Trabajo Fin de Grado

Grado en Ingeniería Electrónica, Robótica y Mecatrónica

IOT – Posicionamiento en mapas de Google empleando bases de datos y coordenadas GPS

Autor: Sergio Real Aragón

Tutor: Alfredo Pérez Vega-Leal

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2018

Proyecto Fin de Grado
Grado en Ingeniería Electrónica, Robótica y Mecatrónica

IOT – Posicionamiento en mapas de Google empleando bases de datos y coordenadas GPS

Autor:
Sergio Real Aragón

Tutor:
Alfredo Pérez Vega-Leal
Profesor Contratado Doctor

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2018

Trabajo Fin de Grado: IOT – Posicionamiento en mapas de Google empleando bases de datos y coordenadas GPS

Autor: Sergio Real Aragón

Tutor: Alfredo Pérez Vega-Leal

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Sevilla, 2018

Agradecimientos

A mi madre, Maria del Carmen, por acompañarme desde siempre, por no permitir que me rinda, por su paciencia y por mostrarme qué es la fortaleza.

A mi padre, Juan Manuel, por su confianza, por sus clases de historia convertidas en consejos y por inspirarme con cada uno de ellos.

A mi abuela, Carmen, por cuidar de mí cuando lo he necesitado.

A mi tía, Ana Belén, por enseñarme a valorar el lado espiritual de la vida, y saber encontrar el equilibrio.

A mi pareja, Claudia, por su fe en mi y por aportarme el valor de la constancia que tanto necesitaba.

A todos ellos, a toda mi familia, y a mis amigos por el constante apoyo y cariño, que hoy me permiten estar aquí a las puertas de terminar una etapa muy importante de mi vida.

Mi más sincero agradecimiento a mi tutor, Alfredo, por permitirme realizar este proyecto junto a él, con la esperanza de que sea la base de otros muchos trabajos.

Gracias a todos, de corazón.

Sergio Real Aragón

Sevilla, 2018

Resumen

El geoposicionamiento y su interés en el desarrollo de aplicaciones para móviles se encuentra actualmente auge y todo parece indicar que así seguirá durante un largo periodo de tiempo, el hecho de que podamos llevar en el bolsillo un ordenador más potente que los que existían en nuestros escritorios hace diez años nos brinda un abanico de oportunidades muy extenso.

Este proyecto consiste en aprender de forma autodidacta, habilidad que hemos entrenado durante el Grado, para posteriormente aplicar los conocimientos en el desarrollo de una aplicación para el sistema operativo de Android para teléfonos móviles. El objetivo principal es que, leyendo el presente documento, cualquier persona interesada sea capaz de configurar y poner en marcha la aplicación, entenderla, y realizar las modificaciones que se estimen. Es un trabajo destinado a servir como herramienta para futuros proyectos que necesiten las funciones de la aplicación que aquí se desarrolla.

La aplicación consiste en mostrar en un mapa de Google Maps una secuencia de posiciones geográficas que previamente han sido instroducidas en una base de datos MySQL. Aquí no solo vamos a tener en cuenta el diseño y programación en Android, sino absolutamente todo lo necesario para que la aplicación funcione. Esto quiere decir que se aplicarán conceptos de Android, de bases de datos MySQL, PHP, FTP y del software necesario.

Abstract

Geolocation and its relationship with the development of mobile applications is becoming more and more popular and it seems that this will continue for a long time. The fact that we can carry in our pocket a more powerful computer than the those that existed ten years ago offers a wide range of oportunities to create and innovate.

This project is based in the DIY (Do it Yourself) philosophy. It consists in learning by yourself, a very important skill that is trained during this Degree, and then apply the knowledge acquired to develop an Android App. The main goal is that, by reading this document, any interested person could be able to create an application that works properly and to understand enough to modify the App in the way they need.

This is a work intended to serve as a tool for future projects based in this technology. The application shows a map where a sequence of geographical coordinates that have been previously inserted in a MySQL database are drawn. Not only we need to program the Android App but also, we have to setup everything behind that like the MySQL database, php query programming, managing a host and learn how to use all the software we need.

Índice

ÍNDICE DE FIGURAS

Notación

GPS	Global Positioning System
S.O.	Sistema Operativo
BBDD	Base de datos
XML	eXtensible Markup Language
App	Applicaction (Como programa informático)
AVD	Android Virtual Device
FTP	File Transfer Protocol
API	Application Programming Interface
SQL	Structured Query Language
TIOBE	The Importance of Being Earnest
IDE	Integrated development environment
APK	Android Application Package

1 INTRODUCCIÓN

La geolocalización es un concepto que nace a principios del siglo XX. En ese tiempo se podía determinar la posición de un barco o un avión utilizando la recepción de señales de radio y aplicando el principio de la triangulación.

Uno de los sistemas más populares de la época, desarrollado por motivaciones bélicas al igual que una gran cantidad de innovaciones e inventos, fue el radiogoniómetro, que era capaz de detectar la dirección de procedencia de una señal de radio. Con este dispositivo, con un lápiz y regla y un mapa se podría conocer la posición de un barco que se encuentre en medio del mar.

Ya en 1993 el Departamento de Defensa de los Estados Unidos terminó de desarrollar el sistema que actualmente se utilizado que se denomina GPS, haciendo uso de satélites que orbitan alrededor de la Tierra.

Utilizando esto como base, aparecen mapas digitales donde podemos visualizar la posición de un determinado dispositivo electrónico. Google es una de las compañías que se interesa en el desarrollo de este tipo de software y hoy día posee el que puede ser el mapa más consultado, no solo para llegar de un punto a otro como antaño, sino para más funciones que se irán mencionando a lo largo de este documento.

1.1 Justificación

Por la importancia que tiene hoy día los mapas de Google y el posicionamiento en mapas/geolocalización resulta interesante indagar más en estos conceptos. Aún contando con la gran cantidad de aplicaciones que se han creado haciendo uso de un mapa, sigue existiendo una gran proyección en el uso de esta tecnología. Estos mapas están disponibles como aplicación de escritorio, como servicio web, como navegador GPS, y en dispositivos móviles como tablets y smartphones.

Hoy día prácticamente toda persona que haya tenido en sus manos un teléfono móvil tiene una ligera idea de qué es Android. Android es un sistema operativo creado por Andy Rubin, Nick Sears, Chris White y Rich Milner, que más adelante Google adquirió. La idea inicial fue usarlo en cámaras digitales, aunque luego derivó a un completo sistema operativo capaz de mantener numerosas aplicaciones ejecutándose simultáneamente.

Tras quince años en el mercado Android se ha convertido en el sistema operativo más popular utilizado en diversos dispositivos además de en smartphones. Gran parte de esta popularidad se debe a ser de código abierto, característica que ha atraído a miles de desarrolladores interesados en crear contenido para esta plataforma. Además, la misma compañía apoya este comportamiento ofreciendo gran cantidad de facilidades a aquellos que quieran comenzar a desarrollar para su sistema operativo.

La gran cantidad de aplicaciones diferentes que pueden crearse para un dispositivo móvil utilizando Android lo convierte en otra razón de peso para realizar un trabajo sobre ello en un Grado altamente relacionado con la innovación.

1.2 Objetivos

Este Trabajo Final de Grado se centra en búsqueda de información e investigación de todo lo relacionado con mapas de Google para finalmente emplear los conocimientos adquiridos y proceder con la implementación de una aplicación de Android que haga uso de mapas de Google y posicionamiento basados en coordenadas GPS.

El objetivo principal es que cualquier persona interesada que lea y entienda el presente documento sea capaz de tener una aplicación de Android completamente funcional en su teléfono móvil, ya que no es algo trivial y hay que dedicar bastante tiempo porque además de tener que aprender qué hace falta para comenzar con el desarrollo de una aplicación en Android, también necesitamos tener unas contundentes bases de programación y conocer uno de los lenguajes soportados por el S.O.

Este proyecto también tiene como intención servir de soporte para futuros trabajos, por ejemplo, usarse para comprobar el funcionamiento de proyectos que envíen coordeadas a una BBDD, pudiendo ser modificado a placer según las funciones que se necesite en ese momento.

Para conseguirlo vamos a proceder con una explicación conceptual, seguida de las herramientas necesarias para poder llevar todo a cabo y por último a la explicación práctica y paso por paso de lo que debemos hacer para tener una aplicación que haga uso de lo mencionado al principio de este aparatado. El apartado práctico será un tutorial guiado, haciendo uso de imágenes para que sea más sencillo seguir los pasos. En mi caso he tenido que aprender los lenguajes que se van a utilizar, pero esto queda excluido de las explicaciones por no extender el contenido del proyecto. Igualmente se proporcionarán todos los códigos aquí utilizados adecuadamente comentados para que se entienda qué realiza cada línea.

2 CONCEPTOS RELEVANTES

Es importante que introduzcamos los conceptos más significativos que vamos a utilizar en esta documentación. Vamos a definir qué es cada ítem que vamos a usar, mencionar detalles importantes como su contexto histórico, quién es el creador o de donde procede su existencia y sus funcionalidades más importantes.

2.1. GPS

2.1.1 Qué es

El GPS es un sistema desarrollado por el Departamento de Defensa de los Estados Unidos que utiliza satélites para obtener información de la geolocalización de receptores electrónicos. Gracias a ellos se podrá determinar la posición geográfica de dispositivos que cuenten con un receptor GPS. Este sistema es, básicamente, una constelación de veinticuatro (treinta y dos desde el año 2016) que están orbitando a una altura de unos 20.000 km con respecto a la superficie de la tierra. Cada uno de estos satélites da una vuelta rodeando la tierra cada doce horas. Todo está diseñado de forma que un dispositivo que quiera conocer su posición terrestre siempre pueda ver al menos cuatro satélites diferentes en todo momento, tres para conocer la posición y uno más como medida redundante a fin de mejorar la precisión.

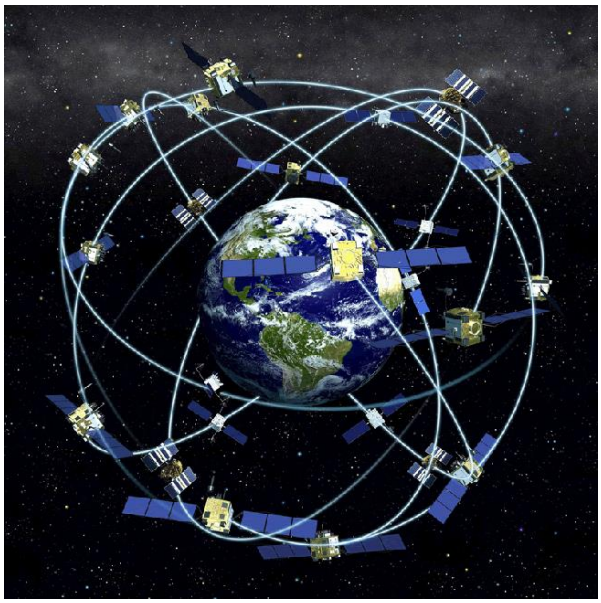


Ilustración 1. Constelación GPS.

Para conocer donde nos encontramos, la tecnología de GPS se basa en el concepto matemático de trilateración, término que se tiende a confundir con la triangulación. Esto consiste en conocer la distancia que nos separa a tres puntos diferentes cuyas ubicaciones conocemos, dibujar tres circunferencias de centro cada punto, y el punto de interesección de las tres circunferencias será el punto donde nos encontramos. Para obtener la distancia del receptor GPS hasta los satélites se calcula el tiempo que tarda la señal en viajar desde los satélites hasta el punto donde está el receptor y en el mismo receptor se realizan las operaciones matemáticas del cálculo de esa distancia.

2.1.2 Principio de funcionamiento

El satélite GPS emite continuamente señales de radio que contienen información sobre su localización, estado y la hora obtenida de un reloj atómico interno. Esta señal viaja por el espacio a la velocidad de la luz y los receptores GPS la reciben, guardando el momento exacto de llegada. Con el dato de la hora contenida en la señal de radio y el dato obtenido a la llegada se conoce el tiempo que ha tardado en recorrer el camino. Esta distancia se calcula como el producto de la velocidad por el tiempo que ha tardado. Este mismo proceso lo realiza varias veces para los distintos satélites que el receptor GPS tenga a la vista y una vez que tenga esta información procede, usando el principio matemático de la trilateración, a determinar su localización. Esta posición obtenida, en el caso de contar con un buen receptor GPS, tiene un error de hasta cinco metros (tanto horizontal como verticalmente).

Es interesante indicar que no existe un límite de usuarios (dispositivos utilizando el receptor GPS) que deseen obtener su posición de manera simultánea. Otro de los motivos que hacen que esta tecnología sea hoy día la más utilizada para conocer la localización de dispositivos es su resistencia a diversas intereferencias y a fuentes de error. Aún no se ha conseguido que este sistema sea totalmente altamente preciso (definiendo preciso aproximadamente 0.5 metros de error), pero se está trabajando duramente en la actualidad día para conseguirlo.

2.1.3 Trilateración

Hay que tener en cuenta que el método matemático de la trilateración y la triangulación son diferentes, aunque hoy día se tiende a denominar a ambos conceptos como triangulación. La principal diferencia entre ambas es con qué datos se realizan los cálculos, en el caso de la triangulación utilizamos los ángulos y en la trilateración las distancias.

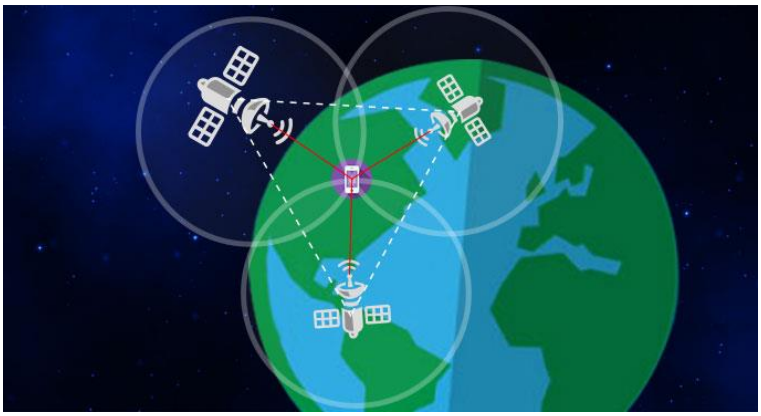


Ilustración 2. Trilateración. (Fuente: blog.semaphore-software.com)

La trilateración es el método usado para posicionar el receptor en los sistemas GPS con lo cual vamos a ampliar la información sobre el mismo. Este método permite posicionar de forma relativa objetos usando la geometría de circunferencias (o esferas) y triángulos.

Para comprender cómo es el proceso se va a realizar un script en Matlab, contando con unos datos de referencia (que son los mismos con los que contaríamos en la realidad), y vamos a obtener las coordenadas un receptor. Se va a obtener en el plano 2D ya que basta para comprender el método de la trilateración y en 3D se complica en exceso y no entra en el alcance de este proyecto.

El código en Matlab es el siguiente:

```
%Posiciones de los satélites
s1 = [0, 0];
s2 = [21 ,0];
s3 = [9, 15];

%Las almacenamos en vectores
lon = [s1(1),s2(1),s3(1)];
lat = [s1(2),s2(2),s3(2)];
```

```
%Distancia al receptor GPS
d1= 13; d2=20; d3=5;

%Cálculo de las circunferencias
th = 0:pi/50:2*pi;

%Circunferencia 1
x1 = d1 * cos(th) + lon(1);
y1 = d1 * sin(th) + lat(1);
%Circunferencia 2
x2 = d2 * cos(th) + lon(2);
y2 = d2 * sin(th) + lat(2);
%Circunferencia 3
x3 = d3 * cos(th) + lon(3);
y3 = d3 * sin(th) + lat(3);

plot(lon,lat,'*');grid;axis([-20 50 -30 30]);
xlabel('Longitud');ylabel('Latitud');
title('Obtener posición receptor GPS por trilateración')
hold on;
plot(x1,y1);
hold on;
plot(x2,y2);
hold on;
plot(x3,y3);
```

Con el que obtenemos la siguiente salida:

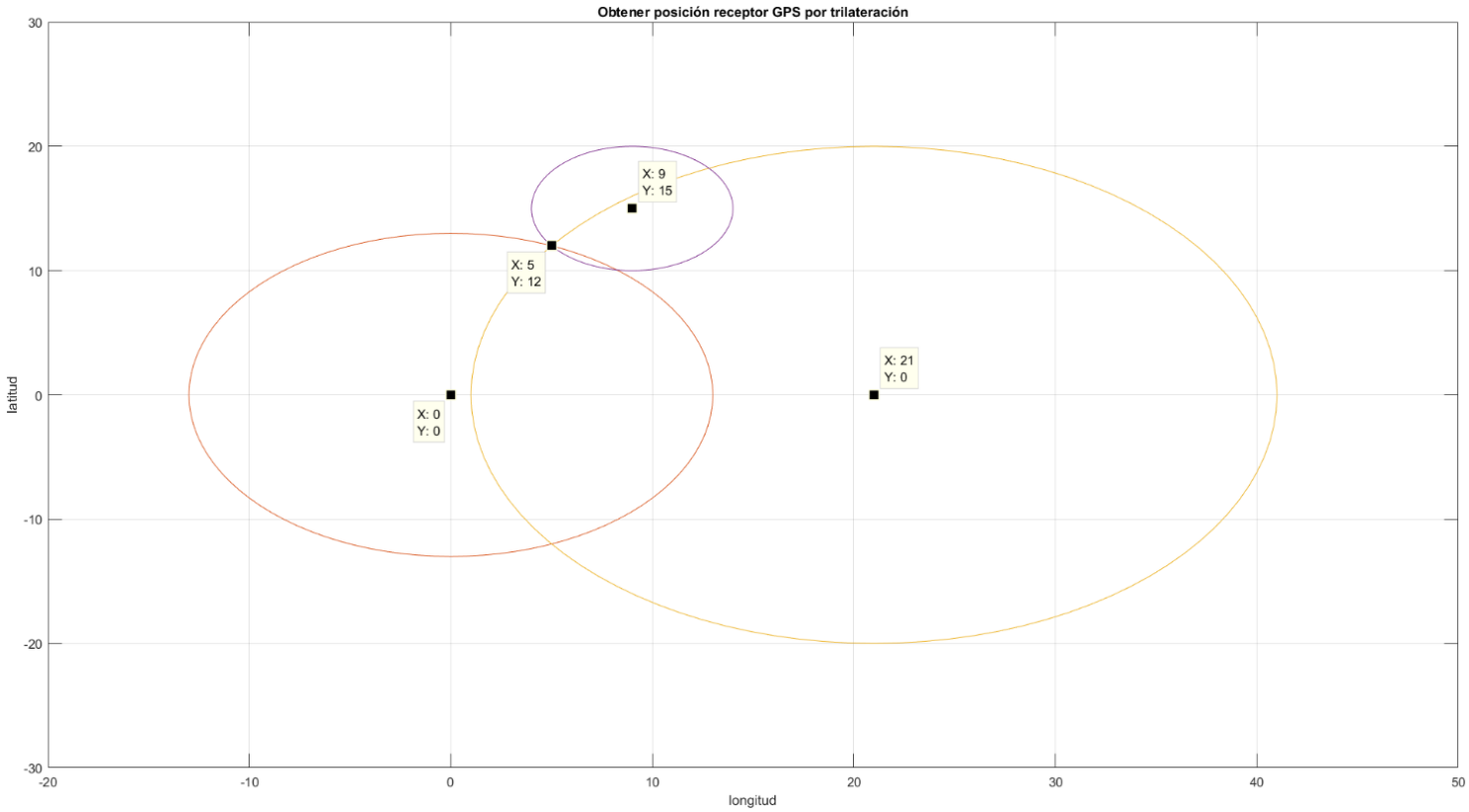


Ilustración 3. Resultados Matlab: Trilateración.

Si usamos la herramienta de selección de punto de Matlab vemos que la intersección de las tres circunferencias es el X =5 e Y =12, esa sería la posición del receptor GPS. Es preciso recordar que en el caso real donde obtenemos las coordenadas en tres dimensiones se necesitan cuatro satélites.

2.1.4 Señal GPS

Los satélites GPS transmiten señales de radio que son percibidas por receptores GPS. Estas señales proveen a los receptores de la información necesaria para poder conocer su ubicación. Todos los satélites GPS transmiten en las mismas dos frecuencias, y utilizando técnicas de modulación spread spectrum permitiendo que varias señales se transmitan en la misma frecuencia sin contaminarse entre ellas, siendo el receptor GPS el encargado de separarlas. Las dos frecuencias principales son:

- L1 (1575.42 MHz): Contiene dos señales, una denominada C/A (Coarse and Acquisition) y otra P(Precise). Esta es la frecuencia que los receptores GPS usados en ámbitos civiles reciben.
- L2 (1227.60 MHz): Contiene una copia de la señal P. En este caso solo lo reciben receptores de aplicaciones militares. Ésto es lo que les permite tener más precisión que los receptores de uso civil. Aún se piensa que lo que provoca la mejora de precisión del receptor de uso militar con respecto al civil era la existencia de la disponibilidad selectiva, pero ésta ha sido eliminada en el año 2000.

La información transportada por la señal GPS contiene tres tipos de datos:

- Código pseudo-aleatorio: Identifica al satélite que transmite la información.
- Datos efemérides: Indica el estado, fecha y hora actual usadas por el receptor GPS para determinar su posición.
- Datos almanaque: Contiene información de la posición del satélite y sobre la órbita de ese satélite.

Toda la información se encapsula en un mensaje con un formato determinado. Los datos se encapsulan en palabras de 30 bits y luego las palabras en grupos de diez, a esta agrupación se le denomina en inglés "subframe". El mensaje completo está formado por cinco subframes, numeradas del 1 al 5, y se transmite por completo en 30 segundos. El satélite transmite el mensaje repetidamente, cambiando el contenido según proceda. La estructura es:

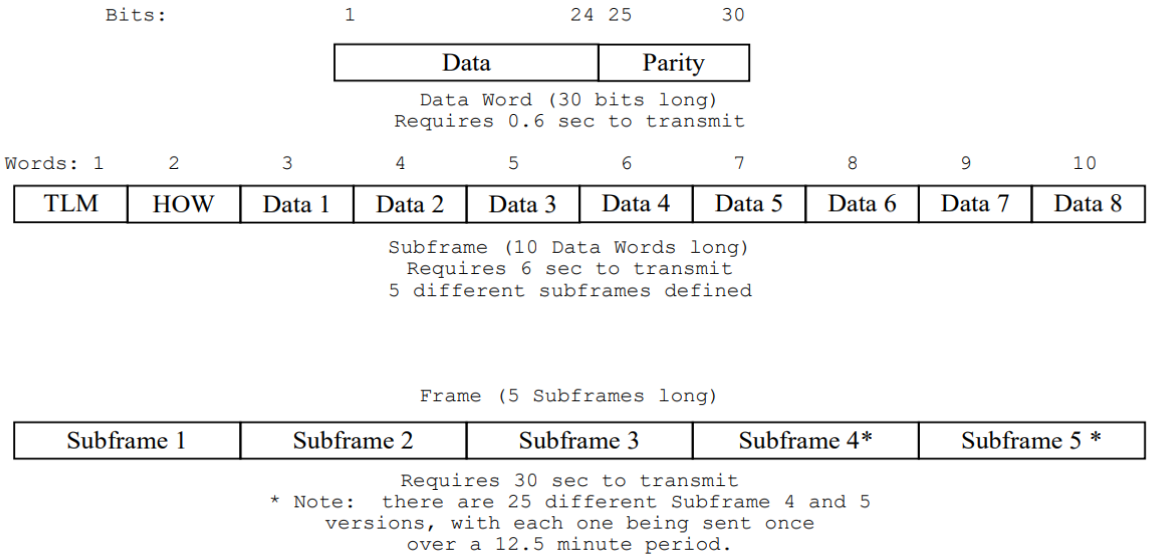


Ilustración 4. Estructura del mensaje contenido en la señal GPS. (Fuente: www.inventeksys.com)

- Subframe 1: Los satélites tienen en su interior un reloj atómico altamente preciso, pero se producen derivas de vez en cuando. Este fragmento del mensaje contiene información relevante para corregir estos pequeños errores en los relojes de los satélites.
- Subframe 2 y 3: Contiene los datos efemérides. Gracias a estos datos el receptor puede obtener la localización del satélite en un momento concreto. Esta información se actualiza una vez a la hora.
- Subframe 4 y 5: Estos fragmentos del mensaje contienen los datos de almanaque, permitiendo que el receptor GPS tenga acceso a información no solo del satélite sino también de los satélites de esa constelación.

2.1.5 Precisión y fuentes de error

Un buen receptor GPS es capaz de obtener su posición con una precisión de unos tres metros, aunque hay métodos para mejorar esta precisión que luego mencionaremos. A continuación, se mencionan las fuentes de error más comunes que provocan el sistema GPS no sea tan preciso como se desearía:

- Refracción atmosférica: Causados por la ionosfera y la troposfera, la velocidad de transmisión de la señal se ve afectada al pasar estas capas, provocando un error en el cálculo de la distancia.

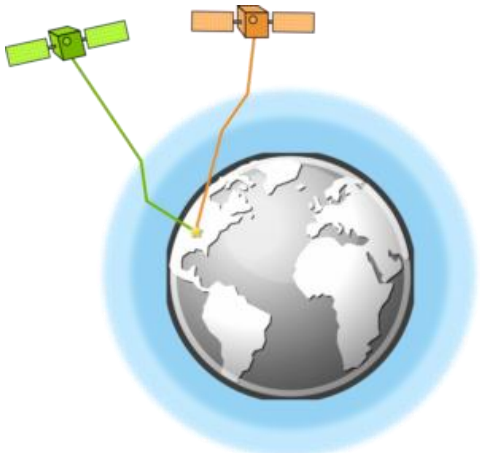


Ilustración 5. Refracción atmosférica. (Fuente gisgeography.com)

- Efecto multitrayectoria: Este error consiste en que la señal no incide directamente en el receptor GPS, sino que llega después de haber "rebotado" en una superficie reflectora, provocando una medición falsa.

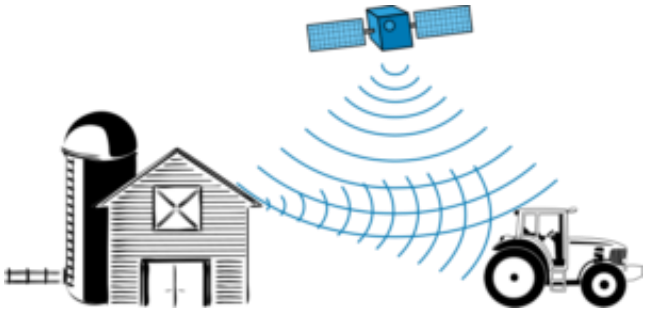


Ilustración 6. Efecto multitrayectoria. (Fuente gisgeography.com)

- Errores en los relojes (tanto del satélite como del receptor): Esto lo hemos mencionado al hablar del subframe 1. Se provocan unas pequeñas derivas en los relojes y en consecuencia un mal cálculo de la posición. Estas derivas se observan continuamente para que no se acumulen.

Ilustración 7. Errores en los datos efemérides. (Fuente gisgeography.com)

- Dilución de la precisión geométrica: Este error aparece a causa de la posición relativa de los satélites usados para el cálculo de la ubicación. Si los satélites están suficientemente separados y bien distribuidos apenas se produce error por esta causa, pero si se encuentran muy pegados puede aumentar la inexactitud en el orden de metros.

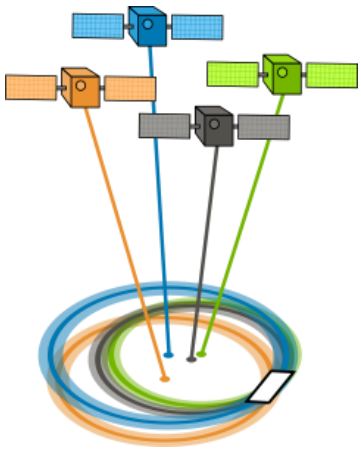


Ilustración 8. Dilución de la precisión geométrica. (Fuente gisgeography.com)

Tras mencionar estas fuentes de errores que provocan que tengamos una localización menos precisa, mencionaremos una de las formas más típicas y efectivas de mejorar la precisión, el uso del GPS diferencial. Esto consiste en que no solo toma las medidas el receptor GPS, también lo hace una estación terrestre cercana al receptor. Estas medidas tomadas en la estación terrestre se utilizan para corregir errores en la ubicación del receptor GPS. Este sistema es capaz de conseguir una precisión incluso de decenas de centímetros y se utiliza mucho para receptores en movimiento continuo y que necesitan gran precisión para ofrecer seguridad, como por ejemplo los aviones a la hora de aterrizar.

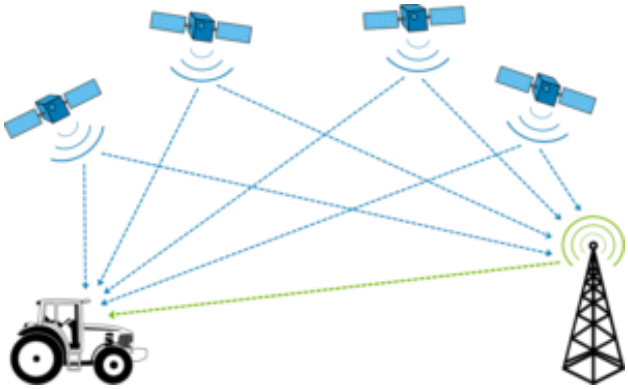


Ilustración 9. GPS diferencial. (Fuente gisgeography.com)

2.1.6 Coordenadas GPS

Las coordenadas GPS están formadas por las componentes de latitud (posición norte-sur) y longitud (posición este-oeste) y una tercera componente denominada altitud (elevación sobre el nivel del mar).

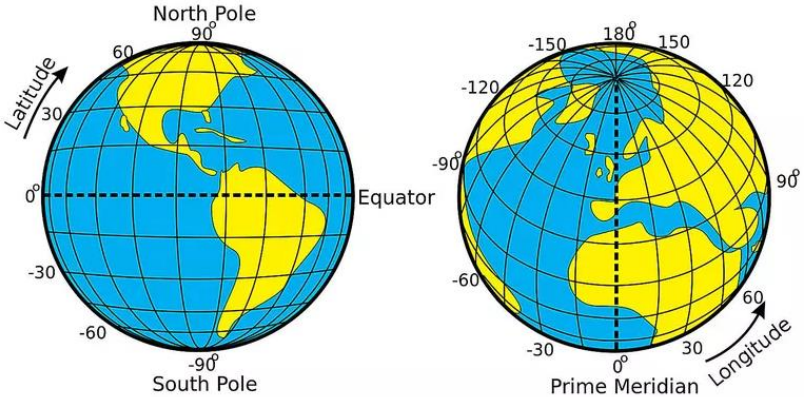


Ilustración 10. Latitud y longitud. (Fuente: www.thoughtco.com)

Hay dos principales unidades de medida, coordenadas decimales y sexagesimales.

- Las decimales, que son las que usaremos en este trabajo, que se expresa en grados º.
- Las sexagesimales que tienen tres componentes: grados, minutos y segundos.

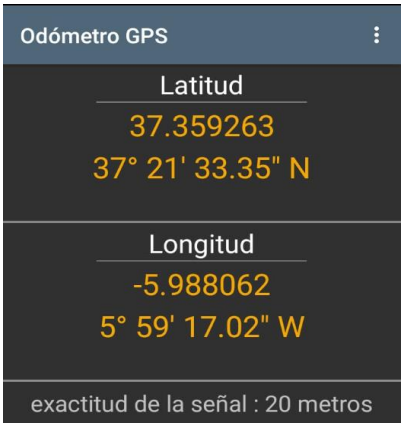


Ilustración 11. Ubicación actual usando “My GPS Coordinates”.

2.1.7 GPS en Android

El servicio de localización en los sistemas basados en Android utiliza la tecnología de GPS para conocer su ubicación en la Tierra. Esta función de localización no es obligatoria y debes activarla si quieres hacer uso de ella. Activar o desactivar esta opción significa que le das o no permiso a tu sistema operativo a que acceda a la lectura del receptor GPS que hoy día incluyen todos los dispositivos móviles.

En la ilustración 2 hemos utilizado una aplicación gratuita de la *Play Store* para obtener nuestra ubicación al momento de estar redactando este documento.

2.2 Google Maps

2.2.1 Qué es

Es un servicio digital que nos provee con información detallada sobre el mapa del mundo. Además de mapas callejeros convencionales, el software de Google Maps nos ofrece vista satélite y vista terreno, que es igual a la vista por defecto a la que se le añade la variable de altitud. Otra vista interesante es la denominada "Street View", que nos permite obtener la vista que tendríamos si nos encontrásemos en ese punto geográfico, de esto se encargan los famosos coches de Google Maps que recorren las calles del mundo tomando imágenes panorámicas por donde pasa.

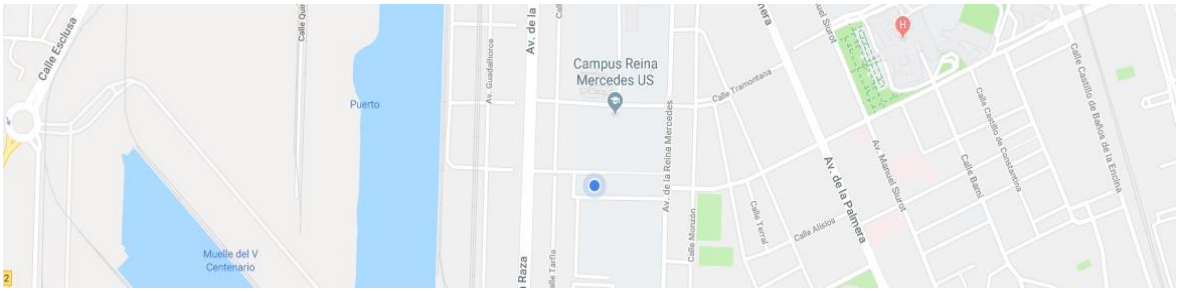


Ilustración 12. Imagen en modo *Mapa* en Google Maps.

Este software nació en principio como una aplicación de escritorio creada por la compañía "Where 2 Technologies", que más adelante Google compró y convirtió en una aplicación web. Hoy día podemos disfrutar de ella en escritorio, web y dispositivos android.

2.2.2 Servicios

Además de las diferentes vistas del mapa del mapa del mundo, Google Maps nos ofrece diversos servicios como:

- Un planificador de rutas para conductores de vehículos, usuarios de transporte público o viandantes, que te indica las diferentes formas de llegar de un punto a otro, con un tiempo estimado de llegada y, en el caso de los conductores, un indicador de tránsito en tiempo real que puede ayudarte a evitar atascos.
- Para móviles, este software incluye, utilizando localización GPS, una herramienta que nos permite tener a otro dispositivo localizado en todo momento en el mapa. Por ejemplo, si alguien va a viajar, o alguien que va a hacer senderismo puede compartir donde se encuentra en cada momento en aras de evitar posibles accidentes.
- Un indicador de sitios de interés, ocio, servicios públicos(hospitales, comisarías de policía, etc ...), estaciones de transporte público para ayudar a aquellas personas que estén de turismo puedan conocer en todo momento dónde se encuentran y a dónde puede ir para conocer la ciudad.
- Además, como punto más fuerte a lo que nos concierne, la propia compañía de Google Maps nos provee de una extensa API tanto para web, ios y android que podremos utilizar con el fin de crear software propio en los que necesitemos hacer uso de algunos de sus servicios.

  Esta documentación, que es totalmente pública, no solo consta de las funciones con la descripción de lo que hacen, sino que también nos ofrecen ejemplos de uso y pequeños tutoriales para facilitar el aprendizaje de aquellos que quieren comenzar a desarrollar aplicaciones para su sistema operativo.

Estos son únicamente los servicios que he considerado relevantes mencionar, pero es importante mencionar que no son los únicos que ofrece este software.

2.2.3 Google Maps en Android

Para los desarrolladores en Android, que es el caso que nos concierne, la API de Google Maps nos ofrece una gran cantidad de servicios. Haciendo uso del software oficial que ellos mismos nos recomiendan, "Android Studio", y utilizando los métodos que están en la documentación que nos ofrecen podemos añadir un mapa a nuestra aplicación y, sobre éste, marcar puntos de interés, colocar marcadores, polilíneas, obtener información, etc...

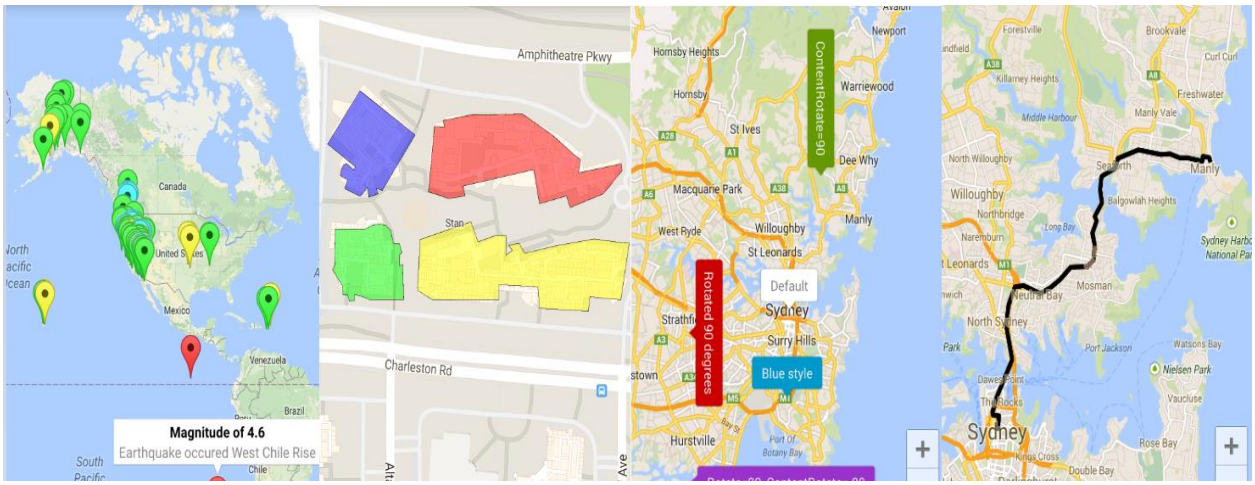


Ilustración 13. Utilidades de dibujo sobre Google Maps en Android. (Fuente: developers.google.com/maps/)

2.3 Bases de datos

2.3.1 Qué son

En la época del nacimiento de los computadores los datos, denominados en ese momento sistemas de ficheros, se codificaban junto con el programa desarrollado, es decir, se diseñaban exclusivamente para que esa aplicación los usara. Esto creaba una gran dependencia ya que si se modificaban los datos se modificaba provocaba que tuviésemos que recompilar el mismo programa. Otro problema surgió cuando se necesitaba compartir esa información, al no ser compatibles los formatos de unas aplicaciones con otras, el proceso de exportación/importación de datos era tedioso. Las bases de datos nacen para arreglar estos inconvenientes.

Una base de datos es, en resumen, una colección de datos que están organizados de forma que se puedan gestionar, actualizar, compartir y acceder fácilmente. Las bases de datos ayudan a organizaciones a mantener correcto control de toda la información que poseían y ayudar a que estos datos no desaparecieran.

Para realizar las funciones mencionadas se suele hacer uso de un Sistema de Gestión de Bases de Datos (SGBD en adelante), que es un software creado exclusivamente para hacer mas triviales todo lo relacionado con la gestión de una base de datos y permitir que un usuario no experto pueda encargarse de esa misma gestión. Access, Oracle, SQLServer, MySQL son conocidos sistemas gestores de base de datos, en nuestro proyecto haremos uso de MySQL. Algunos de los servicios que una SGBD proporciona son los siguientes:

- Definición de la base de datos mediante un lenguaje de definición de datos.
- El SGBD permite la inserción, eliminación, actualización y consulta de datos mediante un lenguaje de manejo de datos (se usa el conocido SQL que más adelante mencionaremos).
- Otro servicio es el acceso de usuarios a la base de datos de forma controlada, contando además con servicios de seguridad contra el acceso no permitido y un sistema de integridad de los datos para que no sean eliminados de forma no autorizada.

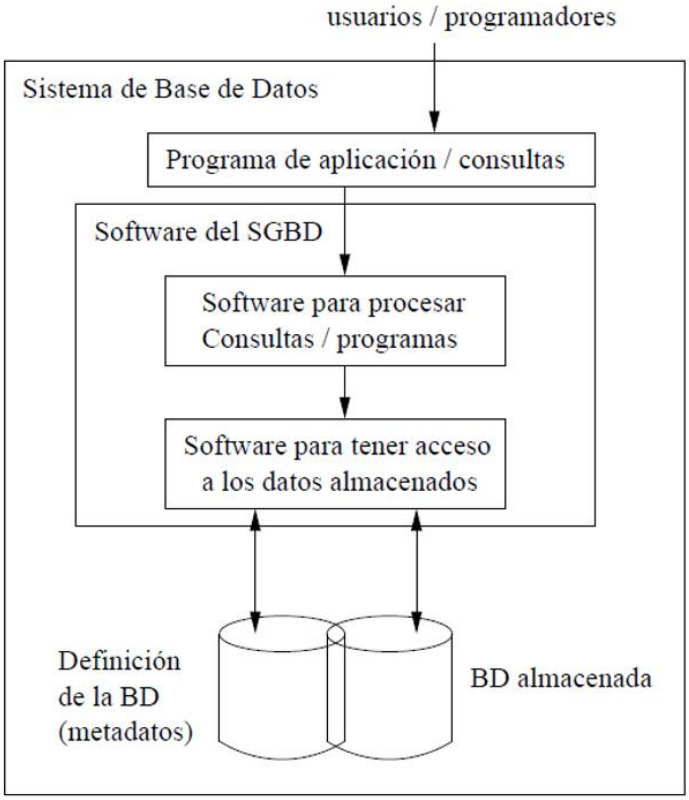


Ilustración 14. Sistema de Base de datos. (Fuente: Ramez Elmasri, Fundamentos de Sistemas de Bases de Datos, 2007)

2.3.2 Bases de datos basadas en modelos relacionales

Estos SGBD se basan en un modelo de datos, en concreto la mayoría de los SGBD que se utilizan hoy día se basan en un modelo relacional. Pero ¿qué es un modelo de datos? Esto es un conjunto de conceptos y reglas que se utilizan para describir la estructura de una base de datos, es decir, para describir los datos, las relaciones existentes entre ellos y las restricciones que deben cumplirse.

La aplicación de este modelo a las BBDD fue presentada por un programador en IBM que propuso cambiar la forma de almacenar los datos, en vez de hacerlo de forma jerárquica se organizarían en tablas formadas por columnas y filas. Cada una de las tablas estaría formada por un conjunto de columnas, que serían atributos, y las filas que serían los registros. La ventaja de estas bases de datos con respecto a las originales es que, teniendo varias tablas podemos relacionarlas entre sí mediante una columna denominada "clave primaria", que será única para cada registro. Esto nos permite manejar una gran cantidad de datos de forma más ordenada y nos ayudará a reducir el coste computacional a la hora de obtener información de la base de datos.

DNI	Nombre	Apellidos	Dirección	Teléfono
11111111X	Juan	Gómez	Robles,1	611111111
22222222Q	María	Rodríguez	Manzano,2	622222222
33333333F	Esther	Sánchez	Peral,3	633333333
44444444D	Rodolfo	Gutiérrez	Naranjo,4	644444444
55555555H	Antonio	Ramos	Alcornoque,5	655555555

Id	Nombre	Créditos	Profesor
1	Bases de Datos	6	1
2	Programación	6	1
3	Álgebra	6	2
4	Automatizació	6	3
5	Electrónica	6	3

Ilustración 15. Ejemplo de una tabla de la entidad *Alumno* (izq) y la entidad *Asignatura* (der.)

Una relación es una asociación entre dos entidades (tablas, como hemos denominado anteriormente) que están relacionadas de alguna manera en el mundo real. Las relaciones que pueden darse entre dos tablas pueden ser de varios tipos:

- Relación uno a uno (1:1): Si cada registro de la tabla A está relacionado con un solo registro de la tabla B. Por ejemplo, una tabla es de los departamentos de una universidad y otra tabla de profesores. La relación uno a uno será que cada departamento tiene un solo encargado y un profesor solo puede encargarse de un departamento.
- Relación uno a muchos (1: N): En este caso, cada registro de la tabla A puede estar relacionado con varios de la tabla B, pero cada uno de los de B solo estaría relacionado con uno de A. Por ejemplo, tenemos una tabla profesores(A) y una tabla asignaturas(B), podemos tener que un profesor imparta varias asignaturas, pero una asignatura solo puede ser impartida por un profesor.
- Relación muchos a muchos (N: N): Cada registro de la tabla A puede estar relacionado con varios de la tabla B y viceversa. Si tenemos una tabla alumnos y una tabla asignaturas, podemos establecer la relación 'matriculado en' que significa que un alumno puede estar matriculado en varias asignaturas y en una asignatura puede haber varios alumnos matriculados. Para llevar a cabo este tipo de relación se crea una tercera tabla.

A continuación, se muestra como quedaría el diseño de la base de datos en Access con las relaciones anteriormente mencionadas.

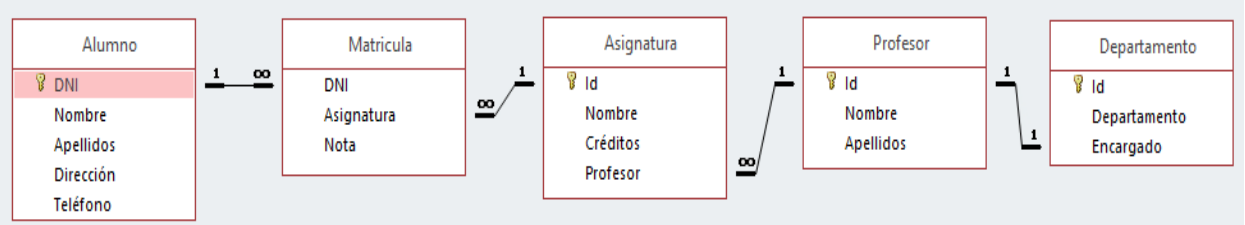


Ilustración 16. Ejemplo de diseño base de datos relacional.

2.3.3 SQL

SQL o Lenguaje estructurado de consultas, es un lenguaje para interactuar con bases de datos relacionales. Este lenguaje, además de realizar consultas puede realizar otras operaciones como crear bases de datos desde cero, crear campos nuevos, eliminarlos, modificar las propiedades de estos, establecer relaciones entre dos tablas de una base de datos, etc...

Como dato relevante hay que mencionar que SQL no es considerado un lenguaje de programación, ya que carece de las estructuras propias que tienen los lenguajes de programación como los condicionales, bucles y uso de variables.

Suponiendo que disponemos de una base de datos en un servidor, mandaríamos nuestra solicitud SQL a ese servidor, este servidor se encargaría de obtener los datos y de devolver sólo aquellos por los que hemos preguntado. El software del servidor encargado de obtener esos datos de la base de datos sería, como ya hemos mencionado en el apartado anterior, un Sistema de Gestión de Bases de Datos. Todos estos programas utilizan SQL como lenguaje para interactuar con las bases de datos.

2.3.4 Diseño y creación de una BBDD

Antes de crear una base de datos habría que diseñarla. En nuestro caso la base de datos que utilizaremos constará de una sola tabla, su diseño es trivial. Igualmente vamos a indicar las fases recomendadas para un diseño correcto de una BBDD.

Fases del diseño:

- Recolección y análisis de requerimientos.
- Creación de un esquema conceptual usando un modelo de alto nivel.
- Traducir el esquema conceptual a un esquema lógico usando el modelo de datos de la BBDD que queremos implementar.
- Traducción del esquema lógico anterior convirtiéndolo en la base de datos requerida.

Las formas de crear bases de datos suelen ser dos:

- Por un lado, utilizando la interfaz proporcionada por el SGBD. Este modo es bastante sencillo e indagaremos en él en la puesta en marcha del proyecto.
- Con SQL, utilizando los comandos existentes para hacerlo. La sintaxis para hacerlo sería un simple "CREATE DATABASE NombreBaseDeDatos", y para crear la tabla que contendrá los datos procederíamos de la siguiente forma:

```
CREATE TABLE nombreTabla(
                columna1 tipodeDato,
                columna2 tipodeDato,
                .....
                columnN tipodeDato,
                PRIMARY KEY( nombre de columna que lo sea )
            );
```

2.3.5 Consultas SQL

Uno de los conceptos más importantes y más útiles en una base de datos son las consultas. Una consulta nos permite extraer la información que necesitemos de una forma ordenada, usando los filtros que estimemos oportunos para obtener únicamente los datos que cumplan ciertos criterios. Es decir, si tenemos una base de datos extensa y queremos sólo conocer datos específicos, utilizaremos realizaremos una consulta a la base de datos para obtenerlos.

Las consultas en SQL están formadas por él comando SELECT, y las diferentes cláusulas que nos permiten filtrar de manera que obtengamos sólo los datos que queremos, estas cláusulas son:

- SELECT Indicamos las columnas que queremos mostrar en el resultado.
- FROM Aquí se especifica la tabla de donde queremos obtener los registros.
- WHERE Indicamos las condiciones que queremos que cumplan esos registros.
- GROUP BY Lo usamos para agrupar los registros en función de un campo.
- HAVING Equivalente al WHERE, pero aplicado a los grupos.
- ORDER BY Ordena los registros en función de un campo.

A la hora de escribir una consulta, estas cláusulas deben escribirse necesariamente en un orden determinado, y ese orden es el descenciente de la descripción de cada una de las cláusulas que se indican arriba.

2.3.6 BBDD en Android

Las bases de datos en Android se clasifican según donde tenemos almacenados los datos, la primera forma es usando almacenamiento externo, que consiste en usar una base de datos que se encuentre en un servidor externo, gestionado con un gestor de bases de datos y a la que realizaremos consultas y obtendremos únicamente los datos que necesitemos.La segunda forma sería una base de datos donde toda la información queda almacenada en el propio terminal de android, usando el gestor SQLite.

Teniendo la base de datos en el mismo terminal nos puede ahorrar tiempo y recursos a la hora de realizarse cualquier proceso que necesite obtener información, pero es ineficiente si lo que queremos es estar almacenando datos constantemente de forma remota.

En nuestro caso, teóricamente, un módulo GPS enviará las coordenadas a la base de datos que se encuentra en un servidor remoto, con lo cual usaremos la opción de tener una base de datos en un servidor e iremos accediendo al mismo a medida que necesitemos información para procesarla.

2.4 Android

2.4.1 Qué es

Android es el sistema operativo más popular en el mundo, es de código abierto, desarrollado en un principio para teléfonos móviles convirtiéndolos en smartphones,pero también incluyen tablets, smartwatches, pantallas para navegador GPS y control de música coches modernos y en la actualidad incluso en televisores, los denominados smart TVs.

Android es, básicamente, un software programado en Java y basado en el sistema operativo de Linux que permite que nuestro hardware funcione. Este sistema operativo nos da acceso a aplicaciones, creadas por Google, por empresas, o por usuarios. El tipo de aplicaciones que soporta el SO de android es muy diversa. Con ellas podemos reproducir música, vídeos, navegar por la web, hacer fotos, jugar, usar el dispositivo como navegador GPS, pagar en un restaurante, pedir comida a domicilio y un largo etcétera.Por todo lo mencionado, este SO se ha convertido hoy día en algo prácticamente indispensable.

Ilustración 17. Logo Android.

2.4.2 Arquitectura Android

El sistema operativo de Android está formado por una gran variedad de componentes. Estos componentes estan divididos en cinco capas relacionadas entre sí. El diagrama correspondiente a las capas es el siguiente:

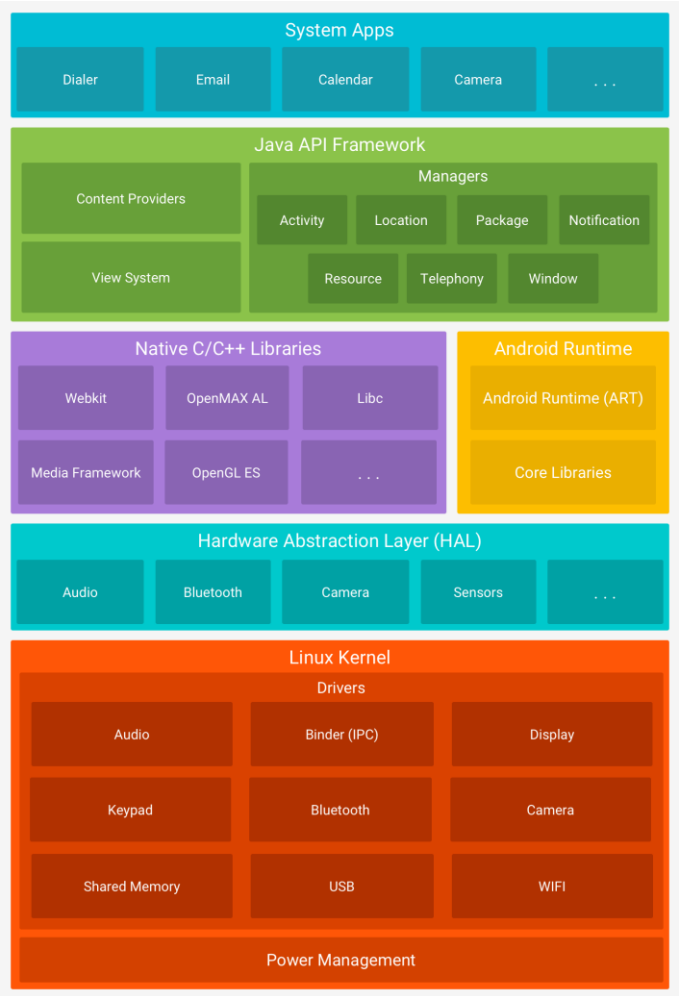


Ilustración 18. Capas de la arquitectura Android. (Fuente: developer.android.com)

Empezando desde la más baja, tenemos:

- Kernel de Linux: El núcleo del SO Android es el kernel de Linux. La elección se basa en la flexibilidad, seguridad y portabilidad que Linux presenta.
- Capa de abstracción de hardware(HAL): Esta capa es la que otorga la independencia del hardware. Esta es la capa que permite que Android se pueda ejecutar sin importar el hardware que se esté utilizando. Es una capa trivial de modificar para que los mismos fabricantes sean los encargados de hacerla funcional sobre su dispositivo.
- Bibliotecas nativas: Aquí se encuentran librerías totalmente disponibles para los desarrolladores como SQLite, Webkit, OpenGL...
- Framework para Apps: Esta es la capa más interesante para las personas que se dediquen a desarrollar aplicaciones para Android ya que es donde se encuentran almacenadas todas las librerías en Java con todos los métodos disponibles para programar nuestras aplicaciones.
- Capa de Aplicaciones: Esta capa se centra en la ejecución y estabilidad de las aplicaciones. Esta es la capa que todos utilizamos al hacer un uso a nivel de usuario del sistema operativo Android.

2.4.3 Evolución histórica

La beta de Android se lanzó en noviembre de 2007, y desde ese día hasta ahora ha estado recibiendo constantes actualizaciones. Estas actualizaciones se encargan de arreglar fallos y de agregar nuevas funcionalidades. Cuando los añadidos de una actualización son de relevancia, Android lleva a cabo un cambio de versión, con su correspondiente nombre. Hasta día de hoy las versiones que han ido apareciendo son las siguientes:

Code name	Version number	Initial release date	API level	Security patches[1]
(No codename)[2]	1.0	September 23, 2008	1	Unsupported
(Internally known as "Petit Four")[2]	1.1	February 9, 2009	2	Unsupported
Cupcake	1.5	April 27, 2009	3	Unsupported
Donut[3]	1.6	September 15, 2009	4	Unsupported
Eclair[4]	2.0 – 2.1	October 26, 2009	5 – 7	Unsupported
Froyo[5]	2.2 – 2.2.3	May 20, 2010	8	Unsupported
Gingerbread[6]	2.3 – 2.3.7	December 6, 2010	9 – 10	Unsupported
Honeycomb[7]	3.0 – 3.2.6	February 22, 2011	11 – 13	Unsupported
Ice Cream Sandwich[8]	4.0 – 4.0.4	October 18, 2011	14 – 15	Unsupported
Jelly Bean[9]	4.1 – 4.3.1	July 9, 2012	16 – 18	Unsupported
KitKat[10]	4.4 – 4.4.4	October 31, 2013	19 – 20	Unsupported[11]
Lollipop[12]	5.0 – 5.1.1	November 12, 2014	21 – 22	Unsupported[13]
Marshmallow[14]	6.0 – 6.0.1	October 5, 2015	23	Supported
Nougat[15]	7.0 – 7.1.2	August 22, 2016	24 – 25	Supported
Oreo[16]	**8.0 – 8.1**	August 21, 2017	26 – 27	Supported
Android P[17]	9	May 8, 2018 (beta)	28	Presupported
Legend: Old version	Older version, still supported	**Latest version**	Latest preview version	

Ilustración 19. Versiones de Android desde su lanzamiento. (Fuente: Wikipedia)

Para más información de los ítems importantes que se añaden en cada versión se puede visitar su página oficial. Según la versión en la que nos encontremos, podremos usar ciertas funcionalidades o no, por eso es importante a la hora de diseñar una aplicación para Android comprobar que se pueden cumplir las especificaciones usando una versión en concreto.

2.4.4 Programación en Android

No son pocas las personas que piensan que Android es un lenguaje de programación. Aprender Android hace referencia a todo lo relativo a este sistema operativo, y si queremos desarrollar alguna aplicación debemos usar algunos de los lenguajes que Android soporta. C/C++, C#, PhoneGap(HTML,CSS,Javascript), BASIC, Java, Kotlin, son varios de los lenguajes de programación en los que podemos desarrollar aplicaciones para este sistema operativo.

El lenguaje oficial dictado por Google es Java, y es el lenguaje que soporta su entorno de desarrollo oficial Android Studio. Toda su API se encuentra programada en Java, aunque se pueden conseguir en otros lenguajes de forma extraoficial y creadas por desarrolladores independientes. El uso de Android Studio no es obligatorio si queremos desarrollar aplicaciones en Android, pero sí es altamente recomendable y nos evitará muchos quebraderos de cabeza, ya que ha sido diseñado con el objetivo de reducir la dificultad de desarrollar aplicaciones en este sistema operativo.

Oracle, actual compañía dueña de Java, impuso una demanda en 2010 a Google por basar su sistema operativo en su lenguaje de programación y pidiendo una indemnización por violación de su propiedad intelectual. Al ser Android de código abierto, y tras comprobarse que no se violaban ninguna de las patentes de Oracle, Google salió ileso de la demanda.

Dos años después apareció el lenguaje de Kotlin, que, aunque pueda operar con código Java, no es compatible directamente con el mismo. La idea de sus creadores es que Kotlin vaya desplazando poco a poco a Java en la industria hasta que todo haya migrado a Kotlin. Desde que salió se comenzó a crear una API en Kotlin para poder usarse como lenguaje de programación en Android. En 2017 Google acepta a Kotlin como el "otro" lenguaje oficial para su sistema operativo, se piensa que todo viene a raíz de la demanda de Oracle de 2010.

2.4.5 Java vs Kotlin

En este apartado vamos a discutir qué ventajas tiene cada lenguaje con respecto al contrario y con esta información vamos a decidir qué lenguaje utilizar para el desarrollo del presente proyecto.

Kotlin

- Más claro y compacto, se puede hacer más trabajo en menos tiempo.
- Los errores se pueden detectar en tiempo de compilación y arreglarlos antes de ejecutar la aplicación y esperar a que falle.
- Es completamente interoperable con Java, puede convivir en una misma aplicación escrita en Java, lo que te puede permitir migrar tu programa poco a poco a Kotlin.
- Tiene un mantenimiento sencillo y puede usarse en diversos entornos de desarrollo, permitiendo a cada programador que utilice su preferido.

Java

- Es el lenguaje actualmente más utilizado en el mundo y lleva desde 2004 en el primer puesto, disputado de vez en cuando con el lenguaje de programacón C, podemos comprobarlo en el índice TIOBE. ¿Quiere decir que por ser el más utilizado sea mejor? Obviamente la respuesta es no, pero sí quiere decir que existen más profesionales que conocen en este lenguaje y esto lleva implícito ciertas ventajas.
- En relación directa con lo anterior, la comunidad de Java posee más herramientas y documentación. Es decir, las fuentes de calidad para aprender Kotlin escasean en comparación con Java. Esto ha sido exhaustivamente comprobado realizando búsquedas en internet y preguntando a profesionales de este campo.
- Hay que tener en cuenta que Kotlin no es Java, todavía existen problemas que Kotlin no es capaz de solucionar y sí Java.

May 2018	May 2017	Change	Programming Language	Ratings	Change
1	1		Java	16.380%	+1.74%
2	2		C	14.000%	+7.00%
3	3		C++	7.668%	+2.92%
4	4		Python	5.192%	+1.64%
5	5		C#	4.402%	+0.95%
6	6		Visual Basic .NET	4.124%	+0.73%
7	9	⌃	PHP	3.321%	+0.63%
8	7	⌄	JavaScript	2.923%	-0.15%
9	-	⌃⌃	SQL	1.987%	+1.99%
10	11	⌃	Ruby	1.182%	-1.25%

Ilustración 20. Diez primeras posiciones índice TIOBE.

Tras analizar ambos lenguajes, y teniendo claro que, aunque Kotlin es sencillo y potente y apunta a que en un futuro pueda llegar a destronar a Java, me he decidido por desarrollar la aplicación de Android utilizando Java ya que será más sencillo encontrar fuentes de aprendizaje, y encontrar soluciones a los problemas que vayan surgiendo a la hora del desarrollo de la aplicación. Además, otra razón por la que me he decantado por elegir Java es la cantidad de ofertas de empleo que requieren de este perfil de trabajador con lo que he considerado importante añadir la realización de un Trabajo Final de Grado utilizando este lenguaje de programación en el currículum.

3 DESCRIPCIÓN DEL SOFTWARE A UTILIZAR

En este capítulo voy a mencionar los programas y servicios que vamos a utilizar durante la puesta en marcha de la aplicación de Android. Se entiende que para conseguir correctamente el objetivo final de este trabajo se debe disponer de todo el software que aquí se va a mencionar. Esto no quiere decir que el objetivo se puede conseguir únicamente utilizando estos programas, si alguien se siente más cómodo con otro software y se ve capaz de adaptar lo aquí explicado a lo que conoce es libre de hacerlo, pero no aseguro que funcione.

3.1 Android Studio

Este software es el entorno de desarrollo integrado, IDE, de Google creado para el desarrollo de aplicaciones en Android.

Aunque se puede programar para Android utilizando otros entornos, hemos elegido este ya que es el oficial y viene con diferentes funcionalidades para hacer que sea trivial comenzar con el desarrollo de una aplicación. A la hora de programar detecta fallos al mismo tiempo que vamos escribiendo, y nos sugiere diversas opciones de autocompletado que nos permite ahorrar tiempo a la hora del desarrollo de la aplicación.

Además de estas funcionalidades también podemos compilar el programa, convertir ese programa en un archivo de extensión .apk para su posterior instalación en un dispositivo Android o incluso crear una máquina virtual con el SO de Android permitiéndonos testear nuestra aplicación directamente en el ordenador y ahorrar el tiempo de conversión a una .apk y tener que probrarlo en nuestro terminal. Esta máquina virtual podemos personalizarla, eligiendo la versión en la que se encuentra, tamaño y diversas opciones que nos ayudaran a crear una aplicación con el menor número de errores posible.

3.2 phpMyAdmin

PhpMyAdmin es una herramienta creada para la administración de bases de datos en MySQL, es gratuita y de código abierto. Ha sido programada en Php y es actualmente el software de gestión de bases de datos MySQL para servidores web más utilizado.

Algunas de las características más relevantes son:

- Interfaz web intuitiva.
- Gestión de BBDD en MySQL.
    - Navegar e introducir datos en las BBDD.
    - Crear, copiar, modificar BBDD.
    - Mantenimiento de la BBDD.
    - Ejecutar y editar sentencias SQL.
- Importar datos desde CSV y SQL.
- Hace que consultas complejas en SQL sean más fáciles.
- Exportar los datos obtenidos en una consulta a diversos formatos.

Hemos elegido esta herramienta por contar con una gran comunidad que lo respalda permitiéndonos encontrar abundante documentación.

3.3 Servicio de hosting con BBDD – 000webhost

Como servicio de hosting de pruebas, para poder tener una base de datos en un servidor externo y no tener que mantener encendido un equipo en nuestra casa para poder consultar la información de la bbdd utilizaremos 000webhost.

Esta compañía ofrece un servicio de hosting gratuito a cambio de utilizar en el dominio de tu servidor web su nombre, de forma que les estás promocionando. Además, al ser gratuito, nuestros servicios se pausan durante una hora todos los días, con lo cual está pensado para servir para testeos a no ser que actualices a la cuenta de pago. En este caso lo utilizaremos para explicar la puesta en marcha de este proyecto.

Si alguien está interesado en mantener la base de datos en un equipo propio, puede utilizar para ello el software *Xampp Control Panel*, aunque en esta documentación no se explicarán los pasos necesarios para poner en marcha la aplicación haciendo uso de éste, aunque resulta sencillo si se ha entendido todo el procedimiento. Cabe indicar que uno de los motivos de porqué no se ha utilizado este software es que para poder acceder a los datos desde una red externa debemos abrir ciertos puertos del router de manera que provocamos que nuestra red sea vulnerable a posibles ataques, con lo cual hemos querido evitar realizar cualquier acción que pueda exponer nuestra seguridad.

3.4 Filezilla

Este software lo utilizamos para conectarnos mediante FTP (File Transfer Protocol) al administrador de archivos del hosting gratuito que estamos utilizando y poder subir ciertos directorios y archivos al hosting para llevar a cabo las consultas en php que necesitemos desde nuestra aplicación Android.

4 PUESTA EN MARCHA DE LA APLICACIÓN

Legamos al capítulo central de este proyecto donde vamos a explicar lo más detalladamente posible todos los pasos que hemos tenido que tomar para llegar a tener una aplicación creada por nosotros en nuestro teléfono móvil que haga uso de Google Maps y posicionamiento GPS.

Vamos a desarrollar una aplicación para Android que será capaz de obtener información de coordenadas GPS que se encuentran en una BBDD externa que gestionamos con phpMyAdmin. Esta base de datos estará formada por coordenadas GPS, siendo uno de los atributos la ruta a la que pertenece. Esto quiere decir que habrá, por ejemplo, seis registros en la BBDD con un mismo valor en el atributo "Ruta", si suponemos que ese valor es el dos esto quiere decir que esas seis coordenadas corresponden a la ruta número dos. Si mostramos la ruta número dos en el mapa se dibujarán los marcadores de cada una de las coordenadas GPS y una línea que conecta cada coordenada consecutiva.

La aplicación consistirá en un mapa y varios botones y textos para interactuar con el mismo. Un botón nos dirá cuántas rutas se encuentran almacenadas en la base de datos, ese número será el máximo valor que podremos introducir en el cuadro de texto donde debemos indicar qué ruta mostrar en el mapa, empezando desde el uno.

4.1 Configuración y códigos iniciales: Android Studio

Para comenzar con el desarrollo del proyecto vamos a la dirección web *https://developer.android.com/studio/* y descargamos el software Android Studio. Una vez lo tengamos, lo instalamos y ejecutamos. Iniciamos el programa y pulsaremos en *New Project*, a continuación, solo modificamos el nombre de nuestra aplicación e indicamos la ruta en nuestro disco duro donde queramos guardarlo. Pulsamos *Next*.

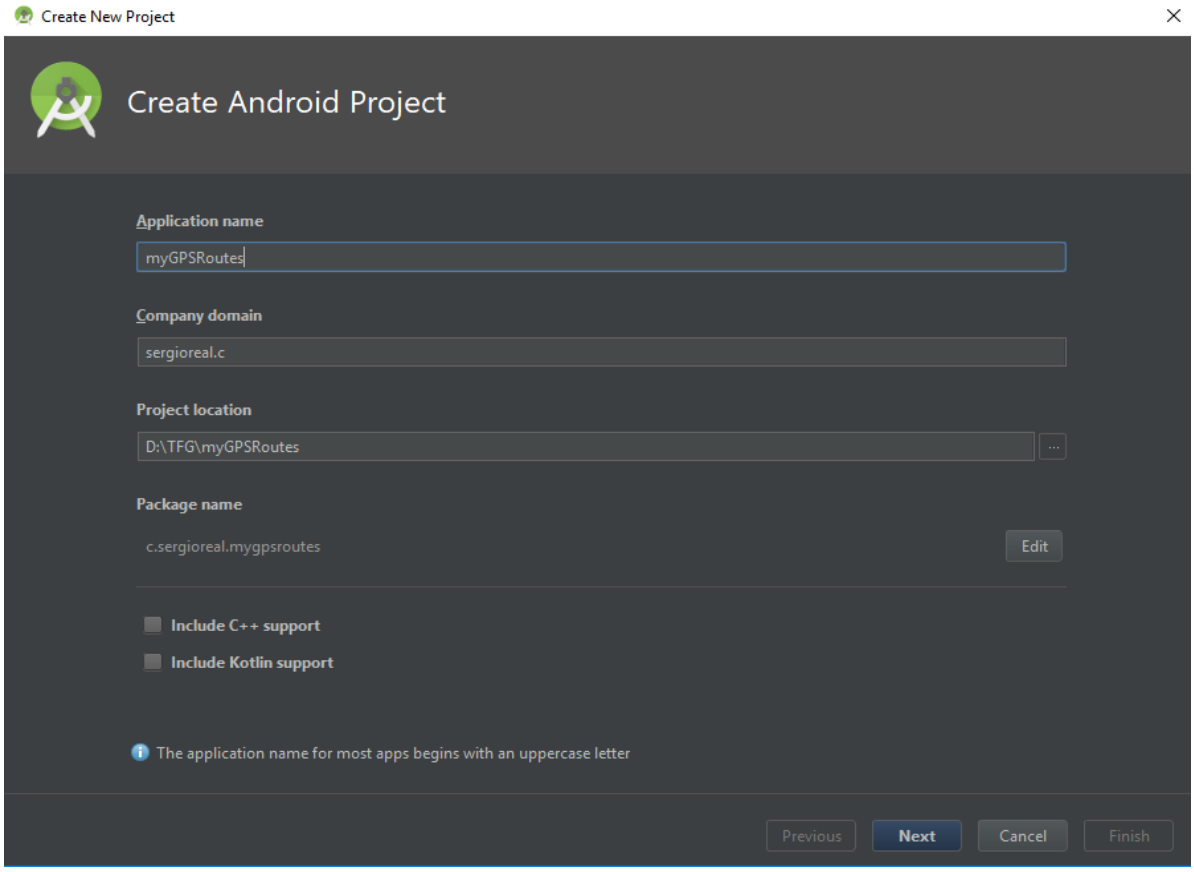


Ilustración 21. Creando nuevo proyecto en Android Studio.

En esta parte debemos indicar para qué dispositivo queremos desarrollar la aplicación, en nuestro caso *Phone and Tablet* y seleccionamos el nivel de API que queremos utilizar para crearla. A más nivel más funcionalidades tenemos, ya que cada nivel de API corresponde a actualizaciones y versiones más nuevas de Android.

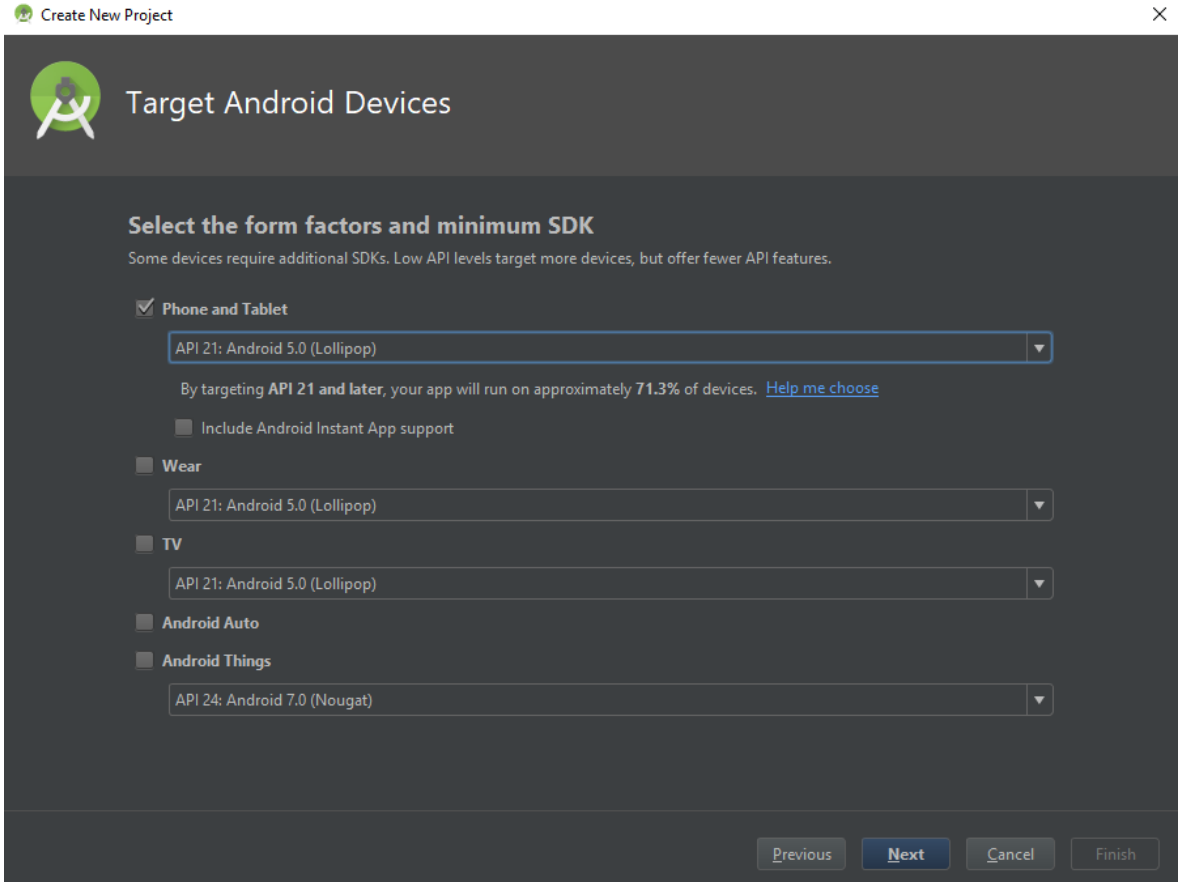


Ilustración 22. Selección de API Android Studio.

Puede pensarse que cuanto más alta sea la API mejor será ya que contamos con más métodos y funciones que utilizar a la hora de programar y hay que tener cuidado, ya que esto es un arma de doble filo. Es cierto que cuanto mayor sea el nivel de API más funcionalidades tenemos, pero también hay menos terminales en el mercado que pueden soportar la aplicación.

Es decir, a más baja sea la API, mayor cantidad de personas serán capaces de usarla. Para tener constancia de qué porcentaje de personas podrían usarla, Android Studio nos proporciona con datos actualizados pulsado el link *Help me choose.*

Como vemos en la Ilustración 8, para la API 21 que tenemos seleccionada podremos cubrir el 71.3% de los dispositivos en activo, con lo cual vamos a desarrollarlo para esta versión. Esto quiere decir que desde esa versión a las menores (4.0-4.4) el funcionamiento de la aplicación que vamos a crear no está garantizado.

Pulsamos siguiente y seleccionamos *Google Maps Activity*, en la siguiente ventana podemos cambiar el título que aparecerá en el mapa si queremos y presionamos *Finalizar*. Debemos esperar a que se configure todo y aparezca la pantalla para comenzar a programar.

El hecho de utilizar *Google Maps Activity* nos ahorra tener que añadir a la lista de permisos el del uso de localización y nos ahorra tener que incluir ciertas librerías que vamos a utilizar al usar métodos de la API de Google Maps.

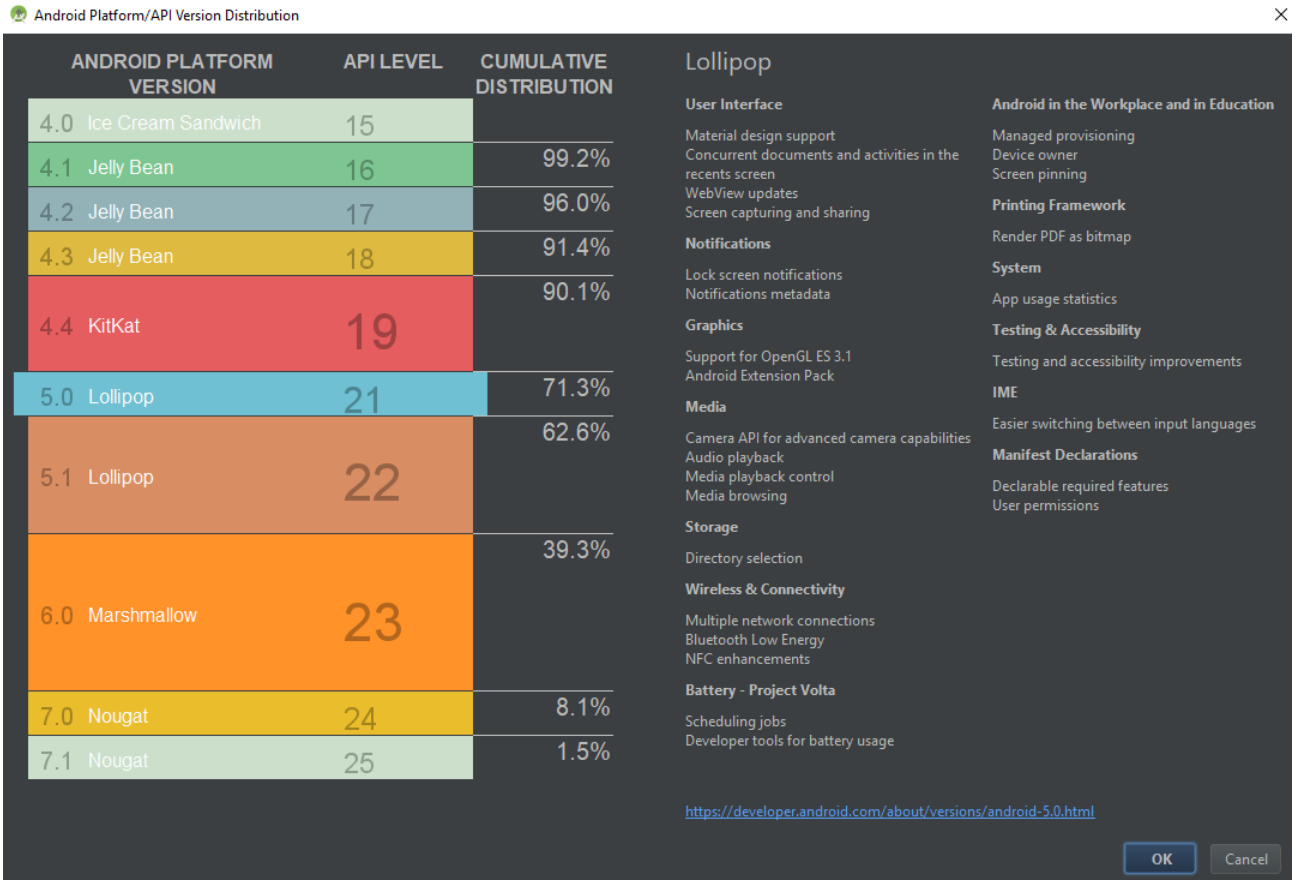


Ilustración 23. Porcentaje de terminales Android usando cierta versión.

Una vez está todo listo para programar, echando un primer vistazo al entorno de desarrollo Android vemos que se trata de un entorno de desarrollo típico, muchos botones, muchos menús, muchas opciones de configuración, etc… En este tutorial no se trata de pararnos a explicar qué hacen todos y cada uno de los botones, nos dedicaremos a mencionar cada ítem que vayamos a usar, pero quedarán infinidad de cosas sin tocar que se dejan al usuario que esté interesado en aprenderlo.

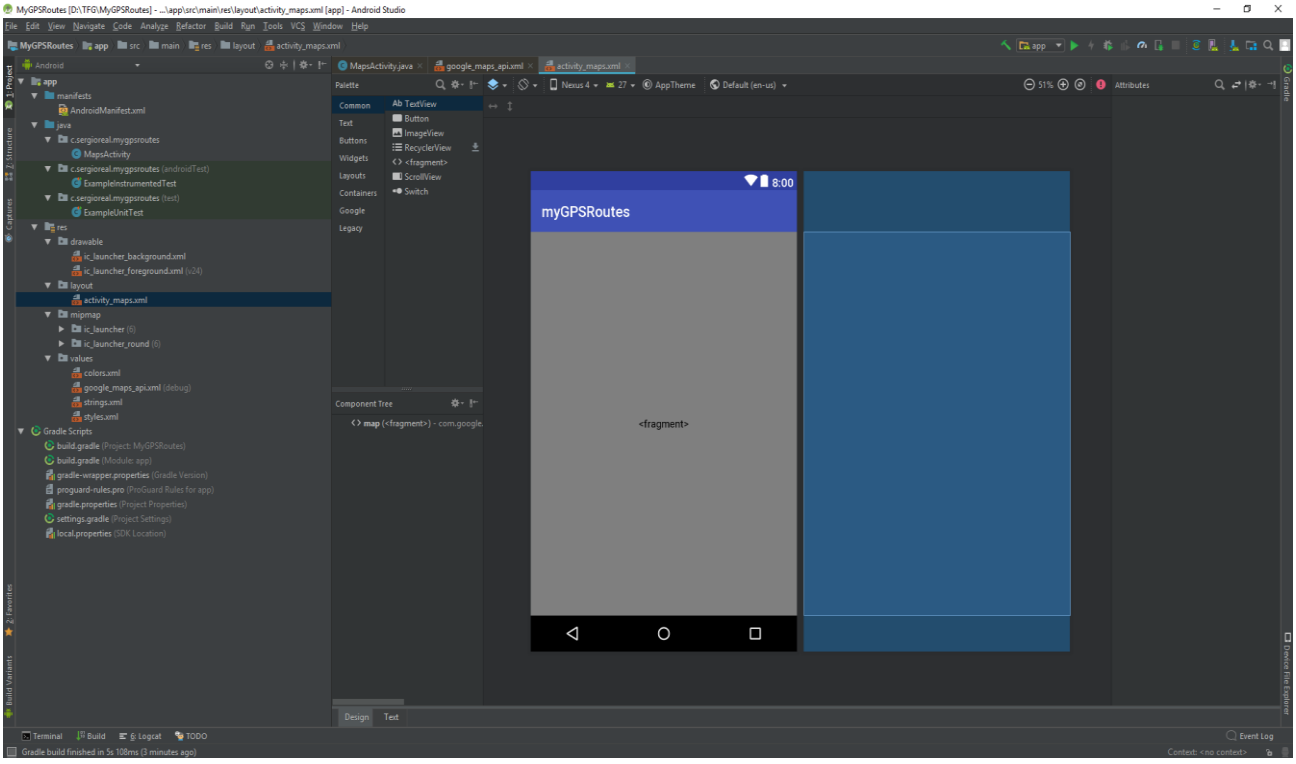


Ilustración 24. Entorno de desarrollo de Android Studio.

Para abrir la vista que tenemos en la Ilustración 9 buscamos en el navegador de los directorios de nuestra aplicación, la columna de la izquierda, la carpeta *res/layout/activity_maps.xml*.

El primer paso que se recomienda a la hora de desarrollar una aplicación es realizar la interfaz que creamos que debe tener, para luego programar la acción que realizará cada botón o cuadro de texto que hayamos introducido, o dibujar en un mapa como haremos en nuestro caso.

Para esto, una vez hayamos abierto archivo indicado en la Ilustración 9, abajo a la izquierda veremos dos pestañas, una que sería para desarrollar la interfaz en vista diseño, que sería la gráfica, y otra pestaña, en modo texto, para crear la interfaz usando lenguaje de marcado XML.

Estando en modo texto debemos pegar el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/relativeLayout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MapsActivity"
    tools:layout_editor_absoluteY="81dp">


    <fragment xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:map="http://schemas.android.com/apk/res-auto"
        xmlns:tools="http://schemas.android.com/tools"
        android:id="@+id/map"
        android:name="com.google.android.gms.maps.SupportMapFragment"
        android:layout_width="match_parent"
        android:layout_height="575dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentStart="true"
        tools:context=".MapsActivity" />

    <EditText
        android:id="@+id/seleccionRuta"
        android:layout_width="69dp"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/NombreRuta"
        android:layout_marginStart="20dp"
        android:ems="10"
        android:inputType="textPersonName" />

    <TextView
        android:id="@+id/NombreRuta"
        android:layout_width="105dp"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true"
        android:layout_marginStart="20dp"
        android:layout_marginTop="11dp" />

    <Button
        android:id="@+id/mostrarRuta"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/seleccionRuta"
        android:layout_toEndOf="@+id/NombreRuta"
        android:text="Mostrar"
        tools:text="Mostrar" />
```

```
    <TextView
        android:id="@+id/nRutasTotales"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBaseline="@+id/seleccionRuta"
        android:layout_alignParentEnd="true"
        android:layout_marginEnd="31dp"
        android:text="0" />

    <Button
        android:id="@+id/contarRutas"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignTop="@+id/seleccionRuta"
        android:layout_marginEnd="59dp"
        android:text="Rutas" />

</RelativeLayout>
```

Una vez llegados a este punto, si cambiamos a pestaña de diseño debemos ver lo siguiente:

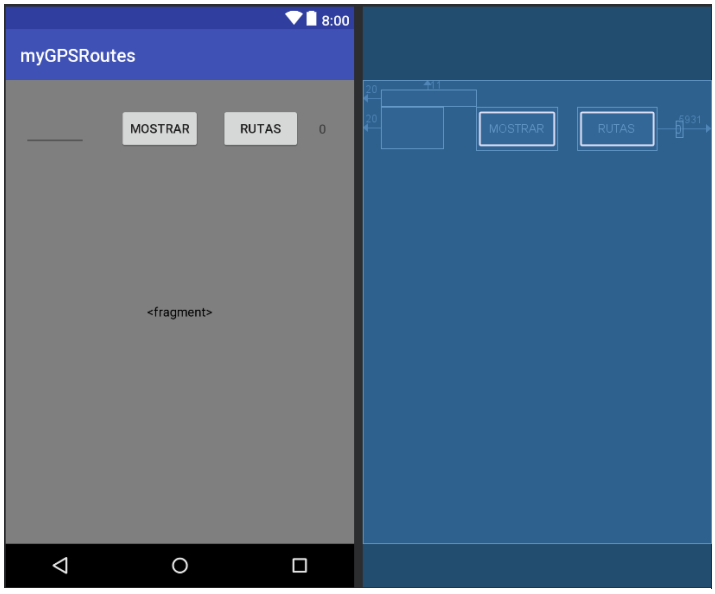


Ilustración 25. Vista en modo diseño tras insertar el XML.

Tras esto, y comprobar que todo va bien, vamos al panel de directorios de la izquierda o en una de las pestañas de arriba y seleccionamos y abrimos el archivo *MapsActivity.java* para editarlo. Una vez abierto copiamos y pegamos el siguiente código en el mismo.

```
/**
 * Trabajo Fin de Grado: IOT – Posicionamiento en mapas de Google empleando bases de datos y coordenadas GPS
 * Grado en Ingeniería Electrónica, Robótica y Mecatrónica con Intensificación en Robótica y Automatización
 *
 * @author Sergio Real Aragón
 * Tutor: Alfredo Pérez Vega-Leal
 * @date Junio 2018
 *
 */

/** Esta línea debemos modificarla según nuestro paquete, lo mejor es dejar la que viene
 * por defecto al iniciar la aplicación y copiar lo demas */
package c.sergioreal.mygpsroutes;
```

```
/** Importación de todas las librerías que usamos durante el proyecto. **/
import android.support.v4.app.FragmentActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.OnMapReadyCallback;
import com.google.android.gms.maps.SupportMapFragment;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.MarkerOptions;
import com.google.android.gms.maps.model.Polyline;
import com.google.android.gms.maps.model.PolylineOptions;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;


public class MapsActivity extends FragmentActivity implements OnMapReadyCallback {

    /** Instancia de la clase GoogleMap -> mMap*/
    private GoogleMap mMap;

    /** Nº máximo de rutas y coordenadas que la aplicación puede almacenar mientras se ejecuta. Puede modificarse según las
necesidades*/
    private static final int N_MAX_RUTAS = 20;
    private static final int N_MAX_COORD = 100;

    /** Declaración de las variables para identificar los diferentes elementos de la interfaz de usuario */
    Button   botonMostrar,botonCuentaRuta; //Botones.
    TextView textoNombreRuta,nTotalRutas;  //Campos de texto.
    EditText nRutaSeleccionada;          //Cuadro de texto para introducir la búsqueda.

    /** Constante de la dirección donde accederemos al archivo .php que realiza la conexión y consulta deseada a la BBDD*/
    /** Es importante cambiar la siguiente línea*/
    private static final String QUERY_URL = "http://www.WEBDONDETENEMOSLABBDD/googleMaps/coordenadas.php";

    /** Matrices donde instroduciremos la información recogida de la BBDD
     * Ejemplo: arrayRutasNombre[2][i]
     * Obtenemos el nombre de la coordenada nº i en la ruta 2.
     * */
    double[][] arrayRutasLat; // Latitud de la coordenada
    double[][] arrayRutasLon; // Longitud de la coordenada
    String[][] arrayRutasNombre; // Nombre de la coordenada

    /** Variables que usamos para ir mostrando una por una las coordenadas obtenidas de la BBDD*/
    double lat,lon;
    String nombre;

    /** Nos indica el número de rutas totales que se encuentran en ese momento en la BBDD*/
    String numRutasTotales;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {

    /** Definición del tamaño de las matrices donde tendremos los datos almacenados mientras se ejecuta la aplicación.
     *
     * Con lo definido actualmente podemos tener hasta 20 rutas diferentes y un máximo de 100 coordenadas GPS por ruta.
     *
     * */
    arrayRutasLat = new double[N_MAX_RUTAS][N_MAX_COORD] ;
    arrayRutasLon = new double[N_MAX_RUTAS][N_MAX_COORD];
    arrayRutasNombre = new String[N_MAX_RUTAS][N_MAX_COORD];

    /** Realiza la carga de los fragmentos que haya en la interfaz gráfica.*/
    super.onCreate(savedInstanceState);
    /** Establece como contenido de la actividad el de un layout, en este caso activity_maps. */
    setContentView(R.layout.activity_maps);
    /** Obtiene el fragmento del mapa y es notificado cuando el mapa está listo para usarse*/
    SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
            .findFragmentById(R.id.map);

    mapFragment.getMapAsync(this);

}
/**
 * Una vez el mapa está listo, desde aquí podemos manipularlo.
 * Desde aquí podemos añadir marcadores o líneas, mover la cámara o añadir elementos que se
 * encuentren a la escucha(botones, cuadros de texto...) para realizar alguna modificación.
 * Para que funcione correctamente el usuario deberá tener instalado en su terminal los
 * los servicios de Google Play.
 */
@Override
public void onMapReady(GoogleMap googleMap) {
    /** Objeto que identifica al mapa activo que estamos usando */
    mMap = googleMap;

    /** Muestra en el mapa la ruta seleccionada */
    botonMostrar = (Button)findViewById(R.id.mostrarRuta);
    /** Una vez seleccionada la ruta, muestra el nombre de la misma encima del cuadro de texto donde hemos seleccionado la
misma*/
    textoNombreRuta = (TextView) findViewById(R.id.NombreRuta);
    /** Cuadro de texto donde introducimos el nº de identificación de la ruta que queremos mostrar en el mapa*/
    nRutaSeleccionada = (EditText) findViewById(R.id.seleccionRuta);


    /** Carga la información obtenida de la BBDD en las variables destinadas a ello(Arrays de lat, lon y nombre)*/ y cuenta el
número de rutas.
    botonCuentaRuta = (Button)findViewById(R.id.contarRutas);
    /** Indica el nº total de rutas que hay en este momento en la BBDD
     *  Ejemplo: Si se muestra un 5, se puede introducir en el cuadro de texto un valor del 1 al 5, según la ruta que queramos
mostrar.
     * */
    nTotalRutas = (TextView) findViewById(R.id.nRutasTotales);

    /** Acciones a realizar si se pulsa el botón mostrar de la interfaz*/
    botonMostrar.setOnClickListener( new View.OnClickListener() {
        @Override
        public void onClick(View v) {

            /** Variable que indica la ruta que ha sido seleccionada en el cuadro de texto*/
            int idRuta;

            /** Se obtiene la información del cuadro de texto y se convierte a entero para usarlo como índice en los arrays */
            String selection;
```

```
            try {
                selection = nRutaSeleccionada.getText().toString();//Obtenemos el texto del cuadro de texto
                idRuta = Integer.parseInt(selection) ; //Convertimos a entero

            }catch (Exception e)//Para que la aplicación no se cierre al detectar un valor no válido en el cuadro de texto
            {
                Toast.makeText(getApplicationContext(),"Introduzca un valor entre 1 y " + numRutasTotales,Toast.LENGTH_LONG
).show();
                selection = "0";
                idRuta = Integer.parseInt(selection); //Convertimos a entero
            }

            /** Borramos los elementos que estén actualmente en el mapa */
            mMap.clear();

            if(idRuta>=1 && idRuta<=Integer.parseInt(numRutasTotales))
            {
                /** Declaración e instancia del objeto donde definimos las opciones de las líneas que vamos a utilizar para unir las
coordenadas obtenidas de la BBDD*/
                PolylineOptions rectOptions = new PolylineOptions();
                /** Mostramos encima del cuadro de texto de selección el nombre de la ruta que ha sido seleccionada*/
                idRuta = idRuta -1; //Para que concuerde con el índice de los arrays
                textoNombreRuta.setText(arrayRutasNombre[idRuta][0]);

                /** En este bucle vamos a ir mostrando todas las coordenadas de una misma ruta(la seleccionada).
                 * Además se establecen en las opciones de polilínea los puntos iniciales y finales donde debemos dibujar una línea,
                 * de forma que al terminar el bucle for podamos llamar al método que dibuja la linea.
                 * */
                for(int i=0; i < arrayRutasLat[idRuta].length ;i++){

                    /** Obtención de una coordenada en concreto*/
                    lat = arrayRutasLat[idRuta][i];
                    lon = arrayRutasLon[idRuta][i];
                    nombre = arrayRutasNombre[idRuta][i];

                    /** Como condición de terminación de ruta usamos que tanto latitud como lontitud sean cero.
                     * Aunque son unos valores totalmente posibles, tomamos como cierto que  trabajando bajo nuestras condiciones
                     * (El vehículo no va a pasar por medio del Océano Atlántico), y así poder establecer una condición de salida
                     * de las matrices donde almacenamos la información de las rutas.
                     * */
                    if(lat==0 && lon ==0) break;

                    /** Se crea instancia de clase LatLng y asignamos a ese objeto los valores de latitud y longitud
                     * Tras esto se añade el marcador con título al mapa, se mueve la cámara a ese lugar y se realiza
                     * un zoom.
                     * */
                    LatLng lugar = new LatLng(lat, lon);
                    mMap.addMarker(new MarkerOptions().position(lugar).title( i + " " + nombre));
                    mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(lugar,15));

                    /** Se añade una nueva coordenada a las opciones de polilínea para luego mostrarla al completo*/
                    rectOptions.add(new LatLng(arrayRutasLat[idRuta][i], arrayRutasLon[idRuta][i]));

                }

                /** Dibuja la línea que va desde la coordenada inicial a la final*/
                Polyline polyline = mMap.addPolyline(rectOptions);
            }    /** Indicamos que no se ha introducido valor correctamente*/
            else  Toast.makeText(getApplicationContext(),"Introduzca un valor entre 1 y " +
numRutasTotales,Toast.LENGTH_LONG ).show();

        }
    });
```

```
    /** La acción de este botón es la de cargar la información que esté almacenada en la BBDD a la aplicación.
     * Si durante el uso de la aplicación se añade datos a la BBDD, pulsando a este botón podemos actualizar la información
     * y mostrar las nuevas rutas añadidas.
     * */
    botonCuentaRuta.setOnClickListener( new View.OnClickListener() {
        @Override
        public void onClick(View v) { //Dentro de éste método OnClick es donde tenemos que escribir el código necesario para que se escriba la información en la BBDD

            /** Función que actualiza la información de los arrays con los datos obtenidos de la BBDD*/
            loadCoordenadas();
            /** Cambiamos el nº de rutas existentes para conocer el valor límite que podemos introducir en el cuadro de texto de mostrar ruta. */
            nTotalRutas.setText(numRutasTotales);

        }
    });

  }

  /** Este método carga la información de la BBDD en los arrays destinados a ello.
   */
  private void loadCoordenadas()
  {
    StringRequest stringRequest = new StringRequest(//4 Parámetros
          Request.Method.GET, //Método tipo GET, ya que queremos obtener la información
          QUERY_URL, //URL donde realizamos la petición, aquí se encontrará el texto en JSON con la información de la BBDD
          new Response.Listener<String>() { //Respuesta obtenida de la petición
            @Override
            public void onResponse(String response) {
              try{

                /** Instancia de un objeto JSONArray del cual descompondremos la información recibida.*/
                JSONArray coords = new JSONArray(response);

                int nRutas = 1; //Variable para detectar cambio de ruta en la información de la BBDD y que se muestra en la interfaz como nº de rutas disponibles.
                int index = 0; //Para colocar el indice correcto en las rutas

                /** Bucle que extrae la informacíon en formato JSON y la almacena en los arrays*/
                for(int i=0; i < coords.length();i++){

                  JSONObject coordObject = coords.getJSONObject(i);

                  int id = coordObject.getInt("id");
                  int ruta = coordObject.getInt("ruta");
                  String nombre = coordObject.getString("nombre");
                  double lat = coordObject.getDouble("lat");
                  double lon = coordObject.getDouble("lon");

                  /** Detección nueva ruta y cambio variable idRuta*/
                  if (nRutas != ruta) {
                    nRutas=ruta;
                    index=0;
                  }

                  Coord coordenada = new Coord(id,ruta,nombre,lat,lon);

                  arrayRutasLat[ruta-1][index] = coordenada.getLatitud();
                  arrayRutasLon[ruta-1][index] = coordenada.getLongitud();
```

```
                    arrayRutasNombre[ruta-1][index] = coordenada.getNombre();

                    /** Depuración: Descomentar si quieres comprobar por consola de Android Studio que las coordenadas han
sido cargadas correctamente.*/
                    //System.out.println(ruta-1 + " Lat = "+ arrayRutasLat[ruta-1][index] + "Lon = " + arrayRutasLon[ruta-
1][index]);

                    index++;
                  }
                  numRutasTotales = Integer.toString(nRutas);//Pasar a entero el número de rutas

                  /** Si ha habido un error en la lectura de la información en JSON lo capturamos aquí y actuamos.*/
                }catch (JSONException e){
                  e.printStackTrace();
                }

              }
            },
            /** Si ha habido un error en la adquisición de los datos avisamos por pantalla con un toast*/
            new Response.ErrorListener() {
              @Override
              public void onErrorResponse(VolleyError error) {
                Toast.makeText(MapsActivity.this,error.getMessage(),Toast.LENGTH_SHORT).show();
              }
            });

      Volley.newRequestQueue(this).add(stringRequest);
    }

}
/** Final de código MapsActivity.java*/
```

Ahora aparecerán una gran cantidad de líneas marcadas en rojo informándonos de la existencia de errores. No te preocupes, se deben a que aún no hemos creado la clase *Coord* y que tenemos que introducir ciertas líneas al *Gradle*, que es el sistema encargado de unir y compilar todas las partes que hemos programado para convertir nuestro trabajo en una aplicación.

Vamos a la izquierda y buscamos *Gradle Scripts* y abrimos *build.gradle(Project:MyGPSRoutes)* y *build.gradle(Module:app)* .

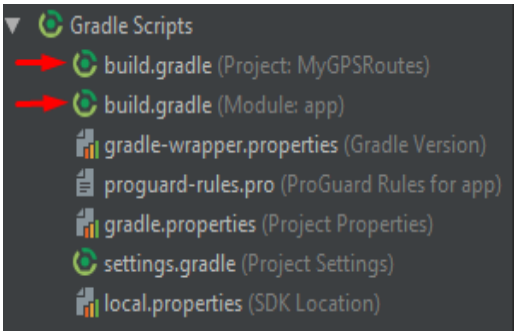


Ilustración 26. Scripts de Gradle a modificar.

En el primer archivo vamos donde dice *Allprojects{repositories{* y bajo *jcenter()* añadimos la línea *mavenCentral()* , en el segundo vamos a *dependencies{* y abajo del todo añadimos la línea *implementation 'com.android.volley:volley:1.1.0'* . Una vez hecho estos dos pasos Android Studio nos informará de que los archivos de Gradle han cambiado y es necesario una sincronización, pulse *Sincronizar ahora*. Gradle realizará ciertas funciones y verás el mensaje de que ha sido completado con éxito si no ha surgido ningún error.

Para seguir arreglando errores vamos a proceder con la creación de la clase *Coord,* que es la clase que luego instanciamos para crear objetos "coordenadas". Para ello situamos el ratón en la carpeta donde se encuentra almacenado el archivo MapsActivity, como vemos en la Ilustración 11, y pulsamos el botón derecho y creamos una nueva clase de Java denominada *Coord*.

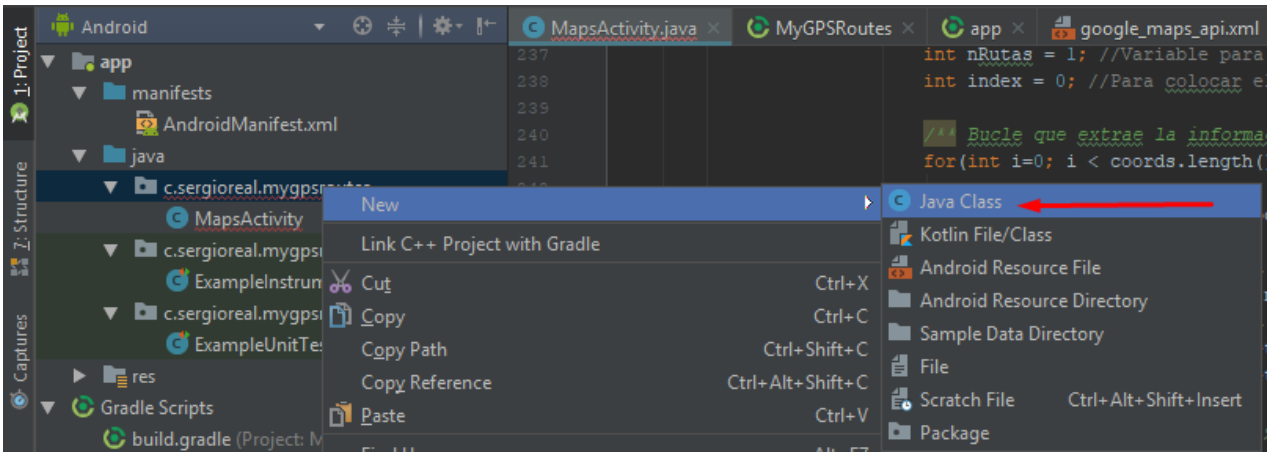


Ilustración 27. Añadiendo nueva clase al proyecto.

Una vez tengamos la clase creada, copiamos el siguiente código y lo pegamos en ella.

```
/** Definición de la clase Coord: se recuerda que el package debe ser el que cada uno tenga en su terminal y no este que se indica.*/
package c.sergioreal.mygpsroutes;

public class Coord {

   /** Atributos de la coordenada*/
   private int id;
   private int ruta;
   private String nombre;
   private double latitud;
   private double longitud;

   /** Constructor */
   public Coord(int id,int ruta, String nombre, double latitud, double longitud) {
      this.id = id;
      this.ruta = ruta;
      this.nombre = nombre;
      this.latitud = latitud;
      this.longitud = longitud;
   }

   /** Getters
    * Métodos para acceder a la información de los objetos de esta clase
    * */
   public int getId() {
      return id;
   }

   public int getRuta() {
      return ruta;
   }

   public String getNombre() {
      return nombre;
   }

   public double getLatitud() {
      return latitud;
```

```
    }

    public double getLongitud() {
        return longitud;
    }

}
```

Por último, aunque no sea un error como tal, tenemos que añadir a la App los permisos de uso de internet, ya que nuestra idea es que podamos consultar la BBDD desde cualquier lugar desde nuestro dispositivo Android, y para permitir que la App haga uso de los datos móviles debemos añadir en el archivo *AndroidManifest.xml* antes de la etiqueta <*application*/> la siguiente línea:

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

En este punto no deben señarlarse más errores, y para asegurarnos de que todo está bien procedemos a compilar el proyecto utilizando la combinación de teclas *CTRL + F9*, abajo en la pestaña *Build* debe indicarse que se ha completado satisfactoriamente.

4.2 Creando dispositivo virtual y probando App: Android Studio

Como ya hemos compilado nuestra App y no hemos tenido errores, el próximo paso es probarla. Para ello vamos a utilizar una de las herramientas más interesantes que nos ofrece Android Studio, la denominada AVD (Android Virtual Device). Una AVD es básicamente una máquina virtual que ejecuta un sistema Android completamente funcional y donde podemos testear nuestras creaciones.

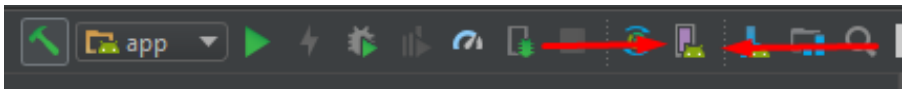


Ilustración 28. Android Virtual Device Manager.

Pulsamos el botón de la barra de herramientas de arriba a la derecha que se muestra en la Ilustración 13 y abriremos el menú de configuración de un dispositivo virtual. Abajo pulsamos en *Create a Virtual Device*, elegimos *Phone* con el modelo que queramos y pulsamos siguiente. En la siguiente ventana nos pedirá que elijamos la imagen a instalar en la máquina virtual, vamos a elegir *Android Nougat 7.0 API Level 24* , que será capaz de ejecutar aplicaciones que hayan sido diseñadas como máximo hasta la API 24, y recordamos que la nuestra es la API 21 con lo cual no tendremos problemas, pulsamos *Next*. En la siguiente ventana pulsamos *Finish* sin modificar nada.

Si todo ha ido bien debemos ver en el mánager de dispositivos virtuales lo siguiente:

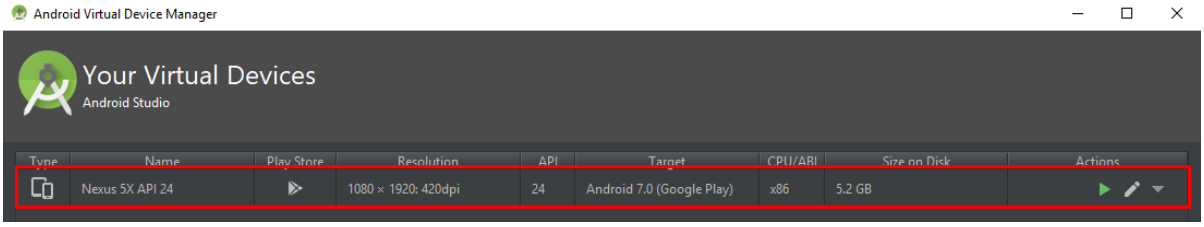


Ilustración 29. Dispositivo Virtual correctamente creado.

Pulsamos el símbolo verde del *play* que vemos en la imagen anterior para iniciar la máquina virtual y esperamos, depende del equipo desde el que lo ejecutemos tardará más o menos ya que es un proceso que consume bastantes recursos. Una vez se inicie debemos pasar por el proceso típico de primer encendido de un terminal Android con el que todos estamos familiarizados y terminando con tener en nuestra pantalla del ordenador una máquina virtual con un dispositivo Android completamente funcional.

Llegados a este punto podemos proceder a ejecutar nuestra App, para ello pulsamos el botón verde indicado en la siguiente figura y luego seleccionamos el AVD que tenemos abierto para que ejecute ahí.

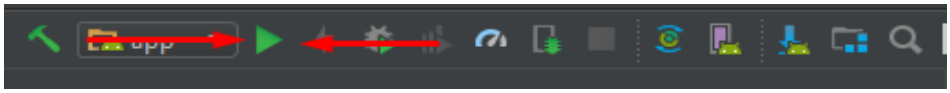


Ilustración 30. Ejecutar aplicación

Se abrirá nuestra aplicación en el dispositivo virtual por primera vez, podremos ver los botones y cuadros de texto que hemos diseñado, pero ¿y el mapa?

Verás que, en lugar del mapa, se verá un fondo gris con el logo de Google abajo a la izquierda, pero no te preocupes, es normal. Si nos paramos a leer el archivo *google_maps_api.xml* nos avisan que para que nuestra que hace uso de un mapa de Google funcione correctamente, debemos hacer uso de una *API Key*, esto es básicamente un código que debemos introducir que está adjunto a nuestra cuenta de Google, con lo que es necesario tener una cuenta creada con ellos.

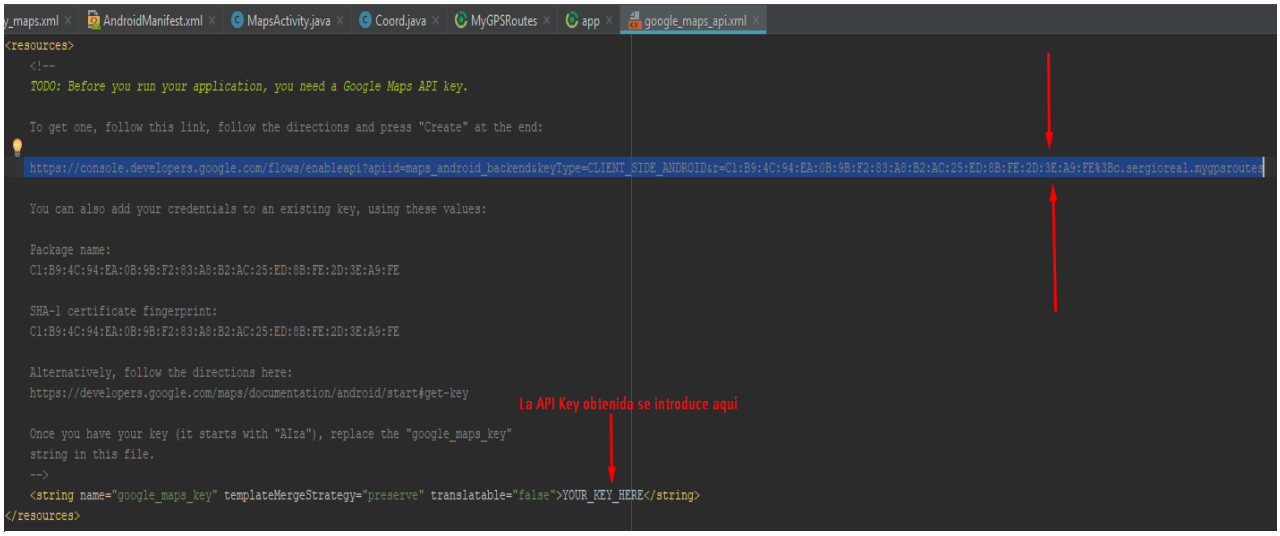


Ilustración 31. Obtención API Key de Google.

Debemos introducir el link que hay marcado en la Ilustración 16 en un navegador web y seguir los pasos que te indiquen hasta obtener la API Key necesaria para ejecutar nuestra App. Una vez obtenida se introduce donde se te indica en el código y tras ejecutar de nuevo la aplicación el mapa debe mostrarse sin ningún problema.

Ya tenemos una aplicación que se inicia correctamente y nos muestra una interfaz. Si pulsásemos ahora el botón *RUTAS* la App no hará nada o se cerrará debido a un error, ya que aún no se ha indicado de dónde obtener los datos que queremos mostrar y para eso debemos crear una base de datos en un servidor que podamos consultar en todo momento y desde cualquier lugar.

4.3 Hosting y BBDD: 000webhost y phpMyAdmin

En este apartado vamos a explicar, para el que no esté familiarizado con esto, la obtención de un servicio de hosting y cómo crear una BBDD en el mismo. La razón de usar un hosting para tener ahí nuestra BBDD ya se ha explicado al principio de este documento.

Hemos elegido como servicio de hosting el perteneciente a la compañía 000webhost que nos ofrece un servicio gratuito bajo ciertas condiciones, pero nos resulta muy adecuado para el testeo de la App. Accedemos a su página web y nos registramos para poder usar sus servicios. También debemos dar un nombre a nuestra "web", que será la dirección donde más adelante accederemos para obtener la información de la BBDD.

Al iniciar sesión con nuestra cuenta veremos arriba un panel de control muy general que nos informa de las diferentes operaciones que podemos llevar a cabo. Pulsaremos en *Manage database*.

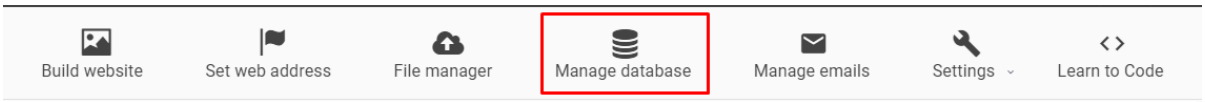


Ilustración 32. Panel de control 000webhost.

Ahí pulsaremos en *New database* y tendremos que rellenar los datos que nos indican. El nombre que queramos y un usuario con contraseña, que será el que más adelante utilizaremos para conectarnos a la BBDD mediante php y obtener la información con lo que es bastante importante que apuntes los datos que vas a introducir aquí. Creamos la base de datos y esperamos a que termine el proceso.

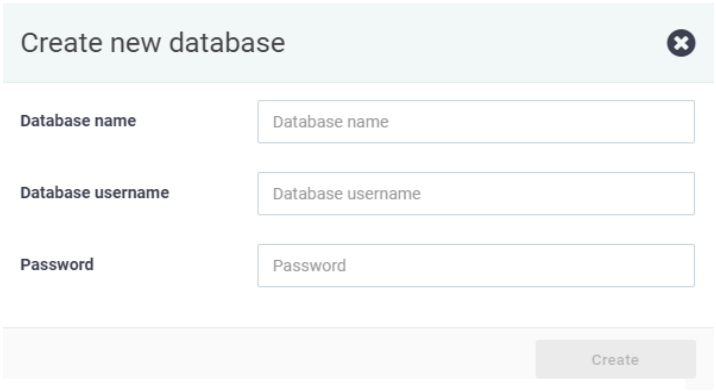


Ilustración 33. Creación BBDD en 000webhost.

Una vez creada podremos seleccionar del menú que aparece gestionar la BBDD utilizando PhpMyAdmin. Pulsamos ahí y nos redireccionará a una interfaz web típica de PhpMyAdmin.

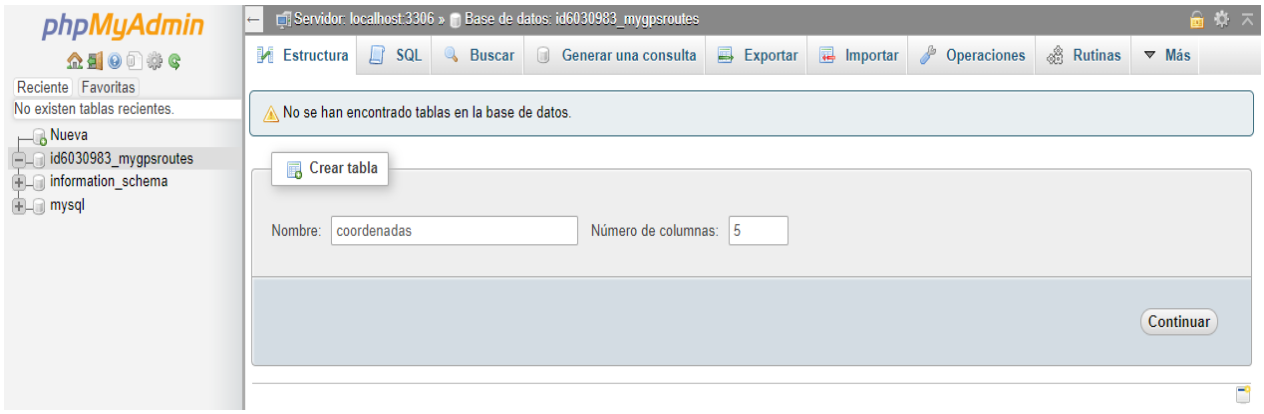


Ilustración 34. Interfaz web phpMyAdmin.

Vamos a crear la tabla donde más adelante introduciremos los datos de las coordenadas. En vez de haciendo uso de la interfaz, iremos a la pestaña SQL e introduciremos las siguientes líneas en SQL:

```
CREATE TABLE `nombreBaseDeDatos`.`nombreTabla` (
    `id` INT(4) NOT NULL AUTO_INCREMENT ,
    `ruta` INT(50) NOT NULL ,
    `nombre` VARCHAR(100) NOT NULL ,
    `lat` FLOAT(10,6) NULL DEFAULT NULL ,
    `lon` FLOAT(10,6) NULL DEFAULT NULL ,
    PRIMARY KEY (`id`)) ENGINE = InnoDB;
```

Tenemos que tener en cuenta que la primera línea no será para todos iguales y en mi caso concreto sería:

```
CREATE TABLE `id6030983_mygpsroutes`.`coordenadas`
```

Con lo cual es importante que mires bien el nombre de tu BBDD y nombres la tabla como prefieras, lo que sí debes dejar tal como está son las demás líneas.

Una vez que hayamos introducido esto ya tendremos una BBDD accesible desde el exterior con una tabla denominada "coordenadas" y diferentes atributos (id, nº de ruta a la que pertenece la coordenada, nombre del lugar de esa coordenada, latitud y longitud), pero no tenemos información almacenada en la misma, que será el paso que vamos a dar a continuación.

Teniendo selecionada la tabla de coordenadas volvemos a pulsar la pestaña de SQL e introducimos las siguientes líneas:

```
INSERT INTO `coordenadas` (`id`,`ruta`,`nombre`,`lat`,`lon`) VALUES
(1,1,"Barqueta",37.404671,-5.997488),
(2,1,"CanalSur",37.406240,-5.998621),
(3,1,"TeatroCentral",37.408266,-5.995937),
(4,1,"JuanBautista",37.410239,-5.997647),
(5,1,"CartujaSport",37.410887,-5.999535),
(6,1,"Etsi",37.411705,-6.001445),
(7,2,"SanTelmo",37.381240,-5.996310),
(8,2,"Mcdonals",37.381521,-5.995087),
(9,2,"PtaJerez",37.382316,-5.993266),
(10,2,"BurguerKing",37.383223,-5.993400),
(11,2,"Starbucks",37.384297,-5.993950),
(12,2,"ArchivoIndias",37.384774,-5.993856),
(13,2,"Catedral",37.385738,-5.993867),
(14,3,"StaJusta",37.390553,-5.976139),
(15,3,"Mediamarkt",37.389607,-5.974307),
(16,3,"CorteIngles",37.387074,-5.972231),
(17,3,"NervionPlaza",37.384279,-5.972730),
(18,3,"SanchezPizjuan",37.384185,-5.970686)
```

Estas coordenadas pertenecen a lugares concretos de la ciudad de Sevilla obtenidas usando la funcionalidad de Google Maps de obtener la coordenada del punto que estemos marcando y las he utilizado para comprobar el buen funcionamiento de la aplicación. Se supone que la tarea de introducir información a la BBDD se haría desde otro dispositivo totalmente ajeno a lo que es la App, ya que el objetivo de la aplicación es obtener la información de esa BBDD externa y mostrar los datos en un mapa.

4.4 Conexión al administrador de archivos del hosting: Filezilla

Ahora vamos a explicar cómo introducir en nuestro hosting un directorio con un archivo php que se encargará de hacer la consulta a la BBDD que tenemos en ese mismo hosting y devolver en formato JSON la información que queremos. Hay que dejar claro que no estamos almacenando toda la BBDD en nuestro terminal ya que esto sería un desperdicio de recursos, sino que recogemos de una dirección web sólo la información que vamos a necesitar desde la App en Android. En nuestro caso vamos a leer todos los campos para ver que están correctos y luego utilizarlos en la App, pero solo estamos leyendo unos datos, no almacenando y gastando memoria de nuestro dispositivo.

Presionamos sobre el botón *Settings* → *General* en 000webhost para ver la información necesaria para conectarnos usando Filezilla a la gestión de archivos de nuestro host.

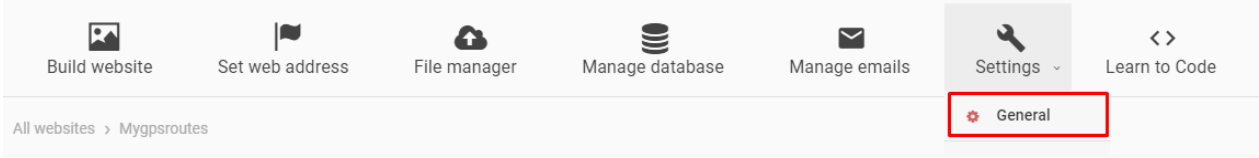


Ilustración 35. Obtener información general de nuestro host.

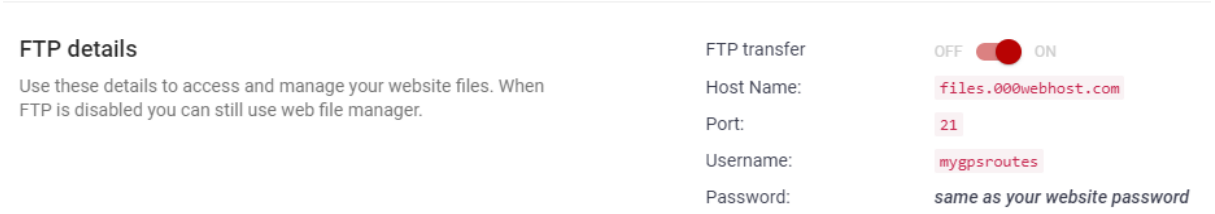


Ilustración 36. Datos para la conexión FTP.

Suponiendo que ya tenemos Filezilla instalado, lo abrimos y accedemos a *File* → *Site Manager* → *New Site* e introducimos los datos de la Ilustración 21. Pulsamos *Connect* y deberemos tener acceso a los directorios de nuestro host.

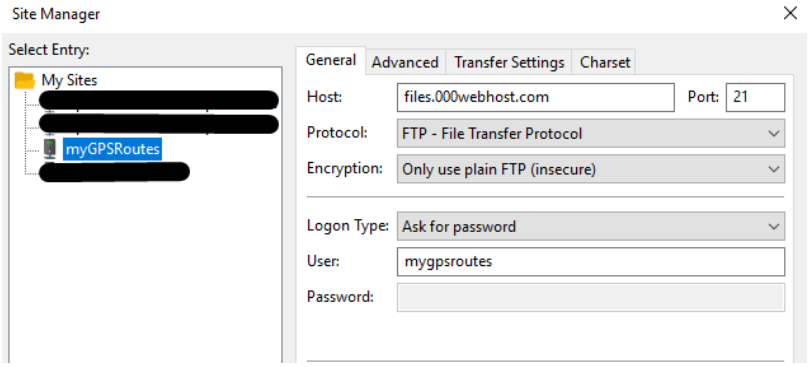


Ilustración 37. Conexión FTP con Filezilla.

Accedemos al directorio *public*_html, que es donde vamos a subir una carpeta denominada googleMaps con un archivo en su interior escrito en php que será el encargado de realizar la consulta.

Vamos a crear, en el escritorio por ejemplo, una carpeta llamada googleMaps. Luego abrimos un bloc de notas (en mi caso he usado Notepad++) y vamos a pegar ahí el siguiente código programado en php:

```php
<?php
  //Datos para conectarnos a la BBDD
  define('DB_HOST', 'localhost');
  define('DB_USER', 'id6030983_sergio'); //Esta línea modificarla con tu propio usuario.
  define('DB_PASS', 'etsi2018');  //Esta línea modificarla con tu contraseña.
  define('DB_NAME', 'id6030983_mygpsroutes'); //Esta línea modificarla con el nombre que diste a la BBDD

  //Conectando con la BBDD
  $conn = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);

  //Comprobación de errores de conexión
  if (mysqli_connect_errno()) {
     echo "Failed to connect to MySQL: " . mysqli_connect_error();
     die();
  }

  //Creando una consulta
  $stmt = $conn->prepare("SELECT id, ruta,nombre, lat, lon FROM coordenadas;");

  //Ejecutando la consulta
  $stmt->execute();

  //Uniendo el resultado a la consulta
  $stmt->bind_result($id,$ruta, $nombre, $lat, $lon);
  $coordenadas = array();
```

```php
//Leyendo cada resultado de la consulta
while($stmt->fetch()){
  $temp = array();
  $temp['id'] = $id;
  $temp['ruta'] = $ruta;
  $temp['nombre'] = $nombre;
  $temp['lat'] = $lat;
  $temp['lon'] = $lon;
  array_push($coordenadas, $temp);
}

//Mostrando el resultado en formato JSON
echo json_encode($coordenadas);
```

A este documento debemos cambiarle la extensión a .php para que funcione, no sirve dejarlo como un .txt, con Notepad++ es sencillo ya que a la hora de guardarlo te permite elegir la extensión, si usas otra herramienta con una búsqueda de diez segundos en Google serás capaz de saber cómo cambiar el formato.

Una vez tengamos el archivo *"coordenadas.php"* lo introducimos en la carpeta que creamos anteriormente *googleMaps*. Ahora debemos ir a Filezilla y subir esa carpeta con el archivo dentro, dentro del directorio *public_html*.

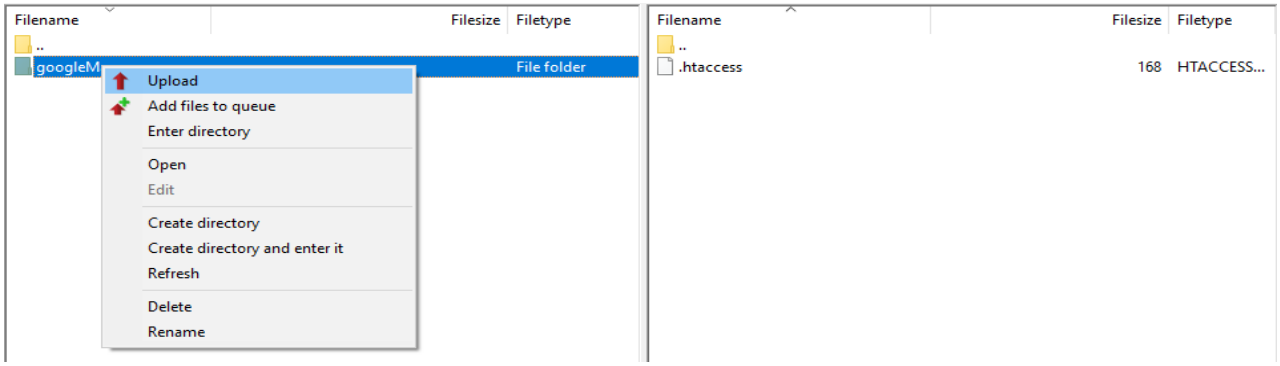
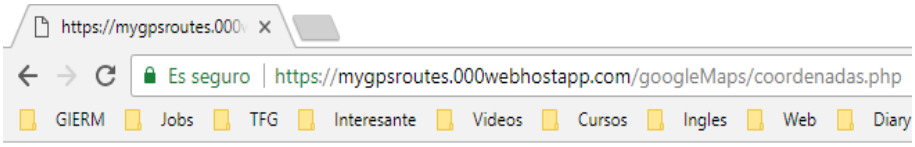


Ilustración 38. Subida carpeta al host usando Filezilla.

Con esto la consulta mediante php a nuestra BBDD externa debería estar realizándose correctamente, para comprobarlo deberías visitar *https://NombreDeTuWeb.000webhostapp.com/googleMaps/coordenadas.php*, y deberás obtener la siguiente salida en tu navegador web:



[{"id":1,"ruta":1,"nombre":"Barqueta","lat":37.404671,"lon":-5.997488},
{"id":2,"ruta":1,"nombre":"CanalSur","lat":37.406239,"lon":-5.998621},
{"id":3,"ruta":1,"nombre":"TeatroCentral","lat":37.408264,"lon":-5.995937},
{"id":4,"ruta":1,"nombre":"JuanBautista","lat":37.41024,"lon":-5.997647},
{"id":5,"ruta":1,"nombre":"CartujaSport","lat":37.410889,"lon":-5.999535},
{"id":6,"ruta":1,"nombre":"Etsi","lat":37.411705,"lon":-6.001445},
{"id":7,"ruta":2,"nombre":"SanTelmo","lat":37.381241,"lon":-5.99631},
{"id":8,"ruta":2,"nombre":"Mcdonals","lat":37.381519,"lon":-5.995087},
{"id":9,"ruta":2,"nombre":"PtaJerez","lat":37.382317,"lon":-5.993266},
{"id":10,"ruta":2,"nombre":"BurguerKing","lat":37.383224,"lon":-5.9934},
{"id":11,"ruta":2,"nombre":"Starbucks","lat":37.384296,"lon":-5.99395},
{"id":12,"ruta":2,"nombre":"ArchivoIndias","lat":37.384773,"lon":-5.993856},
{"id":13,"ruta":2,"nombre":"Catedral","lat":37.385738,"lon":-5.993867},
{"id":14,"ruta":3,"nombre":"StaJusta","lat":37.390553,"lon":-5.976139},
{"id":15,"ruta":3,"nombre":"Mediamarkt","lat":37.389606,"lon":-5.974307},
{"id":16,"ruta":3,"nombre":"CorteIngles","lat":37.387074,"lon":-5.972231},
{"id":17,"ruta":3,"nombre":"NervionPlaza","lat":37.384277,"lon":-5.97273},
{"id":18,"ruta":3,"nombre":"SanchezPizjuan","lat":37.384186,"lon":-5.970686}]

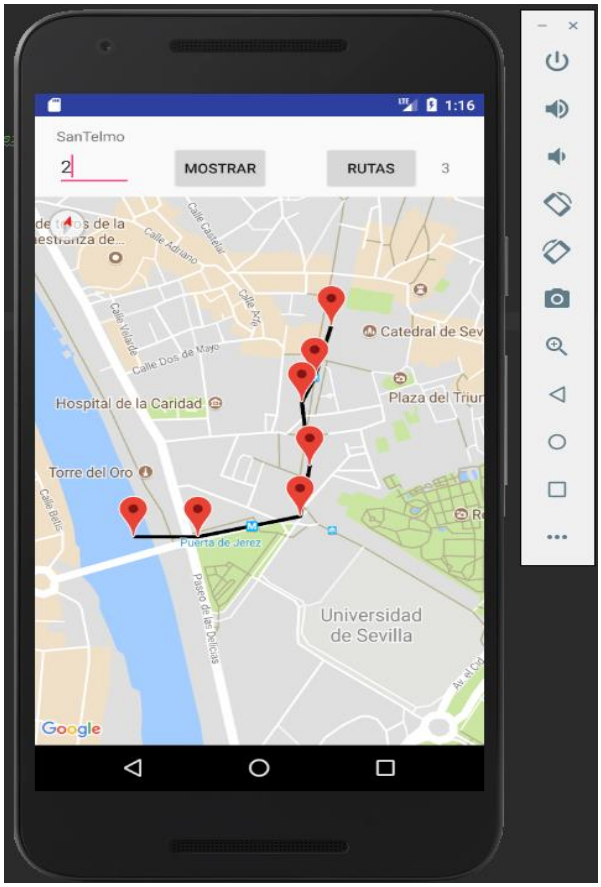
Ilustración 39. Resultado de la consulta web devuelta en formato JSON.

4.5 Comprobación final de la App: Android Studio

Si hemos llegado hasta este punto obteniendo todos los resultados que se han ido mencionando durante las explicaciones solo nos queda comprobar que la App recoja y opere con los datos de la BBDD. Para ello vamos a comprobarlo en el AVD.

Abrimos el AVD y de nuevo ejecutamos la aplicación como se ha hizo en los primeros apartados. Al abrirse la App mostrará el mapa y los botones, en este caso al pulsar el botón *RUTAS* esta vez sí que ocurre algo, el número de la derecha de este botón cambia, mostrando un número que indica la cantidad de rutas que se han leído de la BBDD. Si durante la ejecución de la App la BBDD se modifica y se vuelve a pulsar el botón la BBDD se leerá correctamente.

En este caso introdujimos tres rutas diferentes para hacer las pruebas, con lo que podemos indicar en el cuadro de texto de la izquierda un número entre el uno y el tres para elegir qué ruta queremos mostrar. Introduciendo la ruta dos, obtenemos lo siguiente:



Como podemos observar cada coordenada se muestra con un marcador rojo típico de Google Maps, y cada una de las coordenadas se unen con una línea negra. Si pulsamos en cada uno de los marcadores se mostrará un índice de menor a mayor indicando lo que sería el orden en el que han sido grabadas las coordenadas (el índice cero correspondería a la coordenada inicial y la mayor a la coordenada final de dicha ruta).

Para comprobar que todo ha ido bien se recomienta escribir todas y cada una de las rutas y ver si se muestran dibujadas en el mapa. Además, también se recomienda introducir una ruta más mientras se está utilizando la App para ver si lee correctamente la BBDD sin tener que reiniciar la aplicación.

Al tener seleccionada una de las coordenadas abajo a la derecha aparecerá un botón que nos conecta directamente con la aplicación *Google Maps* y nos permite establecer una ruta hasta ese punto (esto último no lo hemos programado nosotros sino ya estamos utilizando una de las funcionalidades del software de *Google Maps*).

Ilustración 40. App mostrando ruta leída de BBDD.

4.6 Generación e instalación de .apk en dispositivo Android

Vemos que todo funciona. Pero lo interesante es, por supuesto, tener la App desarrollada en nuestro terminal con Android. Para esto necesitamos el instalador de la aplicación, que en Android estos ficheros tienen extensión .apk. Para ellos, en *Android Studio* buscamos en el menú de arriba la opción *Build* → *Build APK(s)* , pulsamos sobre ella y esperamos a que se genere.

Una vez la tenemos buscamos el directorio donde se ha almacenado en nuestro ordenador y lo enviamos al terminal donde queramos probar la App, mediante correo electrónico, usando Dropbox, conectando el dispositivo al ordenador, etc…

Cuando tengamos acceso al archivo de extensión APK desde nuestro teléfono e intentemos instalarlo no tendremos permisos para hacerlo ya que es un fichero de origen desconocido. Para poder instalar archivos con extensión .apk en Android debemos activar la opción de *Aplicaciones de origen desconocido* , esta opción suele encontrarse en *Ajustes* → *Ajustes avanzados* → *Seguridad*, pero puede cambiar de un terminal a otro con que se recomienda buscar en internet como hacerlo, es muy sencillo.

En mi caso concreto he probado la App en un terminal *Sony Xperia Z5 Compact* usando la versión de *Android 7.1.1 Nougat*. Aporto captura de pantalla:

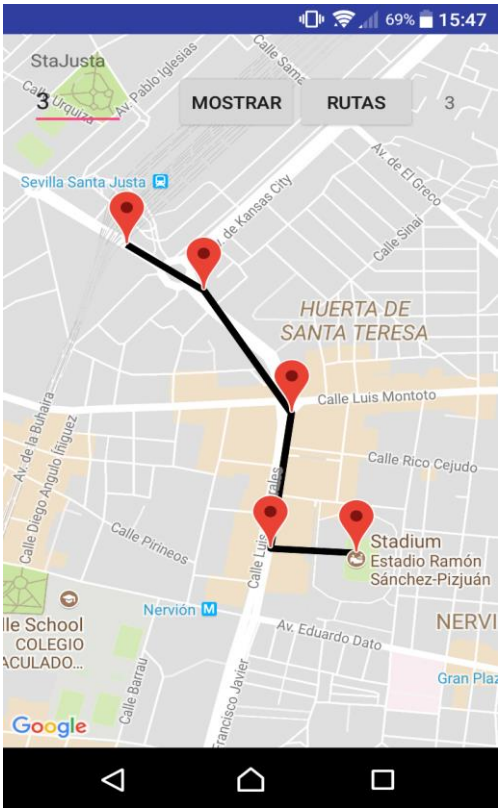


Ilustración 41. App funcionando en dispositivo físico Android.

4.7 Manual de usuario de la App

Aunque es bastante trivial entender el uso vamos a explicar el manejo de la interfaz de la aplicación que se ha desarrollado:

Ilustración 42. Manual de usuario.

- En rosa: Al pulsar este botón se cargará la información de las coordenadas en de las variables destinadas a ello de la App y escribirá en el cuadro verde un número. A veces sólo se carga los datos y no actualiza el número del cuadro verde, entonces debemos volver a pulsarlo y saldrá.
- En verde: Indica el número de rutas totales que se encuentran actualmente en la BBDD.
- En morado: Cuadro de texto donde introducimos qué ruta queremos mostrar, en este caso si en el cuadro verde se muestra un tres, se podrá introducir un número entre el uno y el tres.
- En naranja: Al pulsar este botón de dibujará la ruta que hemos indicado en el cuadro morado. En el caso de que haya una dibujada la borrará y dibujará la nueva.
- En rojo: Indica el nombre de la primera coordenada de la ruta como manera de informar al usuario de la ruta que se está consultando.

5 CONCLUSIONES

5.1 Lo aprendido

Tras la realización del presente proyecto hemos podido comprender lo extenso que es el mundo de la creación de aplicaciones en Android. Por algo existe una Formación Profesional de Grado Superior que se dedica exclusivamente al desarrollo de de aplicaciones en este S.O. .

El objetivo inicial ha sido conseguido con éxito. Para acabar este trabajo he tenido que dedicar horas de estudio, prácticas y corrección de errores en todo lo referente al aprendizaje del lenguaje de programación Java. Esta, sin duda, ha sido la pendiente con más inclinación en lo que se refiere a esfuerzo invertido. También he tenido que aprender lo básico del manejo de bases de datos y el lenguaje SQL, lo fundamental de la programación php, administrar un hosting y a utilizar diferentes programas con el que manejar todo lo mencionado.

Una de las habilidades que he mejorado durante el desarrollo del proyecto ha sido la capacidad de búsqueda correcta en internet. Otra de las habilidades que he tenido que poner en práctica ha sido el aprendizaje autónomo. Estas dos habilidades son, hoy día, sumamente importantes para personas del campo de la ciencia, ingeniería e investigación .

Es cierto que internet está inundado de información, pero es importante saber cómo buscarla, qué información descartar y con cual quedarte para aplicar a la idea que estés desarrollando. Tienes que buscar en libros, en apuntes realizados por expertos en la materia, tutoriales, webs dedicadas a la enseñanza, en cursos y ser capaz de aplicar de forma correcta toda la información recolectada.

Lo interesante de este proyecto ha sido que, aunque al final lo único que veamos es una App funcionando en un teléfono móvil, para conseguirlo se ha tenido que unir diversas tecnologías que funcionan como apoyo a esa aplicación. Esto está directamente relacionado con lo que es la ingeniería y nos hace reflexionar con lo que nos encontraremos en un futuro.

En proyectos de mayor escala que el que hemos realizado suelen unirse varias ramas de la ciencia e ingeniería, de forma que siempre se necesita de un equipo de profesionales, cada uno en su materia, que sean capaces de llegar a un acuerdo combinando sus conocmientos para conseguir un mismo objetivo.

En resumen, a la hora de realizar un como el que en este documento se redacta, se pueden diferenciar varias etapas. Una primera etapa de investigación, donde hay que obtener y leer toda la documentación posible. Una segunda etapa de comprensión y aprendizaje de lo que se necesita para poder empezar. Una tercera, donde vamos aplicando los conocimientos adquiridos en los dos primeros pasos y el proyecto va adoptando forma. Un cuarto paso en el que comprobamos que lo aplicado tiene sentido y se comprueba que cumple las especificaciones y una última etapa en la que se muestra en un documento todo el proceso que se ha llevado a cabo para llegar hasta el final.

5.2 Resultados y mejoras propuestas

Los resultados obtenidos han sido satisfactorios y ha sido testeado en varios terminales funcionando a la perfección en todos los casos.

Con las coordenadas que hemos introducido en la BBDD tenemos hasta tres rutas diferentes que se muestran a continuación:

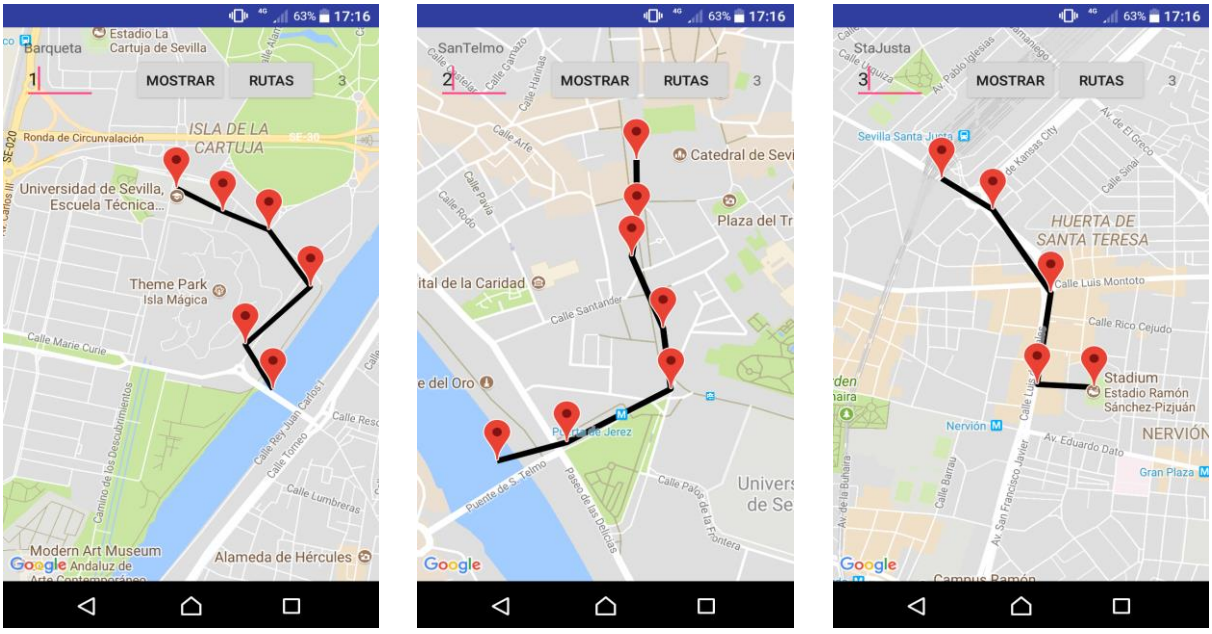


Ilustración 43. Visualización de las rutas de prueba.

Mostramos también dos funcionalidades de la App que se ha creado:

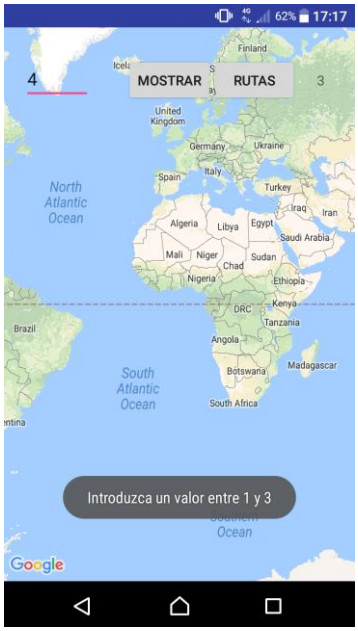


Ilustración 45. Mensaje emergente en App.

Ilustración 44. Información al seleccionar un marcador.

Dos mejoras de la aplicación desarrollada que son interesantes pero que no se han conseguido, aunque sí intentado son:

- Cargar los datos en la App de forma síncrona nada más otro dipositivo mande nueva información a la BBDD.
- En vez de la interfaz de botones, un menú de selección de ruta que cree dinámicamente botones según el nº de rutas que haya, indicando el nombre de la ruta antes de seleccionarla.

REFERENCIAS

[1] John Horton, (2015). Android Programming for Beginners, Birminghan: Packt Publishing Ldt.

[2] Rick Boyer, (2016). Android Application Development Cookbook, Birminghan: Packt Publishing Ldt.

[3] Ramez Elmasri y Shamkant B. Navathe, (2007). Fundamentos de Sistemas de Bases de Datos, Madrid: Pearson Eduación S.A.

[4] Android Developers, (2018). API Android.
Recuperado en abril 2018 de: https://developer.android.com/guide/

[5] Google Maps, (2018). API Google Maps.
Recuperado en abril 2018 de: https://developers.google.com/maps/

[6] Píldoras Informáticas, (2012-2018). Curso online de SQL.
Recuperado en abril 2018 de: http://www.pildorasinformaticas.es/course/curso-sql/

[7] NASASpacePlace, (2015). What is GPS.
Recuperado en mayo 2018 de: https://spaceplace.nasa.gov/gps/en/

[8] GPS: The Global Positioning System, (2018).
Recuperado en junio 2018 de: https://www.gps.gov/

[9] GISGeography, (2018). Remote Sensing.
Recuperado en junio 2018 de: https://gisgeography.com/

[10] Radio-Electronics, (2018). GPS signals by Ian Poole.
Recuperado en junio 2018 de: http://www.radio-electronics.com/

[11] Stack Overflow, (2018). Preguntas y Respuestas entre desarrolladores.
Recuperado en mayo 2018 de: https://stackoverflow.com/questions

Se han utilizado las siguientes herramientas para el desarrollo completo de la aplicación y realización de contenido del presente documento:

- Android Studio para la programación de la App.
    - < www.android.com/ >
- Hosting en 000webhost como servicio para almacenar nuestro directorio.
    - < es.000webhost.com/ >
- PhpMyAdmin para la gestión de la BBDD del hosting.
    - < www.phpmyadmin.net/ >
- Filezilla para conexión FTP con el hosting.
    - < filezilla-project.org/ >
- Notepad++ para la programación en php.
    - < notepad-plus-plus.org/ >
- Matlab para el ejercicio de cálculo de posición mediante trilateración.
    - < es.mathworks.com/products/matlab.html >
- Microsoft Access para el diseño de la BBDD de ejemplo.
    - < products.office.com/es-es/access>
- Microsoft Word para la edición del presente documento.
    - < products.office.com/es-ES/word >