Treball Fi de Carrera

Enginyeria Tècnica de Telecomunicació
Especialitat en Sistemes Electrònics

Desarrollo de una aplicación Web para representación de datos de posicionamiento

Ana Sierra Calderón

Director: Jose A. Lopez Salcedo

Departament de Telecomunicació I Enginyeria de Sistemes
Área de Teoria del Senyal I Comunicacions

Escola Tècnica Superior d'Enginyeria (ETSE)
Universitat Autònoma de Barcelona (UAB)

Junio 2009

UAB

El tribunal d'avaluació d'aquest Treball Fi de Carrera, reunit el dia *data_de_la_presentació*, ha acordat concedir la següent qualificació:

President: Joan Garcia

Vocal: Gonzalo Seco

Secretari: Jose A. Lopez Salcedo

El sotasignant, Jose A. Lopez Salcedo, Professor de l'Escola Tècnica Superior d'Enginyeria (ETSE) de la Universitat Autònoma de Barcelona (UAB),

CERTIFICA:

Que el treball presentat en aquesta memòria de Treball Fi de Carrera ha estat realitzat sota la seva direcció per l'alumne Ana Sierra Calderón.

I, perquè consti a tots els efectes, signa el present certificat.

Bellaterra, 27 de Mayo de 2009.

Signatura: Jose A. Lopez Salcedo

ÍNDICE

1. Introducción 1

1.1. Introducción, motivación y objetivo del proyecto 1
1.2. Estructura de la memoria 3

2. GPS 4

2.1. Introducción al sistema GPS 4
2.1.1. El segmento espacial 4
2.1.2. El segmento de control 5
2.1.3. El segmento usurario 6
2.2. Cálculo de la posición 7
2.3. Formatos de intercambio de datos 8
2.3.1. Fichero RINEX de Observación 10
2.3.2. Fichero RINEX de Navegación 10

3. Herramienta MATLAB para la lectura de archivos RINEX/SwRx 12

3.1. Introducción a MATLAB 12
3.2. Descripción de las funciones MATLAB utilizadas 13
3.2.1. Funciones creadas en este proyecto 15
3.2.2. Resto de funciones utilizadas 17
3.3. MATLAB Compiler, creación del ejecutable Linux 23

4. Google Maps 25

4.1. Introducción a Google Maps 25
4.2. API de Google Maps 27

5. Creación de la página web 30

5.1. Lenguajes de programación utilizados 30
5.1.1. PHP 30
5.1.2. HTML 31
5.1.3. JavaScript 32
5.2. Descripción de las páginas utilizadas 33
5.3. Método de utilización de la Web 36

6. Entorno de trabajo 40

6.1. Sistema operativo Unix/Linux 40
6.2. Sistema operativo Ubuntu 41
6.3. Servidor Apache 41

7. Ejemplo práctio de la aplicación 44

8. Conclusiones 49

Referencias 51

Apéndice: contenido del CD 52

Agradecimientos

En primer lugar, me gustaría agradecer a Jose A. Lopez Salcedo la dedicación y el tiempo invertido en resolver dudas, aconsejar, reunirse conmigo.. en definitiva en hacer de guía en la elaboración y creación de este proyecto.

A mis amigos de la facultad, “las tres marias”, gracias por escucharme en todo momento y apoyarme incluso en la distancia.

A mis amigas/compañeras de piso, gracias por estar ahi, por vuestro apoyo y ayuda, las reinas del photoshop, del word y de la traducción, aunque a partir de ahora ya no seamos compañeras, espero que sigamos siendo buenas amigas.

Por último dedicarle esta memoria a mis padres y a mi hermano, sin vosotros nada de esto hubiera sido posible. Os quiero.

A todos, muchas gracias.

CAPÍTULO 1:
INTRODUCCIÓN

1.1 Introducción, motivación y objetivo del proyecto

El desarrollo y despliegue del sistema GPS ha supuesto un hito tecnológico cuya importancia va creciendo según se sucede la aparición de aplicaciones en las que interviene, con esta perspectiva podemos afirmar que al sistema GPS, pero engeneral, a los sistemas de posicionamiento por satelite (Galileo, Glonass,Beidou, etc) les depara un gran futuro, por lo que es obvio que las aplicaciones de soporte de esta tecnología tienen una gran utilidad.

Con este proyecto se ha pretendido desarrollar una herramienta útil que permita al usuario visualizar en la aplicación Google Maps los datos GPS, frutos de la observación y recopilación de un receptor, con rapidez y sin necesidad alguna de manipulación previa de los mismos.

Esto lo hemos conseguido desarrollando una aplicación Web en la que el usuario tiene que rellenar un sencillo formulario, al que por un lado debe aportar los datos de referencia de la sesión GPS que desee visualizar, y por otro debe seleccionar los archivos de Observación (formato RINEX[1] o el formato propio manejado por el Grupo SPCOMNAV[2]) y de Navegación (formato RINEX) de dicha sesión, que desee subir al servidor. Una vez enviados estos datos, el usuario tan sólo deberá esperar a que la aplicación haga los cálculos pertinentes y refleje las coordenadas calculadas en el mapa de Google Maps.

En el momento en que el servidor detecta los datos suministrados por el usuario en su directorio, ejecuta la aplicación encargada del cálculo de las coordenadas.

Esta aplicación no es más que un ejecutable de MATLAB® que interpreta los datos suministrados por el usuario y mediante una serie de funciones calcula las coordenadas de dicha sesión GPS, éstas son almacenadas por la misma aplicación en un fichero con formato .xml, que será el que posteriormente Google Maps interprete.

Una vez se descubre el archivo .xml con las coordenadas en el directorio, Google Maps procede a la implementación de las mismas en el mapa incorporado a la

[1] Ver Apéndice I
[2] Ver Referencias

Web gracias al API que Google ofrece. De esta forma el usuario ya puede visualizar los datos almacenados por el receptor GPS en un mapa.

Una vez el usuario contempla las posiciones marcadas en el mapa de nuestra página Web, dispone de una serie de herramientas que le permiten variar la visualización del mismo. Por un lado el API de Google Maps nos ha permitido agregar al mapa dos controles: un control para el zoom, que nos permite obtener una vista más o menos general de las posiciones, y otro control para el tipo de mapa, ya que aunque en nuestro proyecto hemos estimado oportuno disponer por defecto el formato de mapa convencional, si el usuario lo prefiere puede cambiar la vista a formato satélite o formato híbrido, que no es más que la combinación del formato convencional y la vista por satélite.

Por otro lado la página Web dispone de un enlace que redirecciona la página a otra, con las mismas posiciones implementadas en Google Maps unidas entre sí en forma de ruta, para que si el usuario lo desea pueda hacerse una idea del itinerario seguido.

En el siguiente esquema damos una idea de las diferentes capas que conforman este proyecto:

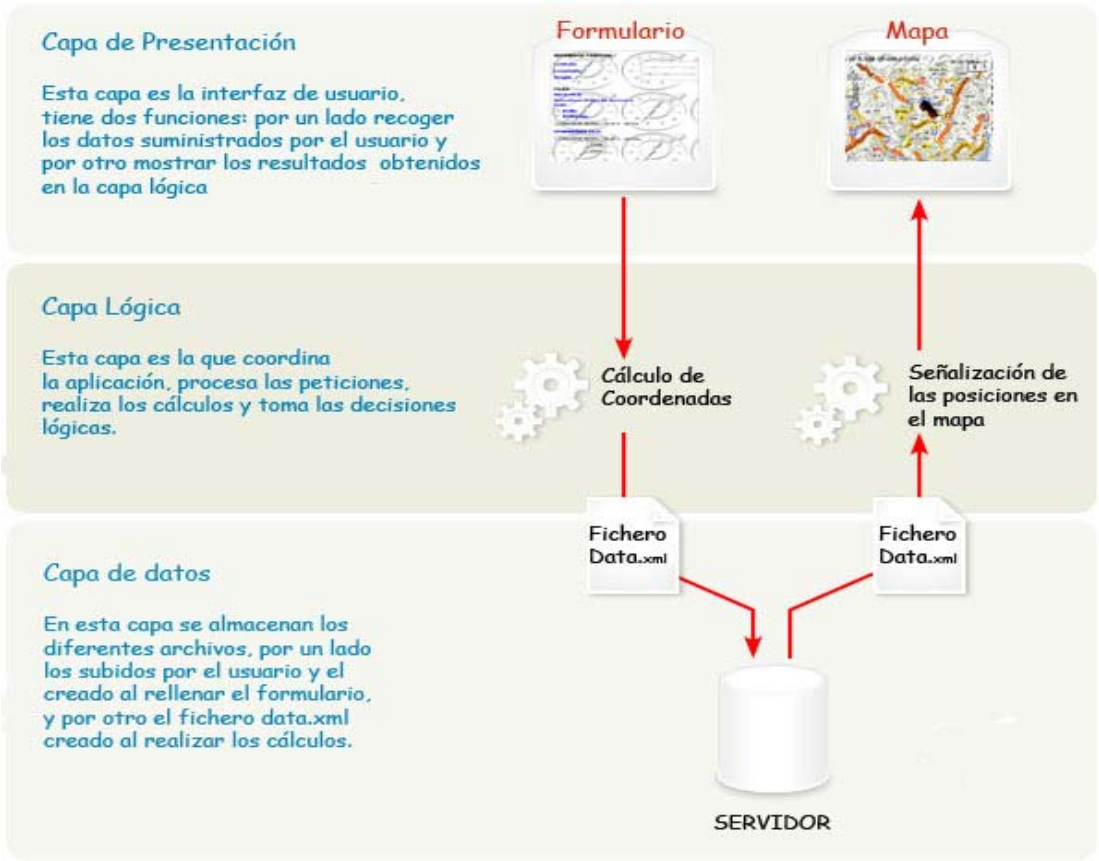

Fig.1.1: Arquitectura del TFC

1.2 Estructura de la memoria

La memoria que hemos hecho se estructura en siete capítulos ordenados de la siguiente forma:

CAPÍTULO 1: En este primer capítulo introducimos el proyecto describiendo la motivación que ha llevado a su creación y el modo de funcionamiento a grandes rasgos.

CAPÍTULO 2: En el segundo capítulo exponemos una breve explicación del funcionamiento de los GPS, así como el formato uniformizado de intercambio GPS, la importancia que este tiene y sus dos principales archivos (el RINEX de Observación y el de Navegación).

CAPÍTULO 3: En el tercer capítulo hacemos una pequeña introducción al software utilizado para calcular las coordenadas a partir de los archivos subidos, en este capítulo también se describen las funciones utilizadas para el cálculo y la manera de crear el ejecutable que funciona en el servidor.

CAPÍTULO 4: En este capítulo introducimos la herramienta creada por Google: Google Maps y su funcionamiento, así como la manera en que ha sido introducida en nuestra aplicación.

CAPÍTULO 5: Este capítulo es el encargado de explicar el funcionamiento de la página Web describiendo las diferentes partes de ella y los lenguajes utilizados en cada una.

CAPÍTULO 6: En este capítulo explicamos el método y la plataforma en la que hemos colgado la aplicación.

CAPÍTULO 7: En este último capítulo introduciremos un ejemplo práctico de la aplicación para aclarar diversos aspectos del proyecto.

CAPÍTULO 8: En este último capítulo plasmaremos las conclusiones derivadas de este proyecto.

CAPÍTULO 2:

GPS

2.1 Introducción al sistema GPS

El GPS (Global Positioning System) es un Sistema Global de Navegación por Satélite (GNSS) que permite calcular con precisión la posición de un objeto (o persona) en cualquier lugar del mundo. Aunque fue inventado por el gobierno francés y el belga, el sistema fue desarrollado por el departamento de Defensa de los Estado Unidos conjuntamente con la agencia espacial. Su motivación era obtener un sistema que cumpliese los requisitos de globalidad, abarcando toda la superficie del globo; continuidad, funcionamiento continuo sin afectarle las condiciones atmosféricas; altamente dinámico, para posibilitar su uso en aviación; precisión y que fuese de bajo coste.

El sistema GPS se divide en tres segmentos, el segmento espacial, el segmento de control y el segmento usuario. A continuación haremos una breve introducción de cada uno.

2.1.1 EL segmento espacial

Los objetivos principales de este segmento son, a partir de las instrucciones que reciben del segmento de control:

- Proporcionar una referencia de tiempo atómico.
- Generar las señales RF pseudo aleatorias
- Almacenar y reenviar el mensaje de Navegación.

Este segmento está formado por una constelación de treinta y dos[3] satélites uniformemente dispersados alrededor de la tierra en seis órbitas aproximadamente circulares a una altitud media de 20.200 Km, éstas orbitas tienen una inclinación de 55º sobre el plano del ecuador y la separación entre ellas es de unos 60º.

[3] Número de satélites operativos a viernes uno de mayo de 2009 según el USNO (U.S. Naval Observatory)

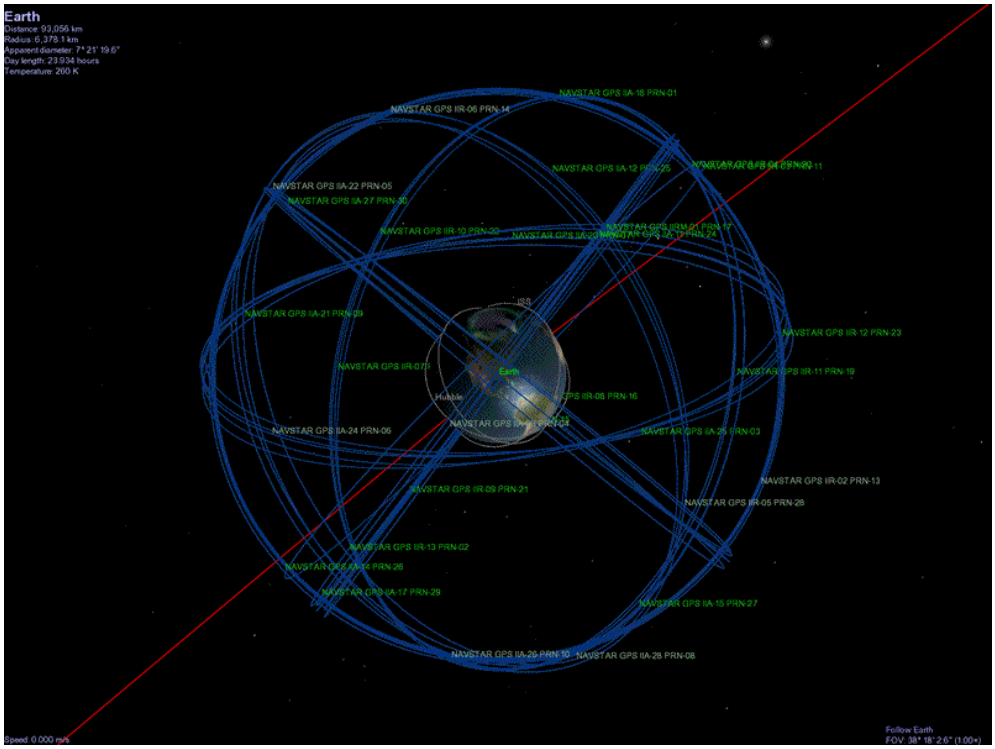

Fig.2.1: Constelación satélites GPS: Sistema NAVSTAR-GPS

Este sistema permite a los usuarios debidamente equipados recibir globalmente señales de radio con información de alta precisión acerca de su posición, velocidad y tiempo.

2.1.2 El segmento de control

El segmento de control está formado básicamente por las estaciones terrestres GPS, las cuales se encargan del control y mantenimiento del estado, así como de la configuración del segmento espacial. Además se encarga de predecir las efemérides y el comportamiento de los relojes de los satélites, mantener la escala de tiempo GPS mediante relojes atómicos y actualizar periódicamente el mensaje de navegación de cada uno de los satélites.

La red de estaciones GPS está formada por tres tipos diferentes de estaciones:

- Master Control Station: Su función es recoger y almacenar los datos de seguimiento procedentes de las estaciones monitoras para utilizarlos posteriormente para el cálculo de las órbitas de los satélites y los parámetros de reloj. Estos resultados son transmitidos a una de las tres Estaciones de Control de Tierra (Ground Control Station) para que sean subidos eventualmente a los satélites.

- Monitor Stations: En un principio fueron cinco a las que posteriormente se añadieron seis más pertenecientes a la Agencia Nacional de Inteligencia Geoespacial (NGA). Cada una de ellas está equipada con un reloj estándar de cesio y receptores que continuamente miden pseudo distancias de todos los satélites que tienen a la vista.
- Ground Control Stations: Las Estaciones de Control de Tierra están ubicadas en las Estaciones Monitoras de la Isla de Ascensión, de Diego García y de Kwajalein. Están equipadas con antenas con la capacidad de transmitir la información de efemérides y de reloj calculada por la Estación Maestra de Control a los satélites.

En el siguiente mapa refleja la posición de cada una de ellas:

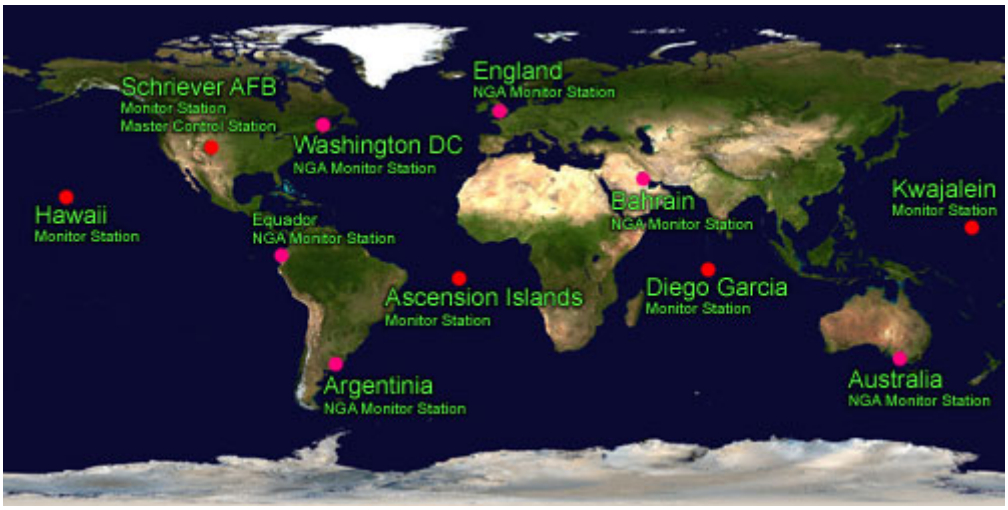

Fig. 2.2: Mapa de situación de las diferentes estaciones GPS

Debido a que existen diferentes perturbaciones que afectan a las órbitas de los satélites, éstas deben ser constantemente actualizadas, y con ello los diferentes cálculos de corrección, lo cual origina un nuevo mensaje de navegación que se envía las Estaciones de Control de Tierra para ser transmitidos a los satélites.

2.1.3 El segmento usuario

Este segmento lo compone el instrumental que deben utilizar los usuarios para la recepción, lectura, tratamiento y configuración de las señales, con el fin de alcanzar los objetivos de su trabajo.

Los elementos que forman este segmento son el equipo de observación y el software de cálculo, que puede ser objeto de uso tras la campaña de observación, o bien realizable en tiempo real, donde se obtienen los resultados in situ.

2.2 Cálculo de la posición

El objetivo de todo sistema GPS no es otro que el de calcular la posición de un punto cualquiera en un sistema de coordenadas (x,y,z), el cálculo de esta posición se basa en el principio de triangulación, partiendo del cálculo de un punto a un mínimo de tres satélites cuya localización es conocida. A partir de ese momento el receptor GPS medirá las distancias que los separan calculando el tiempo que tarda cada señal en viajar de los satélites hasta él, para ello multiplicará el tiempo de vuelo de la señal transmitida por su velocidad de propagación. Para poder medir el tiempo de vuelo de la señal el receptor y el satélite deben estar sincronizados, pues deben generar simultáneamente el mismo código, en el caso del satélite esto no es un problema puesto que posee un reloj atómico de Cesio extremadamente exacto, pero en el caso del receptor se trata de un reloj normal de Cuarzo lo que genera una desviación que añade una incógnita más al cálculo, por lo que se hace necesario al menos cuatro satélites para estimar correctamente las posiciones.

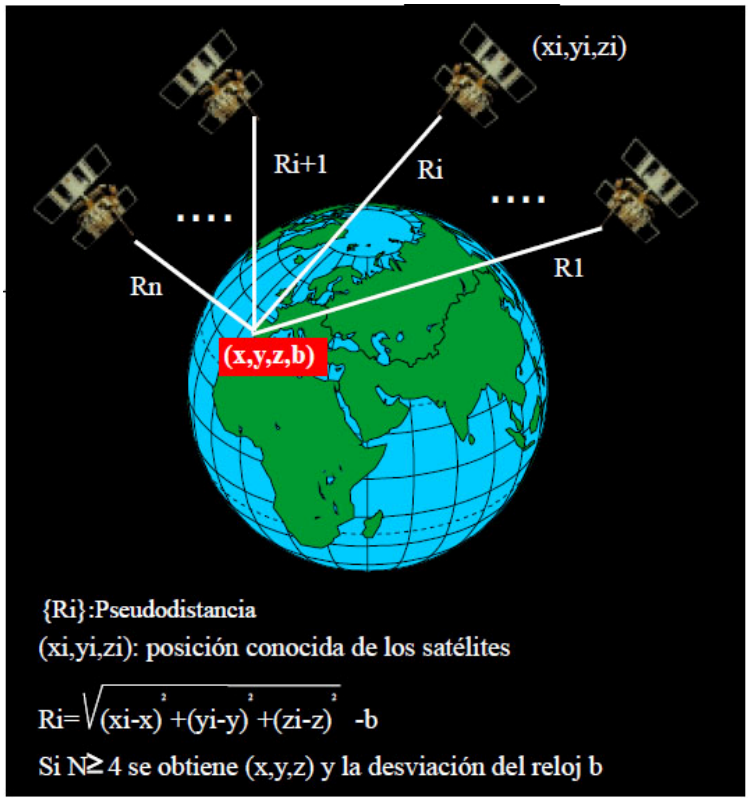

Fig.2.3: Método de triangulación

Las distancias con errores debidos al sincronismo se denominan pseudo distancias. En el cálculo de estas pseudo distancias es importante tener en cuenta que las señales GPS son muy débiles y se hallan inmersas en el ruido de fondo debido al planeta en la banda de radio, este ruido natural está formado por una serie de pulsos aleatorios, lo que motiva la creación de un código pseudo-aleatorio artificial para los receptores GPS, de esta manera en cada instante un satélite transmite una señal con el mismo patrón que la serie pseudo-aleatoria generada por el receptor, esto garantiza que el receptor se esté comunicando con el satélite correcto. La existencia de este código, permite el control de acceso al sistema de satélites, de forma que en situaciones conflictivas se podría cambiar el código, obligando a todos los satélites a utilizar una banda de frecuencia única sin interferencias pues cada satélite posee un código GPS propio, y lo que es más importante, respecto a lo que a coste económico se refiere permite la aplicación de "teoría de la información" para amplificar las señales GPS, por lo que estas señales pueden ser captadas por los dispositivos sin el uso de grandes antenas.

En base a la sincronización mediante código pseudo-aleatorio del satélite y el receptor, este último calcula la distancia realizando un desplazamiento temporal de su código hasta lograr la coincidencia con el código recibido; este desplazamiento corresponde al tiempo de vuelo de la señal, que en definitiva es lo buscado, para que como se ha dicho anteriormente al multiplicarla por la velocidad de transmisión nos de la posición.

En esta memoria no ahondaremos más en el cálculo de la posición, es un tema complejo que requiere la explicación de una serie de ecuaciones que nosotros no hemos desarrollado. Las funciones que en nuestra aplicación realizan estos cálculos han sido cedidas por el grupo SPCOMNAV[4].

2.3 Formatos de intercambio de datos

Los datos GPS, los frutos de la observación y la recopilación hecha por un receptor, normalmente se almacenaban en un formato binario específico que era reconocido sólo por los programas ofrecidos por el fabricante.

[4] Ver referencias.

Para resolver este problema en 1989 la Asociación Internacional de Geodesia recomendó el uso del RINEX (Reciever Independent Exchange Format) como formato estándar de intercambio de ficheros GPS.

El archivo RINEX está basado en que la mayoría del software GPS emplea los siguientes observables:

- La medida de la portadora de fase en una o dos frecuencias (L1 o L1 y L2).
- La medida de Pseudo distancia o código.
- El tiempo obtenido en el instante de validar las medidas de fase y código.

Esto hace que la mayoría de la información que recogen los receptores sea innecesaria, pues únicamente con estos tres observables y alguna información adicional relativa al estacionamiento (altura de la antena, nombre de la estación...) sería suficiente.

El RINEX implica que los datos binarios propios de cada tipo de receptor pueden ser transformados a formato independiente universal ASCII7 durante el proceso de descarga, permitiendo así usar otro tipo de software o intercambiar datos procedentes de otros receptores. Dado que la estructura de los datos fuente (binario) difiere de cada receptor, es necesario que cada proveedor de software GPS genere un interprete para este formato.

Actualmente el formato RINEX se compone de seis tipos de archivos:

- EL fichero de Observación.
- El fichero Meteorológico.
- El fichero de Navegación.
- El fichero de Navegación del sistema GLONASS.
- El fichero de Navegación del sistema GEO.
- El fichero con los datos de hora y fecha del satélite y el receptor.

Cada fichero RINEX se compone de una cabecera y de una sección de datos, la cabecera contiene la información general del fichero como puede ser la relativa a la estación, el receptor o la antena, la sección de datos contiene los datos referentes al tipo de archivo. En nuestro caso para el cálculo de las coordenadas, tan solo utilizaremos el fichero RINEX de Observación y el fichero RINEX de navegación puesto que son los más utilizados, ya que en las versiones RINEX anteriores a la 2.0 no existen el resto de archivos.

2.3.1 Fichero RINEX de Observación[5]

El fichero de observación puede ser de cuatro tipos según el sistema de satélites del que provenga, tipo R el correspondiente al sistema GLONASS, el S para el sistema GEO, el tipo T para el sistema NNSS, el M para el mixto y por último el tipo G el correspondiente al sistema GPS.

En el RINEX de Observación se almacenan las medidas de pseudo rango (desviación del reloj del receptor respecto al del satélite), fase, efecto Doppler y el SNR (medida de la calidad con la que llega la señal de un satélite al receptor).

En el caso del pseudo rango se aceptan tres tipos de medida, la C1 (correspondiente al código C/A estándar sobre la frecuencia L1), la P1 (código Precise en L1) y la P2 (código P en L2). La fase por su parte también dispone de dos tipos de medidas la que se puede realizar sobre la frecuencia L1 y la que se realiza sobre la frecuencia L2 que se denominan igual que las frecuencias. Para el efecto Doppler también se dispone de dos medidas (una para cada tipo de frecuencia) la D1 y la D2 y otras dos medidas para el Doppler de tránsito integrado, T1 a 150 MHz y T2 a 400 MHz. En el caso del SNR también nos encontramos con dos tipos de media la S1 para el SNR medio en la frecuencia L1 y el S2 para el medido en la frecuencia L2.

En nuestro caso el archivo RINEX de Observación que interpreta la función read_rin_ob.m debe proceder del sistema GPS y sus medidas tener el formato C1 L1 D1 S1.

2.3.2 Fichero RINEX de Navegación[6]

El fichero RINEX de Navegación contiene los datos de orbitales, los parámetros del reloj y la precisión de las medidas de pseudo rango de los satélites observados. Su cabecera puede contener opcionalmente datos del mensaje de navegación tales como los parámetros del modelo ionosférico para aparatos de una sola frecuencia y términos de correcciones relacionados con el tiempo GPS y UTC10. Una gran parte de este fichero está basado en el formato ARGO de la NGS.

Se transmite a un régimen binario de 50 bps y se tarda 12.5 min. en enviarlo completamente.

[5] Ver Apéndice I.
[6] Ver Apéndice I.

En la cabecera del fichero se detalla la procedencia del mismo, N para el sistema GPS, G para el sistema GLONASS y H para el sistema SBAS de navegación, así como los parámetros de la ionosfera. En el cuerpo del fichero encontramos los datos sobre el PRN, el tiempo de satélite, la época y la predicción de cada órbita.

CAPÍTULO 3:

Herramienta MATLAB® para la lectura de archivos Rinex/SwRx

3.1 Introducción a MATLAB®

MATLAB® es un software de cálculo técnico de altas prestaciones para cálculo numérico y visualización, es un entorno fácil de usar, donde los problemas y las soluciones son expresados como se escriben matemáticamente sin la programación tradicional, puesto que consta de lenguaje de programación propio.
El nombre proviene de "MATrix LABoratory" (Laboratorio de Matrices). Fue escrito originalmente para proporcionar un acceso sencillo al software matricial desarrollado por los proyectos LINPACK y EISPACK, que juntos representan lo más avanzado en programas de cálculo matricial.

MATLAB® es un sistema interactivo cuyo elemento básico de datos es una matriz que no requiere dimensionamiento. Esto permite resolver muchos problemas numéricos de forma más rápida que con los lenguajes de programación tradicionales, como C, C++ y Fortran.

En los últimos años ha evolucionado a partir de la colaboración de muchos usuarios. En entornos universitarios se ha convertido en la herramienta de enseñanza estándar para cursos de introducción en álgebra lineal aplicada, así como cursos avanzados en otras áreas. En la industria, se utiliza para investigación y para resolver problemas prácticos de ingeniería y matemáticas, con un gran énfasis en aplicaciones de control y procesamiento de señales.
MATLAB® también proporciona una serie de soluciones específicas denominadas toolboxes, que no son mas que conjuntos de funciones que extienden el entorno para resolver clases particulares de problemas.

El lenguaje de programación que utiliza (lenguaje M) es el utilizado por nosotros para implementar las funciones que detallaremos a continuación.

3.2 Descripción de las funciones MATLAB® utilizadas

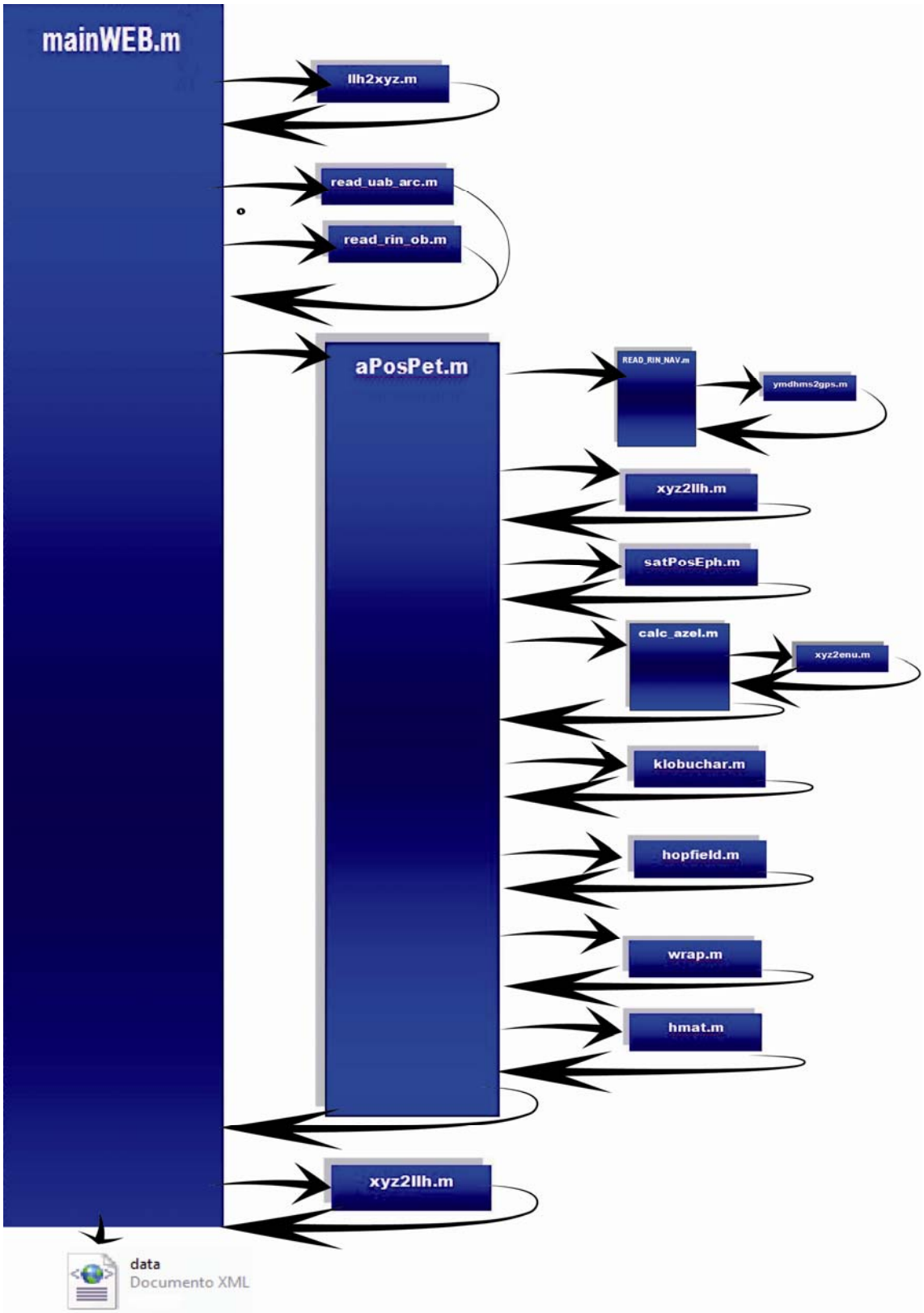

Fig. 3.1: Esquema de las funciones MATLAB ® utilizadas

En este esquema podemos observar el orden real en el que son llamadas las funciones y que función es la responsable de su arranque.

En este apartado pasaremos a explicar las diferentes funciones .m utilizadas, éstas en su conjunto sirven para leer el archivo de navegación (formato Rinex Internacional) y el archivo de observación (formato Rinex Internacional y formato SwRx), y con ello calcular las coordenadas del usuario.

En primer lugar describiremos detalladamente las realizadas por nosotros (mainWEB.m, read_rin_ob.m y read_uab_arc.m) tras lo que pasaremos a explicar las cedidas por el grupo SPCOMNAV[7].

[7] Ver Referencias.

3.2.1 Funciones creadas en este proyecto:

mainWEB.m	
Descripción	Como su nombre indica se trata de la función principal, es la encargada de recoger los datos suministrados por el usuario mediante el formulario de la aplicación Web, y llamar al resto de funciones para que realicen los cálculos. En primer lugar llamará a llh2xyz.m a la que le pasará los datos de referencia: latitud, longitud y altura (pasados previamente a grados). En función del tipo de archivo de observación descargado llamará para su lectura a: • read_rin_ob.m →caso de que el archivo sea formato Rinex. • read_uab_arc.m →caso de que el archivo sea formato SwRx. Tras lo que llamará a la función aPosPet.m para que opere con los datos hallados hasta ahora. Después llamará a la función xyz2llh.m a la que pasara las coordenadas calculadas por aPosPet.m Por último mainWEB.m creará un archivo .xml en el que se reflejarán los datos de latitud y longitud en radianes devueltos por xyz2llh.m que se acabarán plasmando en el mapa de Google Maps.

Read_rin_ob.m	
Código de llamada	`[acq, numSnapShots]= read_rin_ob(obsfile)`
Descripción	Esta función es la encargada de leer el archivo Rinex de Observación, e interpretar los datos del mismo, almacenando los necesarios en las diferentes variables que se utilizarán para el cálculo de las coordenadas.
Variable de entrada	**Obsfile**→Fichero de observación subido por el usuario a la Web.
Variables de salida	**numSnapShots**→Numero de SnapShots capturados por el gps. **acq**→ Struct en el que se almacenan los resultados de adquisición: -acq.SV.CN0: Relación C/N0 estimado. -acq.SV.Fe: Error doppler. -acq.SV.PRN: Número PRN (Ruido Pseudos Aleatorio) de los k-th SV. -acq.SV.pseudoRange: Pseudo rango. -acq.SV.TOW: TOW (Time Of Week) inicial. -acq.SV.estTOW: TOW estimado. -acq.SV.OK: Marcador OK.

Read_rin_uab_arc.m	
Código de llamada	`[acq, numSnapShots]= read_rin_arc(obsfile)`
Descripción	Esta función es la encargada de leer el archivo utilizado por el software de la UAB, e interpretar los datos del mismo, almacenando los necesarios en las diferentes variables que se utilizarán para el cálculo de las coordenadas.
Variable de entrada	**Obsfile**→Fichero de observación subido por el usuario a la Web.
Variables de salida	**numSnapShots**→Numero de SnapShots capturados por el gps. **acq**→ Struct en el que se almacenan los resultados de adquisición: -acq.SV.CN0: Relación C/N0 estimado. -acq.SV.Fe: Error doppler. -acq.SV.PRN: Número PRN (Ruido Pseudos Aleatorio) de los k-th SV. -acq.SV.pseudoRange: Pseudo rango. -acq.SV.TOW: TOW (Time Of Week) inicial. -acq.SV.estTOW: TOW estimado. -acq.SV.OK: Marcador OK.

3.2.2 Resto de funciones utilizadas:

APosPet	
Código de llamada	**`[Position, PosError, UserTime, TruePosition, SVPosition1, a_info]= aPosPet(a_info, acq, flag, RefTOW, TruePosition, navfile)`**
Descripción	Esta función se encarga de calcular la posición utilizando los datos GPS-asistido del archivo subido por el usuario.
Variables de entrada	**a_info→** Struct en el que se almacena información asistida: -a_info.ionpar: parámetros ionosféricos. -a_info.Nsv: Número de satélites visibles para los que se dispone de información sobre la asistencia. -a_info.refLocation.xyz: coordenadas ECEF (sistema de coordenadas cartesiano centrado en la Tierra y que rota fijado en su superficie) de la estación de referencia. -a_info.SV.PRN: Número PRN (Ruido Pseudos Aleatorio) de los k-th SV. **acq→** Struct en el que se almacenan los resultados de adquisición: -acq.SV.CN0: Relación C/N0 estimado. -acq.SV.Fe: Error doppler. -acq.SV.PRN: Número PRN (Ruido Pseudos Aleatorio) de los k-th SV. -acq.SV.pseudoRange: Pseudo rango. -acq.SV.TOW: TOW (Time Of Week) inicial. -acq.SV.estTOW: TOW estimado. -acq.SV.OK: Marcador OK. **Flag→** Estructura con banderas para la selección de diferentes funcionalidades del software. -flag.plot: parcelas permitidas **RefTOW→**TOW (Time Of Week) de la estación de referencia como estimación inicial de la posición del usuario. **Navfile→**Nombre del fichero de navegación. **TruePosition→**Verdadera posición del usuario.
Variables de salida	**Position→**Posición estimada del usuario en coordenadas ECEF. **PosError→** Error en la posición del usuario. **UserTime→**Tiempo estimado del usuario. **SVPosition1→**Posición de los satélites. **a_info→**Struct en el que se almacena información asistida.

Calc_azel	
Código de llamada	`[az, el]= calc_azel(sv_pos, user_pos)`
Descripción	Esta función calcula el azimut (ángulo que nos indica la posición del satélite respecto al sur real) y la elevación del satélite.
Variables de entrada	**Sv_pos→** Posición del satélite en WGS-84 (**W**orld **G**eodetic **S**ystem 19**84**) xyz. **User_pos→** Posición del usuario en WGS-84 xyz.
Variables de salida	**az→** Azimut de los satélites en radianes. **el→** Elevación de los satélites en radianes.

hmat.m	
Código de llamada	`[h]= hmat(svmat, usrpos)`
Descripción	Esta función es la encargada de calcular la matriz de cosenos directores.
Variables de entrada	**Svmat→** Matriz de la posición de los satélites definida por el usuario en coordenadas cartesianas. **Usrpos→** Posición estimada definida por el usuario en coordenadas cartesianas
Variables de salida	**h→**Matriz de cosenos directores para posicionamiento GPS

hopfield.m	
Código de llamada	`[trop_dry, trop_wet]= hopfield(elev, trop_model)`
Descripción	Esta función calcula los retrasos troposféricos (húmedo y seco) usando el modelo Hopfield.
Variables de entrada	**Elev→** Ángulo de elevación de los satélites GPS (en radianes). **Trop_model→** Entrada para el modelo troposférico (opcional), matriz nx3 de la forma [p t e]: -p: Presión atmosférica de superficie (en mb). -t: Temperatura de superficie (en^{o}K). -e: Presión parcial del vapor de agua (en mb).
Variables de salida	**Trop_dry→** Retraso troposférico seco. **Trop_wet→** Retraso troposférico húmedo.

klobuchar.m	
Código de llamada	`[dTiono]= klobuchar(LATU, Lonu, Az, El, Ttr, ion)`
Descripción	Esta función es la encargada de calcular los retrasos ionosféricos de acuerdo con el modelo Klobuchar.
Variables de entrada	**Latu→** Latitud del usuario (en radianes). **Lonu→** Longitud del usuario (en radianes). **Az→**Azimut de los satélites calculados en sentido positivo de las agujas de reloj desde el norte verdadero. **El→** Elevación de satélite. **Ttr→** Tiempo pedido en GPS SOW. **Ion→** Array de los parámetros ionosféricos.
Variable de salida	**dTiono→** Retraso ionosférico.

llh2xyz.m	
Código de llamada	`[xyz]= llh2xyz(llh)`
Descripción	Con esta función convertimos de coordenadas geográficas (latitud, longitud, altura) a coordenadas ECEF (sistema de coordenadas cartesiano centrado en la Tierra y que rota fijado en su superficie).
Variable de entrada	**llh→** array compuesto por: -llh(1): Latitud en radianes. - llh(2): Longitud en radianes. - llh(3): Altura sobre elipsoide en metros.
Variable de salida	**xyz→**array compuesto por: -xyz(1): Coordenada X ECEF. -xyz(2): Coordenada Y ECEF. -xyz(3): Coordenada Z ECEF.

READ_RIN_NAV.m	
Código de llamada	`[navmes, ionpar]= READ_RIN_NAV(navfile)`
Descripción	Esta function es la encargada de leer el fichero Rinex de Navegación[1] y cambiar el formato a una matriz con 21 filas y una columna para cada satélite.
Variable de entrada	**Navfile→** Fichero Rinex de Navegación.
Variables de salida	**Navmes→** Matriz mensaje de navegación con información dependiendo del tipo de fichero de navegación utilizado[2]. **Ionpar→** Array con los parámetros ionosféricos.

satPosEph.m	
Código de llamada	`[satp, navpar, no_ephem_flag]= satPosEph(treqwk, treqsow, svreq, navmes, flagA)`
Descripción	Con esta función calculamos la posición de los satélites.
Variables de entrada	**Treqwk→** Época del tiempo de la posición requerida (semanas GPS). **Treqsow→** Época del tiempo de la posición requerida (segundos GPS). **Svreq→** PRN (Ruido Pseudos Aleatorio) del satélite actual. **Navmes→** Struct con información teniendo en cuenta el mensaje de navegación. **FlagA→** (0) = sin efecto (1)= comprobar que toc-treq<8 horas (error).
Variables de salida	**Satp→** Vector de posición del satélite en coordenadas cartesianas ECEF (metros). **Navpar→** array con parámetros del mensaje de navegación **No_ephem_flag→** (0)=efemérides encontradas. (1)= efemérides no encontradas.

[1] Ver apéndice I.
[2] Ver apéndice I.

wrap.m	
Código de llamada	`[x]= wrap(x, x_max)`
Descripción	Esta función es la encargada de asegurar que una variable pertenezca a un umbral determinado, si éste se sale de él lo que hace es restarle dos veces el mismo.
Variables de entrada	**x**→variable que se evaluará. **x_max**→ valor máximo posible.
Variable de salida	**x**→ valor válido de la variable.

xyz2enu.m	
Código de llamada	`[enu]= xyz2enu(xyz, orgxyz)`
Descripción	Esta función es la encargada de convertir las coordenadas cartesianas ECEF WGS-84 en coordenadas rectangulares local-level-tangent (Este, Norte, arriba).
Variables de entrada	**xyz**→array compuesto por: -xyz(1): Coordenada X ECEF. -xyz(2): Coordenada Y ECEF. -xyz(3): Coordenada Z ECEF. **Orgxyz**→array compuesto por: -orgxyz(1): Coordenada X ECEF del origen local. -orgxyz(2): Coordenada Y ECEF del origen local. -orgxyz(3): Coordenada Z ECEF del origen local.
Variables de salida	**Enu**→ vector columna formado por: -enu(1,1)=Este---coordenada relativa del origen local. -enu(2,1)=Norte---coordenada relativa del origen local. -enu(3,1)=Arriba---coordenada relativa del origen local.

xyz2llh.m	
Código de llamada	`[llh]= xyz2llh(xyz)`
Descripción	Con esta función covertimos de coordenadas ECEF (sistema de coordenadas cartesiano centrado en la Tierra y que rota fijado en su superficie) a coordenadas geográficas.
Variables de entrada	**xyz**→array compuesto por: -xyz(1): Coordenada X ECEF. -xyz(2): Coordenada Y ECEF. -xyz(3): Coordenada Z ECEF.
Variable de salida	**llh**→array compuesto por: -llh(1): Latitud en radianes. -llh(2): Longitud en radianes. -llh(3): Altura en metros.

ymdhms2gps.m	
Código de llamada	`[outmat]= ymdhms2gps(year, month, mday, tour, minute, second)`
Descripción	Esta función se encarga de transformar la fecha (año, mes, día, hora, minuto, segundo) en tiempo GPS (semana GPS y segundo GPS).
Variables de entrada	**Year**→ Año. **Month**→ Mes. **Day**→ Día. **Hour**→ Hora. **Minute**→ Minuto. **Second**→ Segundo.
Variable de salida	**Outmat**→ array compuesto por: -Outmat(1): Semana GPS. -Outmat(2): Segundo GPS.

3.3 MATLAB® Compiler™, creación del ejecutable Linux

MATLAB® Compiler™ es una herramienta contenida en el software de MATLAB®, su aplicación principal es crear archivos ejecutables a partir de código m, de forma que la función principal y todas sus auxiliares se encuentren en un solo paquete, y que con ello no tengamos que abrir el programa ni el directorio, sino que directamente podamos observar los resultados al ejecutarlo.

Al compilar un función y sus auxiliares MATLAB® añade la MCR (MATLAB® Component Runtime) que no es mas que un conjunto de librerías y archivos necesarios a la hora de ejecutar un programa compilado con MATLAB®.

En nuestro caso el MATLAB® Compiler™ fue utilizado para la creación de un ejecutable Linux que nos sirviese para la lectura de los datos ofrecidos por el usuario, el calculo de las posiciones y la posterior creación del fichero .xml en el servidor, de esta manera el usuario en ningún momento se tiene que descargar ningún archivo ni por supuesto tener instalado MATLAB® en su ordenador puesto que la aplicación "corre" en el propio servidor.

Los archivos que se generaron al compilar nuestra función fueron:

- mainWEB.ctf→ archivo biblioteca de extracción rápida de la MCR.

- mainWEB→ aplicación ejecutable.

- Run_mainWEB.sh→ shell script (interprete de comandos para Linux) que se ejecuta para establecer la dirección del entorno y así poder ejecutar la aplicación.

- mainWEB_main.c→ archivo C intermedio que contiene la función principal.

- mainWEB_mcc_component_data.c→ archivo C intermedio que contiene las funciones auxiliares.

El esquema siguiente da una idea del proceso de compilación llevado a cabo:

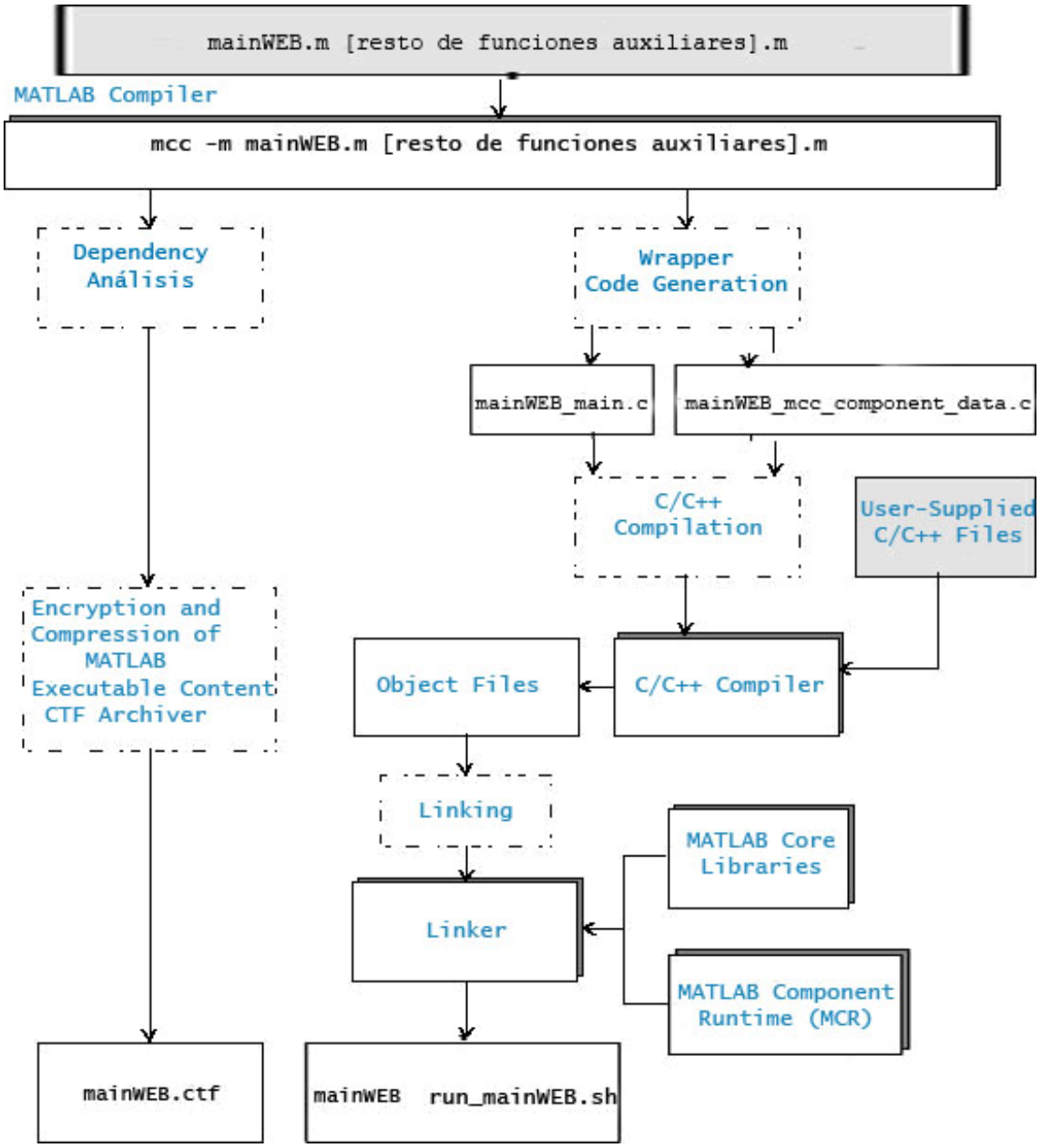

Fig.3.2: Esquema del método de compilación

CAPÍTULO 4:

Google Maps

4.1 Introducción a Google Maps

Google Maps es un servicio gratuito ofrecido por Google desde el 6 de octubre de 2005, se trata de un servidor de aplicaciones de mapas en la Web, que nos permite movernos por la geografía del planeta, la aplicación nos permite imágenes en formato vectorial, en satélite y en forma de Tour virtual, incorpora un buscador que localiza posiciones (poblaciones, calles, negocios..) en el mapa a través del nombre o las coordenadas.

El lenguaje en que está desarrollado es JavaScript e incorpora AJAX lo que da al usuario mayor interactividad con la aplicación.

El hecho de que se puedan crear rutas, o mapas propios a los que se le pueden añadir marcadores con cualquier tipo de información, ha convertido esta herramienta en una de las más utilizadas en la Web.

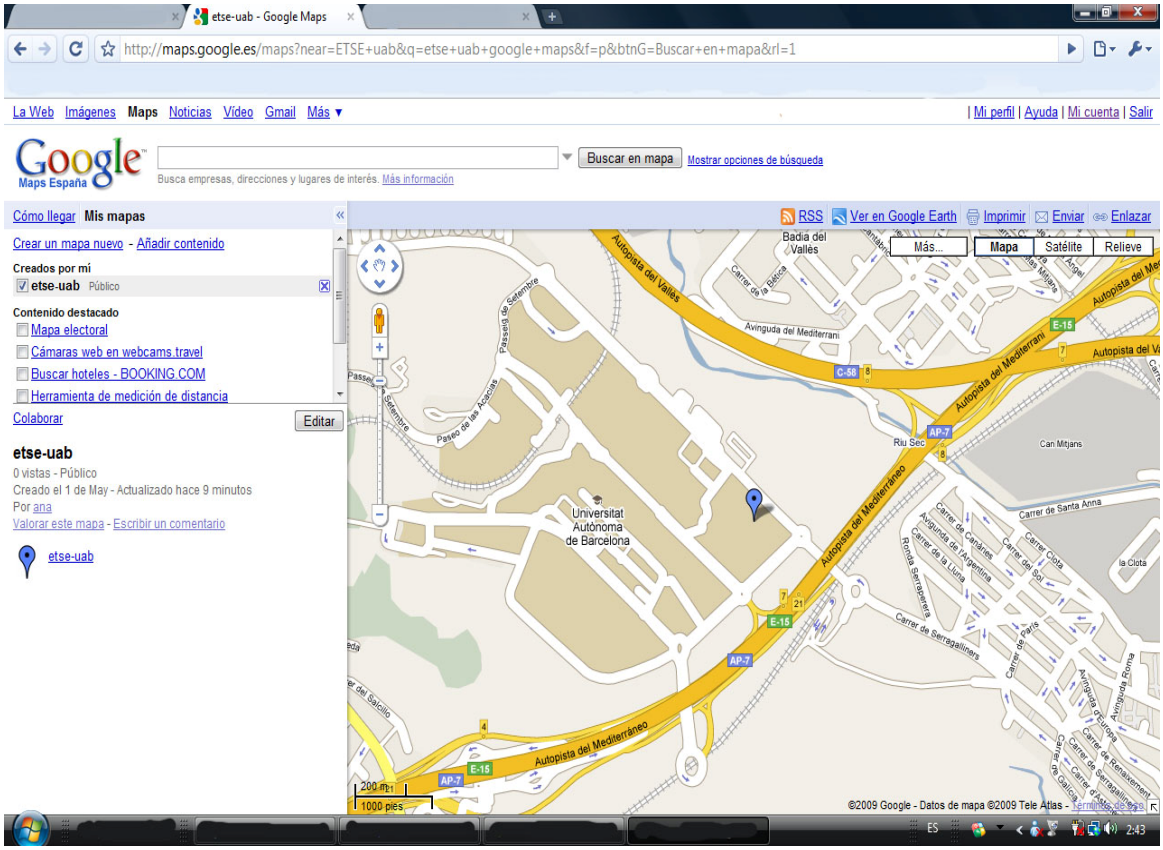

Fig. 4.1: Captura de Google Maps

Como prestaciones básicas Google Maps ofrece las siguientes posibilidades:

- El control sobre el mapa: translación en todas las direcciones y zoom.
- La búsqueda: los usuarios pueden introducir el nombre de una calle, población o la intersección, gracias a la opción de Google Local podemos restringir los resultados a una zona determinados, por añadidura Google Maps también permite la búsqueda de negocios, empresas, instituciones… etc. Similar al concepto de puntos de interés que tienen los GPS.
- Enlace Web: Como resultado de una búsqueda Google Maps también ofrece el enlace de la entidad que se ha buscado, de forma que nos permite ampliar la información de lo buscado.
- Posteriormente Google Maps ha aprovechado la librería para desarrollar diferentes variantes como por ejemplo: Google Moon, Google Mars y Google Earth, este último permite el uso más personalizado y en local (sin necesidad de Internet) del servicio de mapas.

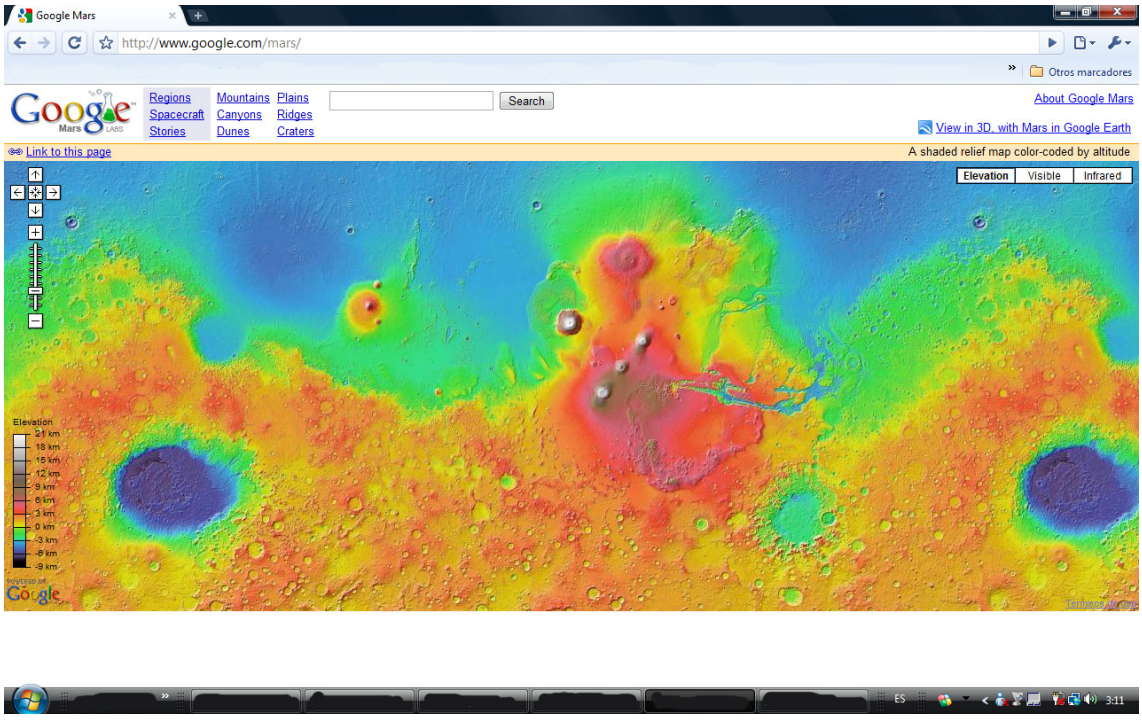

Fig. 4.2: Captura de Google Mars (planeta Marte)

- API: Por último Google Maps nos ofrece su API (Interfaz de Programación de Aplicaciones), que permite a los desarrolladores Web insertarlo en sus páginas.

4.2 API de Google Maps

El API de Google Maps ha permitido gran variedad de aplicaciones que explotan diferentes variantes del código y personalizaciones, el API de Google Maps nos permite insertar la aplicación en nuestra propia página Web con JavaScript, de esta manera podemos reflejar en el mapa las coordenadas calculadas por MATLAB® en nuestro servidor, permitiendo así que el usuario pueda visualizar las posiciones captadas por los satélites en la sesión GPS que subió al servidor.

En este apartado describiremos las diferentes funciones del API de Google Maps que se utilizan en nuestro proyecto.

Para ello explicaremos el código proporcionado por Google Maps al dar de alta la aplicación en nuestro servidor. Para ello seguiremos los pasos tomados en nuestro proyecto:

- En primer lugar como hemos comentado antes, hemos dado de alta la aplicación para poder incluir el mapa en nuestra página Web, para nuestra URL Google Maps nos proporcionó la siguiente clave:

  "ABQIAAAA1Mva-SoetL-ErkTVRbO1zRSctkfHrS9mIs8KjEnA60IulV _HzRRCgAJROeow3Fsv2FVWimOHhWztsw"

- A continuación tuvimos que cargar el API incluyendo la clave obtenida, para ello incluimos en nuestra página HTML el siguiente código:

```
<script
src="http://maps.google.com/maps?file=api&amp;v=2&amp;key=ABQIAAAA1
Mva-SoetL-
ErkTVRbO1zRSctkfHrS9mIs8KjEnA60IulV_HzRRCgAJROeow3Fsv2FVWimOHhWztsw
"
type="text/javascript"></script>
```

En src se indica la ubicación del archivo JavaScript, que incluye todos los símbolos y definiciones que se necesitan para utilizar el API de Google Maps. En key ponemos la clave obtenida de Google para el proyecto.

- Tras lo que procedimos a implementar el mapa en sí:

```
function load() {
if (GBrowserIsCompatible()) {[1]
var map =new GMap2(document.getElementById("map"));[2]
map.addControl(new GLargeMapControl());[3]
map.addControl(new GMapTypeControl());[4]
map.setMapType(G_NORMAL_MAP);[5]
[6]
var iconoMarca = new GIcon(G_DEFAULT_ICON);
iconoMarca.image =
"http://maps.google.com/mapfiles/kml/pal4/icon49.png";[7]
var tamanoIcono = new GSize (20,20); [8]
iconoMarca.iconSize = tamanoIcono; [9]
[10]
iconoMarca.shadow
="http://maps.google.com/mapfiles/kml/pal4/icon49s.png";
var tamanoSombra = new GSize(40,20);
iconoMarca.shadowSize = tamanoSombra;
iconoMarca.iconAnchor = new GPoint(10,20);
[11]
GDownloadUrl("objetos/data.xml", function(data) {[12]
var xml = GXml.parse(data); [13]
var markers =
xml.documentElement.getElementsByTagName("marker");[14]
for (var i = 0; i < markers.length; i++) {[15]
var point = new GLatLng
(parseFloat(markers[i].getAttribute("lat")),

parseFloat(markers[i].getAttribute("lng")));[16]
var marker = new GMarker(point, iconoMarca); [17]
map.addOverlay(marker); [18]
  }
var pcenter = new
GLatLng(parseFloat(markers[0].getAttribute("lat")),
parseFloat(markers[0].getAttribute("lng")));[19]
map.setCenter(pcenter,15); [20]
```

[1]→En primer lugar comprobamos si el mapa es compatible con el explorador Web utilizado.

[2]→Tras lo que creamos una variable map (el mapa en sí) que será la que después insertemos en el código HTML.

[3]→Con esta función añadimos a la aplicación el control de zoom situado en el lateral del mapa.

[4]→Esta función añade el control de tipo de mapa, de esta manera el usuario podrá cambiar el mapa normal (puesto por defecto), a la vista por satélite o al mapa en formato híbrido.

[5]→Con esta orden fijamos el tipo de mapa que por defecto queremos que aparezca en la Web, en nuestro caso el formato mapa normal.

[6]→A continuación pasamos a definir el tipo de marca y tamaño que queremos:

[7]→Fijamos el tipo de marca que queremos, vinculándolo a una dirección Web que lo almacena, en nuestro caso hemos puesto un simple punto  puesto que la marca que viene puesta por defecto para Google Maps  nos parecía un obstáculo a la hora de representar posiciones muy cercanas.

[8]→Con este comando fijamos el tamaño del icono.

[9]→Con este otro comando vinculamos el tamaño fijado a nuestra marca.

[10]→Asociamos una sombra al icono con el mismo procedimiento anterior.

[11]→A continuación pasamos a leer los datos que tenemos en el fichero .xml generado por MATLAB®, para poder implementarlos en el mapa.

[12]→Esta función es la encargada descargar los datos del fichero data.xml generado por MATLAB®.

[13]→Fijamos el método estático de lectura para analizar el fichero XML.

[14]→Esta función hace que se adquieran una serie de datos de la etiqueta maker.

[15]→Hacemos un ciclo for para la lectura de todo el fichero.

[16]→Extraemos de cada etiqueta maker la latitud y longitud.

[17]→Con esta función creamos un marcador en cada punto con su latitud y longitud correspondientes y con el formato que designamos antes.

[18]→Esta función es la encargada de dibujar en el mapa cada marcador.

[19]→Con esta función fijamos la variable centro en el primer punto de nuestro fichero xml.

[20]→y por ultimo esta función es la encargada de centrar el mapa en ese punto con un zoom inicial de 15.

CAPÍTULO 5:
CREACIÓN DE LA PÁGINA WEB

5.1 Lenguajes de programación utilizados

5.1.1 PHP

El PHP (PHP Hypertext Pre-processor), es un lenguaje de programación interpretado de alto nivel y código abierto. El PHP es un lenguaje de servidor, lo que quiere decir que cada vez que se visita la página, éste interpreta y ejecuta el código, en nuestro caso hemos utilizado código PHP para por un lado, tratar los datos aportados por el usuario subiéndolos al servidor cuando envía el formulario (recoger.php) y por otro lado, para borrar del directorio del servidor los archivos subidos y creados por la aplicación en cada sesión (formulario.php).

El usuario no es partícipe de estas acciones pero sí de las consecuencias de ellas, puesto que una vez es implementado el código PHP el servidor redirecciona la página a otra escrita en código HTML, de esta manera se cumple el principal objetivo del código PHP, que no es mas que crear páginas Web dinámicas.

Fig. 5.1: Esquema de funcionamiento código PHP

El lenguaje PHP es muy parecido a los lenguajes de programación más conocidos como C, Perl o Java, lo que facilita a los programadores experimentados realizar aplicaciones

PHP complejas rápidamente. Además el hecho de que sea soportado por todos los sistemas operativos del mercado y la mayoría de los servidores Web, hace de PHP el lenguaje de servidor más utilizado del mundo.

5.1.2 HTML

HTML (Hipertexto Markup Language) es el lenguaje utilizado en la creación de páginas Web. Su sintaxis está basada en marcas o etiquetas que permiten definir la estructura y ubicación de los elementos que vayan a mostrarse en una página Web, y determinar las relaciones que éstos puedan tener con otras partes de la misma mediante hipervínculos.

El objetivo del lenguaje HTML es especificar en el texto la estructura lógica de su contenido, para así permitir la representación del mismo en cualquier navegador. Con esto no pretende describir la apariencia final del documento, sino facilitar que cada sistema operativo pueda darle la forma que sus posibilidades y su navegador le permitan.

Una de las principales características del lenguaje HTML, es la de que gracias a su estructuración en etiquetas (muy intuitivas y de nombre reducido) es muy fácil de entender y aprender. Eso y el hecho de que se admitido por todos los navegadores, ha hecho del HTML el lenguaje estándar para la creación de páginas Web.

HTML por sí mismo no es un lenguaje de programación, pero permite incluir código de otros lenguajes, con lo que se hace más extensa su capacidad y funcionalidad.

En nuestro proyecto contamos con cuatro páginas implementadas en HTML, la primera de ellas es la que sirve al usuario como presentación de la aplicación (index.html), la segunda es la que expone el formulario que debe rellenar con los datos de referencia y los archivos GPS (formulario.html) y por último, están las dos páginas que sirven para mostrar las posiciones resultantes de los cálculos realizados en el mapa de Google Maps (googlempas.html y lineas.html). En estas dos páginas HTML hemos hecho uso de una de las principales características de este lenguaje, la de posibilitar el que sean incluidos códigos de lenguajes de programación en él, en nuestro caso hemos incluido el código JavaScript necesario para insertar la aplicación Google Maps en la Web.

Fig. 5.2: Index.html

5.1.3 JavaScript

JavaScript es un lenguaje de programación interpretado, es decir, no requiere compilación, se utiliza principalmente en el desarrollo de páginas Web y aunque su nombre puede llevar a equívocos, no tiene nada que ver con Java, puesto que a diferencia de éste, no está orientado a objetos, sino que es un lenguaje basado en prototipos[1], ya que las nuevas clases se generan clonando las clases base (prototipos) y extendiendo su funcionalidad.

Como ya hemos citado en el apartado anterior, JavaScript se puede insertar en HTML, esto se debe a que el lenguaje está proveído de una implementación del DOM[2], de manera que se ejecuta en el navegador al mismo tiempo que las sentencias van descargándose con el código HTML cuando el cliente solicita la página. De esta manera en nuestro proyecto, el mapa de Google Maps se carga en la página cuando llamamos a las páginas HTML que lo contienen (Googlemaps.html y lineas.html). Haciendo de esta manera que se cumpla el principal objetivo de JavaScript, que no es otro que interactuar con HTML para enriquecer las páginas Web, añadiendo efectos, dinamismo y en definitiva hacerlas mas interactivas. En resumen, podemos decir que JavaScript tiene dos funcionalidades principales, en primer lugar, crear efectos especiales en las páginas Web para establecer contenidos dinámicos (elementos con movimiento, cambios de estilo, eliminación de elementos…), y en segundo lugar, permitir crear páginas interactivas que ejecuten instrucciones como respuesta a órdenes del usuario, como por ejemplo calculadoras, agendas, tablas de cálculo.

Para los programadores que ya conocen Java o C, JavaScript es fácil de aprender, puesto que su sintaxis es muy similar aunque más simplificada. Esta simplicidad no resulta fácil para los inexpertos, ya que se basa en una disponibilidad de objetos reducida, por lo que algunos aspectos que aparentemente parecen sencillos, pueden requerir una programación bastante compleja.

JavaScript es totalmente exportable de una Web a otra, ya que se trata de código abierto. Por este motivo es el lenguaje de scripts más utilizado del mundo, lo que hace que todos los sistemas operativos lo toleren.

---

[1] Programación basada en prototipos: es un estilo de programación orientada a objetos en el cual, las "clases" no están presentes, y la re-utilización de procesos se obtiene a través de la clonación de objetos ya existentes, que sirven de prototipos, extendiendo sus funcionalidades.

[2] DOM es una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos HTML y XML.

5.2 Descripción de las diferentes páginas utilizadas[3]

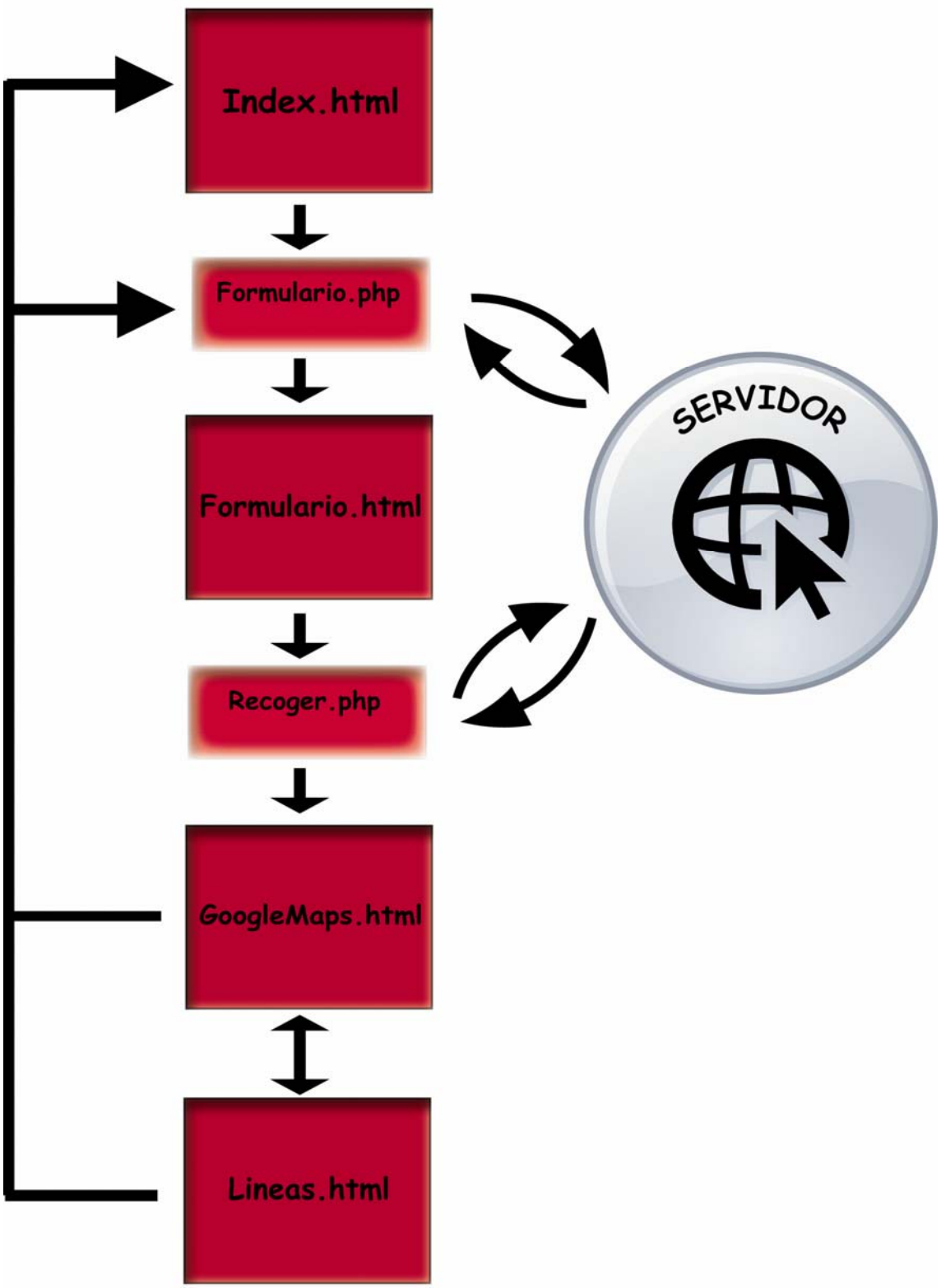

Fig. 5.3: Esquema con el orden de llamada de las páginas

[3] Para ver los códigos de cada página vaya a: Apéndice III

En el esquema anterior podemos observar las diferentes páginas que se utilizan en la aplicación, así como desde que página son llamadas y a que páginas podemos acceder desde ellas.

Nombre de la página	**Descripción**	**Páginas a las que tenemos acceso desde ella**
Index.html	Es la página encargada de presentar al usuario la aplicación.	La imagen central de la página nos redirecciona a la página: **Formulario.php**
Formulario.php	Es la página encargada de eliminar (en el caso de que exista) el archivo data.xml creado por el ejecutable de MATLAB®.	Una vez comprueba que no existe el archivo data.xml en el directorio del servidor, muestra el código incluido en: **Formulario.html**
Formulario.html	Esta página (incluida en **Formulario.php**), es en la que el usuario debe ingresar los datos iniciales, en ella debe rellenar la latitud, longitud y altura inicial, así como, elegir que tipo de archivo de observación va a subir a la aplicación (RINEX o el utilizado por el grupo SPCOMNAV[4]), y seleccionar el archivo de observación y el de navegación que desea cargar en el servidor.	Al pulsar el botón enviar, nos redirecciona a la página de comprobación de recogida y comprobación de datos: **Recoger.php**
Recoger.php	Es la página encargada de recoger la información suministrada por el usuario en el formulario. En primer lugar almacena en	Una vez comprueba que el ejecutable de matlab ha creado el documento data.xml con las coordenadas geográficas redirecciona automáticamente

[4] Ver Referencias.

	un archivo de texto los datos de referencia iniciales (latitud, longitud y altura) y el tipo de archivo de observación que va a ser subido. Y en segundo lugar descarga en un directorio determinado del servidor los dos archivos subidos por el usuario. Al comprobar que todo está correcto ejecuta la aplicación encargada de los cálculos (el ejecutable de MATLAB®). En el caso de que falte algún elemento, aparece en la pantalla un mensaje en el que indica el archivo o el dato que falta.	a: **Googlemaps.html**
Googlemaps.html	Esta página (incluida en **Recoger.php)** nos muestra las posiciones calculadas en un mapa convencional de Google Maps. El usuario una vez se encuentra aquí tiene la opción de acercar más o menos el mapa, o de cambiar el tipo de mapa a la vista por satélite o el híbrido (mezcla del mapa convencional y satélite).	Posee tres enlaces entre los que el usuario puede elegir: •Clicando en *MOSTRAR RUTA* nos redirecciona a: **Lineas.html** •Clicando en Rellenar otro Formulario redirecciona a: **Formulario.php**
Lineas.html	En esta página el usuario puede hacerse una idea de la ruta seguida en la sesión GPS	Posee tres enlaces entre los que el usuario puede elegir: •Clicando en *VOLVER A*

	que ha subido al servidor, puesto que puede ver las diferentes posiciones de ésta unidas mediante líneas geodésicas (línea de menor longitud entre dos puntos dados). Al igual que ocurre en **Googlemaps.html,** el usuario tiene la opción de acercar más o menos el mapa, o de cambiar el tipo de mapa a la vista por satélite o el híbrido (mezcla del mapa convencional y satélite).	*MAPA* nos redirecciona a: **Googlemaps.html** •Clicando en *Rellenar otro Formulario* redirecciona a: **Formulario.php**

5.3 Método de utilización de la Web

Para hacer uso de nuestra aplicación el usuario deberá realizar una serie de acciones, nuestro principio ha sido el de la simplicidad, por lo que la interfaz de la aplicación es muy sencilla.

En este apartado explicaremos paso a paso las acciones que debe realizar el usuario valiéndonos de diagramas de árbol.

- En primer lugar el usuario debe abrir la aplicación:

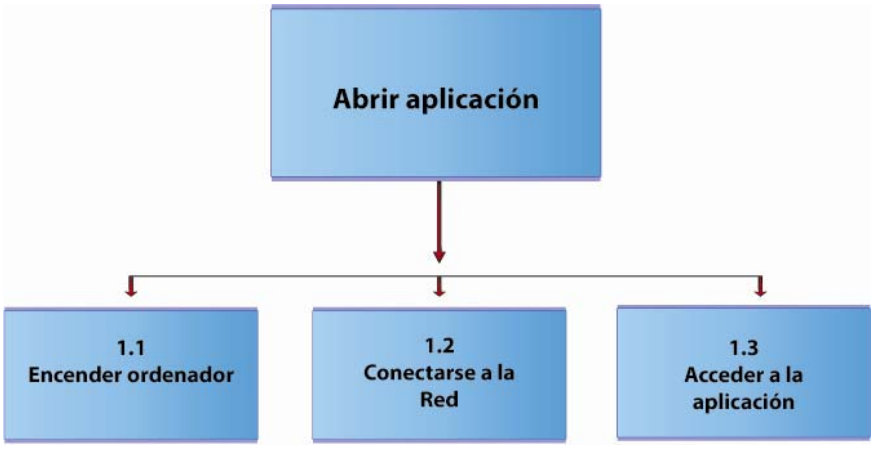

Fig. 5.4: Árbol de apertura de la aplicación

- Seguidamente deberá acceder al formulario:

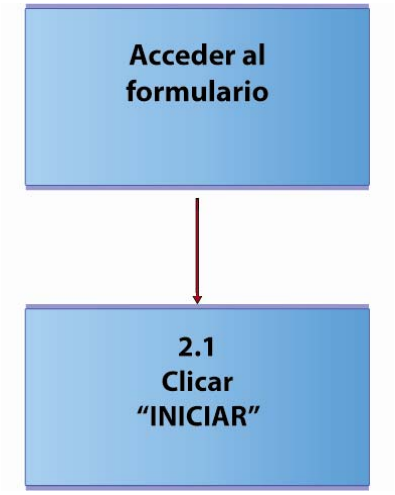

Fig. 5.5: Árbol de acceso al formulario

- Tras lo que deberá rellenarlo con los datos y los archivos de la sesión GPS que desee visualizar en el mapa

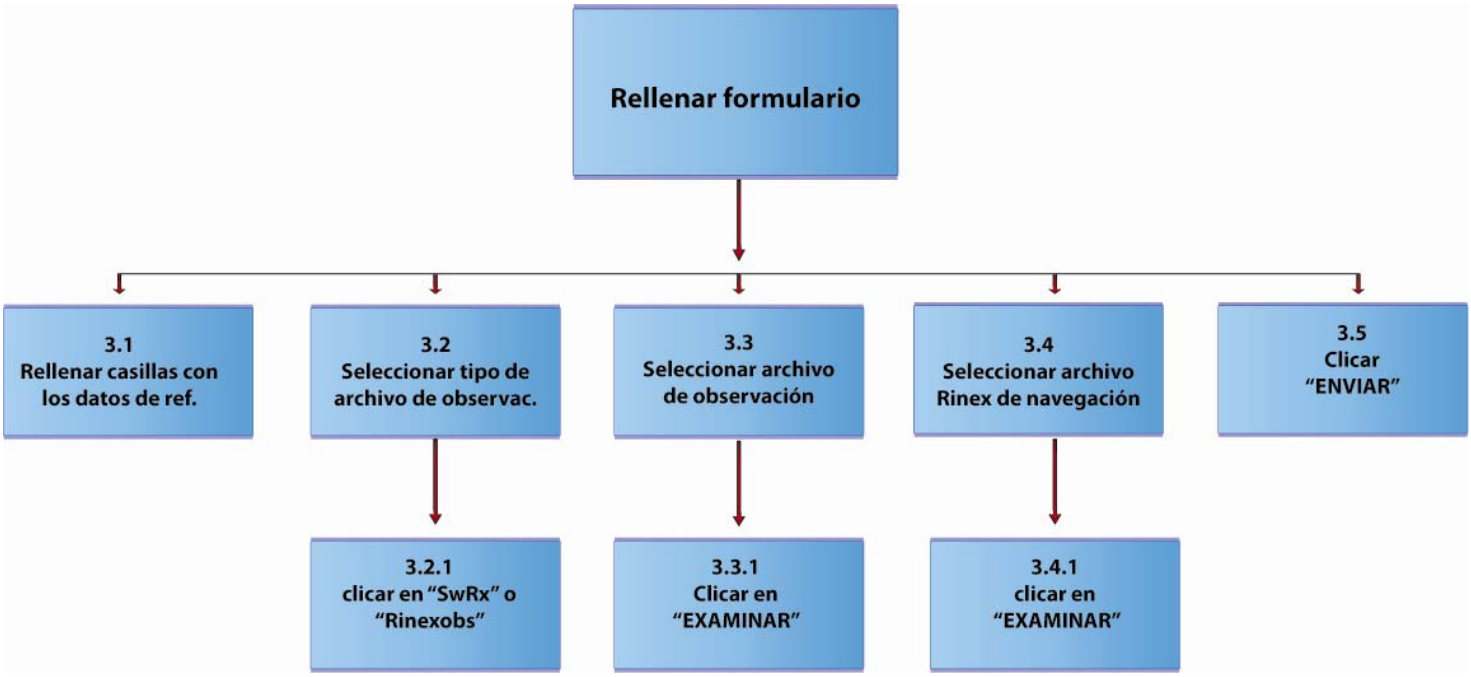

Fig. 5.6: Árbol de relleno de formulario

Tras haber rellenado el formulario, el usuario tan sólo deberá esperar a que la aplicación realice los cálculos pertinentes y los implemente en el mapa de Google Maps.

- Una vez el usuario visualiza el mapa, tiene varias opciones de redireccionamiento

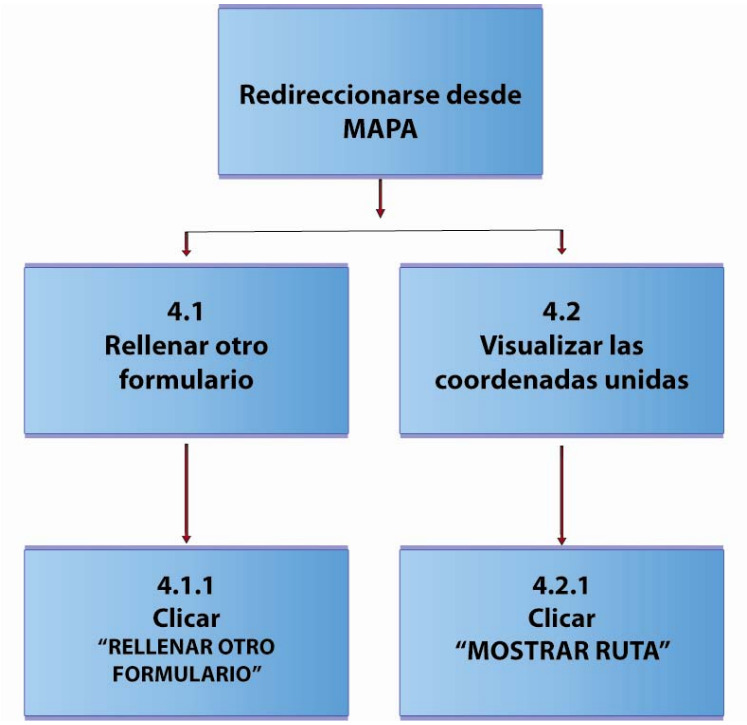

Fig. 5.7: Árbol de redireccionamiento desde MAPA inicial

- Si el usuario ha decidido visualizar la ruta, una vez se halle en esa página también podrá redireccionar

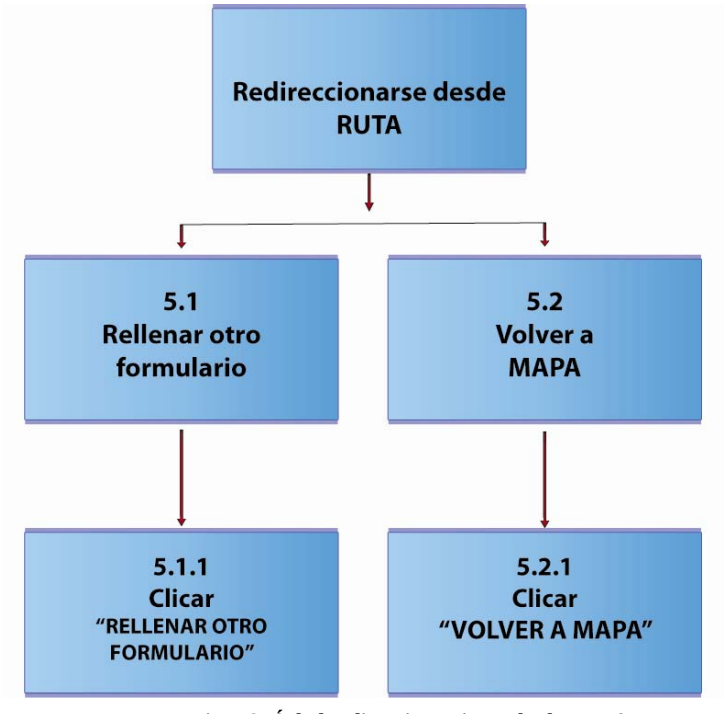

Fig. 5.8: Árbol redireccionamiento desde RUTA

- Tanto si se encuentra en la página que muestra el mapa inicial, como si se encuentra en la que muestra la ruta, el usuario podrá manipular la aplicación para obtener una representación a su gusto.

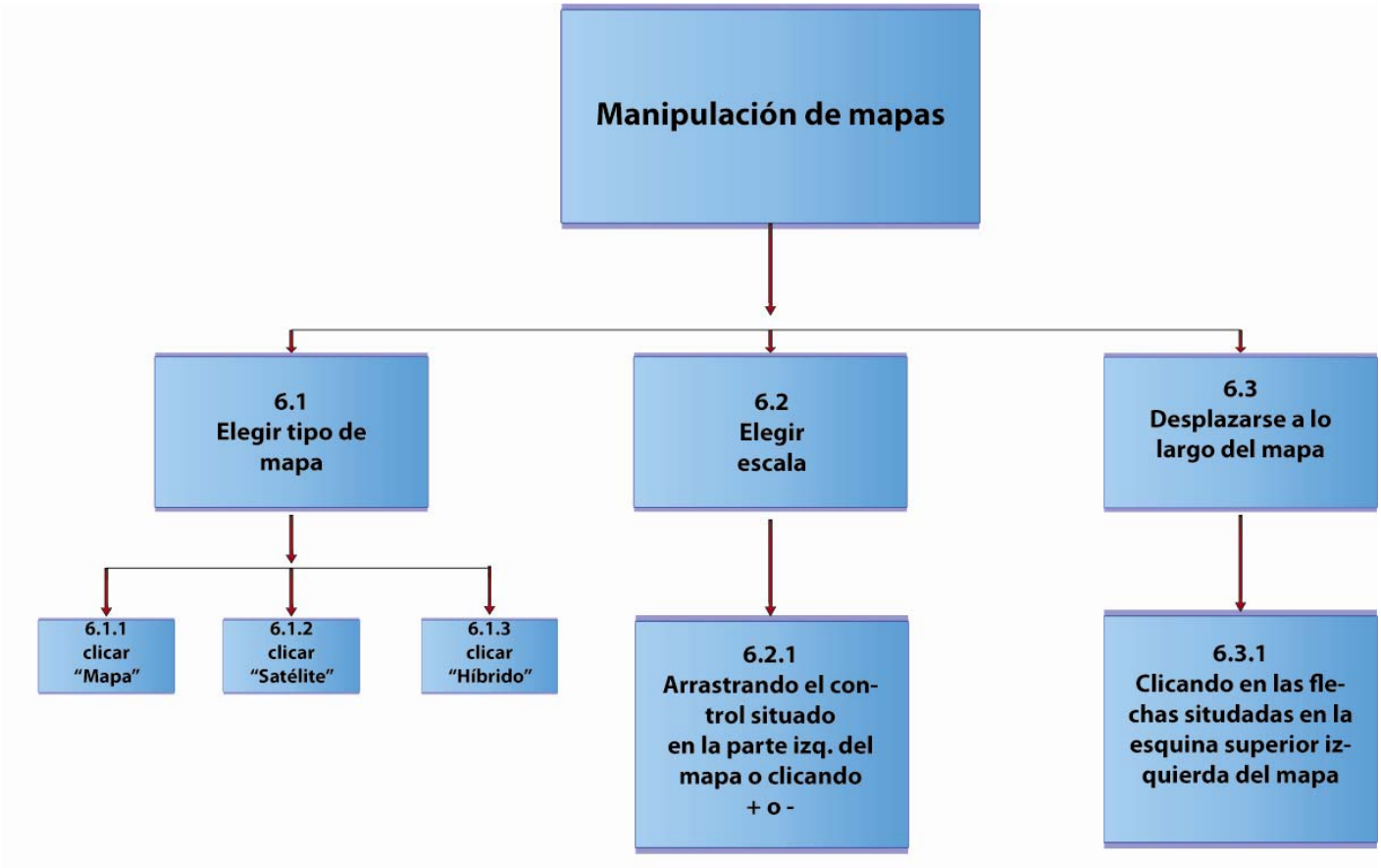

Fig. 5.9: Árbol de manipulación de mapas

CAPÍTULO 6:
ENTORNO DE TRABAJO

6.1 Sistema operativo Unix/Linux

Linux es un sistema operativo gratuito y de libre distribución inspirado en el sistema operativo Unix. La idea de su creación surgió del estudiante finlandés Linus Torvalds cuando modificaba Minix, una versión de Unix con limitadas funcionalidades.

Linus pidió ideas y colaboración a otros desarrolladores en un foro de Internet y el proyecto ha terminado aglutinando a miles de programadores de todo el mundo que todavía hoy siguen desarrollando Linux para hacer de él un sistema operativo mejor.

Linux hereda las principales ventajas de Unix: portabilidad a múltiples plataformas (desde Pc's y Mac's a estaciones de trabajo y superordenadores), flexibilidad a todos los niveles y una altísima configurabilidad; sin olvidarnos de la multitarea, multiusuario, gran capacidad para la gestión de redes, soporte de varios sistemas de ficheros y un largo etcétera. Sin embargo, no está pensado para ser fácil de usar por lo que todavía hoy, y a pesar de los grandes esfuerzos realizados en esta dirección, existen muchos usuarios reacios a considerarlo una alternativa real a Windows.

Linux cuenta con una amplia colección de programas y utilidades que lo acompañan: entorno gráfico X Window (compuesto de dos partes: el servidor X y el programa para la gestión de las ventanas), procesadores de texto como el Vi, compiladores para varios lenguajes de programación (C, C++, Ada, Fortran, etc…), entre otros.

El núcleo de Linux, al igual que la mayoría del software que lo complementa está amparado bajo los términos de la denominada GNU General Public License (GPL). Esta licencia permite básicamente al usuario final de un programa el poder usar, compartir, copiar y modificar con toda libertad el software que se acoge a esta licencia con ciertas restricciones. Algunas partes de este núcleo y otros programas que lo acompañan son propiedad de sus autores, con lo que Linux no es shareware ni freeware como se podría llegar a pensar. La actual versión es la 2.6.28 (publicada en diciembre del 2008) y su desarrollo continúa.

El primer concepto a conocer de Linux es el de distribución. Una distribución es un agrupamiento del núcleo del sistema operativo Linux y otra serie de aplicaciones de uso general o no tan general. En principio las empresas que desarrollan las

distribuciones de Linux están en su derecho al cobrar una cierta cantidad por el software que ofrecen, aunque en la mayor parte de las ocasiones se pueden conseguir desde Internet, de revistas o de amigos, siendo todas estas formas gratuitas y legales.

Las distribuciones más conocidas son RedHat, Debian, Slackware, SuSE, Knoppix y Mandrake. En este proyecto hemos desarrollado las simulaciones de la página Web en la distribución Ubuntu, basada en Debian, pues es el sistema operativo utilizado en los servidores de UAB.

6.2 Sistema operativo Ubuntu

Ubuntu es una distribución de Linux de tipo escritorio, basada en Debian. El proyecto se encuentra patrocinado por Canonical Ltda. y económicamente se sostiene con aportaciones de la misma empresa posee por dueño al sudafricano Mark Shuttleworth.

Ubuntu debe su nombre al movimiento homónimo encabezado por el obispo Desmond Tutu, el cual ganó el Premio Nobel de la Paz en 1984 por su lucha en contra del Apartheid en Sudáfrica. Mark Shuttleworth, el mecenas del proyecto, es sudafricano y por lo tanto se encontraba muy familiarizado con la corriente.

Tras ver similitudes entre los ideales de los Proyectos GNU, Debian y en general con el movimiento de software libre, decidió aprovechar la ocasión para difundir los ideales de Ubuntu.

Ubuntu esta basado en la distribución Debian pero a diferencia de esta, cuenta con una serie de lanzamientos muy fluida. Otra diferencia es el empleo del escritorio Gnome como escritorio principal. No obstante, esta distribución pretende no romper la compatibilidad con debian, de modo que puedan intercambiarse paquetes sin problemas. Por todo ello, se hace una distribución muy orientada al escritorio, pero con bastante estabilidad. En nuestro proyecto el sistema operativo Ubuntu, sirve de soporte del servidor Apache en el que hemos desarrollado la aplicación.

6.3 Servidor Apache

Apache es un servidor de red para el protocolo HTTP, elegido para poder funcionar como un proceso standalone, sin que eso solicite el apoyo de otras aplicaciones o directamente del usuario.

Para poder hacer esto, Apache, una vez que se haya iniciado, crea unos subprocesos, llamados children processes, para poder gestionar las solicitudes.

Los principales objetivos, ya cumplidos, de Apache eran lograr el servidor Web más rápido, más eficiente y con mayor funcionalidad desde un enfoque del Software Libre.

Apache está diseñado para ser un servidor Web potente y flexible que pueda funcionar en la más amplia variedad de plataformas y entornos. Las diferentes plataformas y los diferentes entornos, hacen que a menudo sean necesarias diferentes características o funcionalidades, o que una misma característica o funcionalidad sea implementada de diferente manera para obtener una mayor eficiencia.

Apache se ha adaptado siempre a una gran variedad de entornos a través de su diseño modular. Este diseño permite a los administradores de sitios Web elegir que características van a ser incluidas en el servidor seleccionando que módulos se van a cargar, ya sea al compilar o al ejecutar el servidor.

Al ser un servidor modular, implica que en el servidor básico se incluyen únicamente las funcionalidades más básicas. Otras funcionalidades se encuentran disponibles a través de

módulos que pueden ser cargados por Apache. Por defecto, durante la compilación se incluye en el servidor un juego de módulos base. Si el servidor se compila para usar carga dinámica de módulos, entonces los módulos pueden ser compilados por separado, e incluidos en cualquier momento usando la directiva LoadModule. En caso contrario, Apache deberá ser recompilado para agregar o eliminar módulos.

Apache es un servidor Web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos. Las principales características de este son:

- Implementa los últimos protocolos, aunque se base en el HTTP/1.1.
- Puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo y con la API de programación de módulos.
- Incentiva la realimentación de los usuarios, obteniendo nuevas ideas, informes de fallos y parches para la solución de los mismos.
- Independencia de plataforma, es decir, funciona sobre diferentes plataformas. Como por ejemplo,Unix, Linux, Vms, Win32, OS2, etc.

- Carga dinámica de los diferentes módulos.
- Posibilidad de personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Incluso, es posible configurarlo para que ejecute un determinado script cuando ocurra un error en concreto.
- Permite la creación de sitios Web dinámicos mediante: CGIs, SSI (Server Side Includes), lenguajes de scripting como PHP o JavaScript y, páginas JSP o Java.
- Permite la creación de ficheros Log a medida del administrador. Utiliza el formato CLF, Common Log Format, para la creación de los Logs de error.
- Soporta DSO, Dinamic Shared Object, para construir módulos que le den nuevas funcionalidades que son cargadas en tiempo de ejecución.

En nuestro proyecto, Apache hace de plataforma para el desarrollo de la aplicación, aparte de almacenar las diferentes páginas que la componen así como el ejecutable de MATLAB®, en él se almacenan los datos suministrados por el usuario, y se ejecuta la aplicación MATLAB® que realiza los cálculos de las coordenadas.

En el siguiente esquema podemos ver, como las páginas PHP son las encargadas de comunicarse con él, haciendo que en este se creen o se destruyan archivos, así como de que se ejecute la aplicación encargada de los cálculos.

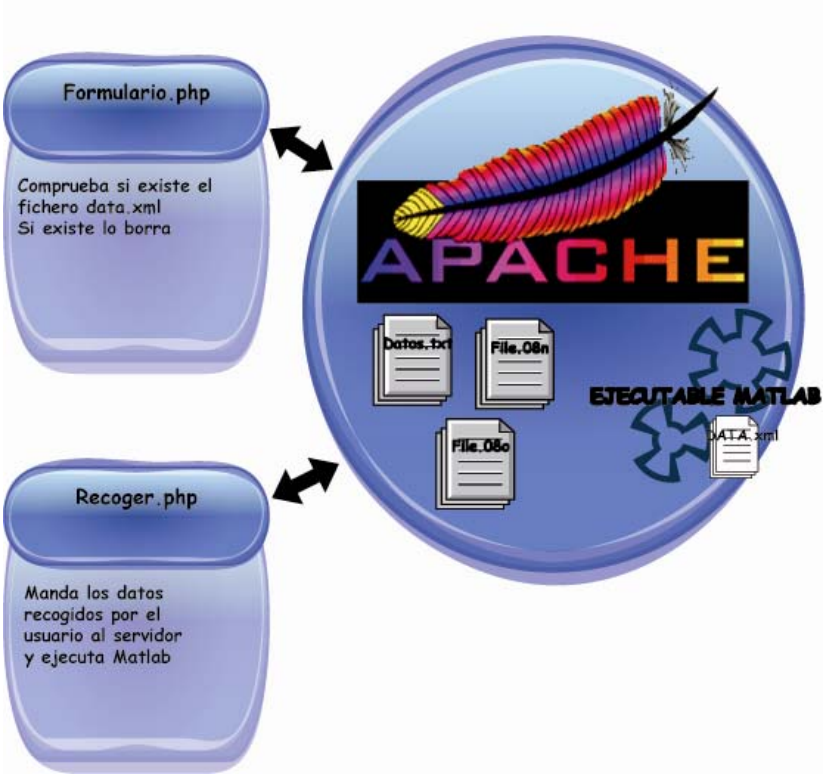

Fig. 6.1: Comunicación de las páginas .php con el servidor.

CAPÍTULO 7:

EJEMPLO PRÁCTICO DE LA APLICACIÓN

En este capítulo describiremos uno a uno los pasos que seguiría un usuario para hacer uso de nuestra aplicación basándonos en lo explicado en los capítulos anteriores.

En primer lugar, el usuario deberá acceder a nuestra aplicación conectándose al servidor en el que está colgada (para este proyecto hemos utilizado un servidor interno de la universidad): **http://158.109.69.251**

Una vez cargada la aplicación el usuario se encontrará con la página inicial (index.html):

Fig.:7.1: Imagen index.html

Tras clicar sobre "INICIAR" se encontrará en el formulario (previamente la aplicación habrá comprobado en si existe el archivo data.xml calculado en la sesión anterior en el servidor mediante formulario.php si es el caso lo borrará):

Fig. 7.2: Formulario.html

Para realizar este ejemplo una vez rellenados los parámetros de referencia, hemos elegido el tipo de archivo que vamos a subir (Rinex) y lo hemos seleccionado, una vez hecho esto hemos clicado en examinar y hemos elegido el archivo Rinex de Observación correspondiente.

En este proyecto como ya hemos comentado, el usuario puede elegir entre dos tipos de archivos de observación, el primero el SwRx es el manejado en la UAB por el grupo SPCOMNAV[5], el segundo es el archivo Rinex de observación. Para que la

[5] Ver Referencias

aplicación pueda interpretar correctamente los datos, en ambos casos los archivos deben tener un determinado formato.

En el caso de elegir el formato SwRx, el usuario deberá subir un archivo de la forma:

```
1 153953.000000 500 2 28800.546538 2721.975000 0.000000 0.000000 0.000000 0.000000
1 153953.000000 500 5 55856.414832 2710.849389 0.000000 0.000000 0.000000 0.000000
1 153953.000000 500 12 238067.861324 2609.999389 0.000000 0.000000 0.000000 0.000000
1 153953.000000 500 14 46420.703924 2062.950000 0.000000 0.000000 0.000000 0.000000
1 153953.000000 500 29 189635.405371 4597.275000 0.000000 0.000000 0.000000 0.000000
1 153953.000000 500 30 228266.121027 3821.999389 4780662.984396 176165.845056 4204526.208497 153953.244639
2 153954.000000 500 2 29319.090317 2721.975000 0.000000 0.000000 0.000000 0.000000
2 153954.000000 500 5 56369.262381 2710.849389 0.000000 0.000000 0.000000 0.000000
2 153954.000000 500 12 238568.187277 2585.000000 0.000000 0.000000 0.000000 0.000000
2 153954.000000 500 14 46821.570403 2062.950000 0.000000 0.000000 0.000000 0.000000
2 153954.000000 500 29 190511.675538 4597.275000 0.000000 0.000000 0.000000 0.000000
2 153954.000000 500 30 228992.932907 3797.000000 4780652.322864 176166.748919 4204499.986483 153954.285516
3 153955.000000 500 2 29831.554500 2721.975000 0.000000 0.000000 0.000000 0.000000
3 153955.000000 500 5 56880.692678 2710.849389 0.000000 0.000000 0.000000 0.000000
3 153955.000000 500 12 239047.410947 2609.999389 0.000000 0.000000 0.000000 0.000000
3 153955.000000 500 14 47209.543708 2062.950000 0.000000 0.000000 0.000000 0.000000
3 153955.000000 500 29 191389.203604 4597.275000 0.000000 0.000000 0.000000 0.000000
3 153955.000000 500 30 229718.291768 3797.000000 4780683.564156 176153.347134 4204553.367463 153955.197022
4 153956.000000 500 2 30352.732023 2721.975000 0.000000 0.000000 0.000000 0.000000
4 153956.000000 500 5 57393.679322 2685.850000 0.000000 0.000000 0.000000 0.000000
4 153956.000000 500 12 239545.415941 2585.000000 0.000000 0.000000 0.000000 0.000000
4 153956.000000 500 14 47592.788839 2062.950000 0.000000 0.000000 0.000000 0.000000
4 153956.000000 500 29 192266.271752 4597.275000 0.000000 0.000000 0.000000 0.000000
4 153956.000000 500 30 230446.497588 3797.000000 4780676.931686 176158.894389 4204546.370688 153956.225020
5 153957.000000 500 2 30870.251975 2721.975000 0.000000 0.000000 0.000000 0.000000
5 153957.000000 500 5 57907.434897 2685.850000 0.000000 0.000000 0.000000 0.000000
5 153957.000000 500 12 240045.151248 2585.000000 0.000000 0.000000 0.000000 0.000000
5 153957.000000 500 14 47993.281923 2062.950000 0.000000 0.000000 0.000000 0.000000
5 153957.000000 500 29 193141.171976 4597.275000 0.000000 0.000000 0.000000 0.000000
5 153957.000000 500 30 231172.245917 3797.000000 4780670.987739 176159.661821 4204526.323917 153957.249948
6 153958.000000 500 2 31385.455602 2721.975000 0.000000 0.000000 0.000000 0.000000
6 153958.000000 500 5 58420.905046 2685.850000 0.000000 0.000000 0.000000 0.000000
6 153958.000000 500 12 240542.395000 2585.000000 0.000000 0.000000 0.000000 0.000000
6 153958.000000 500 14 48381.311519 2062.950000 0.000000 0.000000 0.000000 0.000000
6 153958.000000 500 29 194016.731430 4597.275000 0.000000 0.000000 0.000000 0.000000
6 153958.000000 500 30 231898.043416 3797.000000 4780669.887188 176164.925064 4204515.206683 153958.274498
7 153959.000000 500 2 31901.481167 2721.975000 0.000000 0.000000 0.000000 0.000000
7 153959.000000 500 5 58932.731260 2685.850000 0.000000 0.000000 0.000000 0.000000
7 153959.000000 500 12 241027.505697 2585.000000 0.000000 0.000000 0.000000 0.000000
7 153959.000000 500 14 48762.824253 2062.950000 0.000000 0.000000 0.000000 0.000000
7 153959.000000 500 29 194890.404058 4597.275000 0.000000 0.000000 0.000000 0.000000
7 153959.000000 500 30 232619.786498 3797.000000 4780684.404145 176160.520847 4204546.415429 153959.225864
8 153960.000000 500 2 32418.616373 2721.975000 0.000000 0.000000 0.000000 0.000000
8 153960.000000 500 5 59444.748386 2685.850000 0.000000 0.000000 0.000000 0.000000
8 153960.000000 500 12 241518.149921 2585.000000 0.000000 0.000000 0.000000 0.000000
8 153960.000000 500 14 49166.947765 2062.950000 0.000000 0.000000 0.000000 0.000000
8 153960.000000 500 29 195765.682499 4597.275000 0.000000 0.000000 0.000000 0.000000
8 153960.000000 500 30 233338.280969 3797.000000 4780682.269123 176151.056452 4204541.985597 153960.219756
9 153961.000000 500 2 32937.292955 2721.975000 0.000000 0.000000 0.000000 0.000000
9 153961.000000 500 5 59957.350893 2685.850000 0.000000 0.000000 0.000000 0.000000
9 153961.000000 500 12 242018.042896 2585.000000 0.000000 0.000000 0.000000 0.000000
9 153961.000000 500 14 49551.480676 2062.950000 0.000000 0.000000 0.000000 0.000000
9 153961.000000 500 29 196639.592910 4597.275000 0.000000 0.000000 0.000000 0.000000
9 153961.000000 500 30 234056.811001 3797.000000 4780660.710952 176161.141412 4204506.051059 153961.291674
10 153962.000000 500 2 33455.620693 2721.975000 0.000000 0.000000 0.000000 0.000000
```

Fig. 7.3: Ejemplo de archivo SwRx

Que consta de 10 columnas que contienen:

Columna 1→ Número de Snapshots.

Columna 2→ TOW inicial.

Columna 3→ Tiempo de integración.

Columna 4→ número PRN del satélite.

Columna 5→ Pseudo rango.

Columna 6→ Error Doppler.

Columna 7→ Estimación coordenada X en metros.

Columna 8→ Estimación coordenada Y en metros.

Columna 9→ Estimación coordenada Z en metros.

Columna 10→TOW estimado.

En el caso de que quiera subir un archivo de observación con formato RINEX, la cabecera deberá ser de la forma:

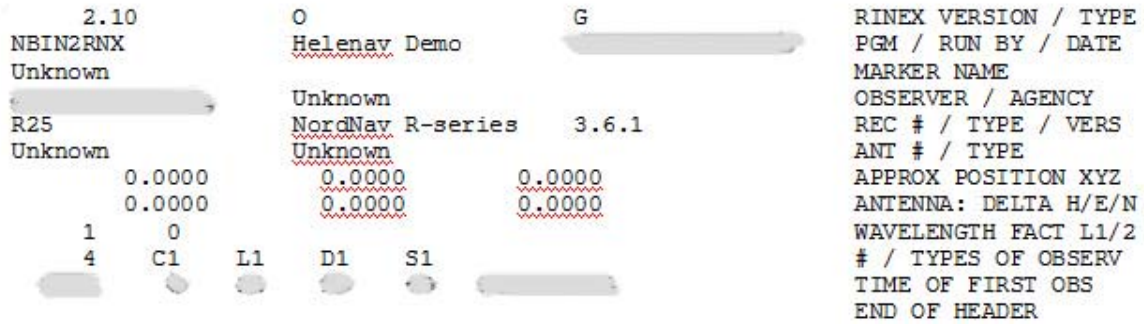

Fig. 7.4: Cabecera del Archivo Rinex de Observación

Por otra parte, para la lectura del archivo de Navegación, la aplicación no tiene ninguna limitación respecto al formato, por lo que el usuario podrá subir cualquier tipo de archivo Rinex de Navegación[6].

Una vez subidos los archivos al servidor, recoger.php se encargará de arrancar la aplicación de MATLAB® y comprobar si ha creado el archivo data.xml y nos redireccionará a Googlemaps.html para que Google Maps interprete las coordenadas y las dibuje en el mapa.

Este archivo data.xml será almacenado en el servidor, y tendrá la siguiente forma:

```
<markers>
<marker lat="   xx.yyy  " lng="  xx.yyy   "/>
.
.
.
.
<marker lat="   xx.yyy  " lng=" xx.yyy    "/>
</markers>
```

Fig. 7.5: Formato del fichero data.xml

[6] Ver Apéndice I.

Como ya hemos comentado, este archivo será el que Google Maps interprete para implementar las diferentes coordenadas:

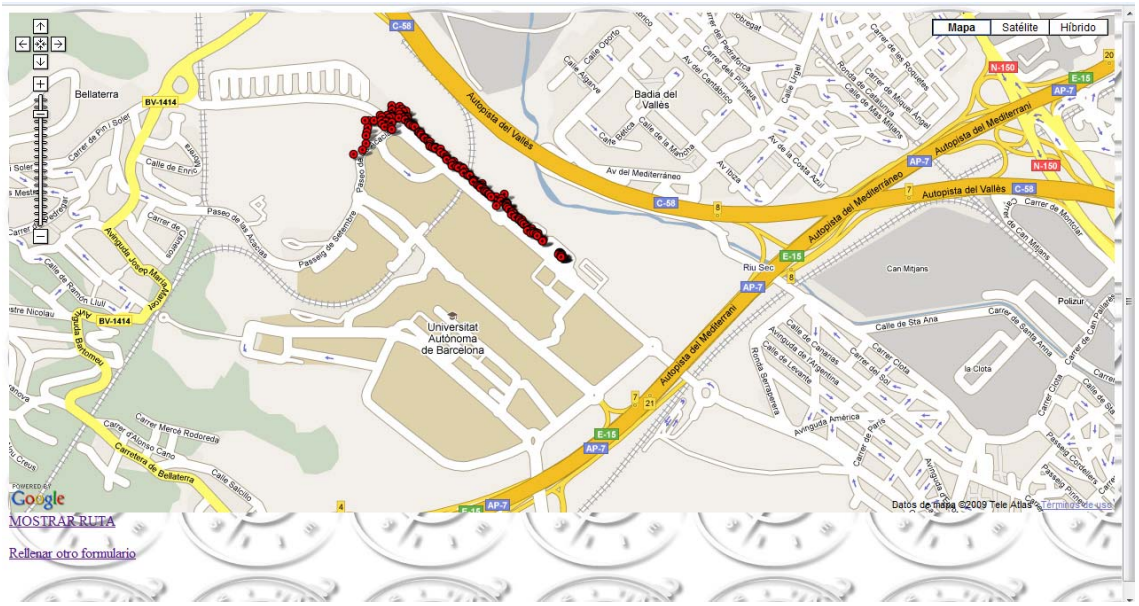

Fig.7.6: Mapa de Google Maps con las coordenadas dibujadas

De esta manera el usuario podrá visualizar el resultado final de las operaciones realizadas en nuestra aplicación. Una vez en esta pantalla, podrá ejecutar las operaciones que estime oportunas para observar el resultado.

CAPÍTULO 8:
CONCLUSIONES

En este proyecto, hemos desarrollado una aplicación Web que permite visualizar en Google Maps los datos de posicionamiento de una sesión GPS.

Para hacer uso de ella, tal y como hemos comentado en capítulos anteriores, el usuario una vez rellenados los datos de referencia de la sesión, deberá subir al servidor los dos archivos característicos de la sesión, por un lado el fichero Rinex de Navegación y por otro el fichero de observación. Éste último deberá ser de una forma específica tal y como hemos explicado en el capítulo 7 en el que mostramos un ejemplo práctico de la aplicación, o bien un Rinex de Observación con las particularidades descritas, o un fichero pre-procesado por el usuario para que tome la forma del utilizado por el grupo SPCOMNAV.

Uno de los objetivos cumplidos en este proyecto ha sido el de la sencillez de manejo, puesto que una vez haya aportado los datos y especificado los parámetros de referencia básicos, el usuario tan sólo deberá esperar a que sean procesados.

Es aquí cuando toma forma la razón de ser de nuestro proyecto, el momento en el que la aplicación convierte en protagonista el script de MATLAB®. Como ya comentamos en el capítulo 3 de esta memoria, este Script es el encargado de procesar los datos aportados por el usuario, y con ello calcular y plasmar en un fichero de datos, las coordenadas capturadas en la sesión GPS. La singularidad del mismo, recae en el hecho de no ejecutarse de manera local, sino remotamente en el servidor Linux. De esta forma la máquina utilizada por el usuario para acceder a la aplicación es liberada prácticamente de la totalidad de la capacidad de cálculo que requiere el procesado de datos, con lo que permitimos que el usuario pueda acceder a nuestra aplicación desde un dispositivo sencillo (por ejemplo, un teléfono móvil o una PDA). Esta parte, ha sido la que nos ha llevado más tiempo desarrollar, aun disponiendo de la gran mayoría de las funciones creadas (gracias al grupo SPCOMNAV), el hecho de que el software debiera ejecutarse en el servidor y de que con el fin de facilitar el uso de la aplicación, el usuario no tomara parte alguna en esta fase, ha hecho de ésta todo un reto. El poder desarrollarla, nos ha supuesto en primer lugar estudiar la manera de compilar el código MATLAB® en el servidor para la creación del script, y en segundo lugar, realizar un estudio exhaustivo del funcionamiento de la ejecución remota en el servidor, así como,

del lenguaje utilizado en la aplicación Web para poder desencadenarla desde allí al comprobar que los datos se encuentren en él.

Puesto que hemos entendido que la mejor manera y la más útil para cualquier usuario es mostrar los resultados de forma gráfica, en este proyecto hemos hecho uso de la aplicación gratuita creada por Google: Google Maps, insertándola en nuestra WEB para que de esta manera, gracias a las funciones que incorpora el API de Google Maps, interprete el archivo de datos creado por MATLAB e implemente las posiciones en el mapa.

Una vez completada esta parte del proyecto, decidimos incorporar a la WEB una serie de añadidos, para que con ellos el usuario pueda ajustar el resultado a la visualización que más le convenga e incluso marcar la ruta seguida mediante la unión de las posiciones implementadas.

Como resultado final de este proyecto, hemos obtenido una herramienta útil y dinámica para la lectura e interpretación de archivos de posicionamiento, que permite al usuario de la misma comprobar la captura realizada por su receptor GPS rápidamente, tan sólo con los conocimientos necesarios para la navegación en Red.

Referencias

[1] Grupo SPCOMNAV (Signal Porcessing for Comunications and Navigation), Departamento de Telecomunicación e ingeniería de Sistemas de la UAB. Proyecto DINGPOS.

[2] Ahmed El-Rabbany, "Introdution to GPS: the Global Positioning System". Artech House, 2006.

[3] Elliotr D. Kaplan, Cristopher J. Hegarty, "Understanding GPS: principles and applications".

[4] David A. Veri, Mark Malseed**, "**La Historia de Google : los secretos del mayor éxito empresarial, mediático y tecnológico de nuestro tiempo". La Esfera de los Libros, 2006.

[5] Jaume Hernández Pajares, Manuel Juan Zornosa, J. Miguel Sanz Subirana, "Tratamiento de datos GPS. Prácticas de laboratorio", Ediciones UPC, 1997.

[6] Web del observatorio nacional de la marina estadounidense: http://tycho.usno.navy.mil/gps.html

[7] Archivo de la base de datos de la NASA sobre el sistema GPS: http://msl.jpl.nasa.gov/Programs/gps.html

[8] Página oficial del API de Google Maps: http://code.google.com/intl/es-ES/apis/maps/

Apéndice: Contenido del CD

El CD contiene tres apéndices diferentes:

- Apéndice I: Contiene la explicación de los formatos RINEX utilizados en este proyecto, el fichero Rinex de Observación y el fichero Rinex de Navegación.

- Apéndice II: Contiene el código de las funciones Matlab diseñadas por nosotros:
  - MainWEB.m
  - Read_rin_ob.m
  - Read_uab_arc.m
- Apéndice III: Contiene el código de las páginas Web implementadas
  - Index.html
  - Formulario.php
  - Formulario.html
  - Recoger.php
  - GoogleMaps.html
  - Líneas.html

Resum:

En aquesta memòria s'explica el desenvolupament d'una eina útil que permet a l'usuari visualitzar en l'aplicació Google Maps les dades de posicionament captats en una sessió GPS.
En aquest projecte, hem dissenyat una aplicació Web en la qual recollim les dades ingressades per l'usuari mitjançant un formulari. Un cop emmagatzemades aquestes dades en el servidor, la nostra eina hi executa l'aplicació encarregada del càlcul de les posicions. Aquesta és un script desenvolupat en MATLAB®, que s'encarrega d'interpretar les dades subministrades per l'usuari, amb les quals es poden calcular les coordenades captades pel receptor GPS. Una vegada calculades, el software les emmagatzema en el servidor, en un arxiu .xml, que serà el que posteriorment interpretarà Google Maps gràcies al seu API.
D'aquesta manera, l'usuari obtindrà el resultat visual de la sessió GPS que hagi decidit carregar sense necessitat des disposar de cap software específic per a la interpretació i el càlcul de les dades que hi ha capturat.

Resumen:

En esta memoria se explica el desarrollo de una herramienta útil que permite al usuario visualizar en la aplicación Google Maps los datos de posicionamiento captados en una sesión GPS.
En este proyecto, hemos diseñado una aplicación Web en la que recogemos los datos ingresados por el usuario mediante un formulario. Una vez almacenados estos datos en el servidor, nuestra herramienta ejecuta en el mismo la aplicación encargada del cálculo de las posiciones. Ésta no es otra, que un script desarrollado en MATLAB®, que se encarga de interpretar los datos suministrados por el usuario, para poder calcular con ellos las coordenadas captadas por el receptor GPS. Una vez calculadas, éstas son almacenadas en el servidor por dicho software en un archivo .xml, que será el que posteriormente interprete Google Maps gracias a su API.
De esta forma el usuario obtendrá el resultado visual de la sesión GPS que haya decidido cargar, sin necesidad de disponer de ningún software específico para la interpretación y cálculo de los datos capturados en ella.

Summary:

This report covers the development of a useful tool that allows the user to visualize in the application Google Maps the positioning data received through a GPS session.
Within this project, a Web application was designed in order to compile the data gathered by the user at the abovementioned GPS session through a form. Once the data is stored in the server, the tool executes the application in charge of the positioning calculation. The application consists on a script developed in MATLAB®, which interprets the data supplied by the user in order to calculate the coordinates received by the GPS receiver. Once calculated, these coordinates are stored in the server as an .xml file by the same software, which is the one which will be later read by Google Maps thanks to its API.
As a result the user will obtain the visual output of the GPS session he has decided to load, with no need to use any additional software for the interpretation and calculation of the data obtained by the receiver