Escola Universitària
Politècnica de Mataró

Enginyeria Tècnica en Informàtica de Gestió

Web per la gestió d'activitats de muntanya

NOM: ALBERT AGUSTÍ COSTA
PONENT: JOAN JOU i MAJÓ

PRIMAVERA 2009

A la meva família i amics per la seva comprensió i companyia.

Els meus sincers agraïments al prof. Joan Jou,
per la seva direcció, ajuda i paciència per aquest projecte

Sumari

En aquest projecte s'ha plantejat una forma de generar les rutes, a peu, en bicicleta o en moto, que els usuaris de Internet realitzen en el seu temps lliure, utilitzant serveis de Internet que estan disponibles pel desenvolupament d'aplicacions. S'ha utilitzat el servei Google Maps per realitzar la gestió de les rutes i el servei Picasa Web Google per emmagatzemar les fotografies que els usuaris vulguin incloure en el seu projecte.

Sumario

En este proyecto se ha planteado una forma de generar las rutas, a pie, en bicicleta o en motocicleta que los usuarios de Internet realizan en su tiempo libre, utilizando servicios de Internet que están disponibles para el desarrollo de aplicaciones. Se ha utilizado el servicio Google Maps para realizar la gestión de las rutas y el servicio Picasa Web Google para almacenar las fotografías que los usuarios quieran incluir en su proyecto.

Abstract

This project has proposed a way to generate routes, on foot, by bicycle or by moto, that Internet users perform in their free time, using Internet services that are available for application development. We used the Google Maps service for the management of routes and services to Google Picasa Web save photos that users want to include in their project.

Índex de continguts

1 Introducció

1.1 Motivació del projecte

La motivació per realitzar aquest projecte era per fer una proposta de com les persones que en el seu temps lliure realitzen activitats de muntanya, com per exemple, rutes de llarga durada, excursions de muntanya, rutes en bicicletes, etc. puguin posar a disposició d'altre persones el seu coneixement del territori.

En un principi es va pensar en el servei de Google Maps[1] com a suport per desenvolupar la part de la creació de mapes, ja que és un servei molt conegut a Internet i on l'usuari ja es troba còmode utilitzant-lo. A més a més, és un servei de Google totalment gratuït i amb les característiques suficients com per dur a terme el projecte.

El primer avantatge de Google Maps és la capacitat d'oferir mapes interactius i raonablement actualitzats. També ofereix la possibilitat de veure els mapes de tres formes diferents: mapa dels carrers, vista satèl·lit, vista terrenal i la combinació d'aquestes vistes. Mitjançant les funcions que proporciona la API de Google Maps[2], s'ofereix al desenvolupador la possibilitat de crear una aplicació amb mapes, ja que permet obtenir informació concreta o col·locar-hi els elements del mapa (marcadors, línies, globus de text,...).

En consideracions inicials del projecte, es va plantejar la possibilitat de pujar fotografies i que es mostressin en el mapa, com un element més de la ruta, però no es volia utilitzar l'espai del servidor per emmagatzemar les fotografies dels usuaris, ja que podien encarir el cost del allotjament. Actualment, a Internet hi ha serveis d'allotjaments d'imatges, com Flirkc o Picasa Web Album[3], Aquest últim, es el que sutilitzarà per emmagatzemar les imatges relacionades amb la ruta de la nostra aplicació.

1 *http://maps.google.es*
2 *http://code.google.com/intl/es-ES/apis/maps*
3 *http://picasaweb.google.es*

L'aparença i l'estructura de la pàgina web s'ha optat per un estil semblant a un bloc, ja que actualment és un estil usable i segueix una estructura on l'usuari se sent còmode i s'acostuma ràpidament.

1.2 Objectius

La idea del projecte es basa en dues parts. Una part destinada a l'administració del site, que estaria destinada a dur una gestió dels usuaris i de les rutes. Aquesta part és d'accés restringit, per tant, només uns cert usuaris amb la credencial d'administrador hi poden accedir. Per tant, en aquesta part ha de complir amb els següents punts:

- Control d'usuaris: Només els usuaris amb la credencial d'administrador poden entrar al site i realitzar les accions d'administració. Poden entrar utilitzant una pantalla de login.
- Administració d'usuaris: s'ha de poden activar o desactivar els usuaris de la part publica, que són els usuaris que faran ús de la web. S'ha de proporcionar un sistema de cerca per facilitar la tasca de l'administrador.

L'altra part del projecte tracta de la part pública de l'aplicació. Aquesta part, és la dirigida als usuaris finals, o sigui, el que faran els usuaris de la web per a la creació de rutes i altres elements de la web que es descriuen a continuació:

- Control d'usuaris: els usuaris han d'estar registrats a la web, per tant, s'ha d'oferir l'opció de registrar-se mitjançant un formulari. Un cop registrat, ha de poder entrar a la web.
- Gestió de rutes: els usuaris registrats han de poder crear una ruta de forma online, usant els recursos que ofereix Google Maps [5]. Els usuaris que no estan registrar poden cercar les rutes, veure els seus detalls i els àlbums fotogràfics pertanyents a cada ruta.
- Gestió d'imatges: durant la creació de la ruta s'ha d'oferir la possibilitat de crear un àlbum fotogràfic i pujar les fotografies de la ruta fetes per l'usuari.

1.3 Requisits tecnològics

És demana que el client disposi d'un servidor capaç d'executar pàgines dinàmiques amb la tecnologia PHP,[4] ja sigui, IIS de Microsoft configurat per PHP però, es recomana que tingui instal·lat un servidor, per exemple, Apache[5].

Referent a la tecnologia de base de dades, al framework Symfony disposa de la possibilitat de treballar amb diversos tipus de base de dades, però es recomana un base de dades [3] MySQL .

Per accedir a l'aplicació es recomana que disposi d'una connexió ADSL per millorar la transferència dels arxius fotogràfics i disminuir l'espera en la resposta entre les peticions.

L'aplicació es pot executar en qualsevol plataforma (Microsoft Windows, *nix)que tingui instal·lat PHP, Apache i MySQL[6]. Serà executable amb qualsevol navegador d'Internet capaç d'interpretar HTML i CSS.

4 *http://www.php.net*
5 *http://www.apache.org*
6 *http://www.mysql.com*

2 Estudi de mercat

Normalment, no hi ha moltes webs dedicades a emmagatzemar rutes i publicar rutes. La majoria d'aquests sites pertanyen a associacions que organitzen activitats i usen la web per notificar el dia de la celebració i, amb un article, explicar mínimament la ruta.

S'ha escollit tres web per fer un estudi comparatiu.

2.1 La web de la FEEC

La primer web triada per l'estudi és de la Federació d'Entitats Excursionistes de Catalunya (FEEC),

Il·lustració 1. Web de la FEEC, http://www.feec.org

El site de la Federació d'Entitats de Catalunya, és una web que ofereix informació de diversos esports relacionats amb l'aire lliure, incloent senderisme, caminades, competicions , refugis, etc. També disposa d'un calendari on pots consultar les dates de celebració d'aquestes

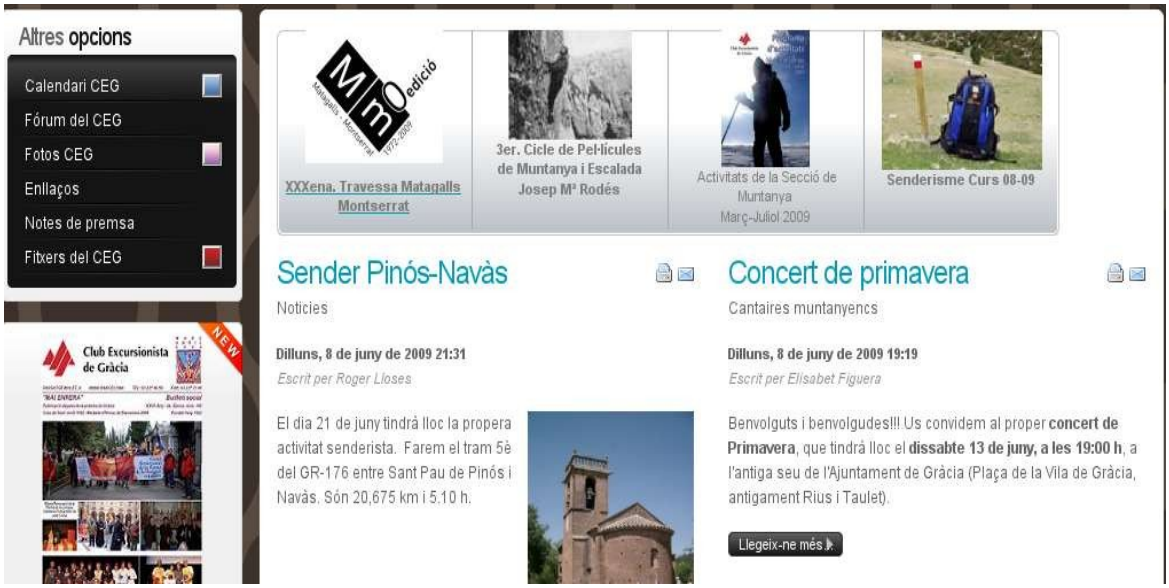
activitats, i ofereix noticies sobre diverses activitats de lleure, com es mostra en la *Il·lustració 1*. També oferix consell i suggeriments de com realitzar-les. Té un apartat d' enllaços d'interès i usa un servei de predicció meteorològic utilitzant el servei web de MeteoCat..

Respecte a l'apartat de cartografia, no mostra cap tipus d'informació ja sigui mitjançant imatges estàtiques o utilitzant mapes dinàmics com Google Maps. Per tant, la informació és unidireccional i no permet que l'usuari interaccioni amb la informació, és a dir, la web no permet rebre informació per part de la persona que navega.

En resum, la pàgina web de la FEEC només ofereix informació sobre les celebracions de les activitats de lleure i consell d'experts per als usuaris que visitin la web.

2.2 La web de CEGràcia

Aquesta és una de les entitats d'activitats d'aire lliure més conegudes a Catalunya que també s'ha inclòs en l'estudi.



Il·lustració 2: Web CEGràcia, http://www.cegracia.cat

Aquesta web es dedica a mostrar informació de les activitats de aire lliure que l'entitat organitza. Utilitza la API de Google Calendar per mostrar la informació però no permet una comunicació bidireccional. Té un disseny ben estructurat i mostra a l'usuari força informació, sense perjudicar l'estructura del disseny ni carregar massa informació la pàgina principal, com es pot veure en la *Il·lustració 2.*

En el moment de l'anàlisi no es va detectar cap mecanisme per mostrar informació cartogràfica de les rutes que es proposa en les seves activitats. Per millorar la participació de l'usuari, té un fòrum on els usuaris poden exposar les seves experiències o suggeriments.

2.3 La web de Wikiloc

Wikiloc és una web que es dedica exclusivament a la generació i gestió de rutes. No pertany a cap entitat o associació., sinó que es basa en la web 2.0 usant serveis de Internet per generar contingut.
Aquesta web no ofereix informació en format article però si que permet obtenir informació de les rutes mitjançant la cartografia de Google Maps

La generació de rutes són aportacions dels usuaris registrats de la web. Aquesta web si que utilitza la API de Google Maps per generar les rutes. Com és pot veure en la *Il·lustració 3.*

Il·lustració 3: Ruta de wikiloc, http://es.wikiloc.com

Wikiloc dóna la possibilitat que els usuaris puguin fer aportacions, pujant les rutes que ells realitzin a la web, També permet l'aportació de fotografies per aquesta ruta.

2.4 Taula comparativa

La taula que tenim a continuació és un resum de les característiques que tenen en comú i les diferències de les web analitzades amb el projecte final de carrera (Guia web).

Característiques	**Webs analitzades**			
	FEEC	Wikiloc	CEGracia	**Guia web**
Gestió administrativa dels usuaris	No	Si	Si	**Si**
Registre d'usuaris	Ni	Si	Si	**Si**
Interacció amb l'API Google Maps	No	Si	No	**Si**
Sistema de visualització de rutes	No	Si	No	**Si**
Gestió d'activitats	Si	No	Si	**No**
Interacció amb l'API Picasa	No	No	No	**Si**
Sistema de visualització de fotografies	No	Si	Si	**Si**
Sistema de cerques de rutes	No	Si	No	**Si**
Creació de rutes	No	Si	No	**Si**
Sessió d'usuari	No	Si	Si	**Si**
Interfície usable	Si	Si	Si	**Si**

Taula 2.4.1: Taula comparativa de les webs analitzades

.

Després d'observar la *Taula 2.4.1* podem concloure que la pàgina de FEEC és de les web analitzades que té menys característiques en comú amb el projecte que hem desenvolupat. La FEEC és una web estàtica amb la missió d'informar a l'usuari d'activitats en diversos àmbits, per tant, esta lluny del concepte web 2.0.

En canvi, wikiloc és la web que està més aprop de les característiques del projecte, ja que sí que té l'acció de crear i mostrar rutes a través de l'API de Google Maps, però amb la diferència que Guia Web guarda les fotografies utilitzant un servei web basant-se en l'API de Picasa Web Album.

3 Metodologia i tecnologia

3.1 Tecnologia i entorn de desenvolupament.

El projecte s'ha desenvolupant amb un sistema operatiu basat en Linux. A continuació es mostra un llistat complet de tot el software que s'ha utilitzat per desenvolupar el projecte:

Sistema operatiu: Linux Ubuntu 8.10

Software addicional inclòs en el sistema operatiu:

Edició de imatge: GIMP
Suite Ofimàtica: Open Office 3.0
Gestió del projecte: Gant project (No inclòs)
Navegador utilitzat pel desenvolupament: FireFox 3.0.X

Software usat pel desenvolupament de l'aplicació web:
Servidor web Apache 2.0.
Llenguatge web PHP, versió mínima recomanable 5.2.6
Framework Symfony ,versió mínima recomanable 1.2.4
MySQL Database 5.X i GUI Tools
IDE Eclipse PDT (es necessari la màquina virtual de Java).

Xdebug

La tecnologia escollida pel desenvolupament del projecte ha d'estar relacionada amb el desenvolupament web, per tant, entre varis llenguatges de programació es va escollir PHP. Desenvolupar un projecte té l'afegit d'utilitzar les altres tecnologies web.

CSS i XHTML, per la presentació de dades.
Javascript / AJAX, per la interacció de l'aplicació amb l'usuari.
XML per guarda informació.
SQL per fer accions amb la base de dades.

PHP/Symfony llenguatge pel desenvolupament de la part del servidor.
API Google Maps.
API Picasa Web Album.

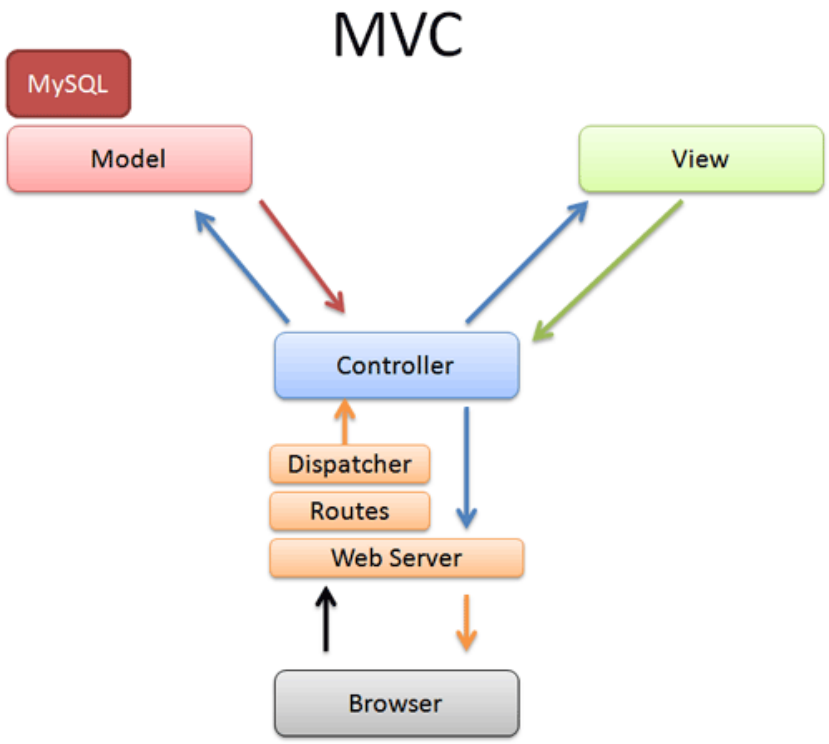
L'API de Google Maps utilitza la tecnologia CSS per mostrar els mapes i el Javascript perquè el programador pugui programar la interacció de l'usuari amb el mapa. Es va decidir utilitzar el Eclipse com interfície de desenvolupament perquè era el que millor s'adaptava al framework. A més a més, després de cercar per Internet i en fòrums relacionats amb Symfony tots acabaven recomanant aquest IDE.

Com ha navegador web utilitzat en el desenvolupaments'ha utilitzat FireFox 3.0.X[7], ja que es considera el navegador més usat en els sistemes operatius Linux. A més a més, té plugins molt útils per desenvolupament web com, el Firebug[8] i el Web Developer Toolbar[9]

3.2 Metodologia

Una de les raons per escollir al framework Symfony és que té implementat el patró Model – Vista – Controlador (MVC). Aquest patró separa les dades de l'aplicació, la interfície d'usuari, i la lògica del negoci en tres components o capes diferents. El patró MVC és un dels patrons més freqüents pel desenvolupament d'aplicacions web, on la vista genera la pàgina HTML. El model correspon a la lògica de negoci i també inclou les sentencies amb la base de dades, i el controlador és el responsable de rebre les accions de l'usuari des de la vista i de controlar el proces de l'acció, tal i com es pot veure a la *Il·lustració 3.0*

7 *http://www.mozilla-europe.org/es/firefox/*
8 *https://addons.mozilla.org/es-ES/firefox/addon/1843*
9 *https://addons.mozilla.org/es-ES/firefox/addon/60*



Il·lustració 3.0: Patró MVC

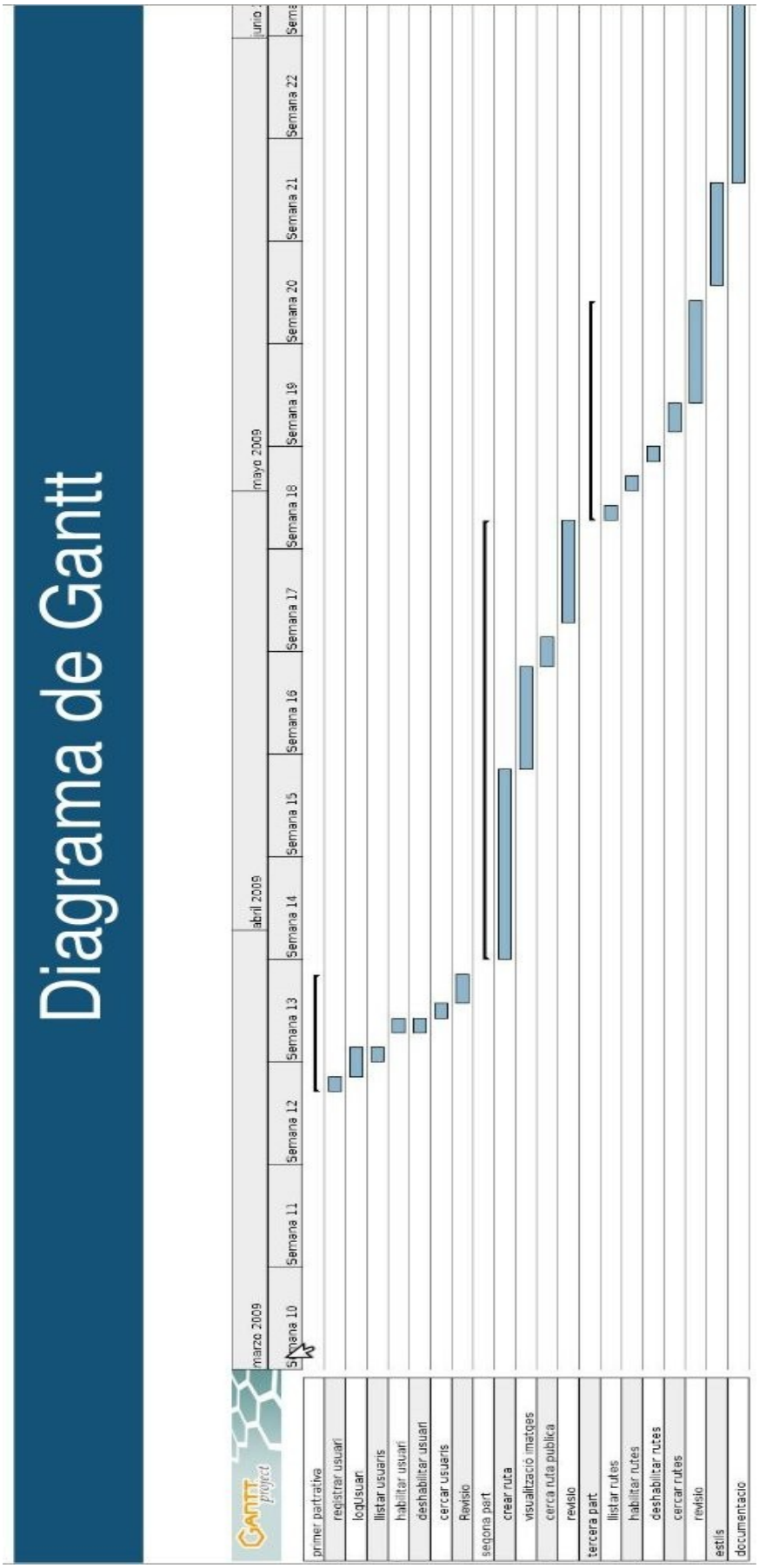
3.3 Planificació i pressupost

Per la planificació del projecte s'ha utilitzat el programa Gantt Project per planificar el temps dedicat a cada iteració. En la *Il·lustració 3.3.1* es pot veure una capturada de pantalla del diagrama de Gantt.

La primera part de la planificació, es basarà en crear l'accés dels usuari a l'aplicació i la seva gestió. La gestió es basar en habilitar-los o deshabilitar-los l'accés a la web. El final de la planificació es comprovarà la que s'hagin assolit en èxit les proves i que s'hagin complert els objectius que es demanen en els requeriments.

La segona part, es desenvoluparà els casos d'ús que es consideren que tenen més complexitat. Per això, és reserva molta part del temps disponible per realitzar per aquesta part de la planificació.

I finalment, la tercera part de la planificació, es desenvoluparà la gestió de les rutes i es revisarà tot el projecte, que compleixi tots els objectius i que funcioni correctament. Un cop comprovat aquest aspectes, es decidirà quina serà l'aparença final del projecte.



Il·lustració 3.3.1: Diagrama de Gantt

3.4 Pressupost

Per la elaboració del pressupost d'aquest projecte, es considera que la temps màxim pressupostat és de 300 hores, (El temps màxim disponible per realitzar el projecte final de carrera.)

Abans d'iniciar el projecte, es va adquirir un ordinador nou pel seu desenvolupament i es va realitzar un estudi de quin software era l'ideal per muntar l'entorn de desenvolupament, es va destinar 150 hores amb un cost de 30 € per hora, fora de les 300 hores destinades pel projecte.

Les 300 hores destinades pel desenvolupament del projecte tenen un cos de 35 € x hora

Es suposa un salari mig de 1200 € d'un enginyer en informàtica.

COST DEL PROJECTE

Cost del personal ***4.800 €***

1 persona durant 4 mesos amb un salari mig 1.200 €.

Estudi previ ***4.500 €***

Estudi del programari pel desenvolupament del projecte.
Proves per correcte funcionament de l'entorn.

Cost informàtic (*) ***499 €***

Cost del hardware	499 €
1 PC Quad Core HD 320 GB 3GB RAM.	
Cost del software	- €
Linux Ubuntu 8.10 LTS.	- €
Navegadors (Mozilla FireFox, Opera).	- €
Eines Ofimàtiques.	- €
Cost de software pel desenvolupament.	- €

Cost de l'aplicació web	***10.500 €***

Cost de 35 € hora pel desenvolupament del project

COST TOTAL DEL PROJECTE	***20.299 €***

4 Anàlisi requeriments

4.1 Estudi de requeriments

L'estudi de requeriments es realitza segons els actors principals de l'aplicació, que ho dividim en l'usuari públic i l'administrador per la part administrativa.

Els requeriments funcionals de l'usuari són:

- *Sessió de usuari*: L'aplicació web no ha de permetre realitzar certes accions si els usuaris no estan registrats, com afegir ruta, afegir àlbum. L'usuari s'ha de registrar a través d'un formulari i ha de tenir una pantalla per iniciar sessió.
- *Cercar rutes:* L'usuari podrà cercar les rutes introduint una o varies paraules per a la cerca. També ha d'haver dos criteris més per la cerca totalment opcionals, aquests són el tipus de la ruta i la dificultat. Els resultats de la cerca de rutes ha de mostrar les dades de la ruta, nom de la ruta, distancia,tipus, dificultat i sí té àlbum.
- *Inserir la ruta:* L'usuari pot inserir la seves rutes, interaccionant amb el mapa de Google Maps. L'aplicació ha de ser capaç de guardar el trajecte dibuixat en el mapa i la informació referent de la ruta. L'usuari, per realitzar aquesta acció, ha d'haver iniciat la sessió. L'aplicació ha de mostrar el trajecte introduït per l'usuari en un mapa generat per Google Maps , el nom dels àlbums fotogràfics i qui l'ha fet. Si es fa clic sobre el enllaç del àlbum, ha de mostrar les fotografies del àlbum.
- *Inserir fotografies:* L'usuari pot afegir les fotografies relacionades amb la ruta. També la de poder esborrar. Un cop acceptades totes les fotografies ja no es poden esborrar. Per realitzar aquestes accions l'usuari ha d'haver iniciat la sessió.
- *Mostrar fotografies:* l'aplicació mostrarà les dades de la fotografia i la miniatura. L'usuari també pot veure la imatge real.
- *Recuperar contrasenya:* L'aplicació ha d'oferir la possibilitat que l'usuari recuperi la seva contrasenya.
- *Registre pel usuari:* L'usuari ha d'omplir un formulari per entrar a ser membre de la web. Introduint el nom, contrasenya, ciutat, correu electrònic.

I els requeriments funcionals del administrador són:

- *Sessió de usuari:* Per accedir a les accions d'administració, el administrador ha d'iniciar sessió introduint un nom i una contrasenya.
- *Administració dels usuaris:* L'administrador ha de poder habilitar o deshabilitar els usuaris, que per certes raons, hagin de ser deshabilitats o habilitats. S'ha de mostrar en el llistat l'identificador de l'usuari, l'email i la ciutat. El sistema ha de permetre habilitar o deshabilitar els usuaris de la forma més fàcil possible.
- *Cerca usuaris:* L'administrador pot cercar els usuaris, d'aquesta manera pot realitzar les accions més ràpidament possible. Els resultats han d'estar paginats. I es poden ordenar per columnes.
- *Administració de rutes:* El administrador ha de poder habilitar o deshabilitar les rutes, que segons l'administrador cregui. S'ha de mostrar en el llistat el títol, localitat, tipus i distància de la ruta. El sistema ha de permetre a l'administrador de realitzar les accions habilitar o deshabilitar rutes de la forma més fàcil possible.
- *Cercar rutes:* L'administrador pot cercar les rutes, d'aquesta manera pot realitzar les accions més ràpidament possible. Els resultats han d'estar paginats.

4.2 Casos d'ús

Cas d'ús - Crear ruta

Actor principal: usuari

Personal involucrat i interessos:

Usuari: Vol crear una ruta utilitzant la interacció amb Google Maps

Precondicions: és necessari que l'usuari hagi iniciat sessió.

Garanties d'èxit: L'aplicació haurà guardat les dades i el trajecte de la ruta, per tant, la ruta sortirà a les cerques de rutes.

Escenari principal:

1 L'usuari executa l'acció de crear ruta.

2 El sistema mostra un mapa centrat en la ciutat que l'usuari ha introduït en el registre i un camp per buscar la ciutat.

3 L'usuari dibuixa el trajecte fent clic sobre el mapa.

3.1 El sistema calcula la distància de la ruta.

4 L'usuari introdueix el nom de la ruta, la descripció, la dificultat, el tipus de la ruta i el sistema mostra la distància del trajecte.

5 L'usuari accepta i el sistema guarda les dades.

Escenari alternatiu:

5.1 Si el sistema no pot guardar les dades, ha de mostrar un missatge d'error advertint que no s'ha pogut guardat la ruta

Cas d'ús – Crear àlbum

Actor principal: usuari

Personal involucrat i interessos:

Usuari: vol pujar les imatges referents a la ruta.

Precondicions: L'usuari ha d'haver iniciat sessió i finalitzat l'execució de crear ruta.

Garanties d'èxit: El sistema ha de mostrar les fotografies de la ruta.

Escenari principal:

1 El sistema dóna l'opció de crear un àlbum fotogràfic. És opcional per l'usuari.

2 L'usuari ha d'introduir el nom de l'àlbum.

3 L'usuari accepta el nom de l'àlbum i el sistema crear l'àlbum.

4 El sistema mostra un mapa i el títol del àlbum.

5 L'usuari introdueix el nom de la fotografia i l'arxiu fotogràfic, les coordenades són opcionals.

6 El sistema guarda la imatge a Picasa i mostra la imatge.

7 L'usuari pot tornar el punt 1.

Escenari alternatiu:

3.1 Si el sistema no pot guarda l'àlbum ha de mostra un missatge d'error informant que no ha pogut crear l'àlbum.

6.1 Si el sistema no pot guardar la imatge mostra un missatge d'error informant que no ha pogut guardar la imatge.

Cas d'ús – Mostrar Ruta.

Actor principal: Usuari.

Personal involucrat i interessos:

Usuari: Vol veure el trajecte de la ruta i la seva informació i àlbums.

Precondicions: L'usuari no fa falta que estigui registrat i ha d'haver buscat la ruta. (acció *cercar ruta*)

Garanties d'èxits: El sistema ha de mostrar el trajecte de la ruta , la seva informació i un llistat dels àlbums de la ruta.

Escenari principal:

1 El sistema ha de mostrar un llistat de les rutes

2 L'usuari fa clic per veure la ruta.

3 El sistema mostra el trajecte de la ruta en un mapa, el nom de la ruta, la descripció, la distància, el tipus de la ruta, la dificultat de la ruta i un llistat de noms d'àlbums de la ruta.

Escenari alternatiu:

3.1 Si el sistema no pot mostra la ruta ha d'avisar amb un missatge d'error dient que no es pot mostrar la ruta.

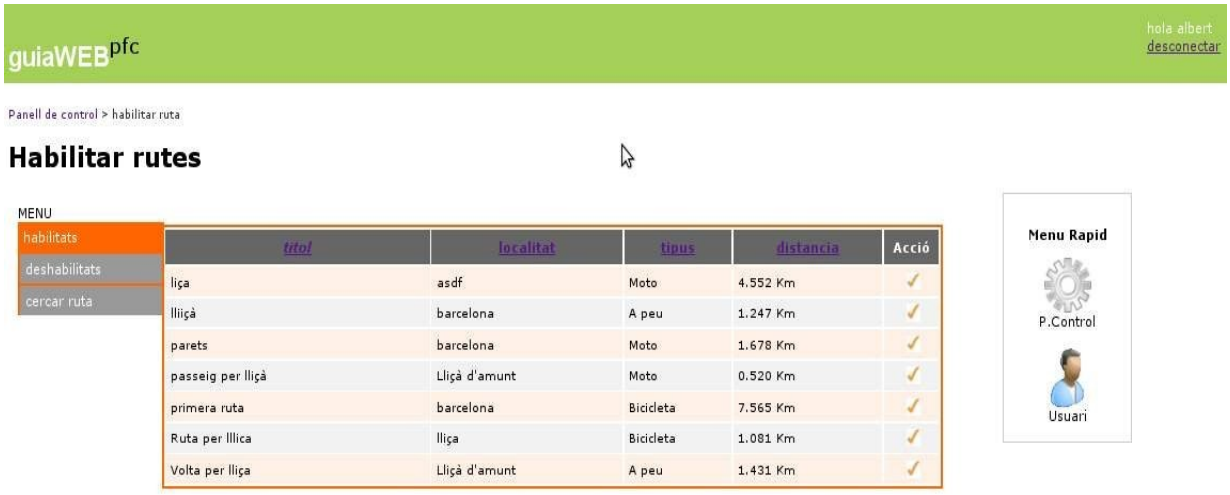
5 Disseny de l'aplicació

5.1 Disseny de la interfície gràfica

L'aplicació consta de dos entorns diferenciats: el de l'usuari i la part administrativa.

En les dues interfícies gràfiques es busca la simplicitat en l'ús.

Per tant, en la pantalla administrativa es va optar per un disseny on agrupar totes les seccions en una pantalla de control. En les seccions interiors, s'ha optat pel fil de Ariadna[10] per millorar la navegabilitat. També s'ha inclòs un menú vertical, a la part esquerra de la pantalla per oferir a l'administrador dreceres, per anar a altres seccions de diferent temàtica de la part administrativa, o sigui, si estem a la part de deshabilitar usuaris en el menú vertical hi ha l'opció de deshabilitar rutes, com es pot veure en la *Il·lustració 5.1.1.*



Il·lustració 5.1.1: Interfície de l'administració

En una sola pantalla, s'ha agrupat les seccions de la mateixa temàtica en pestanyes per millorar la usabilitat, és a dir, que si estem en la secció de deshabilitar rutes, sota de la

10 *El terme en anglès és breadcrumb, que prové del conte clàssic Hansel i Gretel. En espanyol, en comptes del rastre d'engrunes de pa, s'ha traslladat el terme evocant el fil que Ariadna que va deixar a Teseu perquè trobés el camí de sortida en el laberint del Minotaure.*

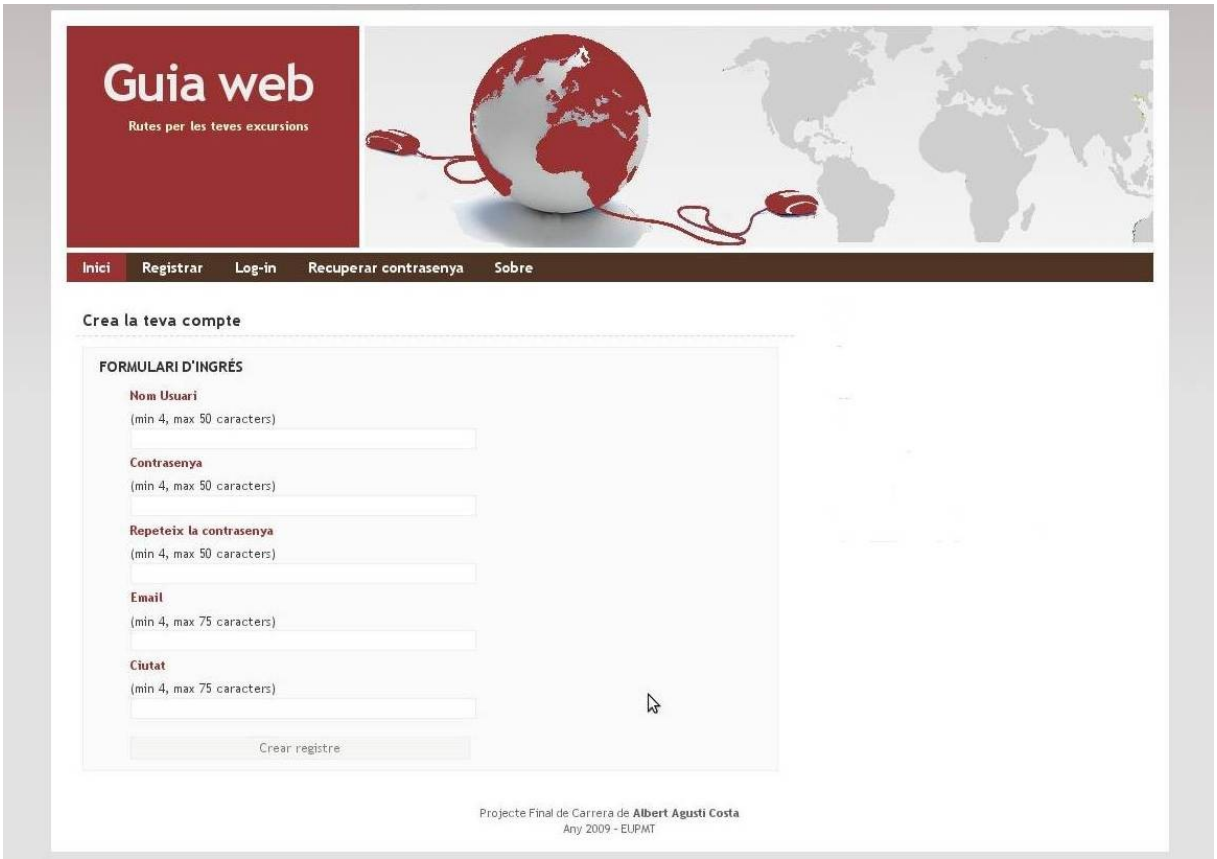
pestanya seleccionada hi ha les altres seccions relacionades amb l'administració de rutes.

S'ha fet varies propostes pel logotip de la web de la part pública. El logotip de la web considero un element important ja que és l'element més representatiu de la web. En la Il·lustració 5.1.2 es presenten dues propostes de logotips. I la proposta utilitzada en el projecte es la primera. Es van adaptar els estils de la web al logotip.

Il·lustració 5.1.2: Propostes de logotips

Navegant per Internet es poden veure diferents estils per distribuir l'espai d'una pàgina web. Podem trobar que el Logotip ocupi un terç de la pantalla, això es degut per l'augment de les pantalles i el menú de navegabilitat estigui situat a la part superior de la pàgina. Normalment aquest dissenys estan destinats a webs empresarials, que aprofiten les capçaleres per promocionar els seus productes estrelles.

També hi ha altres web, en format bloc (sobretot de tipus personals) que només disposen d'un títol i no tenen logotip, això provoca que la taxa d'abandonament dels usuaris sigui major, ja que la web pot transmetre un sensació de poca serietat perquè no hi ha un element representatiu en pàgina web.

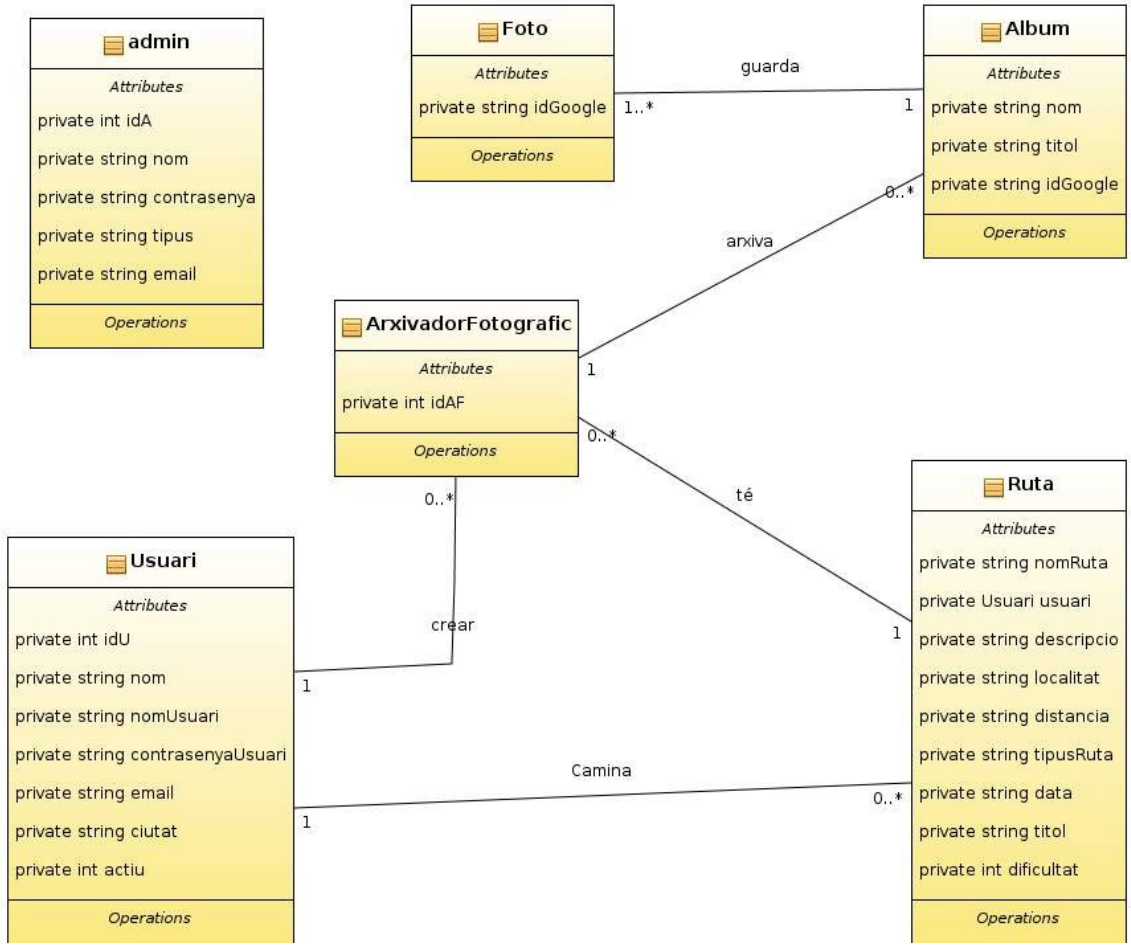


Il·lustració 5.1.3: Web de Guia Web - Secció Registrar usuari

El disseny web utilitzat en el projecte és un entre mig d'aquest dos explicats en el paràgraf anterior. En aquest disseny el logotip ocupat un 20% de la pantalla i el menú de navegabilitat és de tipus horitzontal i situat sota el logotip. Aquest disseny és el més comú de la web. En la *Il·lustració 5.1.3* es pot veure el disseny de la web amb la secció de registrar usuari.

5.2 Disseny de domini

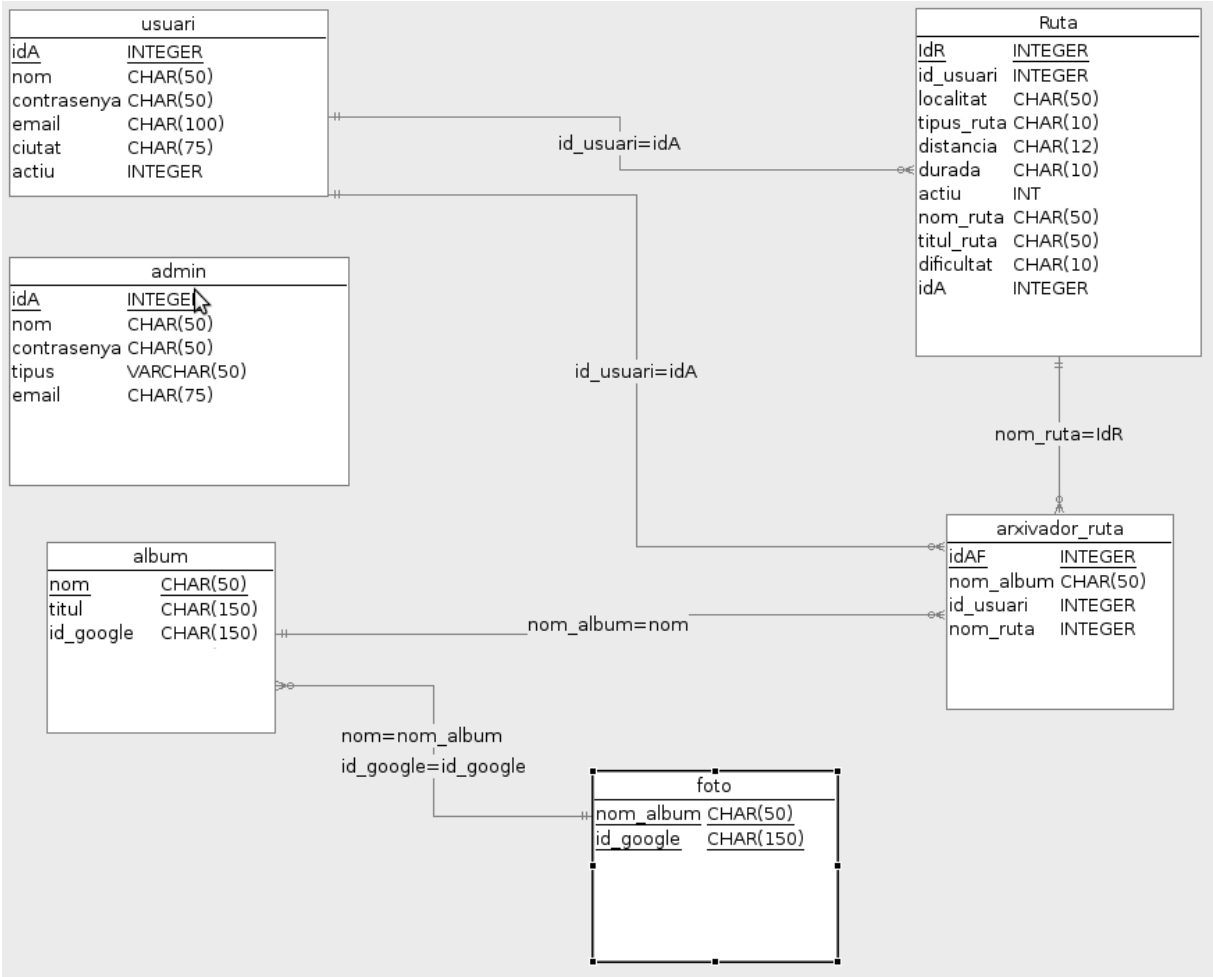
En la *Il·lustració 5.2.1* es pot veure el domini de l'aplicació, les classes que són necessàries perquè l'aplicació funcioni. Es pot veure com la classe *Admin* no té cap relació amb cap altre classe i només 'utilitza per controlar l'accés de l'administrador. La classe usuari és la que crea la classe ruta, que emmagatzema les coordenades de l'excursió i altres dades. La classe ArxivadorFotografic serveix els àlbums fotogràfics de la ruta. La classe Foto emmagatzema l'identificador de la foto de Google, ja que totes les altres dades es guarden en el servidor de Picasa Web Album.



Il·lustració 5.2: Domini de l'aplicació

5.3 Disseny de la base de dades

La *Il·lustració 5.3.1* és el disseny de la base de dades que guarda la informació del projecte .La taula *admin* guarda la informació del administrador. La taula *usuari* guarda la informació dels usuaris. La taula ruta guarda la informació de la ruta però no guarda les coordenades de la ruta, ja que aquesta informació es guarda en un fitxer apart. La taula *album* guarda en nom, el títol i l'identificador de Google. La taula Foto guarda la referencia a quin album pertany i l'identificador de Google. Les altres dades relacionades amb la fotografia són extretes de Google.



Il·lustració 5.3.1: Diagrama de la base de dades

6 Desenvolupament amb Symfony

6.1 Què és Symfony?

Symfony és framework basat en PHP patrocinat per Sensio, una agència web francesa que va desenvolupar un framework propi per desenvolupar les seves aplicacions web. Amb la iniciativa de publicar el seu framework la comunitat va començar a fer suggeriments per anar-lo millorant. Actualment, Symfony [1] disposa d'una versió molt estable i les versions que surten només són per substituir o afegir funcionalitats. Com es veu en la *Il·lustració A1.1*

Il·lustració A1.1 : Logo de Symfony

Symfony es considerat com l'alternativa a PHP de Ruby.

6.2 Introducció de Symfony

Symfony és uns dels molts frameworks [2] en PHP que han aparegut en els últims anys però és el que esta atraient més adeptes.
La idea de Symfony és simplificar el desenvolupament de les aplicacions web automatitzant alguns dels patrons de programació, com el Model – Vista - Controlador, per resoldre les tasques més comunes. A més a més, Symfony facilita la programació d'aplicacions, ja que encapsula les operacions complexes en instruccions molt més senzilles, com es podrà veure més endavant.

Filosofia de Symfony

- Re-utilitzar el codi
- Desenvolupar aplicacions de manera àgil i entenedora.
- Utilitzar el millors components actualment disponibles.

- No re-inventar la roda, el programador no ha de tornar a programar el que ja esta solucionat.
- S'ha de concentrar en el problema no en el codi.

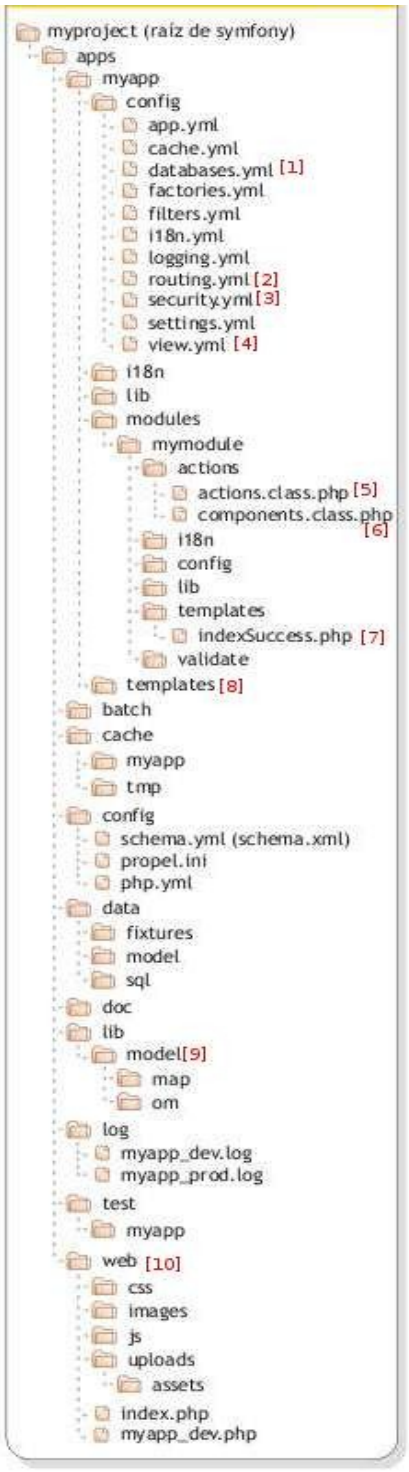
6.3 Prerequisits de Symfony:

Per començar a desenvolupar aplicacions web amb Symfony [9] s'ha de complir uns certs requisits:

- L'equip ha de tenir instal·lat Apache o qualsevol servidor de aplicacions amb soport PHP.
- L'equip ha de tenir instal·lat PHP, com a mínim la versió 5.2.4.
- L'equip ha de tenir instal·lat algun gestor de base de dades (MySQL, SQLite, MS-SQL entre molts altres).
- Per simplificar la instal·lació es recomanable tenir al framework de components de PHP anomenat PEAR.
- L'equip ha de tenir instal·lat Maquina Virtual de Java.
- Tenir instal·lat Xdebug per debugar per l'aplicació.

6.4 Estructura d'una aplicació amb Symfony

L'estructura de directoris d'un projecte Symfony [10] es molt llarga i per defecte ja té un gran número de fitxers de configuració, En la Il·lustració A1.2, es pot observar l'estructuració de directoris i també s'indica els més utilitzats en un projecte Symfony.

Estructura de directoris d'un projecte Symfony	**Fitxes més usats:**
 *Il·lustració A1.2: Directori Symfony*	[1]: Es on es guarda la connexió i la contrasenya de la base de dades. [2]: És on es configuren els enrutaments Http del mod_rewrite de Apache. Cada vegada que crees una acció s'ha de crear un enrutament per l'acció, si no ten una regla general. [3]: si l'aplicació utilitza permisos per realitzar accions, en aquest fitxer guarda les accions associades a les credencials. [4]: Guarda la configuració de quin layout pertany a casa acció. Si la nostra aplicació te diferents layout s'ha de configurar per que aquella acció carregui el layout que li correspon. [5]: Hi ha les accions del controlador. [6]: Es com un controlador que no te model de negoci, com per exemple, carregar ultimes rutes inserides. [7]: En aquest directori es creen les plantilles per cada acció. [8]: Hi ha els layouts de l'aplicació. [9]: Hi ha les classes del domini [10]: Es creen les pàgines inicials de l'aplicació i és on es guarden les fulles d'estils, el Javascript i les imatges de la aplicació web.

Desenvolupament en Symfony

En aquest apartat, es descriu la manera de desenvolupar un projecte amb Symfony i quins arxius s'han de modificat. Aquesta guia no es vol ser un manual de referència sinó, només, una introducció a aquest framework.

Aquesta guias explica els passos essencials de com desenvolupar un projecte amb Symfony. Esta dividida en tres parts, com el patró MVC: la primera és la part del controlador, la segona el domini i la tercera la vista.

6.5 Controlador de Symfony

El controlador és el que respon a les accions dels usuaris i crida al domini i vistes. Les accions es troben en el fitxer *actions.class.php,* en el directori *app/nomdelmodul/modules/actions.*

La nomenclatura de la classe controlador ha de ser **nomAcció*Actions*** , per exemple *rutaActions*, en aquest fitxer es guarden totes les accions relacionades amb les rutes, com , per exemple: executeCrearRuta, executeGuardarRuta, etc. La declaració de cada acció es declara com ***execute*NomAcció,** com es pot veure el *Codi 6.1.1* Symfony es sensible a les majúscules i minúscules, per tant, s'ha de tenir una especial atenció en no cometre errors d'aquest tipus.

```
<?php
class rutaActions extends sfActions{
      public function executeGuardarRuta(sfWebRequest $request){
```

Codi 6.1.1: Nomenclatura del fitxer accions ruta.

6.5.1 Paràmetres de la petició

El objecte *$request* que rebem a través de les accions es l'encarregat de dur els paràmetres de la petició http a l'acció. Per tant, si volem accedir al valor dels paràmetres hem d'utilitzar el mètode *getParameter('Nom del parametre')* de l'objecte *request.* Aquest mètode facilita molt la tasca de recuperar els paràmetres de la petició. Peró Symfony ofereix una recuperació de paràmetres més precisa segons les nostres necessitats. Ens pot retornar un vector de tots els paràmetres de petició amb el mètode *GetParameters*(), els paràmetres tipus POST es poden

recuperar amb el mètode *getPostParameter('Nom del paràmetre')* o recuperar els paràmetres tipus GET amb el mètode *getGetParameter('Nom del paràmetre').* Aquests mètodes també tenen els seus respectius mètodes per tornar tots els paràmetres de la petició en un vector.
En el *Codi 6.1.1.2 é*s un exemple complet de com recuperar paràmetres en una acció.

```
public function executeLlistarRutaHabilitada(sfWebRequest $request){
       $this->camp = $request->getParameter('camp');
       $this->ordenacio = $request->getParameter('ordenacio','descendent');
       $this->pagina = $request->getParameter('page', 1);

       $this->ruta_pager = RutaPeer::llistar(1, $this->pagina, $this->camp,
       $this->ordenacio);

       return sfView::SUCCESS;

  }
```

Codi 6.1.1.2 Exemple de com recuperar els paràmetres en Symfony.

Com es pot veure el *Codi 6.1.1.2,* hi ha un segon paràmetre en la funció *getParameter(),* aquest segon camp és el valor per defecte si la petició no portés al paràmetre. També existeix la funció *hasParameter('Nom del parametre')* per comprovar si en la petició hi ha al paràmetre que busquem.

6.5.2 Tractament de la petició

Symfony també disposa de mètodes per distingir quin tipus de petició en arribar al controlador, si és del tipus GET o POST.

```
if($this->getRequest()->getMethod() == sfRequest::POST){
```

Codi 6.1.2.1: Determina si es una petició POST

En aquest punt, ja sabem recuperar els paràmetres i distingir quin tipus de petició rep l'acció. Una acció pot acabar produint una vista o anat a una altra acció. Per redirigir cap a una altra acció, hi ha dos mètodes, *redirect* i *forward*.

El mètode *forward* redirigeix cap una altra acció sense produir cap canvi de pantalla del navegador del usuari. En canvi, el mètode *redirect,* també redirigeix la acció cap a una altra acció, però aquest redireccionament produeix un canvi de pantalla en el navegador de l'usuari.

```
$this->redirect('@upload_imatge?idArx='.
                          $arxivadorFoto->getIdAf().'&nomRuta='.
                          $this->nomRuta.'&nomalbum='.$album->getNom().
                          '&referer=crearAlbum');
```

Codi 6.1.2.2: Redireccionament amb el mètode redirect

En el *codi 6.1.2.2* és pot veure, com un exemple de redireccionament i com s'afegeixen nous paràmetres a la petició, unint el valor de la variable amb el format que té una petició http. La ruta cap on s'ha de redirigir la petició es representa amb la cadena @upload, que es una reescritura o aliàs de la ruta. La llista de tots els aliàs estan en el fitxer routing.yml

6.5.3 El fitxer routing.yml

Tots els fitxers de configuració de Symfony estan en format YMAL[11]. Aquest format de configuració és més fàcil de llegir pels humans ja que simplifica la lectura i l'escriptura del fitxer, ja que s'estructura i és tabula amb espais, què és un caràcter estàndard per tots els editors de programació.

Symfony també està preparat per interpretar urls amigables, que són aquelles que no tenen paràmetres. sinó que els paràmetres formen part de url. Aquestes urls són conegudes com urls amigables..
El fitxer routing.yml és l'encarregat d'emmagatzemar totes les regles de l'aplicació i saber a quin aliàs esta associat.

L'objectiu del aliàs només és facilitar la programació al programador.

```
upload_imatge:
  url:  /imatges/uploadImatge/*
  param:  { module: imatges, action: uploadImatge }
```

11 *http://www.yaml.org*

Codi 6.1.3.1: Una regla del fitxer Routing.yml

En el C*odi 6.1.3.1*, es pot veure l'àliess *upload_imatge* utilitzat per crear una url amigable per l'acció *executeUploadImatge*. En el fitxer routing.yml, és molt important que l'aliàs no tingui cap espai de tabulació, sinó, dóna error de Routing en executar l'aplicació.
Al paràmetre URL correspon a la url que es veu en la barra de direcció del navegador i l'asterisc representa que la url por variar (accepta varis paràmetres).
Al paràmetre PARAM és per indicar quin mòdul i acció es refereix la url. En el *Codi 6.1.3.1,* la url es dirigeix el mòdul imatges a,'acció executeUploadImatge.
El fitxer routing.yml també pot obligar quins paràmetres són obligatoris, que apareguin a la URL i definir el valor per defecte.

```
cercarRuta:
  url:  /ruta/cercarRutaAdvance/:page/*
  param:  { module: ruta, action: cercarRutaAdvance, page: 1 }
```

Codi 6.1.3.2: Url amb paràmetres

En el *Codi 6.1.3.2,* és un exemple de com es crear una regla amb camps obligatoris en el fitxer routing.yml, en aquest cas, s'obliga que en la url hi hagi un paràmetre obligatori, és el número de la pàgina que es representa en la paràmetre URL com *:page* i s'obliga que tingui el valor per defecte 1, encara que nosaltres no li passem al paràmetre *page.*

En aquí volem recordar que els espais són molt importants, si escrivim *"page 1}"* dóna error en el routing.yml i la forma correcte és "*page: 1 }"*, darrera del 1 ha d'anar un espai.

6.5.4 Classe myUser.php

La classe myUser.php correspon a l'objecte Session de PHP, però s'ha ampliat les funcionalitats, per exemple, afegint-li seguretat. Com es mostra en el *Codi 6.1.4.1*

```
class myUser extends sfBasicSecurityUser{

      public function signIn($admin){
            //el posat con logar
            $this->setAuthenticated(true);
```

```
            //afegim el credencial de admin
            $this->addCredential('admin');
            //guardem el nom i l'id en les seves credencials
            $this->setAttribute('admin_id', $admin->getIdA(), 'admin');
            $this->setAttribute('admin_nom',$admin->getNom(), 'admin');
      }
```

Codi 6.1.4.1: Mètode de la classe myUser

En el C*odi 6.1.4.1*, és un exemple que mostra el mètode utilitzat per controlar l'accés, quan l'administrador s'identifica a l'aplicació web. En aquest cas només s'utilitza una credencial, Si en l'aplicació es volgués afegir més credencials, només seria fer una consulta a la base de dades per conèixer la seva credencial i posar el valor de la consulta en el mètode *addCredencial($variable).* Afegir un altre credencial a una acció es tan senzill com editar el fitxer security,yml i escriure la nova credencial.

Les variables que es guarden a la sessió de PHP, en Symfony es corresponen el mètode *setAttribute(nom varialble).*

6.5.5 Fitxer security.yml

El fitxer *security.yml* és on es configuren quines accions tenen credencials. Si una acció no té credencial l'usuari pot accedir sense identificar-se, en canvi, si una acció té una credencial només l'usuari que tingui aquella credencial hi podrà accedir.

Per tant, en el fitxer security,yml s'ha d'escriure el nom de l'acció i dos punt: posar en *on* en al paràmetre *is_secur*e i el nom de la credencial en al paràmetre *credentials.* És molt important respectar la tabulació amb espais. El *Codi 6.1.5.1* és un exemple del fitxer *secutiry.yml*.

```
logout:
  is_secure: on
  credentials: admin
```

Codi 6.1.5.1: Exemple de regla de seguretat pel métode logout

Aquesta regla diu que per executar l'acció *logout* l'usuari ha de tenir la credencial *admin*. Si volguéssim afegir una nova credencial a l'acció seria escriure:

credencials: admin, novacredencial

6.5.6 Fi de les accions

Les accions poden acabar redirigint-se cap a una altra acció, com ja hem vist, o generant una vista. Per generar una vista, l'acció a d'acabar en *return sfView::SUCCESS;* si tot ha anat correctament, Symfony buscar la vista *nomAccióSuccess*.php coincidint en l'acció *executeNomAcció*. Si acaba en return *sfView::ERROR;* hi ha hagut algun problema a l'acció, Symfony buscar la vista nomAccióError.php

Per exemple, si tenim l'acció *executeCercarRuta* i tot ha anat bé, Symfony buscarà una plantilla anomenada *cercarRutaSuccess*.

Symfony també permet personalitzar els resultats de les plantilles escrivint el nom return 'resultatPlantilla', Llavors, Symfony buscar la plantilla *nomAccióResultatPlantilla*. Per exemple, si tenim l'acció *executeCercarRuta* i l'acció retorna return Resultats; Symfony buscarà una plantilla anomenada cercarRutaResultats. És molt important respectar les majúscules i les minúscules.

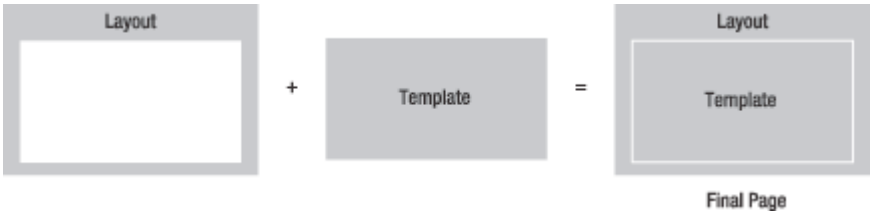
Si el final d'una acció no s'escriu el retorn, Symfony interpreta que tot ha anat bé per defecte.

6.6 Vista de Symfony

La vista s'encarrega de produir les pàgines que es mostren com a resultat de les accions. La vista en Symfony està composta per diverses parts, cadascuna d'elles especialment preparada perquè pugui ser fàcilment modificable per la persona que normalment treballa en l'àmbit del disseny de les aplicacions.
Les vistes també han de seguir una nomenclatura establerta pel framework. El nom de la vista esta format pel nom de l'acció i el resultat si es success o error, entre altres. Per tant, la plantilla de l'acció *executeCercarRuta* sería *cercarRutaSuccess.php* si tot ha anat bé.

Els dissenyadors web normalment treballen amb les plantilles (que són la presentació de les dades de l'acció que s'està executant) i amb el layout (que conté el codi HTML comú a totes les pàgines), com es pot veure en la *Il·lustració 6.2.1*

Els programadors normalment centren el seu treball a la vista en els fitxers de configuració YAML (que permeten establir opcions per a les propietats de la resposta i per altres elements de la interfície) i en l'objecte resposta.

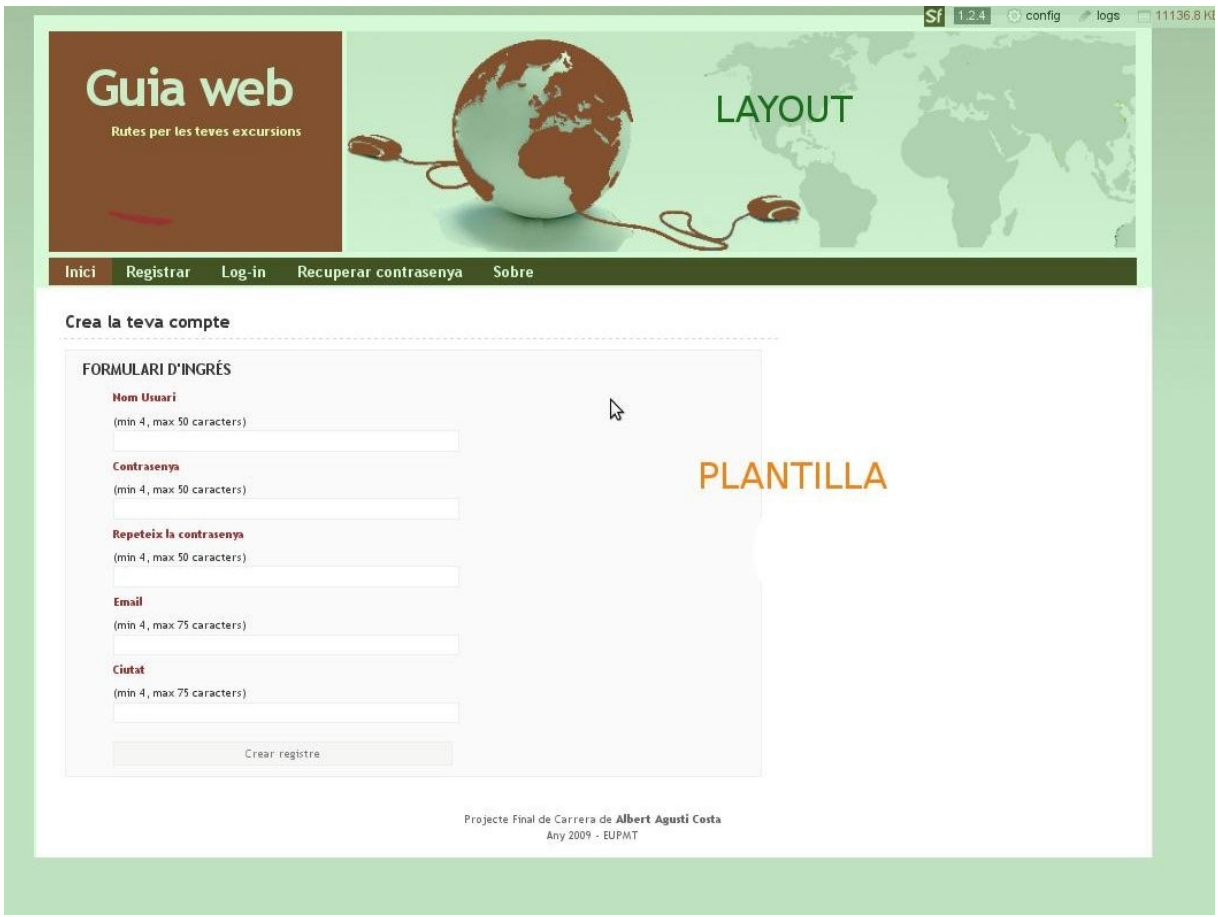


Il·lustració 6.2.1: Plantilla decorada amb un layout

Les plantilles es poden dividir en més trossos anomenat elements parcials.
Un element parcial és un tros de codi de plantilla que es pot reutilitzar. Per exemple, en l'acció cercar ruta produeix un resultat que s'ha de mostrar en HTML, aquest codi HTML és el mateix per cada resultat. Aquest és un cas perfecte per utilitzar un element parcial, ja que es reutilitzar el codi HTML per duplicar.

Il·lustració 6.2.2: Elements parcials

Igual que les plantilles, els elements parcials són fitxers que es troben al directori *templates /*, i que contenen codi HTML i codi PHP. El nom del fitxer d'un element parcial sempre comença amb un guió baix (_), la qual cosa permet distingir als elements parcials de les plantilles, ja que tots es troben en el mateix directori *templates /*.



Il·lustració 6.2.3: Capa vista del projecte

A continuació en la *Il·lustració 6.2.3* es mostra com l'aplicació web esta formada per capa vista.

Es pot veure com el logotip imatge, el nom del projecte i el menú de navegabilitat general pertanyen el layout, ja que no sofreix canvi en tot el projecte. Peró la part principal de l'aplicació, que canvia constantment s'ha utilitzat una plantilla.

Perquè des de la vista es pugui accedir les variables del controlador han de ser un atribut del controlador, o sigui, tot element que es vulgui tenir accés des de la vista s'han de declara de la següent manera: *$this->variable1*, i des de la vista es fa la crida com *$variable1*

6.7 Model de Symfony

En la capa Model de Symfony ens centrarem en dues classes que es on hi ha la lògica del domini i en troben en el domini lib/model

Per tant, per cada taula de base de dades tenim 4 fitxers

```
lib/model/
Ruta.php
 RutaPeer.php,
lib/model/map
 BaseRuta.php
 BaseRutaPeer
```

El programador només s'ha de centrar en les classes del model, o sigui, la Ruta.php i la RutaPeer.php.

Les classe Ruta.php és la que conté els mètodes de la lògica de negoci, per exemple, si Ruta.php li passem les coordenades en un fitxer aquest és l'encarregat d'interpretar-lo. Symfony genera per defecte el mètodes getters i setters dels atributs del objecte que hereden de la classe BaseRuta.php.

Les classes peer, com RutaPeer.php, són les encarregades d'interrogar la base de dades i obtenir els registres. En el *Codi 6.3.1* es pot veure com el mètode de existeixRuta funciona.

```
public static function existeixRuta($nomdelaruta){
            $nomdelaruta= str_replace(' ','_',trim($nomdelaruta));
            return self::retrieveByPK($nomdelaruta);

      }
```

Codi 6.3.1: Mètode de RutaPeer.php

En aquest mètode es pot veure com recuperen un objecte ruta a través del mètode de Symfony retrieveByPK(), que retorna un objecte ruta si el troba a la base de dades.

Les classes Ruta.php y RutaPeer.php hereten de les seves respectives classes Base. La raó d'aquesta herència entre classes, es mantenir una independència total amb la base de dades, ja que si la base de dades sofreix canvis constants, afegint o treien camps, amb la instrucció següent es torna a generar la capa persistència. Això es possible gràcies el ORM [11] de Symfony.

```
> php symfony propel:build-model
```

6.7.1 Formularis de Symfony

La classe formulari

Aquesta secció no es vol convertir en una guia de referència dels formularis[12], però si un punt de partida per entendre com funciona aquest element en Symfony. En el model de Symfony hi ha un altre element molt utilitat, els formularis. Symfony dóna resposta a una des les necessitats de la programació web, la programació de formularis. La classe formulari hereta de la classe sfForm de Symfony.

Tots els formularis s'ha de situar a el directori lib/form. La creació d'una classe formulari en Symfony , es crear un arxiu amb NomFormulariForm, com es pot veure al *Codi 6.3.1.1.*

```
class LoginusuForm extends sfForm{
```

12 *http://www.librosweb.es/symfony_formularios/*

```
        public function configure(){
                ...
        }
}
```

Codi 6.3.1.1: Estructura d'un formulari en Symfony.

Com es pot veure en el *Codi 6.3.1.1* la funció *configure()* és la funció on es configura el formulari. La configuració del formulari es basa en 2 punts. El camps que té el formulari i el validadors dels camps.

Tot formulari té un esquema de objectes anomenat `widgetSchema` en aquest objecte s'afegeixen els elements dels camp i també es poden modificar els atributs HTML, com es pot veure en el *Codi 6.3.1.2.*

```
$this->setWidgets(
 array(
  'nom'      => new sfWidgetFormInput(
    array(),
    array('id'=>'nom',
          'class'=>'element text medium',
          'maxlength'=>'50')),
  'contrasenya'=> new sfWidgetFormInputPassword(
       array() ,
      array('id'=>'contrasenya',
            'class'=>'element text medium',
            'type'=>'password',
             'maxlength'=>'50')),
));//fi camps
```

Codi 6.3.1.2.: Configuració del formulari Login

La classe `sfWidgetFormInput` és el que equival l'etiqueta HTML **<input type="text"/>** i el classe `sfWidgetFormInputPassword` és el que equival l'etiqueta HTML **<input type="password"/>.** Tots les etiquetes HTML tenen el seu equivalent en Symfony.

També permet configurar les etiquetes del formulari segons el camp, com es pot veure en el *Codi 6.3.1.3*

```
            $this->       setLabels(array(
                  'nom'   => 'Nom Usuari',
                  'contrasenya' => 'Contrasenya',
            ));//fi labels
```

Codi 6.3.1.3: Configuració de les etiquetes del formulari Login.

Els identificadors dels elements, per exemple "contrasenya", han de coincidir tant el mètode `setWidgets` i el mètode `setLabels` perquè la classe formulari associi bé els elements.

Les validacions de les elements incorporats a l'esquema del formulari es fan a través del mètode *setValidators*(), on es configuren les validacions dels camps, com per exemple, el camp nom i contrasenya són camps que obligatòriament no poden ser buits, han de tenir una longitud mínima i una longitud màxima, En el *Codi 6.3.1.4* es pot veure com es configura els validadors pels camps nom i contrasenya amb els missatges de alerta.

```
$this->setValidators(array(
 'nom'=>newsfValidatorString(
   array('required'=>true,
         'min_length' => 4,'max_length'=>50)
   ,array(
              'min_length'=>'El nom "%value%" es massa curt. Longitud minima 4
caracters',
          'max_length'=>'Longitud mÃ xima 50 caracters',
          'required'=> 'El camp nom es obligatori',
   )),
 'contrasenya' => new sfValidatorString(
   array('required'=>true, 'min_length' => 4, 'max_length'=>50)
  ,array(
       'required' => 'La contrasenya no es obligatoria.',
       'min_length'=>'Longitud minima 4 caracters',
       'max_length'=>'lLongitud mÃ xima 50 caracters',
   )),
));//fi dels validadors
```

Codi 6.3.1.4: Validadors del formulari Login

El Formulari en el controlador.

Les crides del formulari en el controlador es fan igual que en un objecte normal del domini. Els formularis disposen d'un mètode *bind()* que serveix per associar les dades que arriben a través de la petició. Un cop associades les dades el formulari pot realitzar les validacions de les dades, o sigui, comprovar que les dades siguin correctes segons els validadors. En el *Codi 6.3.1.5* es pot veure com es treballa en un formulari en el controlador.

```
public function executeLogin(sfWebRequest $request){
$this->getRequest()->getMethod() != sfRequest::POST;
$this->formulariLogin = new LoginusuForm();


if($this->getRequest()->getMethod()== sfRequest::POST){
  $this->formulariLogin->bind($request->getPostParameter('usuarilogin'));
              if( $this->formulariLogin->isValid() ){
                    ...
              }
}
```

Codi 6.3.1.5 : *Controlador Login.*

És molt important que el formulari sigui un atribut del controlador amb el *$this->formulariLogin,* ja que es la forma de com accedeix des la vista.

El Formulari en la vista.

En la vista, la crida del formulari que hem declarat en el controlador es fa mitjançant la crida de la variable del formulari, per tant, en el controlador hem declarat la variable *$this->formulariLogin,* la crida en la vista ha de ser `echo $formulariLogin->render()`

El mètode *render()* s'utilitza crear el codi HTML del formulari. Per defecte, el formulari t'ordena els formularis pel ordre que els hem posat en el mètode *configure()* i en una llista desordanda (**
**).

Per més detalls de com mostrar els formularis de Symfony heu de consultar el capitol 3 del llibre de formularis de Symfony[13].

13 *http://www.librosweb.es/symfony_formularios/capitulo3.html*

6.8 Components de Zend en Symfony.

Registrar un component Zend en Symfony

Uns dels lemes de Symfony es "*no s'ha de reinventar la roda*", per això, és molt senzill incorporar components d'altres frameworks a Symfony. Un dels frameworks amb majors components és Zend Framework, per això, és molt senzill incorporar elements d'aquest framework.

Els components de Zend [18] s'han de guarda en el directori *lib/vendor/Zend.* Un cop guardats en el directori indicat s'han de registrar en el projecte, en el fitxer de configuració de Symfony *ProjectConfiguration.class.php* situat en el directori *config* de l'arrel del projecte. En el *Codi 6.4.1* es pot veure com es registre el component *Zend Mail.*

```
static protected $zendLoaded = false;

static public function registerZend(){
 if (self::$zendLoaded){
  return;
}
 set_include_path(sfConfig::get('sf_lib_dir').'/vendor'.PATH_SEPARATOR.get_include
_path());

 require_once sfConfig::get('sf_lib_dir').'/vendor/Zend/Loader.php';
 Zend_Loader::registerAutoload();
 self::$zendLoaded = true;
}
```

Codi 6.4.1: Registrar un component de Zend en Symfony.

Utilitzar un component Zend

Un cop que el component de Zend s'ha registrat en el fitxer de configuració del projecte, ja el podem utilitzar. Fer la crida del component és molt senzill, només s'ha d'escriure "`ProjectConfiguration ::`" i el nom del mètode declarat en el *ProjectConfiguration.class.php* i començar ha utilitzar normalment el component. Com es pot veure en el *Codi 6.4.2*

```
ProjectConfiguration::registerZend();
...
$tr = new Zend_Mail_Transport_Smtp('smtp.gmail.com',$config);
...
```

Codi 6.4.2: Ús de l'objecte de Zend Mail en Symfony.

Com es pot veure, s'ha utilitzat el mètode de registerZend() per carregar l'objecte Zend Mail, un cop carregat aquest objecte el tractament del component és el mateix que si estiguéssim utilitzant al framework de Zend.

6.9 ORM de Symfony

Un ORM consisteix en una capa entre el domini de l'aplicació i la base de dades que permeten accedir a les dades a través d'unes classes.

Un dels avantatges d'utilitzar aquestes capes d'abstracció d'objectes / relacional és que evita utilitzar una sintaxi específica d'un sistema de bases de dades concret. Aquesta capa transforma automàticament les crides als objectes en consultes SQL optimitzades per al sistema gestor de bases de dades que s'està utilitzant en cada moment.

D'aquesta manera, és molt senzill canviar a un altre sistema de bases de dades completament diferent a la meitat del desenvolupament d'un projecte. Aquestes tècniques són útils per exemple quan s'ha de desenvolupar un prototip ràpid d'una aplicació i el client encara no ha decidit el sistema de bases de dades que més li convé. El canvi es pot fer modificant només una línia en el fitxer de configuració database.yml.

La capa d'abstracció utilitzada encapsula tota la lògica de les dades. La resta de l'aplicació no ha de preocupar-se per les consultes SQL i el codi SQL que s'encarrega de l'accés a la base de dades és fàcil de trobar.

El ORM proporciona mètodes per fer les consultes més fàcils pels programadors i el més optimitzades possibles. La llista següent mostrar els mètodes més comuns utilitzats en el projecte:

retrieveByPK: Recupera un objecte pel seu id.

retrieveByPKs: Recupera un array d'objectes segons les seves ids.

doSelectOne: Recuper un únic registre segons les condicions.

doSelect: Recupera varis registres segons les condicions.

doCount: Compte els registres segon les les condicions.

En el *Codi 6.4.1,* es veu un exemple de com una acció crida a un mètode peer, per recuperar l'administrador de la part administrativa. Com que és un mètode estàtic es pot cridar en el controlador sense crea un objecte, i com que estàen el domini no trenca el patró MVC

```
public function executeLogin(sfWebRequest $request){

$this->formulariLogin = new LoginusuForm();

if($this->getRequest()->getMethod()== sfRequest::POST){
 $this->formulariLogin->bind($request->getParameter('usuarilogin'));
  if($this->formulariLogin->isValid()){
                        //miro si existeix l'usuari
   $admin = AdminPeer::getAministrador($request->getPostParameter('usuarilogin'));
              if ($admin){//existeix el administrador?
                    $this->getContext()->getUser()->signIn($admin);
                                // redirigeixo a la ultima pagina
                                return $this->redirect('@panell_control');
                          }
              $this->getUser()->setFlash('ErrorLogIn', 'El nom o la contrasenya són
incorrectes.');
                          }
              }
              return sfView::SUCCESS;
      }
```

Codi 6.4.1: Mètode login de la part administrativa

En el *Codi 6.4.2* es veu com esta implementat el mètode getAdministrador de la classe estàtica peer .

```
class AdminPeer extends BaseAdminPeer
{
      public static function getAministrador($dadesUsu){
            $c = new Criteria();
```

```
            $c->add(AdminPeer::NOM, $dadesUsu['nom']);
            $c->add(AdminPeer::CONTRASENYA, $dadesUsu['contrasenya']);
            return AdminPeer::doSelectOne($c);
        }
}
```

Codi 6.4.2: Implementació del mètode getAdministrador.

En aquest mètode es crea un objecte *Criteria*() que es l'objecte del ORM que interroga la base de dades. Amb el mètode *add* afegim les condicions (equival la clàusula *WHERE* en una sentència SQL), se li indica quina columna i el valor i amb el mètode doSelectOne es seleccionarà el primer registre que coincideixi amb les condiciona afegides en el criteri.

7 API Google Maps.

En aquest apartat es descriuen les funcions de l'API de Google Maps que s'utilitza en el desenvolupament del projecte, concretament la part pública. En l'apart de l'API Google Maps no es vol convertir en un manual de referencia sobre aquesta API. Per això, tenim la pàgina web de referencia de Google Maps[14]. En aquest apartat s'explicaran les funcions més utilitzades per programar la part del mapa del projecte.

Abans d'entrar s'ha de conèixer els conceptes inicials de l'API de Google Maps:

- Tot el codi per manipular les classes de l'API és a través del llenguatge javascript, per tant, s'ha col·locar a l'espai web reservat per aquest llenguatge entre l'etiqueta **<script></script>..**

- El servei de mapes de Google Maps utilitza una clau d'accés. Aquesta es proporcionada per Google a través d'una inscripció[15] i una compte de correu de Google que s'associa a la clau. El procés és molt senzill s'introdueix la direcció web del nostra projecte i et retorna una clau. En aquest projecte, s'ha d'introduir http://localhost, i la clau obtinguda és:

  "ABQIAAAANH9jXoi0rori1VyBPEGG-xT2yXp_ZAY8_ufC3CFXhHIE1NvwkxQL5FeQElnpPlgSV4wOGf4BSrKs3A"

- Un cop copiada la clau, s'ha de carregar el codi a la pàgina web. Per això, s'ha d'inserir en el següent script:

```
<script
 src="http://maps.google.com/mapsfile=api&amp;v=2&amp;sensor=false&amp;key=A
BQIAAAAnfs7bKE82qgb3Zc2YyS-oBT2yXp_ZAY8_ufC3CFXhHIE1NvwkxSySz_REpPq-
4WZA27OwgbtyR3VcA" type="text/javascript">
```

14 *http://code.google.com/intl/es/apis/maps/documentation/reference.html*
15 *http://code.google.com/intl/es/apis/maps/signup.html*

```
</script>
```

El atribut *src* indica la direcció on esta l'arxiu de Javascript, que inclou tots els símbols i definicions que es necessiten per utilitzar l'API de Google Maps. En al paràmetre *key* es posa la clau obtinguda de Google per aquest projecte.

- El següent pas, es mostrar el mapa en la web. Per fer això, s'ha de reservar lloc per mostrar el mapa en la pàgina. S'ha d'utilitzar de la etiqueta HTML ** ** on se li dóna un identificador com el mostra en el *Codi 7.1*. Les dimensions del mapa es corresponen a les dimensions de la capa ** .** Les dimensions de la capa es recomana modificar-les a través de CSS modificant les propietats de la etiqueta HTML ** ,** com es pot veure en el *Codi 7.2*.

```
<div id="mapa"></div>
```

Codi 7.1: HTML on es carrega el mapa de Google Maps.

El mida del mapa venen definits pels atributs *height* i *witdh,* sinó es defineixen aquest atributs de forma explicita mitjançant GMapOptions.

```
#mapa {
        width: 800px;
        height: 595px;
        margin-left:18px;
}
```

Codi 7.2: Propietat DIV a través de CSS

- Un cop reservat l'espai en la pàgina, s'ha de crear una instància de la classe de GMap2 de Google Maps. Com es mostra en el *Codi 7.3*.

```
        var capaMapa =  document.getElementById("mapa");//capa del mapa html
        var map;//objecte mapa
        ...
        map = new GMap2(capaMapa, {
                draggableCursor:"auto", draggingCursor:"move"});
```

Codi 7.3: Instància d'un objecte GMap2

Podem veure com es crea una instància *GMap2* passant-li la referència de l'objecte HTML ** **. El constructor *GMap2* se li poden passar varis paràmetres, entre ells podem destacar *draggableCursor,* que permet que el cursor pugui arrossegar el mapa. I l'altre propietat és *draggingCursor*, que el cursor s'ha de mostrar mentre s'està arrossegant el mapa.

- Un cop creat el mapa, el primer que hem de fer és inicialitzar-lo per poder treballar amb ells. La forma de inicialitzar el mapa es cridant el mètode *setcenter()* que és de la classe GMap2 . Aquest mètode necessita com a paràmetres unes coordenades geogràfiques per situar-se en un punt del planeta. L'últim paràmetre del mètode és el nivell de zoom, determina l'aproximament de la vista sobre el mapa com es mostra en la *Codi 7.4*

```
map.setCenter(new GlatLng(40.2459915041,-3.6474609375),7);//centrem el mapa
```

Codi 7.4: Inicialització del mapa.

Les coordenades latitud 41,24599 i la longitud -3.6474609 corresponen la localització de la ciutat de Madrid amb un zoom de 7.

- Per executar la funció que carregar el mapa, com s'ha vist en el *Codi 7.3* s'ha de situar en una funció Javascript , que en el nostre projecte s'anomenat *buildMap(),* en l'etiqueta HTML <*body*> .en l'atribut *onLoad*. La forma de carrega el mapa de Google es realitzar quan s'acaba de carregar totalment la pàgina web, d'aquesta manera es pot controlar quan es dibuixa el mapa, i així, evitar comportaments impredictibles. En el *Codi 7.5,* es pot veure com es fa la crida en cos del document PHP i la funció *buildMap*() desenvolupada.

```
...
<body onload="buildMap()" onunload="GUnload()">
...
function buildMap() {
        map = new GMap2(capaMapa,
              {draggableCursor:"auto", draggingCursor:"move"});
      map.setCenter(new GlatLng(40.2459915041,-3.6474609375),7);
...
```

```
        }
```

Codi 7.5: Carrega del mapa.

Com ja hem dit, dins l'etiqueta <**body>** hi ha el mètode *onload* que serveix per inicialitzar la funció *buildMap()* en el moment que s'acaba de carregar la pàgina.

Peró la classe GMap2 disposa d'una funció anomenada *Gunload()* que es l'encarregada de destruir les variables que estan en memòria del navegador. Aquest mètode ja esta implementat per Google Maps i només s'ha de realitzar la crida en el atribut *onunload* de l'etiqueta *<body>,* que és el moment que es crida una altre pàgina i s'abandona l'actual.

- Finalment, dir que tots els navegadors més usats pels usuaris o totes les seves noves versions suporten l'API de Google Maps. De totes maneres, API de Google Maps disposa d'un mètode per comprovar si el teu navegador és compatible.

.Fins aquí si executéssim aquest codi ja podríem veure un mapa molt senzill com es mostra en la *Il·lustració 7.1*

Il·lustració 7.2: Mapa senzill

A continuació, s'expliquen les funcionalitats de Google Maps que s'han necessitat pel desenvolupament del projecte. Per conèixer tots els mètodes que ofereix Google Maps, s'ha de consultar la seva pàgina oficial de referència de l'API[16].

16 *http://code.google.com/intl/es/apis/maps/documentation/reference.html*

La classe *GMap2*, com ja s'ha vist es l'encarregada de crear els mapes de Google, i amb el mètode *setCenter()* serveix per inicialitzar el mapa a una posició geogràfica i a un zoom determinat.

Peró la classe *GMap2* disposa d'altres mètodes que permeten canviar la configuració. Es pot definir, per exemple, el tipus de mapa que volem. En el nostre mapa hi ha tres tipus de mapes i un d'ells esta combinat amb un altre.

La vista inicial és la vista de Mapa de carrer, la segona és la vista satèl·lit combinada amb la vista de carrers ,i l'última vista, és la territorial. Es poden definir el tipus de mapa que volem amb el mètode *setMapType(tipus),* on *tipus* pren els valors següents dels mapes : *G_NORMAL_MAP, G_SATELLITE_MAP, G_HYBRID_MAP* respectivament amb els mapes abans comentats. En el *Codi 7.6* podem veure com anem configurant la configuració de les vistes en el mapa.

```
function buildMap() {
 map = new GMap2(capaMapa,
       {draggableCursor:"auto", draggingCursor:"move"});


        // INICIALITZEM ELS CONTROLADORS DEL MAPA
        obtenirCentratMapa(40.2459915041,-3.6474609375);
        map.addMapType(G_PHYSICAL_MAP);//afegim la vista física en el mapa
         /* posem la opció vista híbrida dins la opció de la vista satelit
        var mapaAnidat = new GHierarchicalMapTypeControl();
        mapaAnidat.addRelationship(G_SATELLITE_MAP, G_HYBRID_MAP,
                                                "Etiquetes", true);
        // afegim les vistes anidades
        map.addControl(mapaAnidat);
}
```

Codi 7.6: Funció buildMap() amb les vistes.

Les vistes s'afegeixen amb el mètode *addMapType(tipus_mapa)* de la classe *GMap2*. Per crear un vista mixta, entre satèl·lit i carrer, es necessitat crear un nou objecte *GhierarchicalMapTypeControl()*, aquest objecte té un mètode *addRelationship(vista1, vista2, etiqueta, true), vista1* i *vista2* corresponen el valor d'una vista. L'atribut *etiqueta,* és el label de l'opció per activar l'herència. En el *Codi 7.7,* es pot veure la instrucció

i, en la *Il·lustració 7.2,* l'opció esmentada anteriorment.

Il·lustració 7.2: Detall de la vista mixta

Google Maps també ofereix la possibilitat de definir els controladors del mapa, en mida petita o mida gran: Aquest controls són els controls amb botons per centrar el mapa, així com una barra lliscant per apropar o allunyar. S'afegeix el mapa l'objecte `GlargeMapControl()` amb el mètode addControl(). En aquest cas s'ha triat el panell tipus llarg. En la *Codi 7.7* es veu com s'afegeix el control en el mapa.

```
function buildMap() {
       map = new GMap2(...);
       obtenirCentratMapa(40.2459915041,-3.6474609375);
       ...
       map.addControl(new GlargeMapControl());
       ...
       }
```
Codi 7.7: Afegeix el control del mapa.

L'objecte `GlargeMapControl()` es la classe que crear el control llarg.. També existeix un altra opció d'un control més petit, que es crear amb l'objecte `GsmallMapControl()`. En la Il·lustració 7.3 es pot veure el control utilitzat en el projecte.

Il·lustració 7.3: Control llarg del mapa

Altres mètodes també molt interessants, com `disableDoubleClickZoom()`, per desactivar l'opció de fer doble clic sobre el mapa. En el mapa també se li pot incloure l'escala, afegint-li l'objecte *GscaleControl* .

Per anar acabant amb la classe *GMap2,* s'ha utilitzat els mètodes *addOverlay() i removeOverlay(),* amb aquest dos mètodes poden afegir o treure superposicions en el mapa. Les superposicions es poden entendre com capes que contenen elements, aquests elements poden ser polilínies , marcadors, sombres, etc.

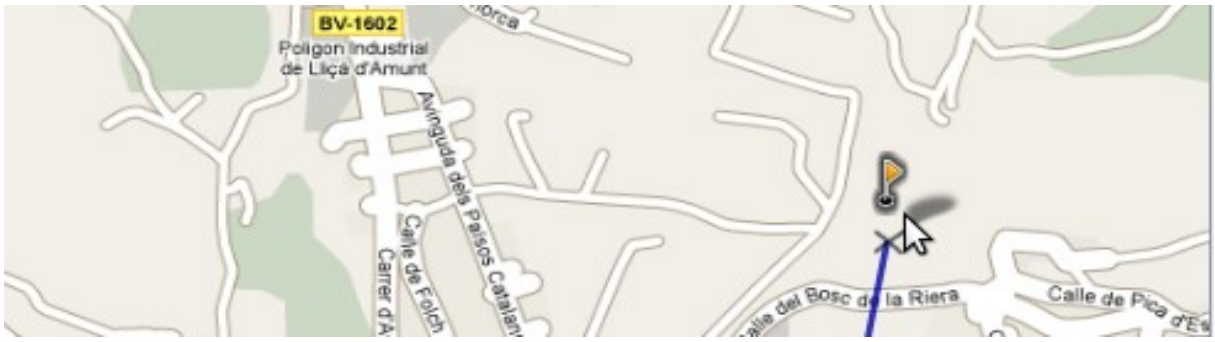
GEvent permet gestionar els events que es produeixen en wl mapa de Google Maps. La classe *GMap2* té els events *click*, *dblclick move i drag*, per controlar el comportament del mapa. És a dir, quan fem clic sobre el mapa, el *GEvent* detecta quin tipus d'event s'ha produit. Un exemple relacionat amb el projecte, quan l'usuari fa clic sobre el mapa per dibuixar una línia, el GEvent detecta aquest aquest clic com un event *click* i executa la funció javascript per dibuixar una línia ja que associada aquest event. La declaració dels events de *GEvent* esta la

funció de carrega del mapa, la *buildMapa(),* tal i com es pot veure en el *Codi 7.8.*

```
function buildMap() {
        map = new GMap2(...);
        ...
        GEvent.addListener(coord, "drag", clickMapa);
}
```

Codi 7.8: Declaració de GEvent.

Tenim la declaració de *GEvent*, que rep el mapa, el segon paràmetre es el tipus d'event que *GEvent* ha de detectar i el tercer paràmetre es la funció Javascript associada l'event *drag.*En la *Il·lustració 7.4* podem veurem en acció d'event *drag.*



Il·lustració 7.4: Arrossegar un punt del la ruta.

GlatLng és l'objecte que permet situar punts geogràfics dins el mapa. Aquesta classe es utilitzada pels objectes *GMarker o GPolyLine.* El constructor d'aquesta classe té els següents paràmetres *GlatLng(latitud, longitud)*, la unitat de mesura són en graus.

Les següents classes permeten representar superposicions o overlays en el mapa. Els overlays més utilitzats per representar les rutes en el mapa són el marcadors (*GMarker*) i les línies (G*polyline*).

Definirem un marcador (*GMarker*) cada cop que l'usuari faci clic sobre un punt del mapa. En aquest moment, intervé *Gevent.addListenner(coord, "click", clickMapa)* que rep les coordenades per situar el marker en aquella posició, i es crea el marcador. En el *Codi 7.10* es pot veure com es crea un marcador.

```
var icon_url ="/images/flag.png";

...

function clickMapa(overlay, point) {

        var icon = new GIcon(G_DEFAULT_ICON, icon_url);

        var marker = new GMarker(point, {icon:icon, draggable:true,
                                    bouncy:false, dragCrossMove:true});
```

Codi 7.10: Declaració d'un marcador

El constructor (*new Gmarker(...)*) del marcador té varis paràmetres que ara comentarem. El primer anomenat com a *point*, són les coordenades que ens arriben a través del mapa, que és on es situara el marcador. Els altres paràmetres del constructor, són per configurar la seva aparença i comportament. L'atribut *icon* és per canviar la icona típica del marcadors per una icona definida pel projecte.

Els altres atributs com `draggable` és per fer que el marcador sigui arrastrable, el `bouncy` és fer l'efecte rebot de la icona quan s'arrossega i es deixa anar. I `dragCrossMove` és quan s'arrossega, mostri una *x* en el punt on es deixa anar.

GIcon és la icona del marcador. En tot mapa hi ha definit una icona per defecte. En el *Codi 7.10* es pot veure com es modificar una icona que estigui allotjada en el teu servidor.

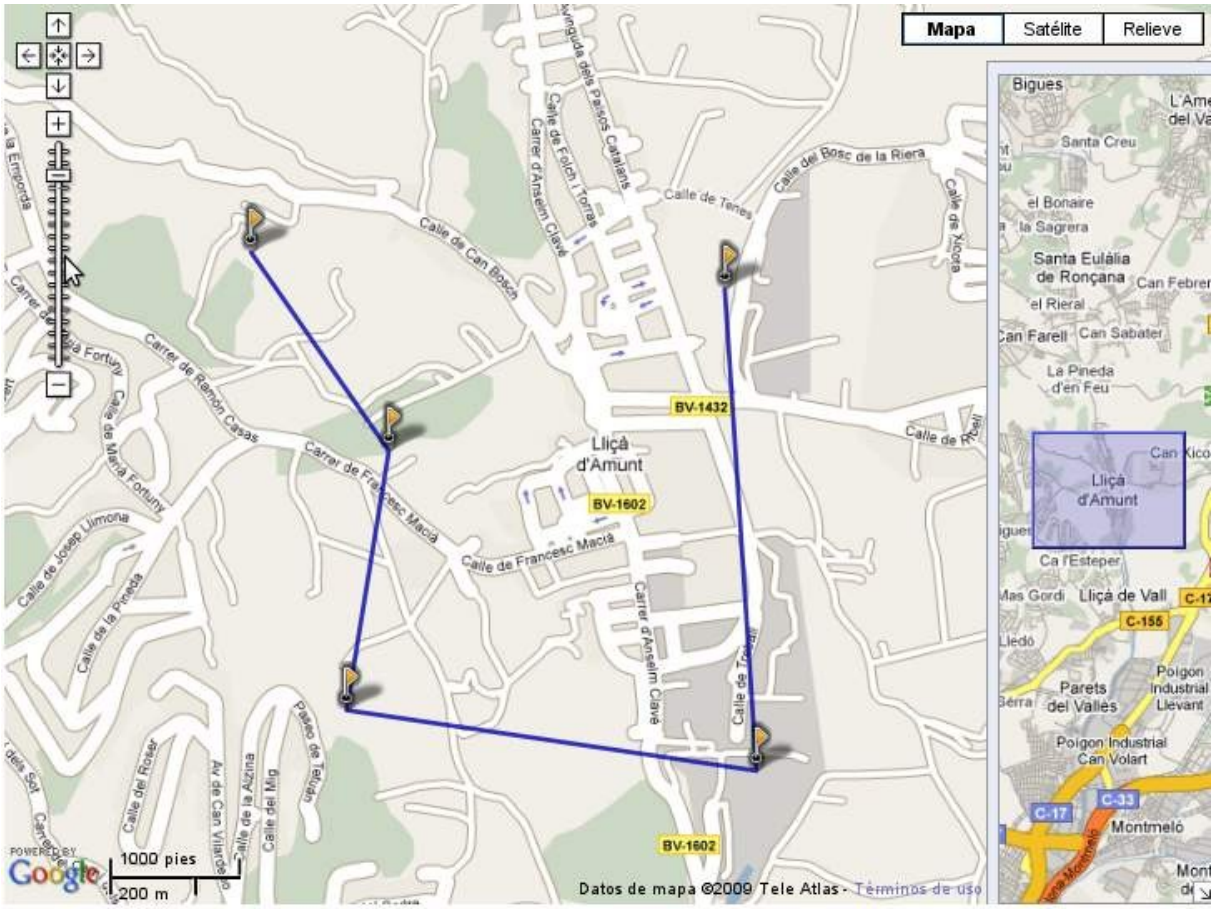
La ruta dibuixada pels usuaris es representa a través de línies (objecte *GPolyline()*) encadenades una darrera l'altre. Durant el procés de dibuix de la ruta, els punt es guarden en un vector. Cada cop que inserixen un nou punt al vector, s'esborra la capa de dibuix, s'inclou el nou punt i es torna a dibuixar les línies. L'objecte GPolyline [12] té un mètode que calcula la longitud de la línia, i ho utilitzem per saber la distància del recorregut. En el *Codi 7.11* podem observar el mètode per dibuixar una ruta.

```
function dibuixarRuta(){

 if(poly) { map.removeOverlay(poly); }
```

```
 punts.length = 0;

 for(i = 0; i < markers.length; i++) {
  punts.push(markers[i].getLatLng());
 }
 poly = new GPolyline(punts, lineColor, lineWeight, lineOpacity);
   var length = poly.getLength()/1000;
   report.value = length.toFixed(3);
   map.addOverlay(poly);
}
```

Codi 7.11: Mètode per diubuixar la ruta amb GPolynile.

L'instrucció `length.toFixed(3)` no és pròpia de l'API de Google Maps, sinó del Javascript, que serveix per arrodonir el càlcul de la distància en tres decimals. L'última línia afegeix la nova capa al mapa. En la *Il·lustració 7.5*, es pot veure com es dibuixa la ruta a través de l'objecte GPolyline de Google Maps.



Il·lustració 7.5: Línies de l'objecte GPolyline en el mapa

Les rutes que es dibuixen en el mapa, vist en la il·lustració anterior, són guardades en un fitxer

XML en el servidor web. Quan un usuari vol veure el mapa amb el recorregut, el servidor dóna la direcció de la ruta. L'API de Google Maps proporciona un mètode per recuperar l'arxiu XML proporcionant la url. Després s'invoca *GXML.paser(recursInvocat)* per analitzar la cadena en qüestió com un text XML, i retorna la representació del DOM. A partir d'aquest moment, es pot recórrer el DOM amb les funcions que proporciona Javascript. En el *Codi 7.12* es pot veure un com recuperar un XML.

```
GdownloadUrl("url/arxiu.xml", function(data, responseCode) {

 var xml = GXml.parse(data);
 var markers = xml.documentElement.getElementsByTagName("marker");
 for (var i = 0; i < markers.length; i++) {
    var point = new
GLatLng(parseFloat(markers[i].getAttribute("lat")),parseFloat(markers[i].getAttrib
ute("lng")));
       points.push(point);
        if(i==0){//centre el mapa
         map.setCenter(point,zoom);
        }
       map.addOverlay(new GPolyline(points, '#ff0000', 5, 0.7));

                }
});
```

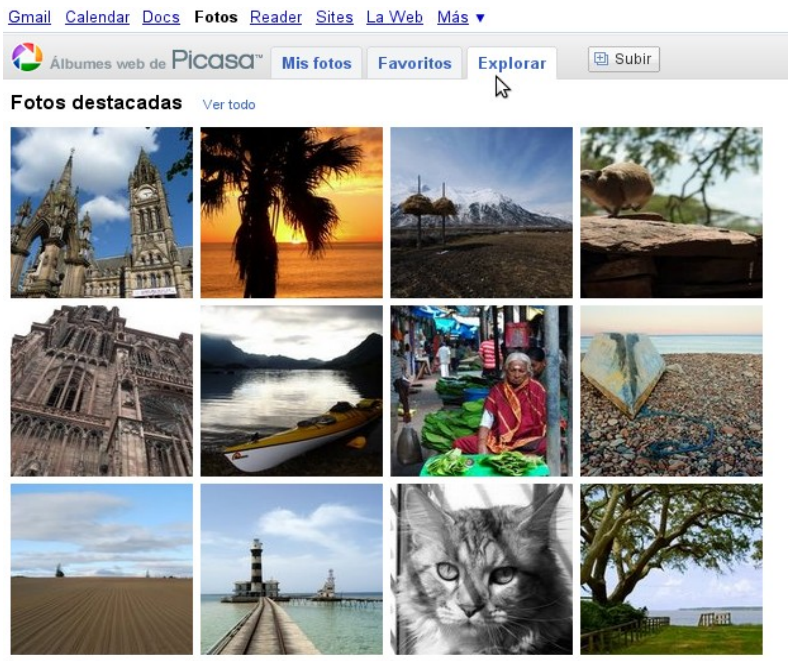
Codi 7.12: Exemple de com es carrega un fitxer xml.

Un cop carregat el fitxer XML, el tractament és mitjançant les funcions per recórrer el DOM amb la funció `documentElement.getElementsByTagName` .

8 API Picasa.

Picasa Album Web[17] és un servei d'Internet per compartir, penjar i publicar imatges. Aquest servei ofereix a l'usuari la possibilitat de crear àlbums i penjar fotografies. A més a més, els usuaris també poden gestionar les seves fotografies i les poden compartir amb altres usuaris.

Picasa Album Web ofereix aquest servei de forma gratuïta i l'emmagatzematge és limitat., per accedir a l'emmagatzematge il·limitat s'ha d'utilitzar la versió de pagament. S'ha escollit aquest servei per emmagatzemar les fotografies dels usuaris, ja que és un sistema molt comú, juntament amb Flirkc[18]. Picasa forma part dels serveis de Google. En la *Il·lustració 8.1* es pot veure la pàgina de Picasa Web Album..



Il·lustració 8.1: Picasa Album Web

17 *picasaweb.google.comn*
18 *http://www.flickr.com*

Picasa Web [13] igual que Google Maps, disposa d'una API perquè els desenvolupadors web puguin desenvolupar les serves aplicacions. En el projecte, es va decidir que les imatges no es guardarien en el servidor web, on s'allotja la nostra aplicació. D'aquesta manera no s'hauria de contractar un allotjament web amb molta capacitat.

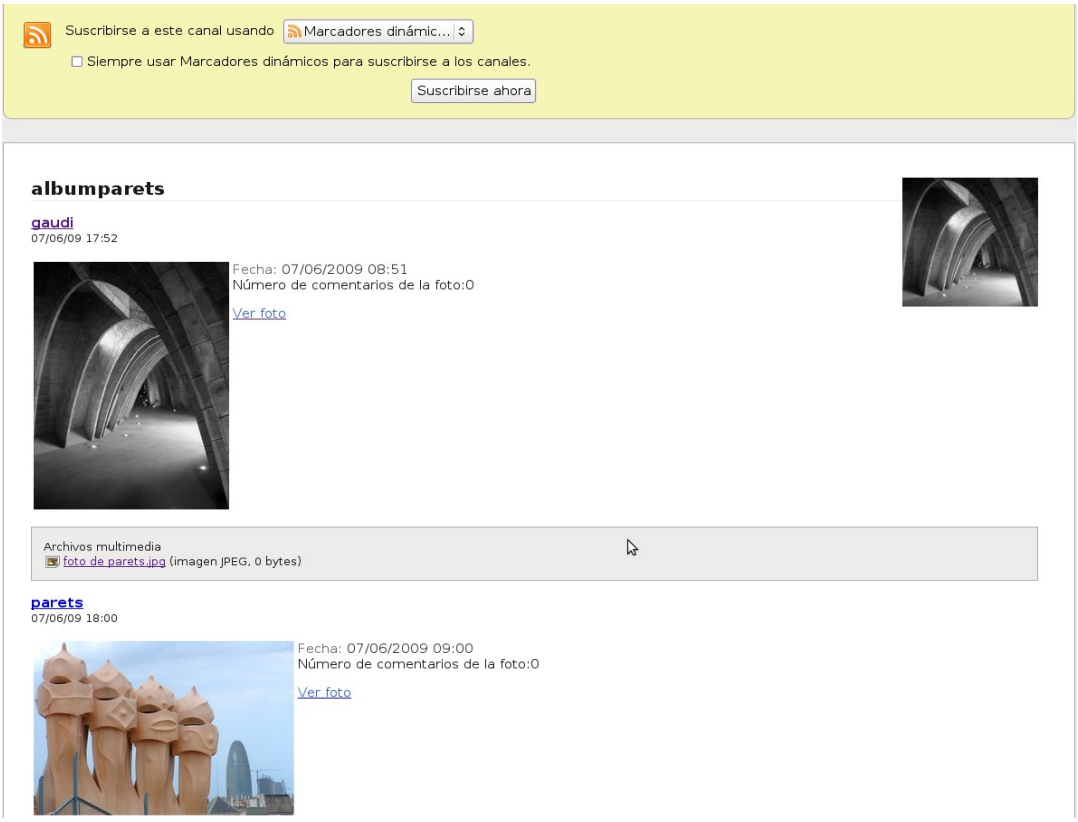
Una altre raó per optar externalitzar el servei d'imatges, es que només s'ha de programar la part de comunicació entre la meva aplicació i Picasa Web, ja que Picasa s'encarrega de tot el relacionat amb la gestió de imatges i la nostra aplicació web només ha d'ordenar a Picasa que realitzi les accions que necessitem. sinó que s'optaria per utilitzar un servei d'Internet capaç de d'emmagatzemar imatges.

Il·lustració 8.2: Symbol Feed

Per tant, el punt crític en la gestió de les imatges és la comunicació entre Picasa i l'aplicació web. La comunicació entre Picasa es realitzar a través del canal rss o Feed (afiliació), que no és res més que un arxiu xml, o sigui, quan nosaltres fem una petició a Picasa des la nostra aplicació ens torna un Feed, que no es res més que un fitxer XML.

Amb aquest fitxer XML l'hem d'interrogar per treure les dades que nosaltres necessitem. Un exemple de les dades que poden haver-hi en el fitxer són, el nom de la fotografia, la mida, la url de la fotografia, el comentari i fins i tot la posició geogràfica.

En la *Il·lustració 8.3* es pot veure el Feed que retorna Picasa Web [14] a l'aplicació utilitzant el navegador web Firefox 3.0.11.Aquest tipus de Fitxer XML es interpretat per la majoria dels actuals navegadors webs.



Il·lustració 8.3: Feed interpretat en el navegador Firefox

En aquesta última il·lustració es pot veure un àlbum que s'ha demana a Picasa Web [15] i com es pot veure, aquest XML conte la informació de l'àlbum amb les fotografies que el component.

```
<?xml version='1.0' encoding='UTF-8'?>
...
 xmlns:georss='http://www.georss.org/georss'version='2.0'>
<channel>
       <atom:id>
http://picasaweb.google.com/data/feed/base/user/pfcagucosal/albumid/53446137282094
53297
       </atom:id>
       <lastBuildDate>Sun, 07 Jun 2009 16:02:53 +0000</lastBuildDate>
       <category domain='http://schemas.google.com/g/2005#kind'>
       http://schemas.google.com/photos/2007#album</category>
       <title>albumparets</title>
       <description/>
       <image>
              <url>
```

```
http://lh6.ggpht.com/_hCp27yQutHk/Sivh3muZwPE/AAAAAAAAA1w/1jaUy2kO3Hw/s160-
c/Albumparets.jpg</url>
```

Codi 8.1: Feed vist en format XML

En el *Codi 8.1* es pot veure algun dels atributs que el Feed conté.

La interacció entre l'aplicació web i Picasa Web es realitzar a través d'una llibreria. El següent llistat mostra el tipus de llibreria que podem utilitzar per interaccionar amb Picasa Web.

1. Java Client Library[19].
2. .NET Client Library[20].
3. PHP Client Library[21].
4. Python Client Library[22].
5. Objective-C Client Library[23]

Totes aquestes llibreries són adaptacions pels llenguatges que es mostren en el llistat de la llibreria de Google Data (*Gdata*). Pel nostre projecte s'ha utilitzat l'opció 3, PHP Client de Libray. La llibreria *Gdata* ha estat adaptada al llenguatge PHP per la companyia Zend, i per tant, s'haurà de registrar aquest component al projecte de Symfony. (Mirar *6.4 Components de Zend en Symfony* per més informació).
La llibreria de GData adaptada per PHP s'ha de descarregar de la pàgina de Zend, aquesta llibreria esta inclosa en al framework de Zend, per tant, per obtenir la llibreria s'ha de descarregar tot al framework o Zend també dona l'opció de descarregar les llibreries de Google GData apart[24] .

GData és una llibreria que facilita la programació el programador per incloure els serveis de Google en l'aplicació a través d'unes classes, d'aquesta manera no ha de crear el arxiu Feed, o sigui, no fa falta que manipuli el fitxer XML ja que això es fa través de les classes.
Aquesta llibreria inclou els següents serveis Google:

19 *http://code.google.com/intl/es-ES/apis/gdata/articles/java_client_lib.html*
20 *http://code.google.com/intl/es-ES/apis/gdata/articles/dotnet_client_lib.html*
21 *http://code.google.com/intl/es-ES/apis/gdata/articles/php_client_lib.html*
22 *http://code.google.com/intl/es-ES/apis/gdata/articles/python_client_lib.html*
23 *http://code.google.com/p/gdata-objectivec-client/wiki/GDataObjCIntroduction*
24 *http://framework.zend.com/download/gdata*

1. Google Search Book.
2. Google Calendar.
3. Google Base.
4. Google Health.
5. Google Spreadsheets.
6. Google Document List Data.
7. Authentification with ClientLogin.
8. Google Apps.
9. Google Picasa.
10. Youtube.

Un cop descarregat i descomprimit el component, s'han de copiar els arxius que estan en el directori */ZendGdata-1.8.3/library/Zend/* (la ruta pot canviar segons la versió) en el directori *lib/vendor/zend* del projecte Symfony.

El components de *Zend_GData* es registren en la classe ProjectConfiguration.class.php en el directori *config* del projecte, en el *Codi 8.2,* tenim com es registre el component Zend_GData.

```
static protected $zendGData = false;
       static public function registerZendGData(){
 if (self::$zendGData)
 {
       return;
 }


set_include_path(sfConfig::get('sf_lib_dir').'/vendor'.PATH_SEPARATOR.get_include_
path());
       require_once sfConfig::get('sf_lib_dir').'/vendor/Zend/Gdata.php';
       Zend_Loader::registerAutoload();
       self::$zendValidate = true;
}
```

Codi 8.2: Funció que registre el Zend_GData.

La classe que s'ha de registrar és Gdata.php, ja que aquesta fa les crides necessàries a les altres classes que hi ha en el directori.

Pel nostre projecte només ens interessa les llibreries relacionades amb Picasa Web Album, però s'ha d'incloure tota la llibreria Zend_Gdata per raons de dependències entre les classes.

Creació d'un àlbum a Picasa Web

Per utilitzar el servei de Picasa Album Web de Google s'ha de tenir un compte de Gmail. Si no es disposa d'aquesta compte no es pot utilitzar la llibreria GData.
Per crear un àlbum a Picasa hem de fer la crida a la llibreria amb l'instrucció *`ProjectConfiguration::registerZendGData().`* Un cop ja tenim el component, es fan les crides com si estiguéssim en al framework Zend.

Per iniciar la creació del àlbum, primer necessitem la classe que controla el servei per autentificar-nos en el servei. La classe *Zend_Gdata_Photos* és la classe que interacciona amb els serveis de Picasa Web.

```
$this->servei = Zend_Gdata_Photos::AUTH_SERVICE_NAME;
```

El següent pas, es autentificar-se al servei de Google introduint el nom de usuari, la contrasenya i el servei que volem autentificar-nos.

```
$this->client = Zend_Gdata_ClientLogin::getHttpClient
        (
                $this->usuariGoogle,
                $this->passwdGoogle,
                $this->servei
        );
```

Un cop autentificats, ja es pot crear l'àlbum, s'utilitza la classe *Zend_Gadata_Photos,* i hem d'estar autentificats per realitzar aquesta acció, per tant, li passem el client.

```
$this->galbum = new Zend_Gdata_Photos($this->client);
```

Les accions que es poden fer en GData són la preparació d'entrada de dades, tant per inserir o modificar o esborrar, la consulta de les dades i obtenir el fitxer Feed. En aquest cas com que hem d'inserir un àlbum l'acció a realitzar es la entrada de dades:

```
$entry = new Zend_Gdata_Photos_AlbumEntry();
$entry->setTitle($this->galbum->newTitle($this->getNom()));
```

La llibreria també permet modificar el tipus d'accés dels àlbums.

```
$nou_acces = $this->galbum->newAccess();
$nou_acces->text = 'public';//per defecte es private
$entry->setGphotoAccess($nou_acces);
```

Un cop ja hem realitzat tots el passos anterior, realitzem la creació de l'àlbum.

```
$entradaAlbum = $this->galbum->insertAlbumEntry($entry);
```

En resum, la funció per crear un àlbum es pot veure en el *Codi 8.3*.

```
ProjectConfiguration::registerZendGData();
$this->servei = Zend_Gdata_Photos::AUTH_SERVICE_NAME;
$this->client = Zend_Gdata_ClientLogin::getHttpClient
(
       $this->usuariGoogle,
       $this->passwdGoogle ,
       $this->servei
);
$this->galbum = new Zend_Gdata_Photos($this->client);
$entry = new Zend_Gdata_Photos_AlbumEntry();
$entry->setTitle($this->galbum->newTitle($this->getNom()));
$nou_acces = $this->galbum->newAccess();

$nou_acces->text = 'public';//per defecte es private
$entry->setGphotoAccess($nou_acces);

$entradaAlbum = $this->galbum->insertAlbumEntry($entry);
```

Codi 8.3: Entrada de l'àlbum.

Crear una foto a l'àlbum

Per inserir una fotografia, abans s'ha de crear l'àlbum. La classe per crear la fotografia és la mateixa que s'utilitza per crear l'àlbum, per tant, el procediment és el mateix fins arribar a l'instrucció on s'utilitza la classe *Zend_Gdata_Photos*. Per no confondré variables, *galbum* serà *gfoto*.

Un cop creada la foto, preparem per crear la entrada de dades.

```
$foto = $this->gfoto->newPhotoEntry();
```

Després recollim la fotografia que l'usuari ens ha enviat. L'arxiu es guarda amb un nom temporal i amb Symfony s'accedeix amb el mètode *getTempName*()

```
$file = $this->gfoto->newMediaFileSource($arxiu->getTempName());
```

També es recull el tipus de la imatge i s'associa a la foto.

```
$file->setContentType($arxiu->getType());
$foto->setMediaSource($file);
```

En la imatges es pot col·locar la posició geogràfica de la imatge que l'usuari a clicat sobre el mapa. Un punt geogràfic composat per una posició en el mapa, que forma part d'una coordenada que també forma part d'un atribut d'un Feed.

```
$gmlPos = new Zend_Gdata_Geo_Extension_GmlPos($this->getCoordImatge());
$gmlPoint = new Zend_Gdata_Geo_Extension_GmlPoint($gmlPos);
$geoRSSwhere = new Zend_Gdata_Geo_Extension_GeoRssWhere($gmlPoint);
$foto->setGeoRssWhere($geoRSSwhere);
```

Un cop ja tenim la fotografia amb totes les dades necessàries, consultem que existeix l'àlbum. Com s'ha dit abans *GData* apart de preparar l'entrada de dades, també pot consultar.

```
	$albumQuery = $this->gfoto->newAlbumQuery();
```

Introduïm el nom de usuari que ha creat la imatge i el nom de l'àlbum.

```
	$albumQuery->setUser('pfcagucosal');
	$albumQuery->setAlbumName($album->getNom());
```

Finalment inserim la fotografia en l'àlbum que hem indicat en la consulta.

```
	$this->feed = $this->gfoto->insertPhotoEntry
	(
		$foto,
		$albumQuery->getQueryUrl()
	);
```

El mètode per inserir una fotografia amb o sense coordenades geogràfiques a l'àlbum indicat és com es mostra en el *Codi 8.4*

```
ProjectConfiguration::registerZendGData();
	$this->servei = Zend_Gdata_Photos::AUTH_SERVICE_NAME;
	$this->client = Zend_Gdata_ClientLogin::getHttpClient
	(
		$this->usuariGoogle,
		$this->passwdGoogle ,
		$this->servei
	);
	$this->gfoto = new Zend_Gdata_Photos($this->client);
	$foto = $this->gfoto->newPhotoEntry();
	$file = $this->gfoto->newMediaFileSource($arxiu->getTempName());
	$file->setContentType($arxiu->getType());
	$foto->setMediaSource($file);

	if(trim($this->getCoordImatge())!=''){
	 $gmlPos = new Zend_Gdata_Geo_Extension_GmlPos
	(
		$this->getCoordImatge()
	 );
	 $gmlPoint = new Zend_Gdata_Geo_Extension_GmlPoint($gmlPos);
```

```
        $geoRSSwhere = new Zend_Gdata_Geo_Extension_GeoRssWhere($gmlPoint);
        $foto->setGeoRssWhere($geoRSSwhere);
       }

      $foto->setSummary($this->gfoto->newSummary($this->titul));

       $albumQuery = $this->gfoto->newAlbumQuery();
       $albumQuery->setUser('pfcagucosal');
       $albumQuery->setAlbumName($album->getNom());

       $this->feed = $this->gfoto->insertPhotoEntry
       (
              $foto,
               $albumQuery->getQueryUrl()
       );
```

Codi 8.4: Inserir Fotografia.

Creació d'un comentari a la fotografia

El comentari no es pot incloure a la imatge ja es considera un element apart, per tant, en aquest cas utilitzem la classe *Zend_Gdata_Photos_Comentary* per crear un comentari.

```
$entry = new Zend_Gdata_Photos_CommentEntry();
```

Quan ja tenim creat l'objecte, inserim el contingut del comentari.

```
$entry->setTitle($this->gfoto->newTitle("comment"));
$entry->setContent($this->gfoto->newContent($comment));
```

Un cop ja tenim el comentari esta preparat per inserir, fem una consulta per buscar la foto de l'àlbum per inserir al comentari. Aquesta consulta és per introduir un comentari, per tant, obrim una consulta per introduir dades.

```
$photoQuery = new Zend_Gdata_Photos_PhotoQuery;
$photoQuery->setUser("pfcagucosal@gmail.com");
$photoQuery->setAlbumId($idalbum);
$photoQuery->setPhotoId($idfoto);
$photoQuery->setType('entry');
$photoEntry = $this->gfoto->getPhotoEntry($photoQuery);
```

Finalment, introduïm el comentari a la foto, utilitzant la consulta de la fotografia.

```
$this->gfoto->insertCommentEntry($entry, $photoEntry)
```

La funció completa per guardar el comentari es com es mostra en el *Codi 8.5*. S'utilitza la funció *$this->initZendGdata()* per no repetir el codi de la connexió amb Picasa Web

```
$this->initZendGdata();
$entry = new Zend_Gdata_Photos_CommentEntry();
$entry->setTitle($this->gfoto->newTitle("comment"));
$entry->setContent($this->gfoto->newContent($comment));

$photoQuery = new Zend_Gdata_Photos_PhotoQuery;
$photoQuery->setUser("pfcagucosal@gmail.com");
$photoQuery->setAlbumId($idalbum);
$photoQuery->setPhotoId($idfoto);
$photoQuery->setType('entry');

$photoEntry = $this->gfoto->getPhotoEntry($photoQuery);
$this->gfoto->insertCommentEntry($entry, $photoEntry);
```

Codi 8.5: Inserir comentari

Per inserir comentari es necessita l'usuari, l'identificador o el nom de l'àlbum i l'identificador de la fotografia.

Recuperar les Fotografies de Picasa Web

Per recupera un àlbum amb les fotografies es necessitat la funció *$this->initZendGdata()* per inicialitzar la connexió amb Picasa web [19]. S'ha de crear una consultar i es necessitat el codi de l'àlbum. La consulta en retorna un Feed amb tota la informació de l'àlbum, tal i com es mostra en el *Codi 8.6.*

```
$this->initZendGdata()
$consulta = $this->galbum->newAlbumQuery();
$consulta->setUser($this->usuariGoogle);
$consulta->setAlbumId($this->getIdGoogle());
$feed = $this->galbum->getAlbumFeed($consulta);
```

Codi 8.6: Funció recuperar Àlbum fotogràfic.

A través de la llibreria Zend_Gdata, l'objecte $feed és un vector de totes les imatges, per tant,

s'ha de recórrer aquest vector i anar accedint fotografia per fotografia.

Per recuperar la miniatura:

```
$foto->getMediaGroup()->getThumbnail()
$thumbnail[1]->url
```

Per recupera la imatge:

```
$f = $foto->getContent()
$f->getSrc()
```

Picasa en el mapa de Google Maps.

Per adjuntar miniatura en el mapa de Google Maps, es necessita una url amb extenció KML [17]. Aquesta extensió pertany el programa Google Earth, i sense aquest programa no pot generar el fitxer i tampoc la url. Picasa web en retorna la següent url:

```
http://picasaweb.google.com/data/feed/base/user/pfcagucosal/albumid/5344613728209453297?alt=rss&kind=photo&hl=es
```

En aquesta url, que és el Feed, podem veure diferents paràmetres de configuració en la url.
Veiem el nom del servei de Google amb qui ens comuniquem:

- `picasaweb.google.com`

El nom d'usuari:

- `pfcagucosal`

`El` identificador `de l'àlbum.`

- `5344613728209453297`

I el tipus de Feed que es compatible, que pot ser rss, alt=json-in-script, json o kml [20] (només és per Picasa Album Web), si canviem al paràmetre alt=**rss** per:

- **`kml`**

Amb aquest canvi, Google *Maps* interpreta aquesta url com un arxiu KML. Per mostrar les

miniatures, Google Maps te un objecte per carregar els fitxer XML. Si la url té al paràmetre alt=kml et mostra les miniatures en el mapa, tal i com es pot veure en el *Codi 8.7.*

```
geoXml = new GGeoXml(url);
map.addOverlay(geoXml);
```

Codi 8.7: Carrega les imatges de Picasa Web a Google Maps.

L'objecte GGeoXml afegeix contingut geogràfic al mapa a partir d'un fitxer XML i el contingut geogràfic que tenim són imatges. En la Il·lustració 8.4 es pot veure unes fotografies carregades en el mapa.

Il·lustració : Imatge carregades en el mapa

9 Proves

Per fer les proves, s'han realitzat diferents execucions en els dos navegadors més utilitzats en Internet amb el sistema operatiu Linux Ubuntu: Mozilla Firefox i Opera. D'aquest dos navegador es queden 23% de la quota de mercat dels navegadors d'Internet, Segons l'estadística del 26 de maig de 2009 per CyberNet.

L'entorn de proves consta d'un PC amb el següent software:

Sistema Operatiu: Linux Ubuntu 8.1.

Navegadors: Mozilla Firefox 3.0.10 i Opera 9.64..

En la següent taula *(Taula 1)* es poden veure amb detall las proves realitzades per la part administrativa de l'aplicació.

Proves de la part administrativa.

Proves realitzades	Resultat desitjat	Navegador	Coincideix amb el resultat esperat.
Login	Inicia sessió	Mozilla FireFox	SI
		Opera	SI
Cercar usuaris	Mostra usuaris en funció del criteri	Mozilla FireFox	SI
		Opera	SI
Habilitar usuaris	Usuaris no estar a la llista de deshabilitats	Mozilla FireFox	SI
		Opera	SI
Deshabilitar Usuaris	Usuaris no estar a la llista de habilitats	Mozilla FireFox	SI
		Opera	SI
Habilitar Ruta	La ruta no estar a la llista dels deshabilitats	Mozilla FireFox	SI
		Opera	SI
Deshabilitar Rutes	La ruta no estar a la llista dels habilitats	Mozilla FireFox	SI
		Opera	SI
Cercar Rutes	Mostra rutes en funció del criteri	Mozilla FireFox	SI
		Opera	SI
LogOut	Tanca la sessió	Mozilla FireFox	SI
		Opera	SI

Taula 1: Taula de proves de la part administrativa.

L'altre part, en la següent taula (*Taula 2*) poden veure amb detall les proves realitzades per l'aplicació de l'usuari.

Proves de la part pública.

Proves realitzades	Resultat desitjat	Navegador	Coincideix amb el resultat esperat.
Login	Inicia sessió	Mozilla FireFox	SI
		Opera	SI
Cercar Rutes	Mostrar les rutes en funció dels resultats	Mozilla FireFox	SI
		Opera	SI
Crear ruta	Mostrar el trajecte en el mapa	Mozilla FireFox	SI
		Opera	SI
Crear àlbum	Àlbum creat a Picasa Web	Mozilla FireFox	SI
		Opera	SI
Pujar Fotografia	Fotografia carregada a Picasa Web Album	Mozilla FireFox	SI
		Opera	SI
Esborrar àlbum	Àlbum esborrat a Picasa Web Album	Mozilla FireFox	SI
		Opera	SI
Esborrar Fotografia	Esborrar Fotografia	Mozilla FireFox	SI
		Opera	SI
Mostrar Ruta	Mostrar la ruta inserida pel usuari	Mozilla FireFox	SI
		Opera	SI
Mostrar Album	Mostrar les fotografies inserides per l'usuari	Mozilla FireFox	SI
		Opera	SI

Taula 2: Taula de proves de la part pública

Totes les pàgines desenvolupades em aquest projecte intenten seguir els estàndards HTML, CSS, XHTML i WAI del W3C.

10 Ampliacions pel projecte

Es proposen dues ampliacions per aquest projecte. Les dues part estan relacionades amb en entorn mòbils.

La primera aplicació seria adaptar la web per entorn mòbil. En aquest moments, l'accés web a través del mòbil no esta gaire estès, ja sigui perquè els dispositius mòbils no ofereixen gaire comoditat a l'hora de navegar l o per les tarifes de les companyies.

Però aquesta tendència ja esta canviant. Estan apareixen mòbils que necessiten tenir connexió a Internet, com el cas del HTC Magic de Google. També les companyies estan oferint tarifes planes per les connexions a Internet. Per tant, basant-nos amb aquesta tendència, sembla que uns ampliació podia ser:

- Estudi de com realitzar les modificacions a la web per adaptar-la als dispositius mòbils. Aquest estudi hauria de contenir les modificacions necessàries. per dur a l'èxit el canvi de la modificació de la interfície.
- Quins requeriments del punt 4.1 es poden o no adaptar a la web mòbil. Consisteix en detectar els requeriments que no són viables o no interessen per l'aplicació web mòbil.
- Utilitzar Google Maps per Mòbil[25]. Seria imprescindible poder consultar les rutes creades a través del mòbil. Google Maps té una versió per Mòbil, per tant es podia programa la visualització de la ruta.

La segona ampliació també esta relacionada amb els dispositius mòbils. Seria oferir un servei a l'usuari amb les següent característiques:

- Si l'usuari disposa d'un telèfon mòbil amb dispositiu gps, l'aplicació hauria de ser capaç de guardar el recorregut que fa l'usuari. I un cop finalitzada, des del mòbil enviar-la a la web.
- Si el dispositiu disposa de càmera, pot enviar les fotos de l'excursió, indicant la posició geogràfica per mostrar-la en el mapa.
- S'hauria de mostra la ruta amb les fotografies geogràficament situades en la posició indicada i la el recorregut de la ruta.

25 *http://www.google.com/intl/es/mobile/gmm/index.html*

11 Conclusió

En l'inici del projecte, l'única cosa que tenia clara era la idea que volia desenvolupar. Un lloc on els usuaris puguin compartir les seves rutes amb la resta d'usuaris, afegint suport gràfic, bàsicament fotografies. Durant el desenvolupament del projecte he viscut moltes dificultats i he passat moltes hores investigant i documentant en la tecnologia que utilitzaria.

Vaig triar una tecnologia que no coneixia pràcticament, i volia que interaccionés amb els serveis de Google, Google Maps i Picasa Web Album, que tampoc coneixia el funcionament. Per tant, ha fet que invertís moltes hores del projecte en aprendre, investigar i resoldre els errors i problemes, que el llenguatge de programació i els serveis provocaven, tant per interaccionar entre ells com per interacciona amb amb l'aplicació.

Crec que el desenvolupament de les aplicacions web, en un futur molt pròxim, tendirà a desenvolupar i adaptar mòduls per a gestors de continguts. O també, que les aplicacions web utilitzin els serveis d'Internet disponibles, com aquest projecte ha fet.

Ha estat un projecte molt profitós, ja que he après com interaccionant els serveis d'Internet, cosa que abans desconeixia, i, he ampliat el meu coneixement en la tecnologia web (referent a Symfony, php, Propel, CSS, Zend Gdata, etc).

Aprendre a gestionar un projecte, encara que no sigui tan gran com els projectes empresarials, també ha estat un repte, que amb més encerts que errades, he pogut finalitzar el projecte final de carrera just a temps. Cosa que amb fa pensar que la meva gestió ha estat prou encertada.

És un projecte interessant per entendre com funcionen les comunicacions entre els serveis d'Internet i com aplicar-les en una aplicació pròpia.

M'agradaria destacar que aquest projecte és una proposta de com resoldre la gestió de les rutes de muntanya, però aquesta documentació es podia utilitzar per desenvolupar altres projectes que utilitzessin aquest serveis.

Amb tot això, poso en disposició a la comunitat educativa un material prou interessant, com per ser consultat per altres projectistes, que necessitin una punt de partida per assolir els coneixements bàsics per iniciar els seus projectes.

12 Bibliografia

[1] http://www.symfony.es/, Blog de Symfony, 30 de gener de 2009.

[2] http://www.symfony-project.org, Symfony Open-Source PHP Web Framework, 30 de gener de 2009.

[3] http://dev.mysql.com/downloads/gui-tools/5.0.html, MySQL GUI Tools Downloads, 31 de gener del 2009.

[4] http://www.eclipse.org/pdt/, Eclipse PDT, 31 de gener de 2009.

[5] http://code.google.com/intl/es-ES/apis/maps/documentation/, Conceptes sobre el API de Google Maps, 1 de febrer de 2009

[6] http://www.php.net ,Pàgina oficial de PHP, 1 de febrer de 2009.

[7] http://www.otrobloggeek.com/blog/2008/05/howto-symfony-framework-en-ubuntu-hardy-840/, Otro blog Geek, 1 de febrer de 2009.

[8] http://www.maestrosdelweb.com/editorial/el-framework-symfony-una-introduccion-practica-i-parte/ , Maestros del web, 1 de febrer de 2009.

[9] http://groups.google.es/group/**symfony**-es ,Grups de Google sobre Symfony, 2 de febrer de 2009.

[10] http://www.symfony-project.org/api/1_2/ , API de Symfony, 2 de febrer de 2009.

[11] http://propel.phpdb.org/trac/wiki/Users/Documentation/1.3/BasicCRUD , Pàgina oficial de Propel, 5 de febrer de 2009.

[12] http://www.svennerberg.com/2008/10/polylines-in-google-maps/ ,Svennerberg, Exemple sobre PolyLines, 5 de febrer de 2009.

[13] http://googledataapis.blogspot.com/2007/06/picasa-web-albums-data-api-now-with-geo.html, Official Google Data APIs Blog, 7 de febrer de 2009.

[14] http://code.google.com/intl/es-ES/apis/picasaweb/overview.html, API de dades de Albums web de Picasa, 9 de maig 2009.

[15] http://code.google.com/intl/es/apis/picasaweb/developers_guide_php.html, Developer's Guide: PHP of Picasa Web Album. 9 de maig de 2009.

[16] http://www.librosweb.es/symfony/index.html, Symfony, la guia definitiva, 30 de gener de 2009.

[17] http://code.google.com/intl/es/apis/kml/documentation/kml_tut.html#placemarks, Tutorial KML Google Earth, 15 de març 2009.

[18] http://framework.zend.com/manual/en/, Programmer's Referents Guide, 17 de març de 2009.

[19] http://code.google.com/intl/es-ES/apis/picasaweb/articles/gme_picasa_web.html, Using Picasa Web Albums Feeds in Google Mashup Editor, 25 de maig de 2009

[20] http://code.google.com/intl/es/apis/kml/documentation/kmlreference.html KML

Reference, 29 de maig de 2009.

ANNEX I Desenvolupament del projecte amb Symfony

12.1 Instal·lació i creació d'un projecte Symfony.

Symfony [16] és independent de la plataforma, per tant, un cop instal·lat PHP [6]. No disposa de entorn gràfic de desenvolupament especific per Symfony, com Java – Eclipse o NetBeans. Per tant, s'ha d'utilitzar la shell per instal·lar al framework. La instal·lació [7] es realitza de la següent manera:

- S'ha de afegir el canal de Symfony a PEAR. El canal es el repositori on es troba les fonts de Symfony:

```
> pear channel-discover pear.symfony-project.com
```

- Un cop afegit el canat s'ha de instal·lar les fonts al equip.(si en la instrucció s'escriu com *pear install symfony/symfony-1.2.4* s'instal·la la versió que has indicat, en aquest cas 1.2.4)

```
> pear install symfony/symfony

downloading symfony-1.2.4.tgz ...
Starting to download symfony-1.2.4.tgz (1,283,270 bytes)
.....................................................................
.....................................................................
.............done: 1,283,270 bytes
install ok: channel://pear.symfony-project.com/symfony-1.2.4
```

- Comprovem que la versió de Symfony s'ha instal·lat correctament. També ens mostra on estan instal·lades les fonts del Symfony.

```
> symfony -V
```

```
symfony version 1.1.0
```

```
(/ruta/on/esta el/directori/lib/dir/de/Symfony/en/PEAR)
```

Un cop realitzat aquest últim pas, ja tenim Symfony instal·lat.

➢ El següent pas a realitzar és la configuració del servidor de aplicacions. Normalment es troba generalment a *apache/conf/httpd.conf , però* en el sistema operatiu Ubuntu es troba en el directori */etc/apache2/sites-available/default* (*default* és el fitxer que s'ha de modificar).

```
 <VirtualHost *:80>
    ServerName localhost                                    /*localhost o nom
del projecte.com*/
    DocumentRoot "/var/www/pfc/web"                        /* adreça on es trobar
el projecte*/
    DirectoryIndex index.php           /* la pàgina de inici, obligatoriament
index.php*/
    Alias /sf /usr/share/php/data/symfony/web/sf       /*ruta on es troba la web
de symfony*/
    Directory "/usr/share/php/data/symfony/web/sf/">
                                  /*donem permisos totals per es puguin executar
les web de Symfony*/
           AllowOverride All
           Allow from All
    </Directory>
    <Directory "/var/www/pfc/web">
                              /*donem permisos totals perquè es puguin executar la
nostra aplicació */
           AllowOverride All
           Allow from All
    </Directory>
    </VirtualHost>
```

Fitxer default

En sistemes Linux, concretament ubuntu, es recomanable crear les aplicacions web en el directori del servidor web, que es */var/www* y despres crear una carpeta on guardarem l'aplicació. En aquest cas, s'ha creat una carpeta anomenada pfc (sigles Projecte Final Carrera) com es pot veure a la *Il·lustració A1.3*.

Il·lustració A1.3: Directori web i del projecte

12.2 Crear un projecte Symfony

Crear un projecte amb Symfony [8] és molt senzill, tot i que s'ha d'utilitzar la Shell.

Els projectes Symfony segueixen una estructura de directoris definida. Les instruccions que té Symfony permeten automatitzar la creació de nous projectes, ja que s'encarreguen de crear l'estructura de directoris bàsica del projecte i amb els permisos adequats. Per tant, per crear un projecte s'ha de crear un directori, en aquest cas es *pfc,* i dir-li a Symfony que creï un projecte en el directori.

```
> cd ~/miproyecto
> symfony generate:project miproyecto
```

El projecte acabat de crear està incomplet, ja que requereix com a mínim d'una aplicació. Per crear l'aplicació, s'utilitza el comandament *symfony generate: app,* al qual se li ha de passar com a paràmetre el nom de la nova aplicació:

```
> php symfony generate:app admin
```

Aquí tenim l'estructura de una aplicació d'un projecte Symfony:

```
apps/
  admin/

    config/

    i18n/
```

```
    lib/

    modules/

    templates/
```

Al directori web del projecte també es creen alguns arxius PHP corresponents als controladors frontals de cada un dels entorns d'execució de l'aplicació. Apache utilitza per defecte index php:

```
web/
  index.php
  admin_dev.php
```

Un cop creat un projecte hem de crear l'aplicació, que es on hi hauran els mòduls (mòduls són les accions més les plantilles), i l'instrucció és:

```
> php symfony generate:module admin ruta
```

I es crear la següent estructura en el directori *app,* en l'aplicació *admin* en la carpeta *modules*

```
/modules
        /ruta
        /actions
                actions.class.php
        /templates
                index.php
```

Recordem que el nom de l'acció ha de coincidir amb el nom de la plantilla tal i com s'explica en el punt *6.1.6 Fi de les accions.* Si executessim l'apliació el navegador ens obria la pàgina de benvinguda de Symfony, tal i com es pot veure en la *Il·lustració A1.4*

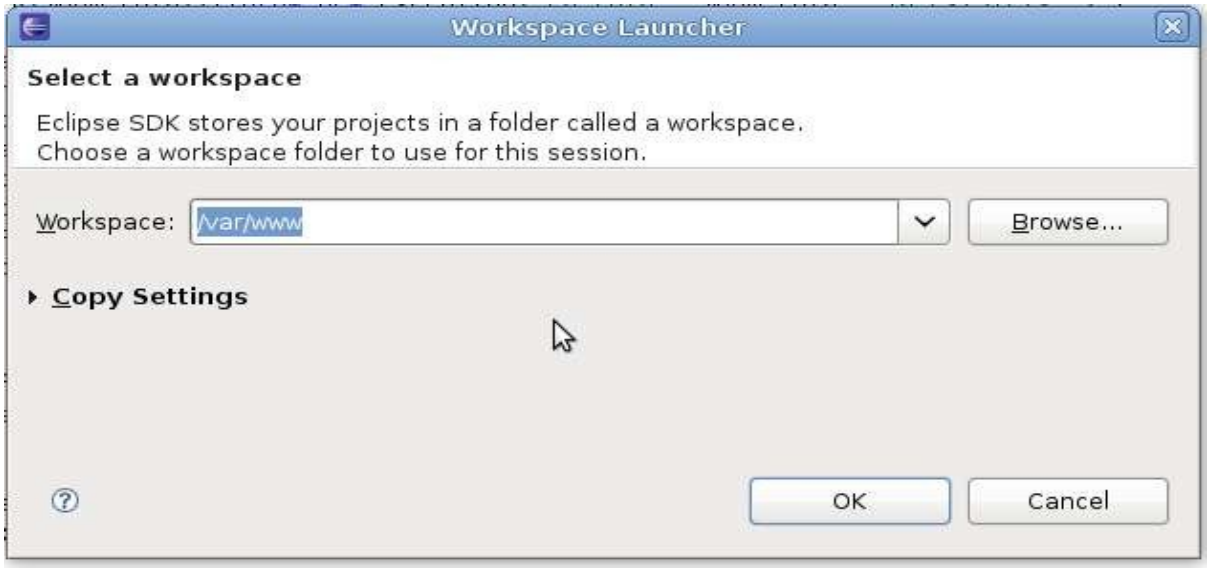
Il·lustració A1.4: Pantalla de benvinguda de Symfony

13 ANNEX II. IDE Eclipse.

13.1 Crear un projecte.

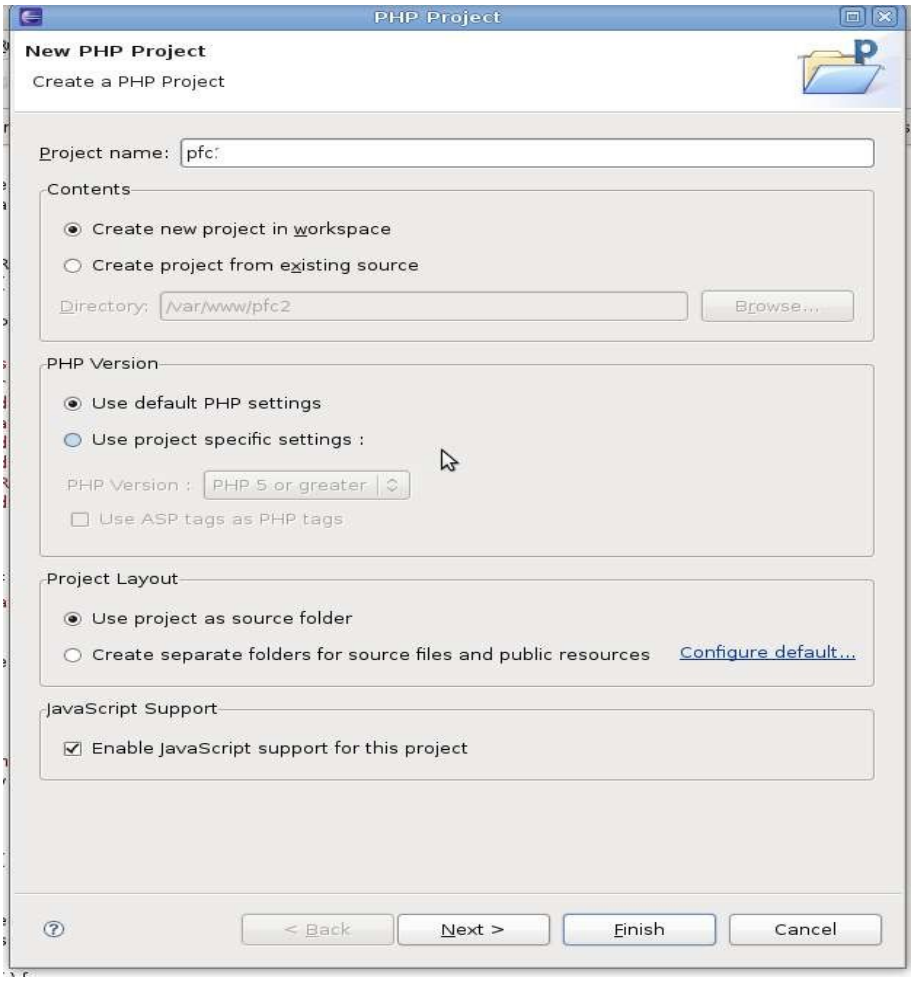
Encara que Symfony no disposa d'un entorn de desenvolupament propi, El programa utilitzat per aquest projecte es Eclipse [4] amb el plugin PDT que es capaç d'executar pàgines PHP i configurar-se amb el servidor Apache. Eclipse necessita la Maquina Virtual de Java per funcionar. Un cop descarregat l'IDE de la seva pàgina de Eclipse[26] es descomprimeix en un directori qualsevol i s'executa l'executable de Eclipse.

26 *http://www.eclipse.org/pdt*



Il·lustració A2.1: Pantalla per trial el workspace

Inicalment ens demanara un espai de treball (Workspace) s'ha d'introduir el directori web de Apache, com s'indica a l'*Il·lustració A2.1* .

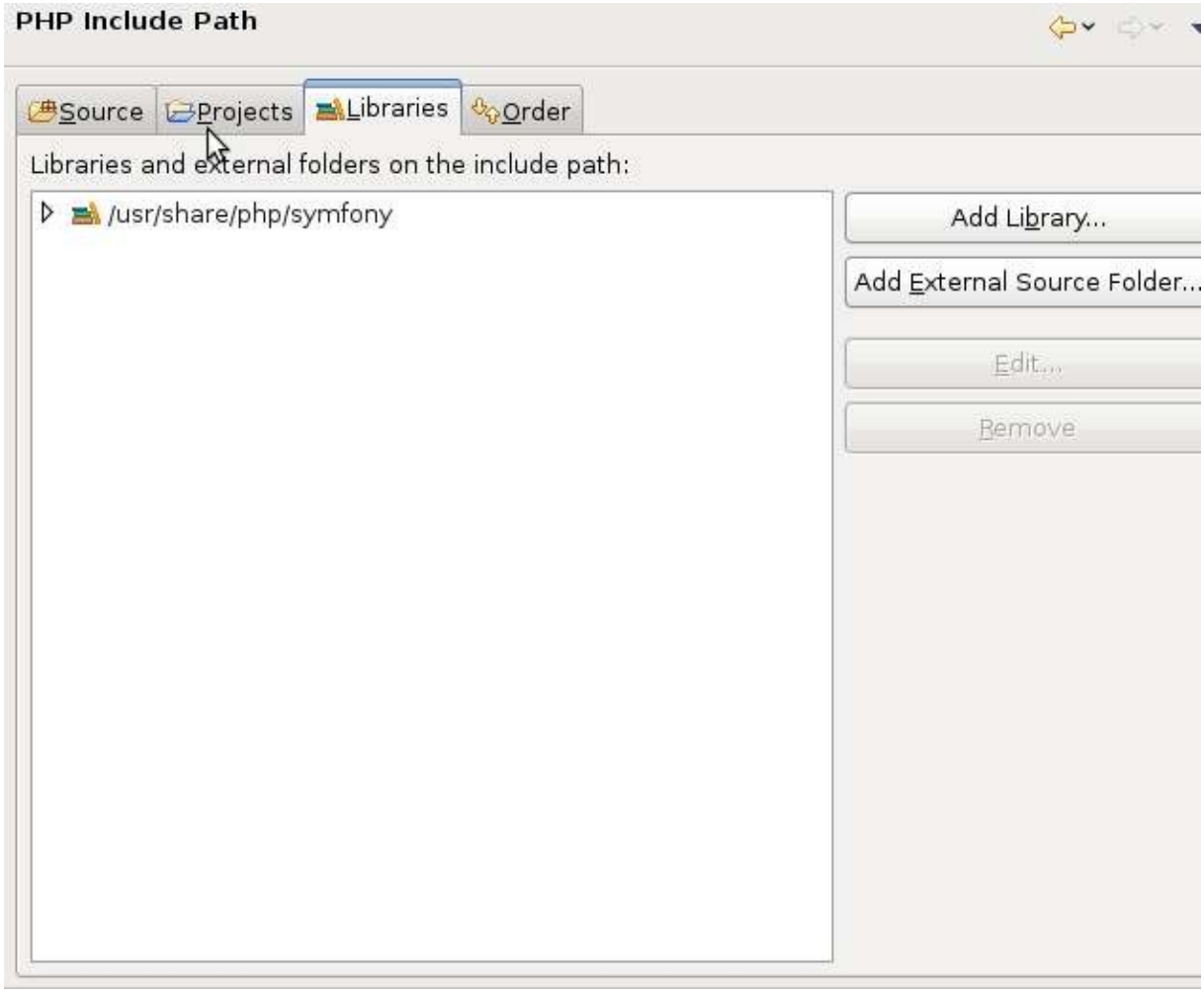


Il·lustració A2.2: Pantalla on s'introdueix el nom del projecte

Un cop triat l'espai de treball, Eclipse demanarà que introduïm el nom del projecte introduirem el nom *pfc,* com es mostra en la *Il·lustració A2.2*. El nom del projecte ha de coincidir amb el nom de la carpeta on guardarem el nostre projecte (que *pfc*) que esta a */var/www.*

Les altres opcions es deixen per defecte.

El següent pas es afegir al framework Symfony a Eclipse. En la següent pantalla que apareix seleccionem la pestanya que posa *Libraries* i un cop dins la pestanya polsem el boto *Add Esternal Source Folder* per importar Symfony. Hem de selecciona la carpeta on es guarda les llibreries de Symfony. Aquesta direcció en Linux Ubuntu es correspon a l'adreça */usr/share/php/symfony ,* tal i com es mostra en la *Il·lustració A2.3*

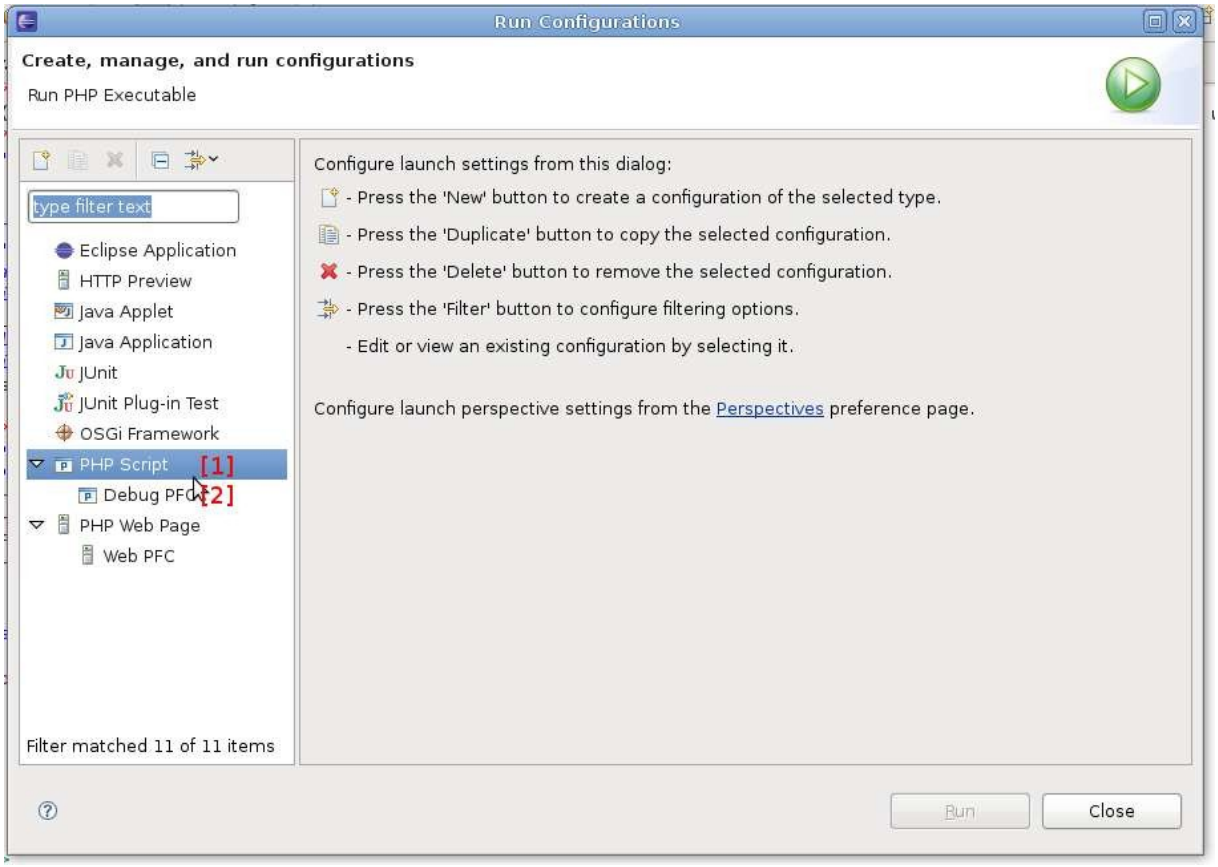


Il·lustracióA2.3 : Pantalla per incloure les llibreries de Symfony

Un cop importat la llibreria de Symfony polsem el boto Finish i ja tenim preparat Eclipse per desenvolupar en Symfony.

13.2 Configuració Xdebug en Eclipse

Per configurar XDebug amb Eclipse s'ha d'anar a menú superior del Eclipse *Run > Run Configurations* per obrir la pantalla per configurar el debugger per Eclipse perquè funcioni amb PHP, com es pot observar la *Il·lustració A2.4*



Il·lustració A2.4: Pantalla de configuració de Eclipse

Un cop en la pantalla que mostra la *Il·lustració A2.4.* Es fa doble clic sobre la opció PHP Script [1], que que sortiu sota la aquesta opció una nova configuració anomenada *New_configuration* i s'ha reanomenat a *Debug PFC* [2]

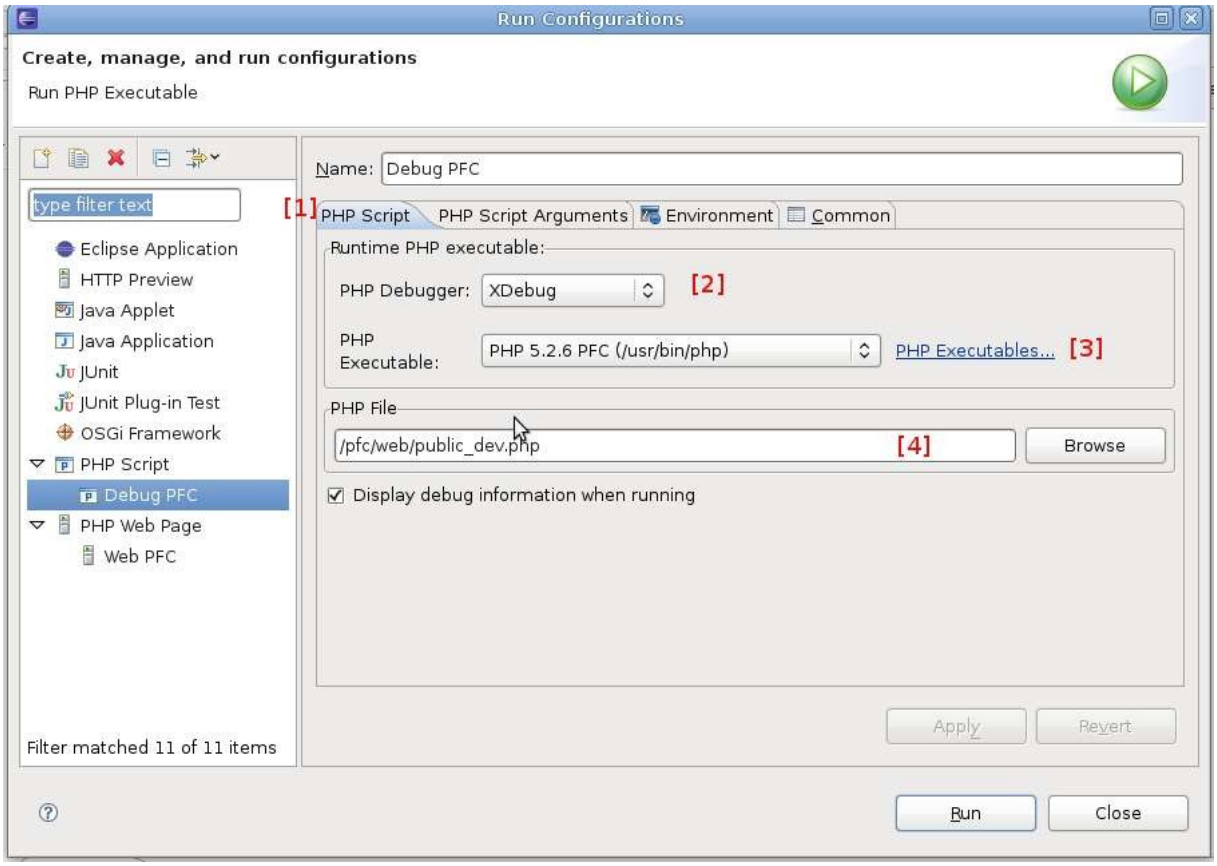
En sortira un formulari on hem de seleccionar les configuracions adients per fer funcionar. Eclipse oferix Zend Debugger com a debugador per defecte, però en aquest haurem de seleccionar la opció de Xdebug instal·lat prèviament.

Eclipse també ofereix el seu propi executable de PHP, que no s'ha de seleccionar ja que Apache no el utilitza.

Com mostra la Il·lustració A2.5 en la pestanya *PHP Script* [1] en la configuració *PHP Debugger* [2] seleccionem Xdebug[27].

En la configuració *PHP File* [4] escrivim la ruta de la pàgina web que volem que s'executi. Aquesta ruta es la corresponen del projecte del Eclipse.

27 *http://www.xdebug.org*



Il·lustració A2.5: Configuració del XDebug

La primera vegada que s'ha obert Eclipse, s'ha de configura el PHP fent clic sobre l'enllaç blau PHP Executables, afegim una nova configuració i surt una pantalla, com es mostra en la *Il·lustració A2.6*. El PHP que s'ha de configurar es el que tenim enllaçat amb l'Apache. Eclipse porta un PHP que no és el que tenim perquè funcioni amb l'Apache, per tant, s'ha de configurar perquè ens agafi el PHP de l'equip.