Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

IMPLEMENTACIÓ D'UNA APLICACIÓ WEB PER REPRESENTAR I DESCARREGAR RUTES DE MUNTANYA

Memòria del Projecte Final de Carrera
d'Enginyeria en Informàtica
realitzat per
Francesc Tomàs Vila
i dirigit per
Pilar Gómez Sánchez
Bellaterra, 26 de Gener de 2010

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Pilar Gómez Sánchez

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat
sota la seva direcció per en Francesc Tomàs Vila

I per tal que consti firma la present.

Signat:

Bellaterra, 26 de Gener de 2010

Universitat Autònoma de Barcelona

Escola Tècnica Superior d'Enginyeria

El sotasignat, Marc Subirà

de l'empresa, MAL DE ALTURA S.C.P.

CERTIFICA

Que el treball a què correspon aquesta memòria ha estat realitzat en l'empresa sota la seva supervisió mitjançant conveni firmat amb la Universitat Autònoma de Barcelona.

Així mateix, l'empresa en té coneixement i dóna el vist-i-plau al contingut que es detalla en aquesta memòria.

Signat:

Bellaterra, 26 de Gener de 2010

Taula de continguts

1 Introducció

El projecte es realitzarà a l'empresa **Mal de Altura S.C.P**, en concret en una de les seves activitats econòmiques, que és la pàgina web Pirineos3000 (www.pirineos3000.com).

1.1 Contextualització del problema

Pirineos3000 és una pàgina web que fa més de deu anys que té presència a Internet, i que dóna serveis als seus visitants i als seus usuaris de temes bàsicament relacionats amb l'alta muntanya, ja siguin d'informació d'ascensions a diferents cims dels Pirineus, expedicions a d'altres països, informació de refugis, parcs naturals, meteorologia, etc... També disposa d'una llibreria de muntanya, així com d'una revista pròpia que s'edita trimestralment.

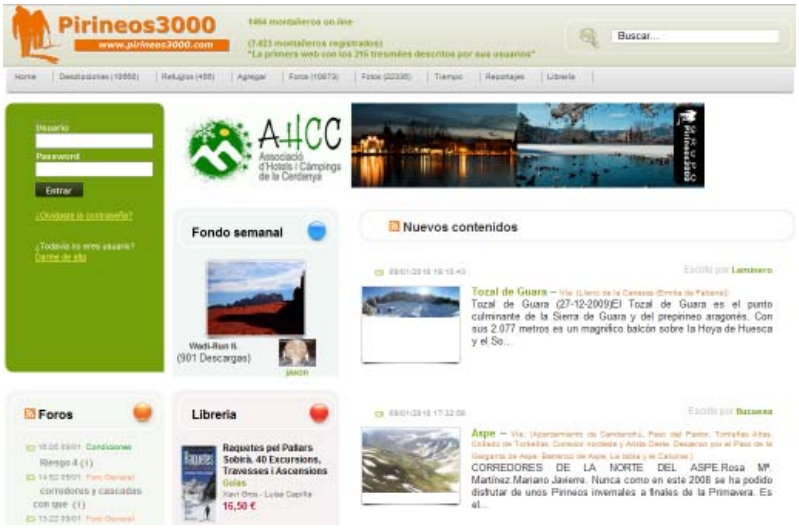

Figura 1. Pàgina principal de www.pirineos3000.com

Un dels grans valors de la pàgina web són els més de 7.000 usuaris que té registrats i que han fet que actualment es disposi de més de 10.000 descripcions de diferents ascensions dels Pirineus i d'altres serralades (inclosos tots els 215 cims més alts de 3000 metres).

També disposa d'una gran base de dades amb més de 22.000 fotografies i d'uns fòrums i un xat molt participatius.

Una de les moltes activitats que poden dur a terme els usuaris registrats de la pàgina és introduir a la base de dades descripcions d'ascensions, rutes entre diferents muntanyes, etc. Aquest fet implica introduir totes les dades necessàries de localització i característiques del recorregut en particular (temps de recorregut, meteorologia, data, dificultat, fotografies...).

Es va pensar que seria una millora per a la pàgina web que també es poguessin penjar i mostrar les rutes i punts d'interès registrats amb els *receptors GPS* [1] que duen molts usuaris a la muntanya. D'aquesta manera el visitant de la pàgina podria veure i descarregar-se aquestes rutes, i seguir-les posteriorment amb el seu receptor GPS.

Aquesta representació hauria de ser mitjançant l'aplicació Google Maps, i s'haurien de poder mostrar rutes provinents de diferents marques de *GPS* [2] , així com les rutes que provenen de programes de cartografia. També s'hauria de mostrar automàticament el perfil de la ruta i algunes dades estadístiques calculades automàticament a partir d'aquestes rutes, i poder descarregar-les en formats diferents de fitxers GPS.

En aquest context es desenvoluparà, doncs, el projecte que estem definint.

1.2 Motivacions

A continuació es detallen algunes de les motivacions que han dut a realitzar el Projecte Final de Carrera de la manera que estem plantejant.

1.2.1 Tecnològiques

Les pàgines web comercials i de serveis són les primeres que van evolucionant i adaptant-se a les noves tecnologies, sobretot si és segur que d'aquesta manera podran oferir més serveis, optimitzar l'eficiència de la pàgina actual o millorar la seguretat o la presentació de les dades.

Google Maps [3] com a aplicació de visualització de mapes cartogràfics

La informació geogràfica i cartogràfica dels mapes augmenten considerablement les capacitats de moltes aplicacions. L'accessibilitat a aquesta informació ha estat possible gràcies a la companyia Google, que l'ha posat gratuïtament a disposició dels seus usuaris a partir de l'any 2005.

A més a més de la capacitat de mostrar mapes topogràfics, de relleu, i imatges per satèl·lit, Google Maps ofereix una *API* [4] que permet desenvolupar i perfeccionar els mapes per defecte, augmentant així el nombre d'aplicacions que es poden dur a terme sobre Google Maps. Aquesta API està implementada en llenguatge Javascript, que és un llenguatge interpretat capaç de ser executat pels navegadors web.

Així doncs, entre moltes altres possibilitats, podem dibuixar línies, polígons, afegir marques i imatges sobreposades, i podem interactuar amb l'usuari. Google Maps treballa amb *AJAX* [5] , i permet realitzar totes aquestes operacions asíncronament amb el servidor i sense necessitat de recarregar la pàgina.

Convé remarcar que malgrat aquesta informació ens arriba mitjançant el protocol *HTTP* [6] (utilitzat per la majoria de navegadors web), la informació es mostra directament integrada a Google Maps, fent el programa independent del navegador que estem utilitzant, i incrementant la interacció entre informació, mapa i usuari.

Malgrat que Google ofereix de forma pública la seva API pel desenvolupament d'aplicacions basades en Google Maps, avui en dia encara no és habitual de veure gaires aplicacions de caràcter cartogràfic basades en ella.

Aquest Projecte Final de Carrera ens pot ser útil per comprovar fins a quin punt són viables aquestes aplicacions.

GpsBabel, eina de conversió de fitxers GPS

Són molts els dispositius GPS que existeixen en el mercat, i molts els programes capaços de tractar amb informació cartogràfica. Cadascun d'aquests elements té les seves particularitats i guarden la seva informació en formats de fitxer diferents.

Per compartir aquesta informació entre els diferents usuaris d'Internet, és necessària una aplicació que ens permeti realitzar conversions entre els diferents formats de fitxers. En aquest context va néixer l'any 2002 el programa lliure **GPSBabel**, on milers d'usuaris arreu del món van actualitzant dia a dia les seves capacitats adaptant-lo als nous dispositius i millorant-ne l'eficiència.

En aquest projecte farem ús d'aquest programa, de manera que els usuaris puguin compartir fitxers de rutes de muntanya sense preocupar-se del format en què s'emmagatzemen.

El navegador web com a interfície general de l'aplicació

Actualment el navegador web és una eina fonamental, present en la majoria de sistemes operatius, i en el qual estan acostumats pràcticament tots els usuaris de sistemes informàtics.

Les grans bases dels navegadors són la comunicació amb sistemes remots basats en el protocol TCP/IP (fonamentalment HTTP) i la possibilitat de mostrar continguts multimèdia, com imatges, àudio, vídeos, ...

En els últims anys han anat apareixen un gran nombre d'aplicacions basades en el navegador web que no només són documents estàtics, com poden ser clients de correu electrònic, reproductors de vídeo, interfícies de xarxes socials, etc.

La majoria dels productes desenvolupats per Google es basen en el navegador web. El fet de situar una nova aplicació sobre un navegador web suposa menys esforç per part de l'usuari, ja que el situa en un entorn més conegut del que seria una aplicació independent.

A més a més, molts desenvolupadors estan apostant fort per aquesta filosofia, fent que els navegadors siguin cada vegada plataformes més refinades i amb majors prestacions.

1.2.2 Econòmiques

Google Maps com a font d'informació cartogràfica gratuïta

Existeixen nombroses empreses dedicades a l'obtenció d'informació geogràfica i cartogràfica. Un exemple serien les empreses dedicades a la comercialització de dispositius de posicionament d'automòbils sobre la xarxa de carreteres mitjançant el sistema GPS. Per l'ús d'aquests dispositius, és necessari prèviament disposar de la documentació cartogràfica en la que l'usuari pugui ubicar el seu vehicle. Generalment aquestes empreses no ofereixen les seves bases de dades públicament, sinó que exigeixen la contractació dels seus serveis.

Google és una excepció, ja que ofereix gratuïtament als usuaris la documentació geogràfica de la que disposa a través de Google Maps. De fet, no només ofereix els mapes, sinó també les eines necessàries per manipular-los.

Sense dubte, l'elecció de Google Maps com a font d'informació cartogràfica, la utilització de diversos programes lliures i llenguatges de programació lliures i amb documentació gratuïta, suposa una gran avantatge de cara a la viabilitat econòmica del projecte.

Augment dels ingressos

És clar que com més serveis ofereixi una pàgina web més visites i més usuaris obtindrà, i es podran traduir en més ingressos provinents de la publicitat, o dels serveis paral·lels que ofereixi la pàgina web.

El cas de Pirineos3000 no és cap excepció, i l'increment de visites i d'usuaris es pot veure traduït també en un increment d'ingressos provinents de la publicitat i també de la botiga virtual.

1.3 Definició del problema

Un cop tenim definides a grans trets les característiques del projecte, anem a veure quines seran les principals línies de treball.

Integració a l'arquitectura del servidor

Pel que fa referència a la programació de la pàgina web, sempre ens haurem d'adaptar a la part actualment ja operativa (a nivell de continguts i disseny), de manera que haurem de seguir el seu model d'arquitectura de tres nivells: Façana, Servei i Accés a dades.

Conversió i emmagatzematge dels fitxers enviats pels usuaris

Serà necessari convertir els fitxers enviats pels usuaris de la pàgina Pirineos3000, a un format estàndard, *GPX* [7] (GPS Exchange format), de manera que des de la nostra aplicació es puguin llegir tots els punts del recorregut, i emmagatzemar-los de manera adient a la base de dades.

Adquisició de dades d'altitud

Alguns dispositius GPS no enregistren dades d'altitud, i en la nostra aplicació haurem de mostrar el perfil de la ruta, per tant, necessitem aquestes dades. En aquests casos el què farem és connectar-nos a un *Webservice* [8] d'ús lliure que tingui a la seva base de dades d'altura de tots els punts del món, i demanarem l'altura de cada punt que necessitem.

Representació de la ruta a través de Google Maps

Un cop familiaritzats amb la API de Google Maps, haurem de llegir les rutes de la base de dades per mostrar a través del mapa el recorregut corresponent mitjançant línies sobreposades al mapa de relleu.

Representació del perfil de la ruta

Donada una ruta, haurem de mostrar el seu perfil, és a dir, un gràfic lineal on a l'eix X mostrarem els quilòmetres recorreguts i a l'eix Y hi mostrarem l'altura de cada punt.

Càlcul d'estadístiques

Per cada ruta, també haurem de calcular algunes estadístiques de forma automàtica. Per exemple, voldrem saber la distància total recorreguda o el desnivell positiu i negatiu.

Descàrrega dels fitxers a múltiples formats

Els usuaris podran descarregar totes les rutes que es mostrin en la nostra aplicació, i ho podran fer en diferents formats de fitxer GPS. Imaginem ja un desplegable on poder escollir el format i un botó *Descarregar* per baixar-se el fitxer.

1.4 Objectius

L'objectiu principal del projecte és realitzar la implementació d'un sistema per carregar, representar i descarregar fitxers de rutes de muntanya a través d'Internet, dins de l'entorn ja operatiu d'aquesta pàgina.

Per assolir l'objectiu fixat, és necessari que el projecte cobreixi les següents funcionalitats:

1. Enviament de fitxers de rutes en diferents formats
2. Tractament dels fitxers i emmagatzematge a la base de dades
3. Representació de les rutes amb l'aplicació Google Maps
4. Representació gràfica del perfil del recorregut
5. Càlcul i mostra d'estadístiques calculades automàticament
6. Descàrrega de la ruta en diferents formats.

Un dels requisits fonamentals és que l'aplicació s'adapti a la pàgina web, tant a nivell de disseny com d'arquitectura del software.

1.5 Viabilitat del projecte

Passem a veure la viabilitat d'aquest projecte, per comprovar si és possible realitzar-lo amb els recursos i temps disponibles.

1.5.1 Recursos humans

A nivell de recursos humans, per la realització del projecte és necessària una persona, amb el treball aproximat de les hores que s'estableixen per a la realització del Projecte Final de Carrera en Enginyeria Informàtica (450 hores en total).

1.5.2 Hardware / Software

Pel que fa referència al Hardware i Software necessari per poder realitzar el projecte, s'instal·larà una rèplica de l'arquitectura del servidor de l'empresa en un portàtil assequible (Intel Core 2 T6400 amb 4GB de RAM) del qual ja es disposa.

Per fer això instal·larem els següents programes, que els podem trobar lliurement a la xarxa:

- Java 1.6
- Eclipse Platform Version: 3.4.0 GANYMEDE
- MySQL 5.1.30, amb Query Browser i Administrator
- Putty o similar
- Client FTP
- GpsBabel 1.3.6

A més a més, a la xarxa es pot trobar abundant documentació sobre tots aquests programes i dels diferents llenguatges de programació.

1.5.3 Tecnologia

D'altra banda, també s'ha de cercar documentació de les diferents tecnologies a utilitzar, ja siguin llenguatges de programació (php, ajax, java, mysql...), algorismes de tractament de rutes, conversors de coordenades, per tal de fer la implementació el més efectiva possible. *Google Maps* també ofereix diferents eines pròpies de desenvolupament, les que permetran interactuar amb el mapa i que estan ben documentades. Són eines que es poden utilitzar lliurement.

Pel que fa a la implementació, també es posaran en pràctica els coneixements adquirits en diverses assignatures de la carrera, com Enginyeria del Software (I i II), Algorismes i Programació, Llenguatges de Programació, Bases de Dades (I i II)... de manera que també disposo de certs coneixements i d'una documentació on poder consultar.

Per tant doncs, amb tota aquesta informació, podem dir que la realització del projecte, i el compliment dels seus objectius, és viable.

1.6 Estat de l'art

Són moltes les empreses i particulars que ja tenen integrades en les seves pàgines web el Google Maps, o alguna aplicació equivalent. Però la majoria d'elles sols l'utilitzen per indicar una adreça o algun punt, i no fan ús de les capacitats de l'aplicació, sobretot pel que fa a la incrustació de més elements i la interacció amb l'usuari.

És cert que en els darrers mesos han anat apareixent pàgines que fan més ús d'aquestes possibilitats i fins i tot s'han arribat a crear jocs a nivell mundial corrent a temps real sobre Google Maps.

Un exemple d'aquestes aplicacions el podem trobar a l'agost del 2009 amb el naixement del *Monopoly City Streets*, una versió del tradicional joc del Monopoly però amb el suport de Google Maps, on es juga a temps real amb milers de contrincants arreu del món.

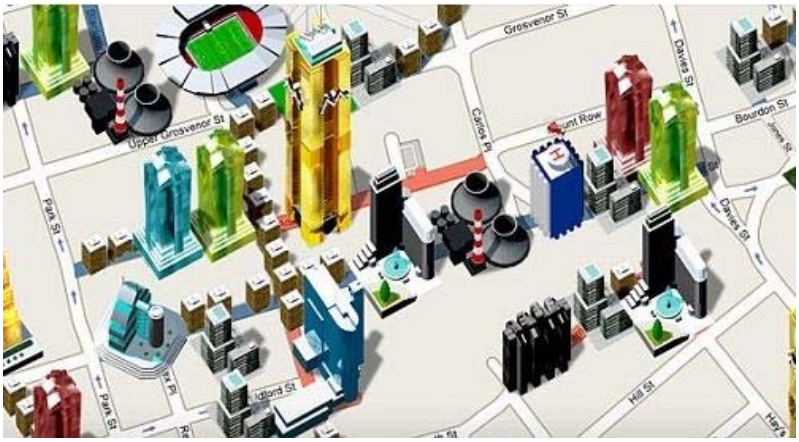

Figura 2. Imatge arbitrària de l'aplicació Monopoly City Streets

També hi ha molts usuaris arreu del món, aficionats a les rutes de muntanya i a la tecnologia GPS que fan ús del programa **GpsBabel**, el conversor de formats de fitxers GPS i fitxers de programes de cartografia. Però l'usen a nivell particular i per convertir algun fitxer en concret.

La majoria de programes que converteixen fitxers i tracten amb mapes s'han d'instal·lar en l'ordinador del client (GpsVisualizer, CompeGps, OziExplorer...) i són força limitats pel que fa a la importació i exportació de fitxers.

Hi ha algunes aplicacions que tracten amb rutes i realitzen algunes conversions de fitxers amb interfície web. Un exemple d'aquestes aplicacions podria ser www.inlinemap.net , una pàgina alemanya que permet crear i modificar rutes de patinadors sobre rodes, tot i que és limitada en aspectes d'importació i exportació de diferents formats de fitxers.

Figura 3. Imatge de la pàgina web www.inlinemap.net

També trobem la web catalana www.senderisme.tk , que descriu algunes rutes de muntanya de Catalunya, i que té la capacitat de representar recorreguts sobre Google Maps, sempre i quan s'hagi enviat el fitxer en un format estàndard. Així i tot, no s'enllacen les representacions amb la base de dades d'ascensions, i les descàrregues són molt limitades

Figura 4. Imatge de la pàgina web www.senderisme.tk

També trobem moltes pàgines on es poden descarregar fitxers de rutes, però en la majoria dels casos ens trobem limitat el format del fitxer de càrrega i descàrrega, i són poques les que representen les rutes enviades, i encara menys, les que mostren el perfil del recorregut.

1.7 Planificació

Veiem la planificació aproximada de tot el procés d'elaboració del projecte.

Juliol**:** Primer contacte amb els responsables de l'empresa. Creació del document "*Anàlisi funcional del projecte*". Realització del l'informe previ del Projecte. Instal·lació de tot el Software necessari per a poder treballar. Començar la part teòrica de la memòria final del projecte.

Setembre: *Recerca i documentació.* Analitzar què s'ha fet ja sobre aquest tema i de quina manera, veure diferències entre diferents implementacions i documentar-se sobre els diferents algorismes i llenguatges de programació utilitzats. Familiaritzar-se amb l'entorn de treball i la API de Google Maps.

Octubre-Novembre-Desembre**:** *Implementació*. Part gruixida del projecte. Elaboració del programa que ha de complir amb els objectius del projecte i amb l'anàlisi funcional. Es realitzarà seguint tots els passos del procés d'enginyeria del software.

Finals Desembre fins Gener**:** *Presentació.* Acabar memòria final del projecte, que ja s'haurà anat redactant a mesura que vagi avançant el projecte, i preparar la presentació.

1.8 Organització de la memòria

Aquest document està organitzat en quatre apartats.

En el primer apartat hem explicat la **introducció**. En la introducció es descriu el problema a resoldre contextualitzat en l'entorn de la pàgina web de l'empresa. Se'n detallen els objectius, les motivacions que han dut a la realització del projecte per part de l'empresa, l'estudi de la viabilitat a nivell de recursos humans, software, hardware i llenguatges de programació, s'estudia què s'ha fet i què no en l'actualitat referent a la tecnologia i a la temàtica del treball, i s'especifica la planificació temporal aproximada del projecte a realitzar.

En la segona part es descriu detalladament com s'ha desenvolupat **el projecte**, des de la fase inicial fins a la instal·lació. S'especifiquen els requeriments de l'empresa, es descriu l'arquitectura del software de l'empresa i s'expliquen els diferents mòduls del programa i tots els passos que s'han seguit per desenvolupar-lo.

En la tercera part veiem els **resultats** del projecte, és a dir, el funcionament de la nova aplicació a través de la pàgina web.

La quarta part fa referència a les **conclusions** un cop realitzat el projecte. Es valoren els resultats contrastant-los amb els objectius inicials, se'n fa una valoració personal, i es detallen possibles ampliacions i línies de continuació relacionades amb la tasca realitzada.

Per últim, tenim les **referències** que ens han ajudat durant el procés d'elaboració del projecte. Al llarg del document ens hi referirem mitjançant lletres entre claudàtors.

Com a **annexes**, tenim el glossari, on hi trobem les definicions de les sigles que s'han utilitzat al llarg d'aquest document. Ens hi referirem mitjançant un número entre claudàtors, corresponent al número d'entrada en el glossari.

2 Descripció del projecte

Per descriure el projecte que s'ha realitzat ho farem seguint el model en cascada d'Enginyeria del Software, és a dir, seguint una seqüència lineal de tots els passos. A diferència d'altres projectes, aquest model és vàlid per aquest projecte ja que és un projecte rígid, es poden especificar correctament els requeriments i les eines a utilitzar.

Primerament farem **l'anàlisi dels requeriments**. L'anàlisi i l'especificació de requeriments descriuen el comportament esperat del software un cop desenvolupat. Gran part de l'èxit d'un projecte de software esdevindrà en la identificació de les necessitats del negoci.

És important assenyalar que en aquesta etapa s'ha de consensuar tot el que es requereixi del sistema i serà això el que se seguirà en les següents etapes, sense que es puguin requerir nous resultats a la meitat del procés d'elaboració del software.

A continuació, després d'haver consensuat els requeriments, passarem a la fase de **disseny**. En aquesta etapa es descomposa i s'organitza el sistema en elements que es puguin elaborar per separat, que els anomenarem mòduls i es representaran mitjançant diagrames.

Seguidament, ja podem passar a la fase d'**implementació**, és a dir, de la programació. En aquí s'implementa el codi font, fent ús de prototips, així com assajos i proves per corregir errors. En aquest document no s'entrarà molt en detall en el codi pròpiament dit, sinó que es descriuran les funcions principals dels diferents mòduls del codi font.

Per últim veurem els **resultats,** amb el suport d'imatges, de tots els passos que han de fer els usuaris per utilitzar l'aplicació.

2.1 Anàlisi de requeriments

L'aplicació a realitzar, com ja s'ha comentat anteriorment, s'ha d'integrar dins la pàgina web Pirineos3000 (www.pirineos3000.com), respectant els continguts i el disseny actuals.

Les funcionalitats, que ha de complir l'aplicació, independentment dels mecanismes interns per dur-les a terme, són:

Enviament del fitxer de ruta

L'usuari de la pàgina web, en el moment de penjar una ascensió o ruta i omplir el formulari amb totes les seves característiques (descripció, dades tècniques, fotografies, etc) – que actualment ja es fa -, també podrà enviar opcionalment un fitxer de text corresponent a la ruta que ha enregistrat el seu receptor GPS (*Global Position System*).

El procés de traspassar l'arxiu del GPS a l'ordinador el durà a terme l'usuari pel seu compte.

Aquest fitxer podrà ser, com a mínim, d'un dels formats de fitxer següents: GARMIN, MAGELLAN, OZIEXPLORER, GOOGLE EARTH. En cas de no ser d'un d'aquests formats, l'aplicació no tractarà la ruta i es demanarà que es torni a enviar.

Representació de la ruta

Els usuaris que visitin la pàgina web (registrats o no), quan consultin la descripció d'una de les rutes o ascensió, si el creador de la ruta havia penjat també un fitxer correcte del GPS, en la pantalla de descripció se li mostrarà un enllaç per veure el recorregut de la ruta i el perfil.

Fent clic en aquest enllaç, s'obrirà una nova finestra on es mostrarà el recorregut de la ruta sobre l'aplicació Google Maps, que serà un requadre amb un mapa de relleu on la ruta es mostrarà mitjançant línies que es ressaltin sobre el mapa. Inicialment la ruta es mostrarà completa al mapa amb la possibilitat d'apropar-se o allunyar-se i de desplaçar-se pel mapa.

A més a més, també s'hauran de mostrar les següents estadístiques calculades automàticament: distància total del recorregut, desnivell positiu/negatiu, i altura màxima/mínima.

Representació del perfil de la ruta

Al mateix temps que es mostra la ruta sobre Google Maps, també s'haurà de mostrar el perfil de la ruta, és a dir, un gràfic de dues dimensions on a l'eix X es mostrin els quilòmetres recorreguts i l'eix Y l'altura en cada moment.

Un exemple de perfil seria el següent:

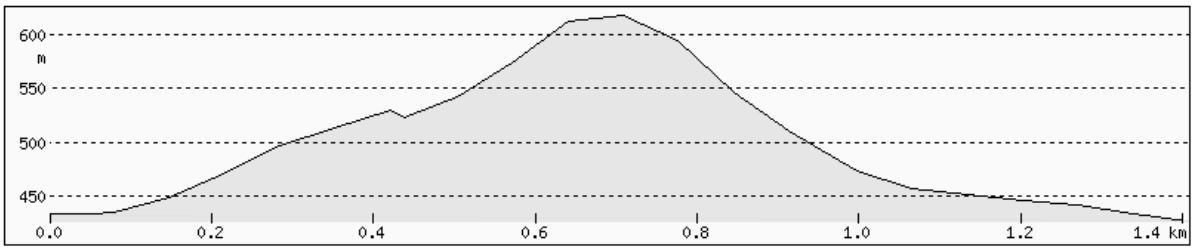

Figura 5. Exemple de perfil de ruta de muntanya

Descàrrega de la ruta

En aquesta mateixa pantalla, a la part inferior es mostrarà un desplegable amb els tipus de formats en què es podrà descarregar la ruta actual. Prement en un d'aquests formats i després en un botó *Descargar ruta* se li haurà de descarregar la ruta representada amb el format escollit, que com a mínim, han de ser els següents: GARMIN, MAGELLAN, OZIEXPLORER, GOOGLE EARTH.

El procés de traspassar el fitxer de text de la ruta al seu receptor GPS, el durà a terme l'usuari.

Aspectes tècnics

Les rutes importades es rectificaran en cas necessari (a vegades el GPS pot registrar punts llunyans a la ruta per error). Un cop validades s'emmagatzemaran en una base de dades interna del servidor, de manera que serà més fàcil i ràpid mostrar-les posteriorment en el mapa.

Les **dades de l'altura** en cada punt de la ruta poden ser les que ha registrat el receptor GPS (si aquest ho permet) o bé les dades de l'altura aconseguides a través d'algun Webservice (en cas que el GPS no permeti enregistrar altituds). Evidentment, en el càlcul de distàncies s'ha de tenir present que no és el mateix el quilometratge mesurat horitzontalment (en un GPS sense altituds, per exemple) que el real havent ascendit i descendit pel terreny.

La implementació de totes aquestes funcionalitats haurà de seguir el **model d'arquitectura** del servidor de l'empresa.

Per últim, hi haurà contacte quan es requereixi amb els responsables de l'empresa per aclarir possibles temes tècnics del model d'arquitectura del servidor, o també per decidir conjuntament qüestions que afectin a l'empresa o al disseny i les funcionalitats de l'aplicació.

2.2 Arquitectura del software

L'aplicació a realitzar ha de seguir el model d'arquitectura del software que segueix fins ara la web ja operativa.

És una arquitectura amb tres nivells (Façana, Servei, Accés a dades) on cada nivell només es pot comunicar amb els veïns de la següent forma:

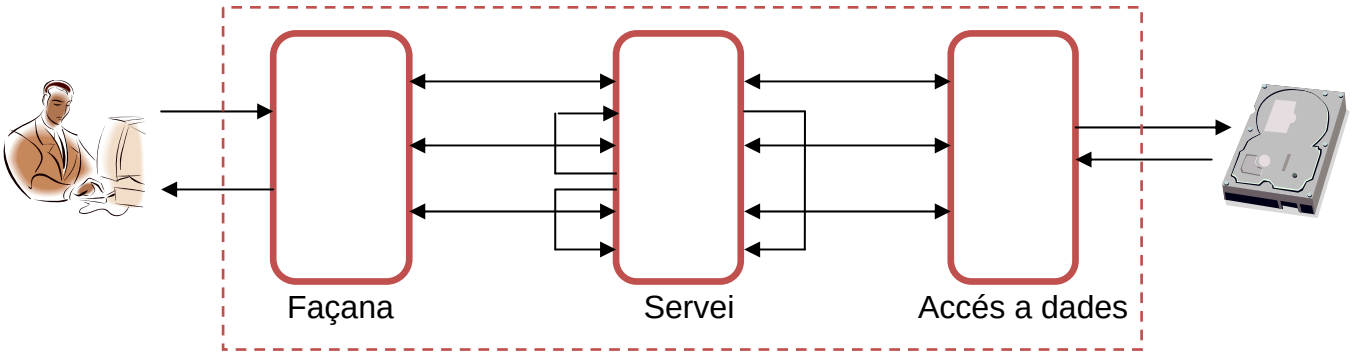

Figura 6. Arquitectura de l'aplicació

Descrivim breument cada capa de l'arquitectura:

Façana

La capa de presentació (Façana) consisteix en la vista que té l'usuari de l'aplicació. Rep peticions dels usuaris (des del navegador web, per exemple), i les processa fent les crides necessàries a diferents serveis de la capa de Servei. Des de la capa de servei es dóna una resposta a aquestes peticions (amb dades, per exemple) i la capa de façana en fa una presentació dels resultats que els retornarà a l'usuari (ja sigui en forma de pàgina web, en forma de fitxer, etc).

Servei

La capa de servei enllaça la capa de presentació amb la capa d'accés a les dades. Rep peticions de la capa de presentació, que les ha de processar ja sigui fent crides a d'altres serveis, o comunicant-se amb les dades a través de la capa d'accés a les dades.

Accés a dades (Data Access Object)

Els objectes de la capa de servei necessiten accedir a les dades, ja sigui per lectura o escriptura. Aquestes poden estar emmagatzemades de diferents formes. A la lògica de negoci no li ha d'importar com estiguin emmagatzemades ni com es realitzin els accessos, d'això se n'ocupa la capa d'accés a les dades.

2.3 Tecnologia utilitzada

Descriurem breument les tecnologies principals, els programes i els llenguatges de programació que utilitzarem en la implementació del projecte.

2.3.1 Servlets de Java

La web actual està desenvolupada principalment amb *Java* [AJ,MJ,JC] , un llenguatge de programació orientat a objectes desenvolupat a principis dels anys 90 per Sun Miscrosystems. Pràcticament tot el Java de Sun avui és software lliure sota llicència GNU GPL [9].

En concret, l'aplicació està implementada mitjançant Servlets. Un Servlet és una classe de java que s'executa al servidor (d'aqui ve el nom de *servlet*, en contraposició amb el de *applet*, que s'executa al client), i que típicament es crida des d'una aplicació web per obtindre un resultat.

A diferència dels *CGI's* [10] , que va ser una de les primeres aplicacions a permetre la comunicació entre un programa client i un servidor, els Servlets no s'executen com un procés separat, i ens evitem de crear un procés nou a cada petició. Com que una sola instància del Servlet s'encarrega de respondre a totes les peticions concurrentment, també ens permet estalviar memòria i gestionar fàcilment dades persistents.

Els Servlets estàn escrits en java i tenen una API ben estandaritzada, com a conseqüència d'això, són portables a qualsevol servidor web.

Aquesta és l'estructura bàsica d'un Servlet. En aquest cas és un exemple molt senzill que crea una pàgina amb el text "Hola món!"

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HelloWorld extends HttpServlet {
          public void doGet(HttpServletRequest request,
                            HttpServletResponse response)
              throws ServletException, IOException {
            PrintWriter out = response.getWriter();
            out.println("Hola món!");
          }
        }
```

Figura 7. Estructura básica d'un Servlet

Veiem que el Servlet *HelloWorld* hereta de la classe *HttpServlet* i a més sobreesciu el mètode *doGet* (també podria sobreescriure el mètode doPost, o ambdós), de manera que les dades són enviades mitjançant GET (provinents d'un formulari, per exemple).

Aquests mètodes reben dos arguments, un *HttpServletRequest* i *HttpServletResponse*. El *HttpServletRequest* té mètodes que ens permeten trobar informació entrant (com dades d'un formulari). El *HttpServletResponse* té mètodes que ens permeten especificar línies de resposta HTTP, capçaleres de resposta

(Content-Type, Set-Cookie...), i ens permet obtenir un *PrintWriter* usat per enviar la sortida al client.

Observem també que hem d'importar les classes dels paquetes *java.io* (per *PrintWriter*, etc.), *javax.servlet* (per *HttpServlet*, etc.), i *javax.servlet.http* (per *HttpServletRequest* i *HttpServletResponse*). I totes les classes dels paquets que ens facin falta per programar el Servlet.

Els Servlets es fan servir com a capa intermitja entre la petició del client HTTP i la base de dades o aplicació resident al servidor HTTP. En la majoria dels casos l'objectiu és la generació de pàgines web, i permeten crear pàgines dinàmiques construïdes a partir de codi Java mitjançant les *Java Server Pages*.

2.3.2 Java Server Pages (JSP)

La tecnologia *Java Server Pages* (JSP) és una especificació oberta i gratuïta desenvolupada per Sun Microsystems com a alternativa a *Active Server Pages* (*ASP* [11]) de Microsoft. L'extensió de fitxer d'una pàgina JSP es *.jsp*. JSP que ens permet mesclar HTML estàtic amb HTML generat dinàmicament. La part dinàmica està escrita en Java (a diferència de ASP, que està escrita en Visual Basic), i és portable a altres sistemes operatius i servidors web. És cert que JSP no ens aporta res que no pogués fer el mateix Servlet, però és molt més convenient escriure i modificar HTML que no pas des del programa en Java haver de tenir nombroses sentències tipus "*println()*". A més a més, ens permet separar el disseny del contingut, amb totes les avantatges que això suposa.

Per executar les pàgines JSP, necessitem que el servidor web tingui un contenidor web que compleixi les especificacions de JSP i de Servlet. Aquest contenidor maneja l'execució de les pàgines JSP i dels Servlets que s'executen en el servidor. En el nostre cas, el servidor web és Tomcat, una completa implementació de referència per les especificacions Java Servlet i JSP.

Veiem com funciona una petició i resposta d'un Servlet amb pàgines JSP en la nostra aplicació.

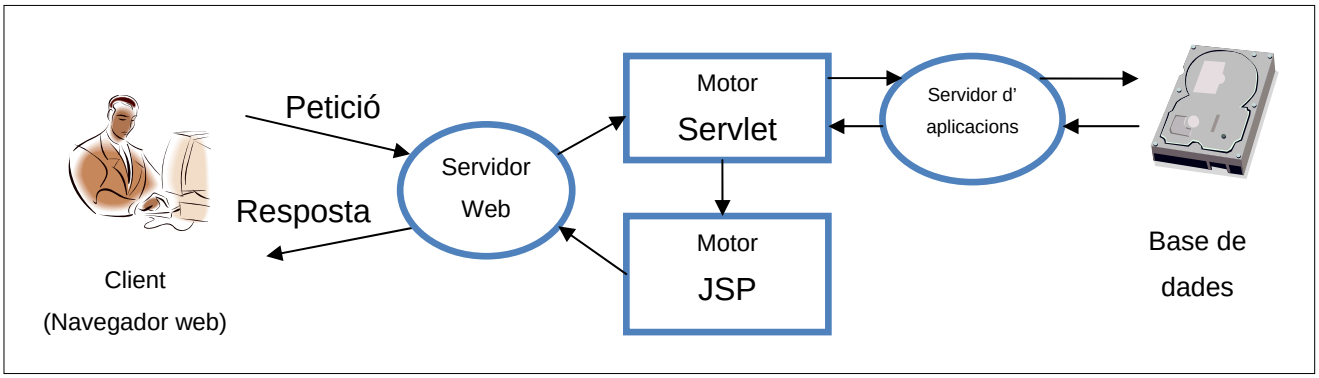

Figura 8. Flux de petició/resposta en la crida d'un Servlet des del navegador web

2.3.3 API de Google Maps

Google ofereix gratuïtament una API de Google Maps, basada en Javascript amb l'objectiu de manipular el mapa mostrat. Per utilitzar aquesta API, hem de carregar la biblioteca remotament. És important remarcar que Google imposa la norma de registrar cada servidor que usarà la API, que implica allotjar el codi en un servidor web públic.

Per usar la API haurem de definir una instància del mapa en la pàgina JSP, i a partir d'aquí cridar a funcions que afectaran a la instància.

Podem dividir el conjunt de funcions en dos grups: funcions per manipular la vista i funcions per definir elements en el mapa.

Les funcions per manipular la vista ens permetran accions com modificar el punt central del mapa, el nivell de zoom, els controls a mostrar, etc.

Pel que fa a les funcions per definir elements, cada vegada que es vulgui afegir un element al mapa, es crea una instància d'aquest, i posteriorment es defineixen els seus atributs, que dependran del tipus d'element que estem tractant. També podrem modificar posteriorment els atributs d'un element per actualitzar-los.

La API també ofereix la possibilitat de definir events, permetent l'execució de certes funcions quan succeeixen algunes accions, com poden ser clicar sobre el mapa, moure el cursor, etc...

En la nostra aplicació, utilitzarem la API de Google Maps per manipular els mapes que mostraran les rutes penjades pels usuaris de la pàgina web.

2.3.4 API de Google Visualization

La *API de Google Visualization* permet accedir a diferents fonts de dades estructurades que es poden visualitzar entre una àmplia gamma d'opcions.

Mitjançant aquesta aplicació es pot mostrar la informació emmagatzemada a qualsevol magatzem de dades connectat al web. Es poden crear informes, gràfics, així com analitzar i mostrar les dades a través d'aplicacions disponibles.

La *API de Google Visualization* també proporciona una plataforma que es pot utilitzar per crear, compartir i reutilitzar visualitzacions desenvolupades per la comunitat de desenvolupadors en la seva totalitat.

En la nostra aplicació, utilitzarem la *API de Google Visualization* per mostrar el perfil de la ruta, és a dir, un gràfic lineal de dues dimensions, on hi representarem les dades que tenim emmagatzemades, en aquest cas, cada punt de la ruta i la seva altitud.

2.3.5 Mysql

Per emmagatzemar les dades ho farem a través de Mysql, un sistema gestor de base de dades relacional, i que s'ofereix sota llicència GNU GPL .

Mysql és utilitzat en molts desenvolupaments web i actualment té milions d'usuaris arreu del món. Permet instal·lar-se en la majoria de sistemes operatius, i entre les seves característiques principals destaquem la velocitat en les operacions i la seguretat referent a la connectivitat.

Per interactuar amb la base de dades utilitzarem el programa **Navicat**, que ens proporciona una interfície gràfica per gestionar la base de dades.

2.3.6 Eclipse

La interfície que fem servir per manejar tot el codi Java i JSP és *Eclipse* [ME] , un entorn de desenvolupament integrat de codi obert i multi-plataforma.

Aquesta plataforma, típicament ha estat utilitzada per desenvolupar entorns de desenvolupament integrat (*IDE* [12]), com l'IDE de Java anomenat *Java Development Kit* (*JDK* [13]) i el compilador (ECJ) que s'entrega com a part d'Eclipse.

Eclipse va ser desenvolupat originàriament per IBM. Actualment és desenvolupat per la *Fundació Eclipse*, una organització independent sense ànim de lucre que fomenta una comunitat de codi obert i un conjunt de productes complementaris, capacitats i serveis.

En el mateix entorn, disposem d'un depurador i d'una consola de missatges que ens ajudaran en el desenvolupament dels diferents mòduls.

Amb la instal·lació d'un servidor *Apache Tomcat* [14] , podrem anar veient i comprovant els resultats a través del navegador.

2.4 Disseny

Un cop fet l'anàlisi de requeriments i coneguda l'arquitectura del software i la tecnologia amb què treballarem, ja podem dissenyar la nostra aplicació.

L'aplicació necessita emmagatzemar i disposar de dades, fet pel qual també mostrarem el disseny de la base de dades necessari per fer córrer l'aplicació.

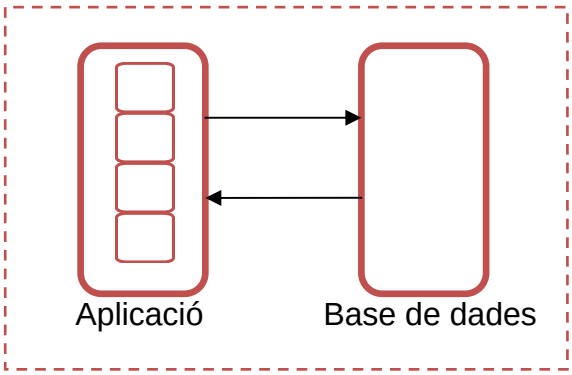

Figura 9. Comunicació de l'aplicació amb la base de dades

2.4.1 Disseny de l'aplicació

Dividirem l'aplicació en mòduls i submòduls independents, de manera que ens permetrà implementar-los de manera separada, i provar-los de manera separada, amb tots els avantatges que això suposa, sobretot a l'hora de trobar i solucionar problemes.

Representarem aquests mòduls mitjançant un diagrama de flux, on es veuen representats tots els passos que realitza un usuari de la web Pirineos3000 des que vol enviar un fitxer de ruta fins que se'l pot tornar a descarregar un altre usuari.

Mòdul d'importació

Aquest mòdul avarca des del moment en què l'usuari de la pàgina web envia el fitxer a través del formulari d'alta o modificació d'ascensió de la pàgina web, fins que la ruta que contenia aquest fitxer és emmagatzemada a la base de dades.

Figura 10. Flux del mòdul d'importació

Enviament del fitxer GPS: La web ja disposa actualment d'un formulari per omplir en el moment de donar d'alta o modificar una ruta o una ascensió. S'afegirà un camp de *fitxer GPS* on l'usuari escollirà qualsevol fitxer del seu equip. Un cop validat el formulari, el fitxer s'enviarà al servidor i es guardarà en un directori temporal, a punt per a ser tractat.

Conversió a format estàndard GPX: D'entre els molts formats de fitxer GPS, existeix el format GPX (GPS eXchange), un format estàndard definit i pensat per a transferir dades GPS entre aplicacions. A través de la crida externa a l'aplicació GpsBabel, amb els paràmetres adequats, convertirem el fitxer original al format GPX, deixant-lo a punt per ser interpretat per la nostra aplicació. Un cop convertit a aquest, ja podrem esborrar el fitxer original.

Parsejador del fitxer i Rectificació: la nostra aplicació serà capaç de parsejar el fitxer GPX, és a dir, llegir totes les dades que aquest contingui de manera ordenada (punts de recorregut i waypoints). En aquest procés s'eliminaran els punts que siguin molt llunyans del punt anterior (degut a algun possible error que hi pugui haver en el fitxer GPS), i també es buscarà l'altura de cada punt en un Webservice en cas que el receptor GPS no hagués enregistrat dades d'altura. En aquest procés també anirem calculant algunes dades que les utilitzarem posteriorment: la distància total recorreguda, el desnivell màxim/mínim i l'altura màxima/mínima.

Emmagatzematge a la base de dades: un cop interpretades totes les dades del fitxer, guardarem la informació a la base de dades de rutes, i enllaçarem la ruta emmagatzemada amb l'ascensió que estàvem descrivint.

Mòdul de presentació

El mòdul de presentació inclou tot el què fa referència a la representació del recorregut de les rutes, del perfil, i de les estadístiques calculades automàticament.

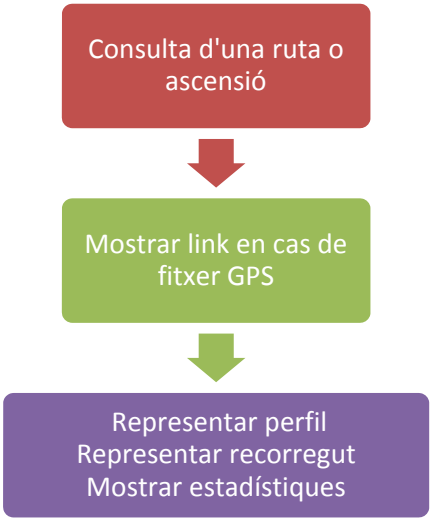

Figura 11. Flux del mòdul de presentació

Consulta d'una ruta o ascensió: Per veure el funcionament de la nostra aplicació, prèviament l'usuari ha d'anar, a través de la pàgina web, a veure la descripció d'una ruta o d'una ascensió.

Mostrar link en cas de fitxer GPS: En cas que l'ascensió tingui una entrada de ruta a la base de dades, mostrarem un link *Ver ruta y perfil*, d'altra manera mostrarem un text *Sin fichero GPS*. Aquest link ens obrirà una nova pantalla de tamany gran en el navegador (sense les barres superiors ni inferiors, de manera que en el moment de mostrar el mapa tindrem més espai a la pantalla).

Representar perfil: A la base de dades ja disposem de la informació de distàncies recorregudes i altura, per tant podem representar el perfil de la ruta. Per fer això, dibuixarem un gràfic lineal a través de l'aplicació Google Visualization.

Representar recorregut: Lògicament, també disposem de les coordenades de cada punt del recorregut, i que ens permetrà enllaçar-los mitjançant línies sobre un mapa de relleu de Google Maps. També hi mostrarem els waypoints que tingui emmagatzemada aquesta ruta, i un punt inicial que representi l'inici del recorregut (d'altra manera costaria apreciar des de quin extrem comença el recorregut).

Mostrar estadístiques: També es mostraran les dades calculades durant el procés d'importació del fitxer: la distància total recorreguda, el desnivell màxim/mínim i l'altura màxima/mínima.

Mòdul de descàrrega

El mòdul de descàrrega avarca el procés des de que l'usuari selecciona un format de fitxer GPS fins que rep l'arxiu en el seu ordinador.

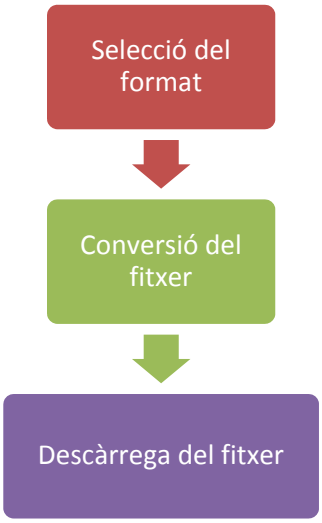

Figura 12. Flux del mòdul de descàrrega

Selecció del format: a la base de dades tindrem emmagatzemats tots els formats de fitxer que la nostra aplicació és capaç de convertir. D'aquesta manera, en la pantalla de representar la ruta, hi mostrarem un desplegable amb tots aquests formats, i al costat un botó *Descargar Ruta*.

Conversió del fitxer: Recordem que el fitxer original de la ruta el tenim emmagatzemat en format GPX. Ara doncs, l'haurem de convertir cap al format escollit per l'usuari, també mitjançant el programa GpsBabel.

Descàrrega del fitxer: en l'aplicació, hi haurà un servei que permetrà a l'usuari descarregar-se aquest nou fitxer a través del navegador web. Un cop descarregat, l'eliminarem del servidor, per no ocupar espai innecessari.

2.4.2 Disseny de la base de dades

Diagrama Entitat-Relació

Per representar el model conceptual de com estaran emmagatzemades les dades per que fa a les rutes i els seus punts del recorregut, ho fem mitjançant el diagrama Entitat-Relació.

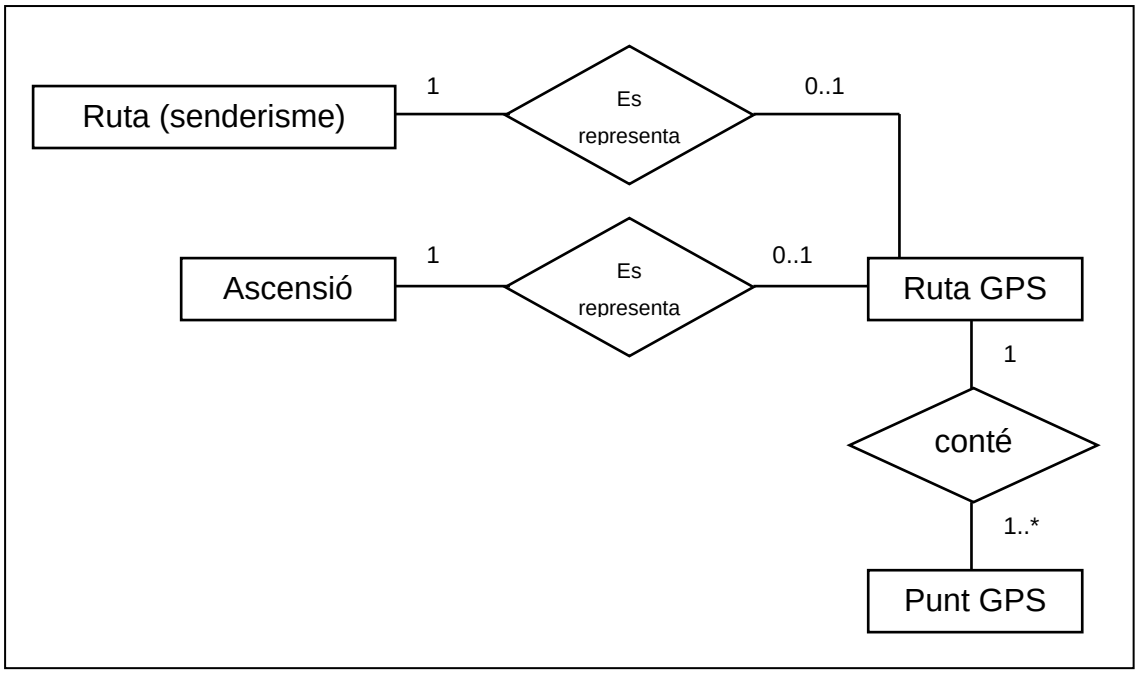

Figura 13. Diagrama Entitat-Relació de la base de dades

Taules de la base de dades

Veiem ara com serà la taula de **rutes GPS** i en descrivim alguns dels seus atributs més importants:

Name	Type	Length	Decimals	Allow Null
V_IDRUTAGPS	int	7	0	☐
V_NPUNTOS	int	11	0	☑
V_NWAYPOINTS	int	11	0	☑
V_MINLON	float	13	10	☑
V_MAXLON	float	13	10	☑
V_MINLAT	float	13	10	☑
V_MAXLAT	float	13	10	☑
V_MINALT	float	10	2	☑
V_MAXALT	float	10	2	☑
V_DISTANCIA	float	13	4	☑
V_ARCHIVO	varchar	255	0	☑
V_DESNIVELPOS	float	10	2	☑
V_DESNIVELNEG	float	10	2	☑

Figura 14. Taula de les capçaleres de ruta GPS

La clau primària és l'identificador de la ruta GPS.

En els atributs s'emmagatzema el número de punts del recorregut i el número de waypoints. També guarda la mínima i màxima longitud, latitud i altura, per fer-ne posteriorment una representació òptima a través de Google Maps, així com la distància total del recorregut i el desnivell màxim i mínim.

A més a més, guarda el nom de l'arxiu GPX que conté aquesta ruta, que ens servirà per exportar el fitxer cap a altres formats.

Pel que fa a la taula de **punts de ruta GPS** és la següent:

Name	Type	Length	Decimals	Allow Null
R_RUTAGPS	int	7	0	☐
V_ITERACION	int	4	0	☐
V_ALTITUD	float	6	2	☑
V_FECHA	datetime	0	0	☑
V_LONGITUD	float	13	10	☑
V_LATITUD	float	13	10	☑
V_NOMBRE	text	0	0	☑
V_WAYPOINT	int	11	0	☑
V_DISTANCIA	float	13	4	☑

Figura 15. Taula dels punts de ruta GPS

La clau primària està formada pel codi de ruta i la iteració.

Els seus atributs emmagatzemen les coordenades del punt (longitud,latitud,altura) i informació extra (nom del punt, data i hora, distància acumulada respecte l'origen...).

També tenim un camp que ens indica si és un punt del recorregut o és un waypoint.

Com que la pàgina web ja està en funcionament, i disposa de rutes de senderisme i d'ascensions a muntanyes, només ens haurem de preocupar de relacionar la part nova (rutes GPS) amb la part ja operativa.

Per fer aquesta relació s'han creat dues taules a la base de dades que ens relacionen aquests elements amb les rutes GPS. Veiem els atributs d'aquestes taules:

Name	Type	Length	Decimals	Allow Null
R_IDASC	int	7	0	☐
R_IDRUTAGPS	int	7	0	☑

Figura 16. Taula que relaciona les ascensions amb les rutes GPS

Name	Type	Length	Decimals	Allow Null
r_idruta	int	7	0	☐
r_idrutagps	int	7	0	☑

Figura 17. Taula que relaciona les rutes (senderisme) amb les rutes GPS

Com hem comentat anteriorment, la base de dades també disposa d'una taula on s'emmagatzemaran els formats de fitxer GPS que l'aplicació podrà convertir.

Aquesta taula és la següent:

Name	Type	Length	Decimals	Allow Null
V_FORMATO	varchar	20	0	☐
V_NOMBRE	varchar	200	0	☑
V_GPSBABEL	varchar	20	0	☑
V_ORDEN	int	3	0	☑

Figura 18. Taula de formats de fitxer GPS

Els atributs d'aquesta taula guarden el format del fitxer (que serà l'extensió), el nom i el codi amb què tractar aquest format a través del programa GpsBabel.

2.5 Implementació

Com hem comentat, el desenvolupament s'ha fet per mòduls i submòduls, i és així com els explicarem en aquest apartat. Descriurem breument la programació de cada submòdul en els seus punts clau.

2.5.1 Enviament del fitxer GPS

La web actual ja disposa d'un formulari de donar d'alta noves ascensions. Veiem una part del codi del fitxer JSP que conté aquest formulari:

```
Gps: <input type="file" name="GPSASCFILE<%out.print((String)
soluciones.get("USER"));%>" value="">
```

Hem afegit un nou camp en el formulari HTML de tipus *file* i on se li indica , com a identificador del camp, el nom *GPSASCFILE* seguit de l'identificador de l'usuari. Això es fa així pel cas que més usuaris al mateix temps estiguin omplint el formulari i per evitar el conflicte entre els dos arxius enviats.

Observem que s'ha incrustat codi Java dins la part escrita en HTML. En aquest cas, *out.print((String) soluciones.get("USER"));* recull les dades que li ha enviat el Servlet corresponents a l'identificador d'usuari actual.

Un cop enviat el formulari, el fitxer es guardarà en un directori temporal conegut, amb el nom que ja hem comentat.

2.5.2 Conversió a format estàndard GPX

En cas que s'hagi enviat un fitxer GPS, el Servlet que dóna d'alta l'ascensió crida el servei *ImportarFicheroGps*.

El primer que farà aquest mètode del paquet de serveis GPS és revisar que el fitxer estigui correcte. Un dels problemes que es va detectar va ser que alguns fitxers provinents del GPS contenien caràcters extranys (degut al traspàs d'un sistema operatiu a un altre), o bé no tenien extensió i se'n desconeixia el format. En aquest procés s'eliminaran els caràcters estranys i es farà una predicció del format del fitxer en cas que no tingui extensió, comprovant les primeres línies del fitxer.

Un cop conegut el format del fitxer, ja podem convertir-lo a format GPX per poder ser interpretat per la nostra aplicació. Això ho farem amb la crida externa a GpsBabel [DG] mitjançant les següent línies escrites en Java:

```
String comanda="gpsbabel.exe -w -r -t -i " + formato1 + " -f " + ruta1 +
archivo1 + " -x simplify,count=800 -o " + formato2 + " -F " + ruta2 +
archivo2;

Runtime run;
Process proc;
run = Runtime.getRuntime();
proc = run.exec(comanda);
proc.waitFor();
```

En el moment de crear la comanda de GpsBabel, li indiquem que converteixi tota la informació del fitxer (els waypoints, les rutes i els recorreguts), especifiquem el format i la ruta del fitxer origen, simplifiquem el número total de punts a 800 i indiquem el format i la ruta del fitxer destí (en aquest cas, el format destí serà GPX).

Amb la resta d'instruccions aconseguim iniciar el procés per executar la comanda. Amb el mètode *waitFor* de la classe *Process* aconseguim que l'execució del nostre programa s'aturi fins que no hagi acabat el procés, és a dir, fins que no s'hagi acabat la conversió, i el nou fitxer existeixi.

2.5.3 Parsejador del fitxer i Rectificació

Una vegada tenim l'arxiu en format GPX [FX], sabem que la informació estarà continguda dins de tags XML, de manera que creem un parsejador que ens interpretarà el seu contingut (waypoints, punts de la ruta, etc), i ens permetrà anar calculant tot allò que ens interessi a mesura que es vagi llegint el document (distància recorreguda, desnivell acumulat, etc).

Java ens ofereix dos models de parsejadors XML: un és SAX (Serial Access Xml) i l'altre és DOM (Document Object Model). Escollim el parsejador SAX perquè és més ràpid i fa un accés seqüencial al fitxer. Com a contrapartida, una vegada llegida una línia no pot tornar enrere, cosa que amb DOM sí. Però és una capacitat que tampoc la necessitem.

El més important en aquest procés és el *DocumentHandler*, que és el manipulador del document, és a dir, on li indiquem què és el què ha de fer el parsejador.

Anem a veure com s'inicia aquest procés.

```
SAXParser sp = spf.newSAXParser();
sp.parse(directorio_destino + archivo_destino, new
GPXReader(connection,directorio_destino,archivo_destino,idascension,tipo));
```

Primerament declarem una factoria parsejadora mitjançant el mètode *newSAXParser*(). D'aquesta factoria li ordenem que creï un parsejador, i en aquest parsejador li passem el *DocumentHandler* que és on li diem què ha de fer en cada moment.

El DocumentHandler és una interfície que respon a events pels quals es cridarà a diferents mètodes quan succeeixi alguna cosa en concret. Els mètodes principals són: *startDocument*, *endDocument*, *startElement* i *endElement*.

Un dels dissenys més senzills és implementar una classe interna que sigui el nostre *DocumentHandler*. En aquest cas creem la classe *GPXReader* i l'extenem de *DefaultHandler* i implementem tots els mètodes anomenats.

Així doncs, la classe *GPXReader* implementa els quatre mètodes comentats. Passem a descriure cadascun d'aquests mètodes:

startDocument

Quan s'inicia el document no fem res. S'ha de dir però que en la instanciació de la classe ja havíem inicialitzat algunes variables.

startElement

En el document GPX, els elements que ens interessen són només aquells que estan entre els tags <wpt> i </wpt>, <trkpt> i </trkpt>, <rtept> i </rtept>, corresponents als waypoints, als punts de ruta i els punts de recorregut. Aquests elements duen en els seus atributs les dades de latitud i longitud (*lat* i *lon*) corresponents a cada punt, com en el següent exemple:

```
<wpt lat="42.370144000" lon="2.130359000">
  <ele>2435.600000</ele>
  <name>COLL DE FONTALBA</name>
  <cmt>COLL DE FONTALBA</cmt>
  <desc>COLL DE FONTALBA</desc>
  <sym>Blue Flag</sym>
</wpt>
```

Per tant, si l'element és un d'aquests, farem el següent:

```
for(int i = 0 ; i < attributes.getLength() ; i++) {
        nombreatributo=attributes.getLocalName(i).toString();
        if ( nombreatributo == "lat" )
                lat=Double.parseDouble(attributes.getValue(i));
        if ( nombreatributo == "lon" )
                lon=Double.parseDouble(attributes.getValue(i));
}
```

Recorrerem els seus atributs i en cas de trobar *lat* o *lon*, guardarem el seu valor en variables.

Pel què fa al valor de la dada d'altura, com podem veure va dins un element addicional (*ele*), i el guardarem en el mètode *endElement*.

endElement

En aquest mètode podem accedir al contingut d'entre dos tags XML. Per exemple, si l'element que finaliza és *ele*, és a dir, s'ha trobat *</ele>,* el contingut serà la dada d'altitud. De la mateixa manera també ens interessa guardar les dades de descripcions, dates, etc.

Quan finalitzin els elements </wpt>, </trkpt>, </rtept>, afegirem el punt amb tota la informació (latitud, longitud, descripció, etc) en un array de punts que tractarem en el mètode *endDocument*.

Ara bé, en aquí farem una petita rectificació de les dades que ens vénen del fitxer.

Primerament, si en el fitxer no venia la dada d'altura del punt, la demanarem al WebService de Geonames [WG] , que és d'accés lliure, a través de la funció següent:

```
altitud=WebService.srtm3(lat,lon);
```

El servei web SRTM3 (Shuttle Radar Topography Mission) és un sistema de radar que ens proporciona les dades d'altura de tots els punts de la Terra separats per 3-arc-segons (aproximadament 90 metres) en una quadrícula latitud / longitud.

Nota: per fer ús d'aquesta funció s'ha d'haver importat prèviament el paquet *org.geonames.WebService*.

Una altra rectificació que fem és que si el punt que està guardant és molt llunyà a l'anterior (més de 50 km), aquest no l'enregistrarem.

En aquest mètode, com que ja tenim tota la informació del punt llegit, podem anar calculant les estadístiques que també guardarem a la base de dades, com són: distància total recorreguda, desnivell mínim/màxim, altitud mínima/màxima, etc. També comprovarem quina és la mínima/màxima altitud i latitud per dur a terme una bona representació posteriorment a través del Google Maps.

endDocument

En el moment que s'acaba el document, disposem d'un array de punts ordenats, que haurem d'emmagatzemar a la base de dades.

2.5.4 Emmagatzematge a la base de dades

La ruta s'emmagatzema a la base de dades en el moment que s'acaba el document que s'estava Parsejant, és a dir, dins el mètode *endDocument* de la classe *GPXReader*.

Primerament s'havia provat d'anar emmagatzemant els punts a la base de dades a mesura que s'anaven llegint del fitxer. Això suposava una llarga espera per tractar tot el fitxer, ja que l'arquitectura actual de l'aplicació obre una connexió a cada accés que es faci a la base de dades. Va suposar una gran millora a nivell de velocitat el fet de realitzar aquest emmagatzematge un cop acabada la lectura del fitxer.

Així doncs, aquest procés el fem de la següent forma:

```
for (int i=0;i<listapuntos.size();i++)
{
        ArrayList punto=new ArrayList();
        punto=(ArrayList)listapuntos.get(i);
        GpsServicio.insertarPuntoRutaGps( -totes les dades del punt- );
}
// Si hi ha punts a la ruta, insertem la capçalera de la ruta
if (listapuntos.size()>0)
{
        GpsServicio.insertarRutaGps(con, numruta, tipo_ruta, ascension,
npuntos, nwaypoints, min_lon , max_lon, min_lat, max_lat,
min_alt,max_alt,distancia_acumulada,nombre_archivo,desnivel_pos,desnivel_ne
g);
}
```

Disposem d'un array amb tots els punts de la ruta o de recorregut. També dels waypoints, que es diferencien en un flag. Per cadascun d'aquests punts, i sabent ja el codi de l'ascensió, inserim les dades a la base de dades. El mateix fem per la capçalera de la ruta amb les estadístiques calculades.

Tota aquesta inserció massiva a la base de dades es realitza de forma unitària, és a dir, si en algun moment es produís un error inesperat, no s'hauria guardat res a la base de dades.

Com podem veure, per inserir dades a la base de dades cridem mètodes de la classe *GpsServicio*, i aquesta cridarà mètodes de la classe *GpsDAO*, corresponents a la capa d'accés a les dades, seguint sempre el model d'arquitectura de l'aplicació.

No entrarem molt en detall, però a mode d'exemple, posem un tall del codi on s'introdueix un punt a la base de dades. Lògicament, el llenguatge que ens permet la interacció amb la base de dades MySql és SQL.

```
stat = dameStatement(connection);
String query = "insert into gps_puntos
(r_rutagps,v_iteracion,v_longitud,v_latitud,v_altitud,v_fecha,v_nombre,v_wa
ypoint,v_distancia) values (" + ruta + "," + iteracion + "," + longitud +
"," + latitud + "," + (float)ele + "," + fecha + ",'" + nombre + "'," +
eswaypoint + "," + (float)distancia + ")";

if (stat.executeUpdate(query) != 1)
{
        throw new Exception("No se ha podido insertar el punto de la ruta
con la siguiente query:" + query);
}
```

Un cop emmagatzemats els punts i la capçalera de la ruta, que corresponen a dues taules diferents de la base de dades, també afegirem un registre a la taula que ens relaciona les ascensions amb les rutes GPS, tal i com es comenta en l'apartat de *Disseny de la base de dades*.

Fins ara hem descrit la implementació pel que fa la importació de fitxers d'ascensions a muntanyes de la pàgina web. Pel que fa a les rutes (senderisme, BTT, escalada...) el procés és exactament el mateix, només canvia que en comptes d'afegir un registre a la taula que relaciona les ascensions amb les rutes GPS, afegirem un registre a la taula que relaciona les rutes amb les rutes GPS.

2.5.5 Consulta de la ruta

L'usuari de la pàgina, per accedir a veure el perfil o la representació de la ruta, primerament haurà d'accedir a la pàgina de descripció de la ruta. En aquesta pantalla comprovem si l'ascensió disposa de fitxer GPS, i en aquest cas mostrem un enllaç a una altra pantalla més gran per tal de mostrar més còmodament tots els elements.

Aquest enllaç el creem a la pàgina JSP de la següent forma:

```
<b>Gps:</b> <%if ( ((Boolean)soluciones.get("GPS")).booleanValue() ){%>
<a href="javascript:pantallacompleta('<%=entorno%>
MostrarRuta?IDASCENSION=<%=idmountain%>&TIPO=0')">Ver ruta y perfil</a>
<%}else{%>Sin fichero GPS<%}%>
```

Veiem que de nou estem mesclant codi HTML i codi Java dins la pàgina JSP. Si el Servlet li ha passat la informació de que l'ascensió disposa de ruta GPS, mostrarem l'enllaç, sinó simplement imprimirà "*Sin fichero GPS*".

Aquest enllaç crida al Servlet MostrarRuta, a través de l'enllaç a *'<%=entorno%> MostrarRuta? IDASCENSION=<%= idmountain%>&TIPO=0'*, on la variable *entorno* guarda l'adreça del servidor i la carpeta de treball. Veiem que al Servlet MostrarRuta se li passa per paràmetre GET l'identificador de l'ascensió que estem consultant, i el tipus, en aquest cas 0, perquè es tracta d'una ascensió.

2.5.6 Representació del perfil

Hem comentat anteriorment que la representació del perfil la desenvoluparem amb Javascript mitjançant la API de Google Visualization [GV] , per la seva simplicitat i el seu ventall d'opcions de configuració en el moment de representar les dades.

Així doncs, en la pàgina JSP de representació del perfil, hi incrustarem tot el codi Javascript referent a la representació del perfil. Passem a explicar-lo simplificadament, en els seus punts més importants.

Abans de tot, carreguem les llibreries necessàries per mostrar el gràfic:

```
// Carga la librería areachart de google visualization
google.load("visualization", "1", {packages:["areachart"]});
google.setOnLoadCallback(drawChart);
```

En aquest cas, necessitem el paquet *areachart*, corresponent a un gràfic lineal on es pinta la part inferior de la línia del gràfic. En el moment d'obrir-se la pàgina es cridarà a la nostra funció *drawChart,* que descrivim a continuació.

Primerament, i previ al tipus de visualització, es crea la taula de dades.

```
// Crea la tabla de datos
var data = new google.visualization.DataTable();
data.addColumn('string', 'Distancia'); // eje x
data.addColumn('number', 'Altitud: '); // eje y
```

Veiem que disposem de dues columnes (camps de dades), un representa la distància recorreguda, i l'altra l'altitud. Passem a omplir aquests camps.

```
// Marcamos eje y con la altitud de los puntos
for (var j=0;j<puntos_perfil.length;j++){
      var punto_perfil=puntos_perfil[j];
      var distancia_punto=punto_perfil[3];
      // Calcula la posicion del eje x a la que se encuentra este punto
      var columna = parseInt (( (distancia_punto*1000) / mpc ) + 0.5);
      // Marcamos la altitud
      data.setCell(columna, 1, punto_perfil[2]);
}
// Dibuja el perfil con las opciones correspondientes
var chart = new
google.visualization.AreaChart(document.getElementById('perfil'));
chart.draw(data, {legend: 'none',pointSize:'2'});
```

Totes les dades dels punts les ha passat el Servlet cap a la pàgina JSP. Per cada punt de la ruta, marca l'altura en la columna 1 de la taula de dades.

A més a més d'això, també s'ha desenvolupat un *Listener* que en cas que es seleccioni un dels punts del gràfic, aquest el centrarà en el mapa de Google Maps, per fer més interactiva la representació i saber a quin punt del mapa es troben un determinat punt del perfil.

Per mostrar el perfil, s'ha d'indicar a la pàgina JSP quin contenidor contindrà el perfil, mitjançant les següents línies:

```
<%if ( hay_puntos ){%>
      <div align=center><div id="perfil" align=center style="width: 100%;
height: 150px; "></div>
<% } else { %>
      <div align=center><div id="perfil" align=center style="width: 100%;
height: 150px; display:none"></div>
<% } %>
```

La variable *hay_puntos* ens indica si la ruta disposa de punts de recorregut, en aquest cas mostrarem el perfil, sinó, el contenidor serà invisible. Això passa a vegades, quan una ruta només disposa de waypoints, llavors no es mostrarà el perfil.

2.5.7 Representació del recorregut

Com sabem, el recorregut el mostrarem mitjançant la API de Google Maps [GM] , també amb Javascript. Veiem algunes de les parts més importants del codi.

Per carregar el Google Maps, és necessari carregar prèviament les llibreries de Google indicant des de quin servidor les utilitzem mitjançant una *key* que ens proporciona gratuïtament Google.

També necessitem que en l'etiqueta *BODY* de la pàgina JSP, indiquem que quan es carregui la pàgina executi una funció *Load(),* corresponent al codi Javascript de Google Maps, així com la funció *GUnload()* pel moment de tancar la pàgina.

```
<body leftmargin="0" topmargin="0" rightmargin="0" bottommargin="0"
marginwidth="0" marginheight="0" onload="load()" onunload=GUnload()>
```

També és necessari definir el contenidor que contindrà el mapa:

```
<div id="mapa"  style="width: 100%; height: 360px; "></div>
```

En la funció Load() fem ús de la API de Google Maps i indiquem com hem d'enquadrar el mapa per mostrar tots els punts, és a dir, quin ha de ser el punt central i quin ha de ser el nivell de zoom.

```
// Calcula el zoom para que se vea toda la ruta
var sw = new GLatLng(info[3],info[1]);
var ne = new GLatLng(info[4],info[2]);
var fronteres=new GLatLngBounds(sw,ne);
var zoomlevel=mapa.getBoundsZoomLevel(fronteres);
// Centra el mapa con el zoom adecuado
var centrox = (info[2]+info[1])/2;
var centroy = (info[4]+info[3])/2;
mapa.setCenter(new GLatLng(centroy,centrox), zoomlevel);

// Controles por defecto de zoom y desplazamiento
mapa.setUIToDefault();
```

També indiquem el tipus de mapa a mostrar i els controls que volem que apareguin:

```
// Botones con los tipos de plano
mapa.addControl(new GMapTypeControl());
// Escala del mapa
mapa.addControl(new GScaleControl());
// Reemplaza el botón de mapa híbrido por el físico
mapa.addMapType(G_PHYSICAL_MAP);
mapa.removeMapType(G_HYBRID_MAP);
// Muestra plano pequeño de la zona
mapa.addControl(new GOverviewMapControl());
// Tipo de mapa a mostrar
mapa.setMapType(G_PHYSICAL_MAP);
// Activación movimiento zoom con la rueda del ratón
mapa.enableScrollWheelZoom();
```

Ara passem a marcar els waypoints, informació de la ruta que li haurà passat el Servlet.

```
// Marca todos los waypoints
var todos_los_waypoints = <%=waypoints%>;
for (var j=0;j<todos_los_waypoints.length;j++){
      var waypoint=todos_los_waypoints[j];
      var nuevo_waypoint = new GLatLng(waypoint[1], waypoint[0]);
      var nom = waypoint[3] + "<BR><BR>Altitud: " + waypoint[2] + "
metros";
      var marker = createMarker (nuevo_waypoint, nom);
      mapa.addOverlay(marker);
}
```

I el més important, el recorregut de la ruta, que ho farem mitjançant línies que uneixen tots els punts.

```
// Marca toda la ruta (mediante líneas de un punto a otro)
var todos_los_puntos = <%=puntos%>;
var antiguo_punto=new GLatLng(0,0);

for (var j=0;j<todos_los_puntos.length;j++){
      var punto=todos_los_puntos[j];
      var nuevo_punto = new GLatLng(punto[1], punto[0]);

      // Define la linea a dibujar, con el color y transperenci
      var polyline = new GPolyline([antiguo_punto,nuevo_punto], "#04B210",
5,0.8);
      // La dibuja en el mapa
      mapa.addOverlay(polyline);
      antiguo_punto=nuevo_punto;
}
```

2.5.8 Mostrar estadístiques

Totes les estadístiques estan calculades i emmagatzemades a la capçalera de la ruta, a la base de dades.

El Servlet que mostra tota la informació també passa aquestes dades a la pàgina JSP, de manera que simplement s'han de mostrar. Per exemple, en el cas de la màxima/mínima altitud, ho fem de la següent forma.

```
<%if ( hay_puntos ){ %>
      <div class="datos">
      <b>Máxima altitud: </b><%=str_est_alt_max%> m.
      <BR><b>Mínima altitud: </b><%=str_est_alt_min%> m.
      </div>
<% } %>
```

En aquest cas, les variables *str_est_alt_max* i *str_est_alt_min* contenen les dades un cop tractades, ja que s'han hagut d'arrodonir i convertir de tipus *float* a *string*.

2.5.9 Selecció del format de descàrrega

A la mateixa pàgina hi mostrarem un petit formulari a la part inferior que bàsicament serà un desplegable amb el llistat de possibles formats de fitxer GPS en el qual ens volem descarregar la ruta. Un cop seleccionat el format, s'haurà de prémer el botó *Descargar Ruta*.

Com sempre, les dades les proveeix el Servlet i des de la pàgina JSP només ens ocupem de la seva representació.

```
<div class="datos" align=right><b><i>Descargar ruta</i></b>
<SELECT NAME="formato" id="formato">
<% for (int i=0; i<formatos_gps.size(); i++) {
      ArrayList formato = (ArrayList) formatos_gps.get(i);
      %>
      <option value=<%= formato.get(0).toString().toLowerCase() %> > <%=
formato.get(1) + " (." + formato.get(0).toString().toUpperCase() + ")" %>
<% } %>
<option value="0" selected>(Escoger formato)
</SELECT>

<input type="submit" value="Descargar"
onclick="javascript:descargarFichero();">
</div>
```

Veiem que un cop escollit el format i premut el botó, s'executa la funció Javascript *DescargarFichero*(), on el què es fa és el següent:

```
function descargarFichero()
{
	// Obtener la referencia a la lista
	var lista = document.getElementById("formato");
	// Obtener el valor de la opción seleccionada
	var formatoSeleccionado = lista.options[lista.selectedIndex].value;

	var info = <%=info_ruta%>;
	var idruta=info[8];

	if (formatoSeleccionado == 0)
		alert("Tiene que seleccionar un formato de archivo GPS");
	else
	{
		var truta=<%=tipo_ruta%>;
		var url="DescargarRuta?idruta=" + idruta + "&formato=" +
formatoSeleccionado + "&tipo=" + truta;
		self.location = url;
	}
}
```

Primer comprova que s'hagi escollit un format de fitxer, i després crida al Servlet que s'ocuparà de descarregar la ruta al client, passant-li com a paràmetres el codi de la ruta i el format de descàrrega.

2.5.10 Descàrrega del fitxer

Per descarregar el fitxer cap al client, disposem del Servlet *DescargarRuta*.

Primerament, el Servlet llegeix les dades dels paràmetres: el codi de la ruta i el format de descàrrega.

```
// Coge los parametros
ruta=request.getParameter("idruta");
formato=request.getParameter("formato");
tipo=request.getParameter("tipo");

// Crea el fichero a descargar
nombre_archivo = GpsServicio.exportarRuta(null, Integer.parseInt(ruta),
formato, Integer.parseInt(tipo));
```

Amb aquestes dades crida el mètode *exportarRuta*, que ens donarà com a resultat la ruta del fitxer resultant.

El mètode *exportarRuta* converteix el fitxer de la ruta (actualment en format GPX) al format escollit per l'usuari. Aquest procés es fa exactament igual que en el procés d'importació.

Un cop disposem del nou fitxer, ja es pot descarregar a través del navegador web.

```
// Si existe lo descarga
File archivo = new File(nombre_archivo);

if (archivo.exists())
{
      response.setContentType("application/x-download");
      response.setHeader("Content-Disposition", "attachment; filename=" +
ruta + "." + formato);
      OutputStream out = response.getOutputStream();

      try{
            dumpFile(nombre_archivo, out);
      } catch (Exception e) {
            PrintWriter esc = response.getWriter();
            esc.println("Se ha producido un error descargando el fichero "
+ nombre_archivo);
      }
}
else
{
      System.out.println("Error descargando ruta. El archivo " +
nombre_archivo + " no existe");
}
archivo.delete();
```

Veiem que en aquest cas el Servlet no retorna una pàgina web, sinó una aplicació *x-download* que enviarà a través d'un Stream el fitxer cap al client. Un cop el client s'ha descarregat el fitxer, s'esborra per no ocupar espai innecessari.

3 Resultats obtinguts

Un cop vist tot el procés de realització el projecte, i després d'haver superat totes les proves i solucionats tots els problemes, ja podem veure els resultats obtinguts, a nivell de l'usuari, de la nostra aplicació.

Primerament, l'usuari, ha de donar d'alta o modificar una de les seves ascensions o rutes.

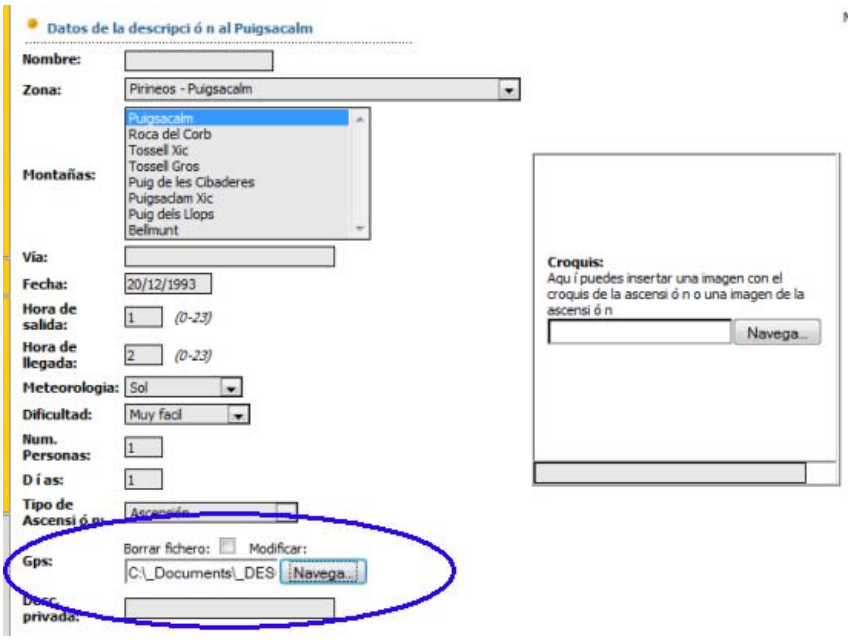

Figura 19. Formulari d'Alta o modificació d'Ascensió

Veiem que en el formulari es disposa d'un camp per poder penjar un fitxer. En aquest cas hi penjarem un fitxer de ruta en format OziExplorer (.plt), i premerem el botó *Modificar* del formulari.

Ara, quan l'usuari consulti la descripció de l'ascensió o ruta modificada, es trobarà amb el següent.

Figura 20. Part de la pàgina de Pirineos3000 de descripció d'una ascensió

En cas que l'ascensió disposi d'una ruta GPS, es mostrarà l'enllaç tal i com es mostra en la figura anterior. Aquest enllaç ens obrirà una pàgina completa amb la representació del perfil i el recorregut.

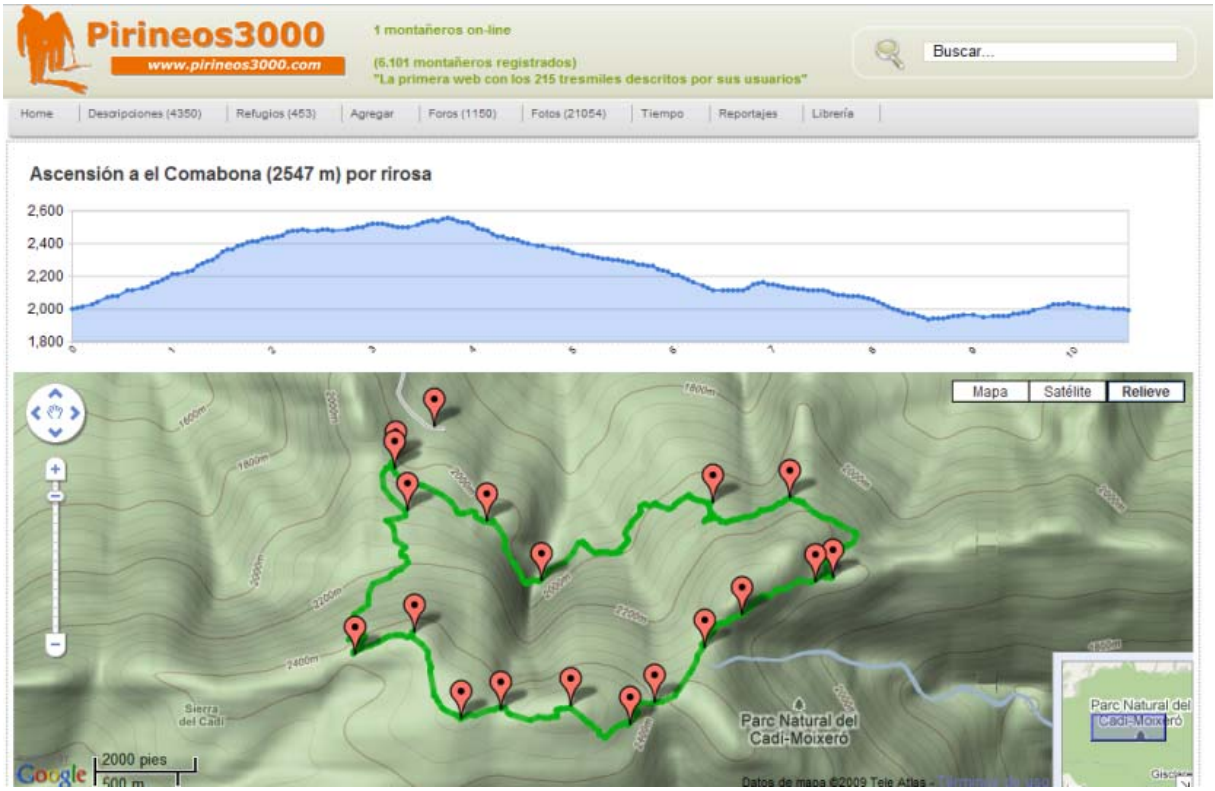

Figura 21. Representació del perfil d'una ascensió i el recorregut

Si la ruta conté waypoints, es mostraran en vermell i podrem fer clic sobre ells i ens mostrarà la informació del punt.

Figura 22. Informació d'un waypoint

També, si fem clic damunt de qualsevol punt en el perfil de la ruta, ens centrarà aquest punt en el mapa, mitjançant un waypoint de color blau..

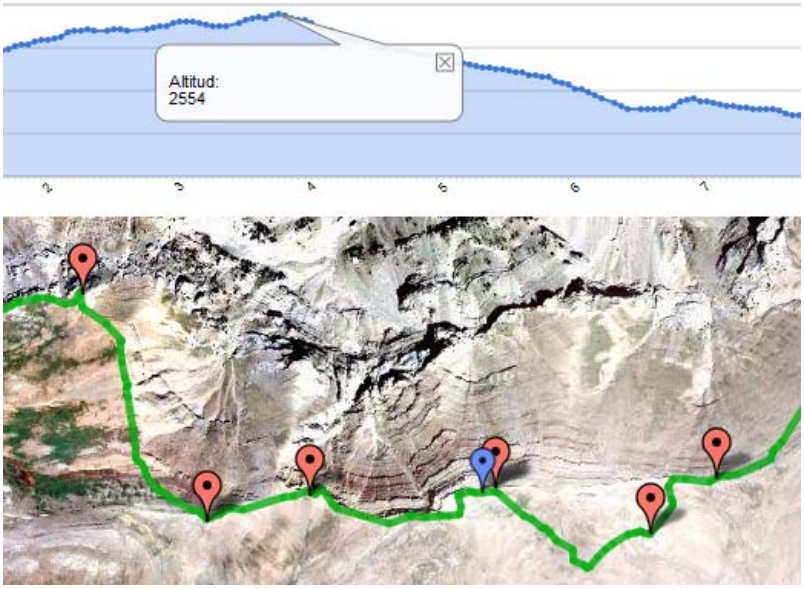

Figura 23. Interacció entre el perfil i el mapa

A més a més, també podem fer ús de totes les capacitats de Google Maps, per exemple, arrastrar-nos pel mapa, fer zoom, o canviar el mapa a imatge per satèl·lit o de relleu.

A la part inferior també es mostren algunes estadístiques de la ruta calculades automàticament en el procés, així com el desplegable amb els diferents formats de GPS amb els quals ens podem descarregar la ruta.

Figura 24. Part inferior de la pàgina

En aquest cas, per exemple, escollim descarregar el fitxer en format .KML (Google Earth), i així que premem el botó, ja ens proposa què fer amb el fitxer.

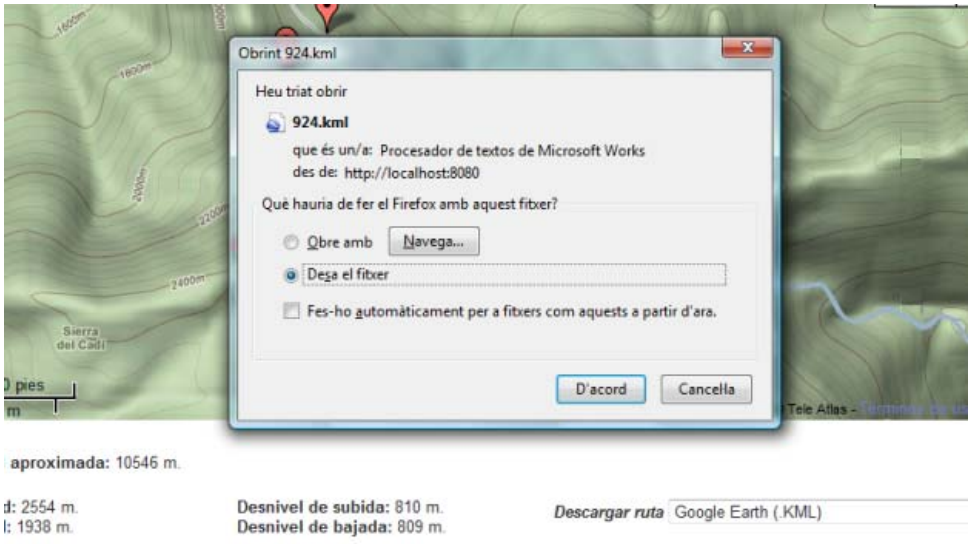

Figura 25. Finestra del navegador de descàrrega d'un fitxer

En el nostre cas, com que tenim el Google Earth instal·lat, podem comprovar que el fitxer s'ha generat correctament.

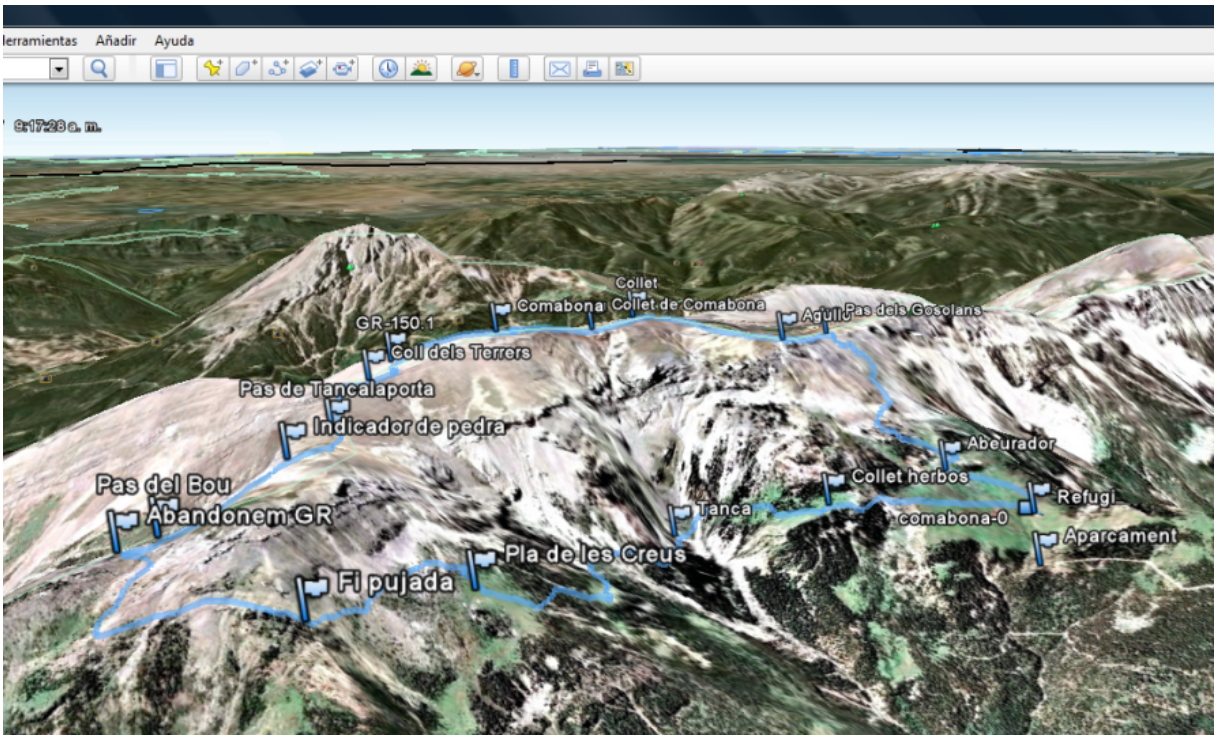

Figura 26. Fitxer de la ruta obert a través de Google Earth

4 Conclusions

Per finalitzar, mostrarem les conclusions del projecte, revisant els objectius inicials descrits en el primer capítol de la memòria a través dels resultats obtinguts. També analitzarem possibles línies de continuació del projecte en un futur.

4.1 Revisió dels objectius

En el capítol 1.4 d'aquesta memòria fèiem referència als objectius establerts a l'inici del projecte.

L'objectiu principal del projecte era realitzar la implementació d'un sistema per carregar, representar i descarregar fitxers de rutes de muntanya a través d'Internet, dins de l'entorn ja operatiu de la pàgina Pirineos3000.

Segons hem pogut veure en l'apartat 3 de resultats, l'objectiu principal està totalment complert.

Fem un repàs de les funcionalitats que havíem establert, tot comprovant si s'han complert els objectius amb més detall.

Enviament de fitxers de rutes en diferents formats

L'usuari pot enviar el fitxer GPS en diferents formats, és a dir, en diferents extensions, i aquests es passen a un programa extern (GpsBabel) que el converteix a un format estàndard (GPX).

Es podria donar el cas que l'usuari intenti *enganyar* l'aplicació fent-li veure que el fitxer original era d'un format diferent (canviant-hi l'extensió), o bé creant un arxiu *fals* amb el qual l'aplicació convertiria el fitxer i en sortiria un fitxer defectuós, és a dir, sense cap punt vàlid. Però aquest fet tampoc suposa cap problema pels usuaris que faran un bon ús de l'aplicació.

Tractament dels fitxers i emmagatzematge a la base de dades

El fitxer estàndard es tracta de manera correcta i els punts i la ruta s'emmagatzemen a la base de dades.

Malgrat tarda alguns segons en alguns casos, es considera que la inserció a la base de dades és molt ràpida tenint en compte que les rutes poden tenir centenars de punts i que cada punt implica inserir un registre a la base de dades.

Representació de les rutes amb l'aplicació Google Maps

Segons hem pogut veure en l'apartat 3 de resultats, la ruta es representa correctament a través de Google Maps. També ha de ser l'usuari qui busqui la seva millor representació, ja sigui fent zoom, o canviant el mapa de relleu per les imatges per satèl·lit, per exemple.

Representació gràfica del perfil del recorregut

El perfil també compleix les especificacions establertes i s'aprecia perfectament la dificultat de la ruta. S'ha volgut anar una mica més enllà de les especificacions inicials, i s'ha fet que en el perfil es poguessin seleccionar els punts, se'n mostrés l'altitud i el mapa de Google Maps es centrés en aquell punt.

Càlcul i mostra d'estadístiques calculades automàticament

Les estadístiques es calculen i es mostren de manera correcta. A més a més, s'han comprovat en diferents portals que realitzen càlculs semblants i efectivament les dades calculades per l'aplicació es corresponen.

Descàrrega de la ruta en diferents formats

L'usuari pot escollir el format del fitxer GPS a descarregar tal i com s'havia previst inicialment, i la conversió cap al nou format es realitza de manera ràpida.

A més a més, si en el futur l'aplicació GpsBabel evoluciona, podent convertir nous formats o millorant les capacitats actuals, la nostra aplicació també es beneficiarà d'aquestes noves característiques.

Adaptació a la web operativa

Un dels requisits fonamentals era que l'aplicació s'adaptés a la pàgina web, tan a nivell de disseny com d'arquitectura del software. Com podem veure en l'apartat 3 de resultats de la memòria, podem veure que manté el disseny de la pàgina web, i pel què fa a l'arquitectura, també la manté, separant en tot moment les tres capes de l'aplicació (Façana, Servei, Accés a dades), tal i com s'ha anat explicant en l'apartat 2.5 d'Implementació.

Sobre la tecnologia utilitzada, el fet d'utilitzar l'aplicació externa GpsBabel amb Java, fa que es perdi un dels objectius de la programació en Java, que és el del funcionament independentment de la plataforma. Per a que això sigui possible, s'hauria de controlar el sistema operatiu i modificar si s'escau la crida a l'aplicació externa.

Per evitar això s'hauria de programar un mòdul de conversió de fitxers GPS a dins l'aplicació, però sense dubte, la complexitat d'aquesta aplicació n'impossibilitaria la realització en un Projecte Final de Carrera.

En conclusió, tal i com hem pogut veure analitzant els objectius inicials proposats, veiem que s'han acomplert tots els objectius.

4.2 Línies de continuació

Durant la realització del projecte es van veient les possibles línies de continuació que se li podrien donar, per tal de millorar l'aplicació i donar-li més funcionalitats.

Tot seguit se'n comenten algunes.

Connexió directe amb el receptor GPS

Mitjançant la instal·lació d'un Plug-in, l'aplicació es podria connectar directament amb el dispositiu GPS connectat a l'ordinador (port USB, sèrie, etc). Amb això, ens podríem estalviar el pas que fa l'usuari de traspassar el fitxer primer a l'ordinador i després a l'aplicació web, i a l'inrevés.

Ara bé, el procés normal actual és revisar la ruta abans de penjar-la a l'aplicació, i cada marca de GPS ja proveeix del seu programa per revisar-la i modificar-la.

Podem imaginar també receptors GPS amb connexió a Internet, i podent descarregar rutes un cop l'usuari està a la muntanya.

Modificació de rutes

Una altra millora que podríem fer, és que les rutes es poguessin modificar a través de la web, i posteriorment poder descarregar la ruta modificada.

Aquest procés s'hauria de dur a terme amb el Google Maps interactuant amb l'usuari, i que es poguessin canviar de lloc els punts del recorregut, així com afegir-ne, eliminar-ne, etc.

Aplicar-hi Google Earth i l'entorn 3D

En els darrers mesos van apareixent pàgines que també tenen implementat el Google Earth en la mateixa finestra on tenim el Google Maps. Aquesta possibilitat requereix la instal·lació del plug-in de Google Earth, però ens permetria mostrar les rutes dins un entorn 3D, poder-les visualitzar des de diferents posicions (a vista d'ocell), cosa que fa que l'usuari percebi millor la dificultat o les característiques de la ruta.

Altres mapes cartogràfics

En diferents regions del món, es disposa de mapes més específics amb millor resolució que la que ofereix Google Maps. Aquest és el cas de Catalunya, on l'ICC (Institut Cartogràfic de Catalunya) proveeix gratuïtament mapes topogràfics i de relleu a una millor escala que Google Maps.

Aquests mapes es poden integrar també dins de Google Maps, millorant-ne així la presentació. En el cas de les rutes de muntanya, és una via a explorar, ja que pel què fa a al marcatge dels camins i corriols, podem dir que la informació que ens proveeixen els mapes de relleu de Google Maps és una mica limitada.

5 Referències

Mostrem algunes de les referències que han ajudat en algun moment del procés de realització del projecte.

[AJ] API de Java

Descripció: API per la Plataforma Java 2 Edició Estàndard 5.0

Adreça: http://java.sun.com/j2se/1.5.0/docs/api/

Idioma: Anglès

Darrer accés: 05/01/2010

[MJ] Manual de Java

Descripció: Manual complet d'utilització de Java

Adreça: http://manual-java.com/

Idioma: Castellà

Darrer accés: 10/11/2009

[JC] Java en castellano

Descripció: Fòrums i solucions per a desenvolupadors en Java

Adreça: http://www.programacion.com/java/

Idioma: Castellà

Darrer accés: 20/12/2009

[ME] Manual de Eclipse

Descripció: Manual bàsic d'Eclipse

Adreça: http://eclipsetutorial.forge.os4os.org/in1.htm

Idioma: Castellà

Darrer accés: 01/10/2009

[GM] API de Google Maps

Descripció: API de Google Maps, documentació, exemples...

Adreça: http://code.google.com/intl/ca/apis/maps/

Idioma: Anglès

Darrer accés: 25/12/2009

[GV] API de Google Visualization

Descripció: API de Google Visualization, documentació, exemples...

Adreça: http://code.google.com/intl/ca/apis/visualization/

Idioma: Anglès

Darrer accés: 25/12/2009

[FX] Format GPX

Descripció: Definició del format estàndard GPX

Adreça: http://www.topografix.com/gpx.asp

Idioma: Anglès

Darrer accés: 10/11/2009

[WG] Webservice Geonames

Descripció: Documentació dels serveis web de Geonames

Adreça: http://www.geonames.org/

Idioma: Anglès

Darrer accés: 10/11/2009

[DG] Documentació de GpsBabel

Descripció: Programa i documentació per a la utilització de GpsBabel

Adreça: http://www.gpsbabel.org/

Idioma: Anglès

Darrer accés: 20/12/2009

6 Annexes

6.1 Glossari

[1] Receptor GPS

Un *receptor GPS* és un dispositiu de butxaca que permet saber la posició geogràfica de longitud, latitud i altura amb una precisió d'uns metres, fent servir la tecnologia GPS.

[2] GPS

El *Global Position System* o Sistema de Posicionament Global és un sistema que permet determinar en tot el món la posició d'un objecte, una persona, un vehicle o una nau, fins a una precisió de centímetres, encara que l'habitual són uns pocs metres. Encara que la seva invenció s'atribueix als governs francès i belga, el sistema va ser desenvolupat, instal·lat, i actualment és operat pel Departament de Defensa dels Estats Units.

El GPS funciona mitjançant una xarxa de 27 satèl·lits (24 operatius i 3 de suport) en òrbita sobre el globus, a 20.200 km d'altura, amb trajectòries sincronitzades per cobrir tota la superfície de la Terra. Quan es desitja determinar la posició, el receptor que s'utilitza per a això localitza automàticament com a mínim tres satèl·lits de la xarxa, dels quals rep uns senyals indicant la posició i el rellotge de cadascun d'ells. Sobre la base d'aquests senyals, l'aparell sincronitza el rellotge del GPS i calcula el retard de les senyals, és a dir, la distància al satèl·lit. Per "triangulació" calcula la posició en que aquest es troba. La triangulació en el cas del GPS, a diferència del cas 2-D que consisteix a esbrinar l'angle respecte de punts coneguts, es basa en determinar la distància de cada satèl·lit respecte al punt de mesurament. Conegudes les distàncies, es determina fàcilment la pròpia posició relativa respecte als tres satèl·lits. Coneixent a més les coordenades o posició de cadascun d'ells pel senyal que emeten, s'obté la posició absoluta o coordenades reals del punt de mesurament. També s'aconsegueix una exactitud extrema en el rellotge del GPS, similar a la dels rellotges atòmics que porten a bord cada un dels satèl·lits.

L'antiga Unió Soviètica tenia un sistema similar anomenat GLONASS, ara gestionat per la Federació Russa. Actualment la Unió Europea està desenvolupant el seu propi sistema de posicionament per satèl·lit, denominat Galileu.

[3] Google Maps

Google Maps és una aplicació web de la companyia Google que es pot incloure a qualsevol pàgina web. Es tracta d'un requadre amb el mapa de la zona del Món que es vulgui, ja sigui de carreteres, de relleu, d'imatge per satèl·lit, etc.

És programable, és a dir, es pot mostrar la informació que es vulgui en el mapa, com adreces, rutes, punts d'interès, fotografies... i es poden afegir botons i events amb diferents funcionalitats mitjançant la interacció amb l'usuari, entre moltes d'altres funcions de l'aplicació.

[4] API

Una *Interfície de Programació d'Aplicacions* o API és el conjunt de funcions i procediments que ofereix certa biblioteca per ser utilitzat per una altra aplicació com una capa d'abstracció.

[5] AJAX (Asynchronous Javascript And XML).

AJAX no es una tecnologia en sí: consisteix en realitzar crides des del script Javascript a un servidor web mitjançant el protocol HTTP durant l'execució de l'script, és a dir, sense para la seva execució i sense recarregar la pàgina.

La crida farà que el servidor respongui amb certa informació que serà tractada per l'script.

[6] HTTP

El protocol de transferència d'hipertext (*HyperText Transfer Protocol*) és el protocol utilitzat en cada transacció de la Web (www). HTTP va ser desenvolupat pel consorci W3C i l'IETF, col·laboració que va culminar el 1999 amb la publicació d'una sèrie de RFC, el més important d'ells el RFC 2616, que especifica la versió 1.1.

HTTP defineix la sintaxi i la semàntica que utilitzen els elements software de l'arquitectura web (clients, servidors, proxies) per comunicar-se. És un protocol orientat a transaccions i segueix l'esquema petició-resposta entre un client i un servidor. HTTP és un protocol sense estat, és a dir, que no guarda cap informació sobre connexions anteriors.

[7] GPX

GPX (GPS eXchange Format) és un format de fitxer utilitzat per intercanviar dades de GPS entre aplicacions.

Al contrari d'altres formats propietaris de dades de GPS, que només poden ser entesos pel programa que els ha creat, els fitxers GPX inclouen una descripció del seu contingut, permetent a qualsevol crear programes que puguin llegir la seva informació.

[8] Webservice

Un servei web (*Web service*) és un conjunt de protocols i estàndards que serveixen per intercanviar dades entre aplicacions. Diferents aplicacions de programari desenvolupades en llenguatges de programació diferents i executades sobre qualsevol plataforma, poden utilitzar els serveis web per intercanviar dades en xarxes d'ordinadors com Internet. La interoperabilitat s'aconsegueix mitjançant l'adopció d'estàndards oberts.

Les organitzacions OASIS i W3C són les responsables de l'arquitectura i reglamentació dels serveis web. Per millorar la interoperabilitat entre diferents implementacions de serveis web s'ha creat l'organisme WS-I, encarregat de desenvolupar diversos perfils per a definir de manera més exhaustiva aquests estàndards.

[9] GNU GPL

La Llicència Pública General (*General Public License*) és un tipus de llicència per a programari que permet la còpia, distribució (comercial o no) i modificació del codi, sempre que qualsevol modificació es continuï distribuint amb la mateixa llicència

GPL. La llicència GPL no permet la distribució de programes executables sense el codi font corresponent o una oferta de com obtenir-lo gratuïtament.

Aquesta llicència va ser dissenyada originalment per Richard Stallman i el grup GNU, com a alternativa al model de programari propietari predominant. Actualment, Linux és el programa sota llicència GPL amb mes difusió.

[10] CGI

La Interfície d'entrada comú (*Common Gateway Interface*) és una important tecnologia de la World Wide Web que permet a un client (navegador web) demanar dades d'un programa executat en un servidor web. CGI especifica un estàndard per transferir dades entre el client i el programa.

Les aplicacions CGI van ser una de les primeres maneres pràctiques de crear contingut dinàmic per a les pàgines web. En una aplicació CGI, el lloc web passa les sol·licituds del client a un programa extern. Aquest programa pot estar escrit en qualsevol llenguatge que suporti el servidor, encara que per raons de portabilitat se solen utilitzar llenguatges de script.

La sortida d'aquest programa és enviada al client en lloc del fitxer estàtic tradicional. CGI ha fet possible la implementació de funcions noves i variades en les pàgines web, de tal manera que aquesta interfície ràpidament es va tornar un estàndard, sent implementada en tot tipus de servidors web.

[11] ASP

Les Active Server Pages són una tecnologia de Microsoft del tipus "costat del servidor" per a pàgines web generades dinàmicament, que ha estat comercialitzada com un annex a Internet Information Services (IIS).

La tecnologia ASP està estretament relacionada amb el model tecnològic del seu fabricant. Intenta ser solució per a un model de programació ràpida ja que "programar en ASP és com programar en Visual Basic", per descomptat amb moltes limitacions i alguns avantatges específiques en entorns web.

[12] IDE

Els IDE (*Integrated Development Environment*) són entorns de desenvolupament integrats. Permeten escriure codi Java, compilar-lo i executar-lo sense haver de canviar d'aplicació.

[13] JDK

Java Development Kit o (JDK), és un programari que proveeix eines de desenvolupament per a la creació de programes en Java. Es pot instal·lar a un ordinador local o en una unitat de xarxa.

En la unitat de xarxa es poden tenir les eines distribuïdes en diversos ordinadors i treballar com una sola aplicació.

[14] Tomcat

Tomcat (també anomenat Jakarta Tomcat o Apache Tomcat) funciona com un contenidor de servlets desenvolupat sota el projecte Jakarta a l'Apache Software Foundation. Tomcat implementa les especificacions dels servlets i JavaServer Pages (JSP) de Sun Microsystems.

Tomcat és un servidor web amb suport de servlets i JSP's. Tomcat no és un servidor d'aplicacions. El motor de servlets de Tomcat sovint es presenta en combinació amb el servidor web Apache.

Tomcat pot funcionar com a servidor web per si mateix. En els seus inicis va existir la percepció que l'ús de Tomcat de forma autònoma era només recomanable per a entorns de desenvolupament i entorns amb requisits mínims de velocitat i gestió de transaccions. Avui en dia ja no existeix aquesta percepció i Tomcat és usat com servidor web autònom en entorns amb alt nivell de tràfic i alta disponibilitat. Atès que Tomcat va ser escrit en Java, funciona en qualsevol sistema operatiu que disposi de la màquina virtual Java.

Francesc Tomàs Vila,

26 de gener de 2010