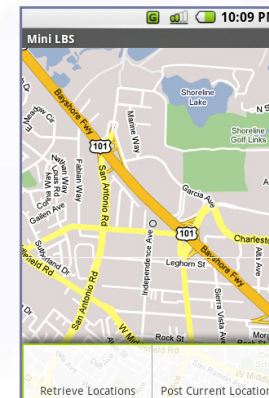




# Desarrollo de aplicaciones: LBS y sensores Android



 **LBS**

 Localización Android

 Google Maps

 Sensores



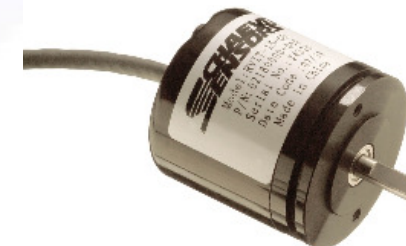
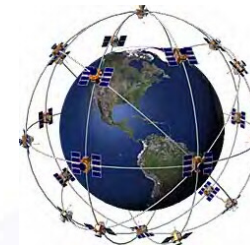
## ■ *Definiciones*

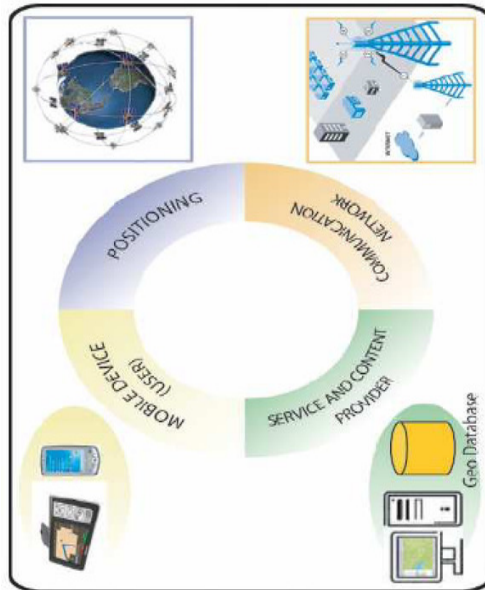
- ▶ Los LBS (Location Based Services) ofrecen distintos servicios en función de la ubicación del solicitante.
- ▶ Son servicios que utilizan la información geográfica para dar valor al usuario móvil.
- ▶ Ayudan a responder distintos tipos de preguntas:
  - ▶ ¿Dónde estoy?
  - ▶ ¿Qué tengo a mi alrededor?
  - ▶ ¿Qué lugares de interés hay cerca de mi ubicación?
- ▶ Con los LBS se intenta dar servicios y aplicaciones personalizadas.



## *Tecnologías de localización*

- ▶ Mecánicas: basadas en detección de presión.
- ▶ Magnéticas: usando magnetismos especialmente para detectar orientación.
- ▶ Radiofrecuencias y microondas: RFID, triangulación temporal, GPS, Galileo
- ▶ Inerciales: giróscopos
- ▶ Ópticas: sensores ópticos, sensores de posición PSD
- ▶ Acústicas: utilizando las ondas sonoras, problemas con ecos, jitter...



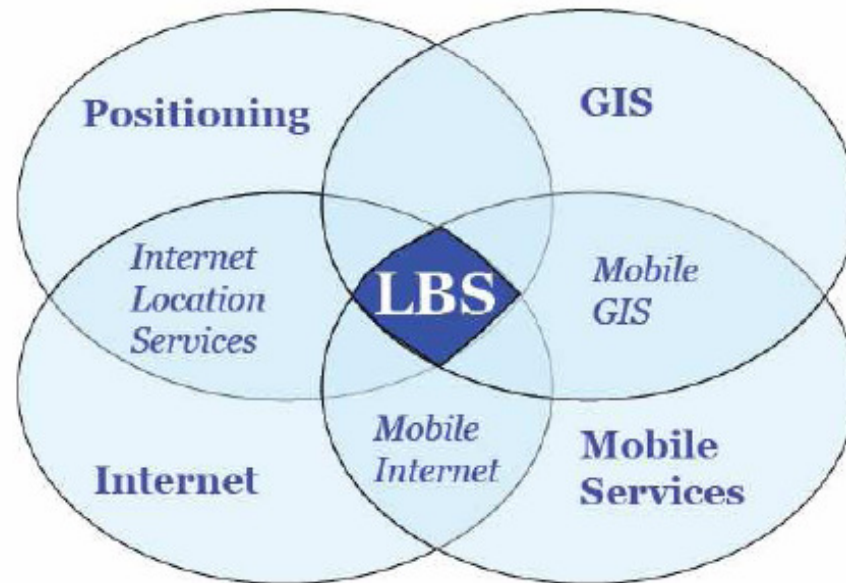


## *Componentes de un sistema de localización*

- ▶ Infraestructura de red: puntos de acceso, sensores, y transmisores de la red. Hardware. Capta información de localización.
- ▶ Función de localización: en función de los datos captados por la infraestructura, se estima la localización.
- ▶ GIS (Geographic Information System): recoge y procesa la información que necesita el LBS usando la función de localización.
- ▶ Gestión de la localización: dada la posición del usuario, se extrae la información de su interés. Esta información se le pasa al LBS.
- ▶ Dispositivos móviles: dispositivo que permitirá utilizar las aplicaciones LBS.

## Componentes de un sistema de localización

### Location Based Services (LBS)



The intersection of technologies is creating LBS

## *Aplicaciones LBS*

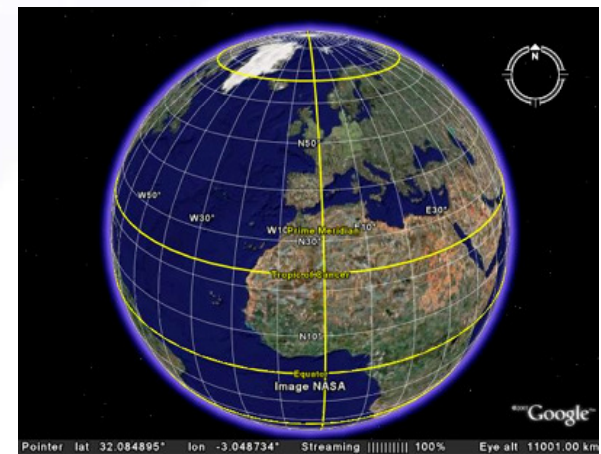
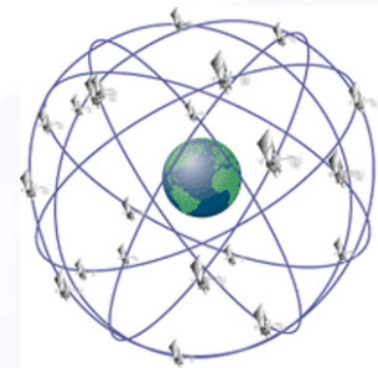
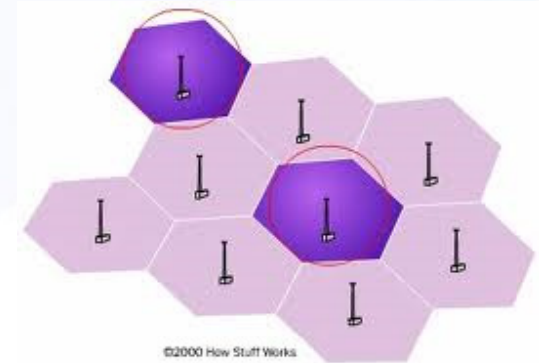
- ▶ Servicios de localización de emergencias
- ▶ Control de tráfico
- ▶ Avisos climatológicos
- ▶ Publicidad selectiva (Digital Signage)
- ▶ Seguimiento de personas
- ▶ Gestión de flotas (mercancías, transportes)



# LBS: Location Based Services

## ■ *Estimación de la localización*

- ▶ Se utilizan técnicas de posicionamiento en tiempo real.
- ▶ Distintos grados de precisión en función de la tecnología
- ▶ Información de localización puede estar en coordenadas absolutas o relativas
- ▶ Métodos:
  - ▶ GSM, GPRS, UMTS: localización basada en celdas
  - ▶ Satélites( GPS, Galileo, GLONASS, MSAS...): GPS utiliza 24 satélites geoestacionarios, triangula en función de señal de radio codificada, precisión de 4 a 40 m.
  - ▶ Sensores y balizas de corto alcance: sensores de presencia, cámaras de infrarrojos...



- LBS
- *Localización Android*
- Google Maps
- Sensores



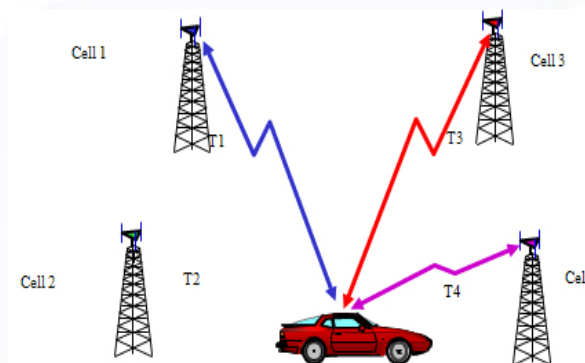
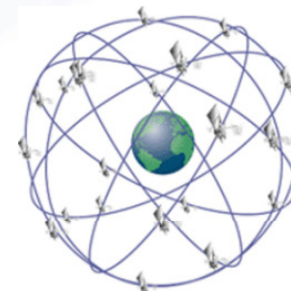
# Localización con Android

## 👉 *Obtener la ubicación del dispositivo en Android*

- ▶ Saber la ubicación del usuario permite a las aplicaciones ser más inteligentes y entregar información más interesante al usuario.
- ▶ Se puede utilizar tanto GPS como los Proveedores de Localización de Red de Android.
- ▶ GPS funciona mejor en espacios abiertos y consume mucha batería.
- ▶ Los Proveedores de Localización de Red determinan la ubicación utilizando celdas, a través de torres y señales wifi. Más efectiva en espacios cerrados, responde más rápidamente y consume menos batería.
- ▶ Se deben dar los correspondientes permisos a las aplicaciones:

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

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



# Ejemplos y ejercicios: *LBS*

## ➤ **LBS-ubicacionSimple:**

- ▶ En este ejemplo se muestra cómo utilizar el LocationManager para obtener la ubicación (última detectada) del dispositivo.
- ▶ Para realizar la simulación de posicionamiento, en el CMD ejecutar:
- ▶ telnet 127.0.0.1 5554 (realiza una conexión directa al emulador, sin pasar por el ADB)
- ▶ Ejecutar *GEO FIX 1.1 43.1*

## ➤ **LBS-ubicacionCriteria:**

- ▶ En este ejemplo se muestra cómo, en función de unos parámetros encapsulados en el objeto Criteria, se le pide al sistema el mejor proveedor de localización (red, GPS...).
- ▶ Por otro lado, se crea un locationListener que se utiliza para recibir los cambios en la posición de ubicación del dispositivo e invocar métodos para la actualización automática de los campos de texto.



## Localización y mapas

- ▶ Android utiliza la API de Google Maps para el desarrollo de aplicaciones basadas en localización.
- ▶ Las clases de la API están ubicadas en el package `com.google.android.maps` package y `android.location` para la localización.
- ▶ Estas librerías se deben instalar de forma separada, aunque existe SDK con librerías de Android+Google Maps integradas
- ▶ Los servicios de localización (GPS, redes...) los aporta el dispositivo.
- ▶ El componente principal de localización es el servicio LocationManager.



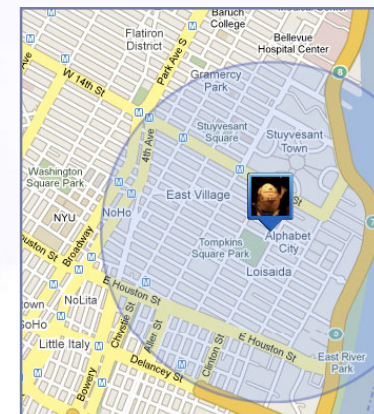
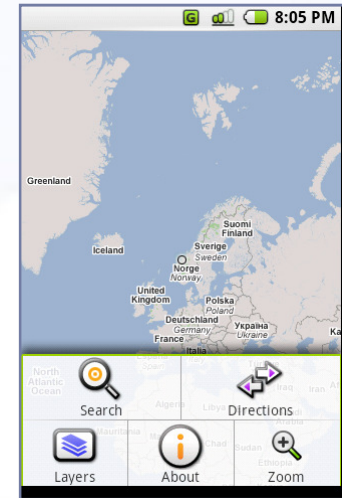
## *LocationManager*

- ▶ Se invoca utilizando `getSystemService(Context.LOCATION_SERVICE);`
- ▶ Este método devuelve una instancia de `LocationManager`
- ▶ A partir de aquí se puede:
  - ▶ Lanzar consultas a los `LocationProviders` para conocer la última posición del dispositivo
  - ▶ Registrar un componente para que reciba actualizaciones periódicas de la ubicación de un `LocationProvider`
  - ▶ Registrar un `Intent` para que se lance si el dispositivo entra en un radio de proximidad de una posición



## *Google Maps Library*

- ▶ Se ubican en el package `com.google.android.maps`
- ▶ Permiten que las aplicaciones puedan descargar, renderizar y cachear zonas de mapas, así como añadir capas, marcadores, etc.
- ▶ La clase principal es un `MapView`, subclase de `ViewGroup`, que dibuja un mapa con datos obtenidos del servicio de Google Maps
- ▶ Esta vista puede capturar eventos de la pantalla y realizar marcados, zoom, etc.



## *Google Maps Library*

- ▶ Para utilizar los mapas, será necesario obtener una API MAPS key.
- ▶ Es necesario registrarse con el servicio de Google Maps y aceptar las condiciones de servicio (indistintamente si es un entorno de pre o producción)
- ▶ Se realiza en dos partes:
  - ▶ Se registra la huella MD5 del certificado con el que vamos a firmar nuestra aplicación.
  - ▶ Con ese MD5 el servicio proporciona una API KEY
  - ▶ Se debe introducir esa API KEY en cada MAPVIEW que utilicemos



# Ejemplos y ejercicios: Localización Android

- ▶ En este ejercicio se va a generar una API KEY para poder utilizar los mapas.
- ▶ Primero hay que localizar un keystore: se puede crear uno o utilizar el del SDK, ubicado en:
  - ▶ Windows Vista: C:\Users\\.android\debug.keystore
  - ▶ Windows XP: C:\Documents and Settings\\.android\debug.keystore
  - ▶ OS X and Linux: ~/.android/debug.keystore
- ▶ Ejecutar: `keytool -list -alias androiddebugkey -keystore debug.keystore -storepass android -keypass android`, esto generará la huella MD5 (44:F4:EB...)
- ▶ Registrar ese MD5 en el servicio de Google Maps: <http://code.google.com/intl/es-ES/android/maps-api-signup.html>
- ▶ Con esto tendremos una API-KEY (0AXf-CzV8uW...) que podremos utilizar para nuestros MapViews
- ▶ Para usar un MapView:

```
<com.google.android.maps.MapView
android:layout_width="fill_parent"
android:layout_height="fill_parent" android:apiKey="0AXf-
CzV8uW..." />
```

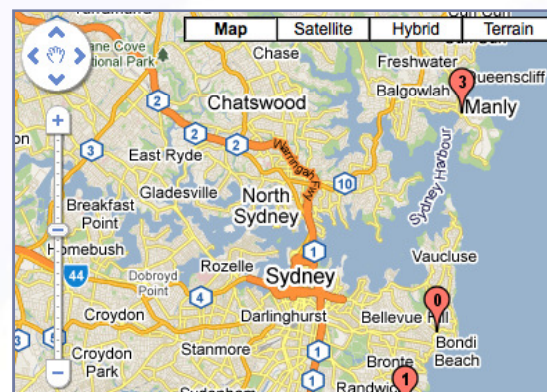


- 👉 LBS
- 👉 Localización Android
- 👉 Google Maps
- 👉 Sensores



## Principales clases de la API

- ▶ MapActivity: Activity que muestra una MapView.
- ▶ MapView: View que muestra un mapa por pantalla, obtenido a través de los servicios de Google Maps. Puede recibir eventos de la pantalla y dibujar capas superpuestas al mapa.
- ▶ MapController: Clase utilidad que permite gestionar el scrollo y zooms del mapa.
- ▶ GeoPoint: Punto que referencia una posición en forma de latitud y longitud, medida en microgrados ( $\text{grados} \times 10^{-6}$ )
- ▶ Overlay: Capas que pueden ser dibujadas encima del mapa.



## Creación del primer mapa

- ▶ En este ejemplo se muestra cómo crear una Actividad con un Map:
  1. Crear un proyecto Android, asignándole en el build target Google APIs para 1.6
  2. En el AndroidManifest.xml añadir: `<uses-library android:name="com.google.android.maps" />` dentro del tag `<application>` ya que va a utilizar librerías externas (no son de Android) y `<uses-permission android:name="android.permission.INTERNET" />` ya que se accede a Internet para acceder a los mapas.
  3. En el main XML debe quedar el MapView del mapa:

```
<?xml version="1.0" encoding="utf-8"?>
<com.google.android.maps.MapView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/mapview"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:clickable="true"
  android:apiKey="Poner aquí vuestra API KEY!"
 />
```

4. Hacer que la Activity principal herede de MapActivity, que nos obligará a crear el método `isRouteDisplayed()`. Con esto ya tenemos una aplicación con un mapa visible.
5. Añadir controles de zoom:  
`MapView mapView = (MapView) findViewById(R.id.mapview);`  
`mapView.setBuiltInZoomControls(true);`
6. Maps-Base: la aplicación del ejercicio anterior debería quedar de esta manera.

## *Overlays (I)*

- ▶ Overlays son objetos visuales que pueden ser añadidos a los mapas
- ▶ Tienen coordenadas asociadas, por lo tanto cuando se mueve el mapa los overlays se mueven con el.
- ▶ Estos objetos pueden designar posiciones, líneas, áreas...
- ▶ Para añadir Overlays:
  1. Crear una clase que herede de ItemizedOverlay que gestionará los Overlays
  2. Añadir el atributo: `private ArrayList<OverlayItem> mOverlays = new ArrayList<OverlayItem>();` que almacenará la lista de Overlays y `private Context mContext;`
  3. Modificar la constructora para que quede: `super(boundCenterBottom(defaultMarker));`
  4. Crear un nuevo método para añadir los Overlays a la list:

```
public void addOverlay(OverlayItem overlay) {    mOverlays.add(overlay);    populate();}
```
  5. En el método `createItem` añadir: `return mOverlays.get(arg0);`
  6. En el método `size()` añadir: `return mOverlays.size();`
  7. Añadir una nueva constructora, que recibirá el objeto Context:

```
public HelloItemizedOverlay(Drawable defaultMarker, Context context) {    super(boundCenterBottom(defaultMarker));    mContext = context;}
```

## *Overlays (II)*

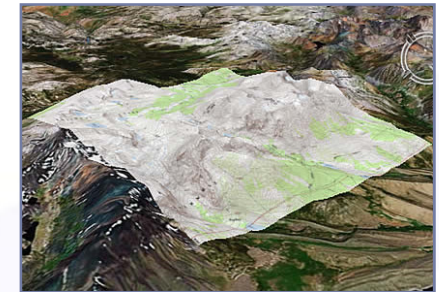
1. Sobrescribir el método de callback onTap:

```
protected boolean onTap(int index) {  
    OverlayItem item = mOverlays.get(index);  
    AlertDialog.Builder dialog = new AlertDialog.Builder(mContext);  
    dialog.setTitle(item.getTitle());  
    dialog.setMessage(item.getSnippet());  
    dialog.show();  
    return true;  
}
```

2. En el método onCreate() de la Activity añadir:

```
List<Overlay> mapOverlays = mapView.getOverlays();  
Drawable drawable = this.getResources().getDrawable(R.drawable.icon);  
HelloItemizedOverlay itemizedoverlay = new  
HelloItemizedOverlay(drawable, this);  
GeoPoint point = new GeoPoint(19240000, -99120000);  
OverlayItem overlayitem = new OverlayItem(point, "Hola, Mundo!", "I'm  
in Mexico City!");  
itemizedoverlay.addOverlay(overlayitem);  
mapOverlays.add(itemizedoverlay);
```

3. En el proyecto Maps-Base-Overlay hay un ejemplo de cómo debe quedar



# Ejemplos y ejercicios: Google Maps

## ➤ **Maps-advanced-overlay**

- ▶ En este ejemplo se muestra cómo utilizar overlays y activar los controles como Zoom y Compass (brújula).
- ▶ También se puede ver cómo dibujar encima del mapa, tanto en una posición fija en la cámara como en una posición geodésica.
- ▶ Este ejemplo captura eventos del teclado, con el método `onKeyDown()`, en este caso captura la posibilidad de cambiar el tipo de mapa y activar los controles de zoom.

## ➤ **Ejercicio**

- ▶ Crear un mapa que implemente las siguientes funcionalidades:

1. Control de zoom y brújula
2. Mapa debe iniciar en la posición inicial del GPS
3. Ubicar un icono distinto para las siguientes ubicaciones: La Estatua de la Libertad, La Torre Eiffel, la Torre de Pisa y dos ubicaciones a elegir.  
Truco: se pueden buscar las ubicaciones utilizando Google Maps y usando el siguiente script en el navegador:  
`javascript:void(prompt("",gApplication.getMap().getCenter()));`
4. Al presionar sobre una posición en el mapa se debe añadir un nuevo marcador.
5. Unir todas las posiciones con una línea (clase `Path`), esas posiciones NO deben quedar FIJAS en la pantalla. Convertir con la clase `Projection`.

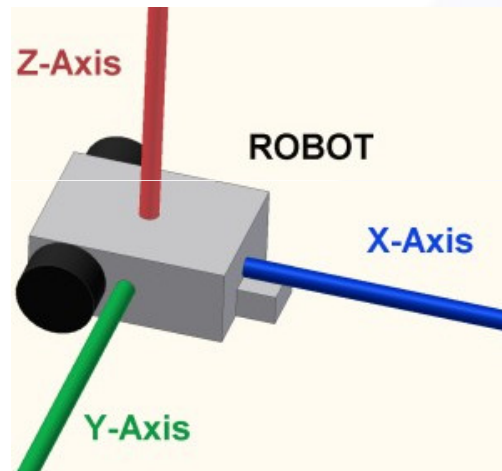


- LBS
- Localización Android
- Google Maps
- Sensores



## Sensores

- ▶ Android está preparado para soportar distintos tipos de sensores:
- ▶ GIROSCOPO: recibe giros en el dispositivo
- ▶ ORIENTACIÓN: orientación respecto al Norte del dispositivo
- ▶ ACCELERÓMETRO: detección de movimiento
- ▶ Otros: Sensor de luz, de temperatura, de proximidad, de campo magnético, de presión
- ▶ La presencia de estos sensores depende de cada uno de los dispositivos. Antes de utilizarlos se debe verificar que el dispositivo tiene estos sensores disponibles.



# Ejemplos y ejercicios: Sensores

## ➤ **LBS-sensores**

- ▶ En este ejemplo se puede ver cómo verificar de qué sensores dispone nuestro dispositivo y sus detalles técnicos.

## ➤ **LBS-sensorBrújula**

- ▶ Aquí se muestra cómo utilizar un dispositivo en concreto, en este caso la brújula. Se muestra numéricamente cual es la desviación en grados respecto al Norte

## ➤ **LBS-Brujula**

- ▶ Ejemplo similar al anterior, pero en el que se puede ver cómo utilizar la brújula y darle una interpretación visual gráfica.

