(){
    LocationSettingsRequest.Builder builder = new
        LocationSettingsRequest.Builder().addLocationRequest(locationRequest);
    SettingsClient client = LocationServices.getSettingsClient(this);
    Task<LocationSettingsResponse> task = client.checkLocationSettings(builder.build());
    task.addOnSuccessListener(new OnSuccessListener<LocationSettingsResponse>() {
        @Override
        public void onSuccess(LocationSettingsResponse locationSettingsResponse) {
            Log.i(IndexActivity.TAG, "GPS is ON");
            settingsOK = true;
            startLocationUpdates();
        }
    });
    task.addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            if(((ApiException) e).getStatusCode() == CommonStatusCodes.RESOLUTION_REQUIRED){
                ResolvableApiException resolvable = (ResolvableApiException) e;
                IntentSenderRequest isr = new IntentSenderRequest.Builder(resolvable.getResolution()).build();
                getLocationSettings().launch(isr);
            } else {
                elevation.setText("No GPS available");
            }
        }
    });
}
```

1. Acceder a la configuración

2. GPS ya está encendido

3. GPS está apagado pero se puede encender

3.1 Lanza actividad con nuevo API

4. Hardware no disponible

Actividad de solicitud para encender GPS

```
ActivityResultLauncher<IntentSenderRequest> getLocationSettings =  
    registerForActivityResult(  
        new ActivityResultContracts.StartIntentSenderForResult(),  
        new ActivityResultCallback<ActivityResult>() {  
            @Override  
            public void onActivityResult(ActivityResult result) {  
                Log.i(IndexActivity.TAG, "Result from settings: "+result.getResultCode());  
                if(result.getResultCode() == RESULT_OK){  
                    settingsOK = true;  
                    startLocationUpdates();  
                }else{  
                    elevation.setText("GPS is off");  
                }  
            }  
        })  
    );
```

Encender y apagar la suscripción

- Suscribirse a la actualización de cambios en la posición es costoso en términos de procesamiento y consumo de energía.
- Sólo se debe hacer cuando la aplicación necesite mostrar actualizaciones de forma constante como parte de su funcionamiento.
- En una aplicación con navegación (Waze, google maps, etc.) sólo se debe escuchar la ubicación si el usuario está viendo algo como un mapa y es consciente de su desplazamiento.


Encender y apagar la suscripción

- Si no se están mostrando los datos la suscripción debería cancelarse:

`@Override`

```
protected void onPause() {  
    super.onPause();  
    stopLocationUpdates();  
}
```

La actividad pasa a un segundo plano, se cancela la suscripción



```
private void startLocationUpdates(){  
    if(ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) ==  
        PackageManager.PERMISSION_GRANTED){  
        if(settingsOK) {  
            fusedLocationProvider.requestLocationUpdates(locationRequest, locationCallback, null);  
        }  
    }  
}  
  
private void stopLocationUpdates(){  
    fusedLocationProvider.removeLocationUpdates(locationCallback);  
}
```

Distancia entre dos puntos

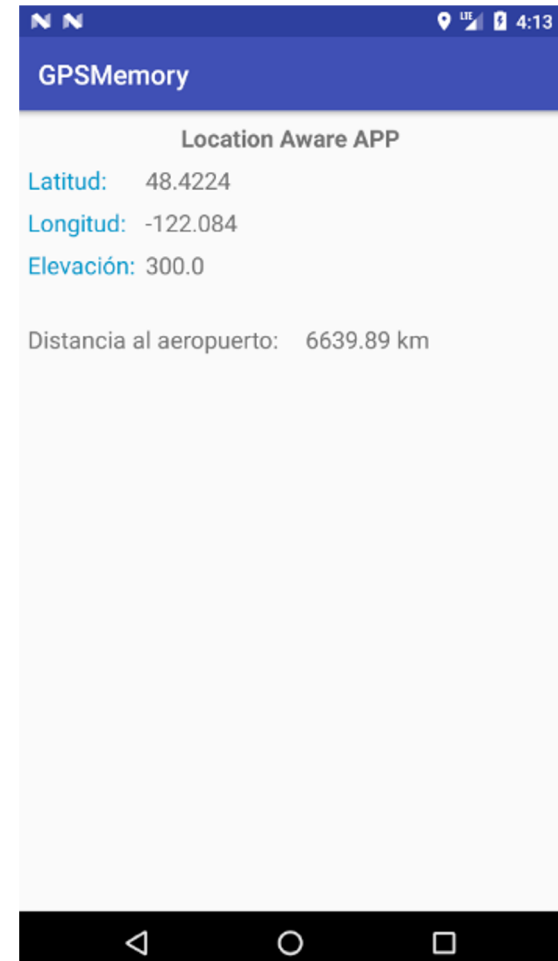
- Es posible calcular la distancia entre dos puntos a partir de la latitud y la longitud de cada punto:

```
public double distance(double lat1, double long1, double lat2, double long2) {  
    double latDistance = Math.toRadians(lat1 - lat2);  
    double lngDistance = Math.toRadians(long1 - long2);  
  
    double a = Math.sin(latDistance / 2) * Math.sin(latDistance / 2)  
        + Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2))  
        * Math.sin(lngDistance / 2) * Math.sin(lngDistance / 2);  
  
    double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));  
    double result = RADIUS_OF_EARTH_KM * c;  
    return Math.round(result*100.0)/100.0;  
}
```

<https://stackoverflow.com/questions/3694380/calculating-distance-between-two-points-using-latitude-longitude-what-am-i-doing>

Ejercicio 2

- Modifique la aplicación del ejercicio 1 para que ahora tenga una suscripción a las actualizaciones de la ubicación del dispositivo.
- Además, calcule la distancia de la posición actual del usuario hasta el aeropuerto El Dorado. Para obtener la latitud y longitud de este sitio utilice Google Maps.
- Si cuenta con un dispositivo, haga desplazamientos para verificar que la distancia se calcula a medida que cambian las coordenadas
- Defina los métodos onResume y onPause para que se active y se cancele la suscripción de acuerdo al ciclo de vida de la actividad.



Escritura en memoria interna

```
private void writeJSONObject(){
    MyLocation myLocation = new MyLocation();
    myLocation.setFecha(new Date(System.currentTimeMillis()));
    myLocation.setLatitud(mCurrentLocation.getLatitude());
    myLocation.setLongitud(mCurrentLocation.getLongitude());
    localizaciones.put(myLocation.toJSON());
    Writer output = null;
    String filename= "locations.json";
    try {
        File file = new File(getBaseContext().getExternalFilesDir(null), filename);
        Log.i("LOCATION", "Ubicacion de archivo: "+file);
        output = new BufferedWriter(new FileWriter(file));
        output.write(localizaciones.toString());
        output.close();
        Toast.makeText(getApplicationContext(), "Location saved",
            Toast.LENGTH_LONG).show();
    } catch (Exception e) {
        //Log error
    }
}
```

El objeto localizaciones es de tipo JSONArray

Si hay almacenamiento externo el archivo se guarda en:

`Android/data/javeriana.edu.co.gpsmemory/files/locations.json`

Nombre del paquete

```
public JSONObject toJSON () {
    JSONObject obj = new JSONObject();
    try {
        obj.put("latitud", getLatitud());
        obj.put("longitud", getLongitud());
        obj.put("date", getFecha());
    } catch (JSONException e) {
        e.printStackTrace();
    }
    return obj;
}
```

Método toJSON del objeto
MyLocation

Ejercicio 3

- Modifique la aplicación del ejercicio 2 para que incluya un botón que guarde los datos del usuario en un archivo JSON en la memoria interna del teléfono, y en una lista con el historial de localizaciones que se muestra en la actividad.

