Universitat Autònoma de Barcelona

escola d'enginyeria

DESENVOLUPAMENT D'UN PORTAL WEB PER UN CENTRE EDUCATIU UTILITZANT COMPONENTS OPEN SOURCE

Memòria del Projecte Fi de Carrera

d'Enginyeria en Informàtica

realtizat per

Xavier Espona Roig

i dirigit per

Diego Javier Mostaccio Mancini

Bellaterra, 09 de Setembre de 2015

Universitat Autònoma de Barcelona

escola d'enginyeria

El sotasignat, Diego Javier Mostaccio Mancini, professor de la Escola d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a que correspon aquesta memòria ha estat realitzat sota la seva direcció per en Xavier Espona Roig.

I per tal que consti firma la present.

Signat: Diego Javier Mostaccio Mancini

Bellaterra, Setembre de 2015

Índex

Capítol 1

Introducció

1.1 Motivació

Avui en dia la gran majoria de centres educatius disposen d'una plataforma LMS *Learning Management System* on els estudiants poden veure els continguts dels cursos on participen, fer entregues de materials, realitzar testos o exàmens, entre d'altres. Tot i la constant evolució d'aquestes plataformes, les disponibles al mercat están totalment desconectades de les altres eines web necessàries per a la gestió d'un centre educatiu, requerint una gran part d'intervenció manual al llarg del procés de formació de l'alumnat.

Aquest projecte vol donar una visió de les diferents plataformes Open Source existents al mercat enfocades construcció d'un portal web per a un centre educatiu i aportar una aproximació de solució unificada per gestionar tot el pas d'un alumne per el mateix, des de la matriculació fins a la obtenció del certificat d'estudis, tot això utilitzant components Open Source.

1.2 Objectiu

L'objectiu del projecte és construïr l'estructura d'una aplicació web que, complint els requisits anteriors, sigui el suficientment flexible com per a adaptar-se a les infraestructures ja existents d'un centre. De manera que la seva implantació no resulti excessivament costosa i permeti anar incorporant diferents funcionalitats gradual-

ment. A continuació es mostra un llistat de les diferents funcionalitats de les que es vol dotar a l'aplicació.

1. Creació i gestió d'usuaris i cursos.
2. Login i Registre d'usuaris.
3. Gestió dels rols d'usuaris.
4. Creació de grups de cursos.
5. Creació de grups d'usuaris.
6. Matriculació a un curs / grup de cursos.
7. Pagament amb tarja de crèdit.
8. Pagament amb una tarja guardada prèviament.
9. Accés als continguts d'un curs.
10. Seguiment de l'estat d'un usuari en un curs / grup de cursos.
11. Restriccins d'accés als cursos d'un grup.
12. Notificacions als usuaris (pròximes entregues, events, notes, etc).

Capítol 2

Estat de l'art

Aquest capítol descriu algunes de les diferents eines Open Source disponibles al mercat que es poden utilitzar a l'hora de construir una aplicació web per a un centre educatiu, diferenciant entre la creació del portal i de la plataforma educativa en si.

2.1 Desenvolupament d'aplicacions web

A l'hora de desenvolupar una aplicació web complexa, podem optar per crear-la des de zero o utilitzar alguna de les moltes opcions que proporciona el mercat.

Per a equips petits on el temps de desenvolupament és molt important la primera opció no acostuma a ser viable si es vol assegurar un mínim de qualitat en el producte que s'està creant.

Les principals eines que podem trobar al mercat es poden agrupar en dos categories ben diferenciades: *Frameworks* i CMS (*Content Management System*). Cadascuna té una sèrie d'avantatges i inconvenients i és important tenir clar les implicacions que suposarà la tria d'una o altra tant a l'hora de construir l'aplicació web, com d'afegir-hi funcionalitats posteriorment.

2.1.1 Frameworks

Un Framework o, més concretament, un *Web Application Framework [1]* és un software orientat a objectes compost de components personalitzables i intercam-

biables per al desenvolupament d'aplicacions web.

Es pot considerar com un conjunt d'eines que proporcionen una estructura de desenvolupament per tal d'ajudar a realitzar les tasques mes comunes a l'hora de construir una aplicació web, ja sigui senzilla o complexa. L'utilització d'un Framework permet accelerar el desenvolupament i la capacitat de reutilització de codi alhora que promou la utilització de bones practiques de programació, com pot ser l'ús de patrons de disseny de software.

Els frameworks proporcionen una sèrie d'eines que, un cop superada la corba d'aprenentatge, permenten als programadors no haver de preocupar-se de programar les tasques més comunes i repetitives. D'aquesta manera, el temps de desenvolupament baixa i s'aconsegueixen aplicacions més robustes ja que aquests components han estat dissenyats i testejats exhaustivament abans de ser desplegats [2].

2.1.1.1 Funcionalitats i característiques més comunes

Seguretat Tasques tant comunes com per exemple la gestió de sessions, contrassenyes d'usuaris, o consultes a la base de dades són unes de les principals causes de forats de seguretat en les aplicacions web. Al utilitzar correctament un framework, el desenvolupador no s'ha de preocupar d'aquests problemes, que acostumen a ser tediosos i causen una gran pèrdua de temps.

Enrutament En una aplicació web complexa, l'enrutament i gestió de les diferents URLs no és una tasca trivial. Quan es vol canviar una ruta a una pàgina s'ha de tenir en compte que també s'haurán de canviar tots els enllaços i redireccions a aquesta. L'us de *routers* converteix aquest problema en una tasca molt més senzilla ja que permet definir una conjunt de rutes per a la aplicació i, a l'hora de crear enllaços o redireccions, fer-ho sobre aquestes rutes. D'aquesta manera, quan canviem la URL d'una ruta automàticament canviaran tots els enllaços de l'aplicació. A més, sovint també hi ha la possibilitat de validar els paràmetres de les rutes automàticament, de manera que sabem que les rutes sempre es generaran correctament.

Gestio de la Base de Dades Escriure consultes 'senzilles' a la base de dades cau-

sa una gran pèrdua de temps a l'hora de desenvolupar una aplicació. Les operacions de tipus CRUD (*Create, Read, Update and Delete*) sobre els objectes de l'aplicació constitueixen la gran majoria de les consultes. El fet de disposar d'una capa DBAL (*DataBase Abstraction Layer*) o un ORM (*Object relational mapper*) que proporcionen funcions per a fer aquestes operacions ajuda a reduïr molt el temps de desenvolupament alhora que proporciona seguretat, ja que aquestes funcions s'encarreguen d'escapar i netejar els paràmetres.

Sistemes de plantilles Aporten flexibilitat a l'hora de canviar la presentació del contingut i permeten reutilitzar blocs de codi per tal de no haver d'escriure cada cop les parts repetitives, com per exemple el header o el footer de la pàgina. Un altre punt important és que permeten crear el conjunt de les plantilles abans de tenir desenvolupada completament l'aplicació, ajudant a pararelitzar el desenvolupament *front-end* i *back-end*. També ofereixen la possibilitat de canviar tota la presentació de la pàgina sense canviar les dades que s'hi mostren i, per tant, permet fer-ho sense necessitat de conèixer el codi de l'aplicació.

Generació de Formularis Alguns frameworks, com per exemple Symfony2, permeten definir una classe de tipus formulari, on anar afegint els diferents camps alhora que es defineix el tipus, validació, etiqueta o atributs de cadascun. Un cop definida la classe, disposen de funcions per a renderitzar els objectes de tipus de formulari. D'aquesta manera es definex quina serà l'estructura del formulari i la seva renderització en codi html es converteix en trivial.

***Cache* de pàgines** La majoria de frameworks creen automàticament una *cache* de les pàgines el primer cop que es visiten, és a dir, guarden una còpia del html resultant un cop l'intèrpret les ha processat. Així, a les pròximes visites, la pàgina no s'ha de tornar a generar sinó que directament es mostra la còpia que s'ha emmagatzemat al disc, millorant de forma molt notable el temps de resposta de l'aplicació.

2.1.1.2 Frameworks dels llenguatges de programació web més utilitzats:

PHP Zend, Symfony, Laravel, Yii, CodeIgniter, ...

Java Spring, Struts, JSF, ...

Python Django, TurboGears, Pyramid, Flask, ...

Ruby Rails, Padrino, Grape, ...

2.1.2 Gestors de Contingut (CMS)

Un Gestor de Continguts [3] és una aplicació desenvolupada per la creació i administració de llocs webs dinàmics, completament preparada per començar a crear contingut, crear i gestionar usuaris i permisos. Disposen d'un ampli conjunt d'extensions per afegir diferents funcionalitats, de manera que un usuari no iniciat en programació sigui capaç de construïr un portal web sense problemes.

Acostumen a estar desenvolupats sobre un framework al que proporcionen accés, de manera que són costumitzables si es coneix el llenguatge de programació en el que estan basats, sempre tenint en compte les limitacions imposades pel fet de ser un software amb una arquitectura molt definida.

2.1.2.1 Funcionalitats i característiques més comunes

Estructura definida Tant a nivell de d'arquitectura del software, com de representació del contingut, l'estructura d'un CMS està completament definida. Això representa alhora un avantatge, doncs es disposa d'una aplicació totalment funcional i preparada per treballar, i un inconvenient, ja que obliga a adaptar-s'hi amb la conseqüent corba d'aprenentatge i l'obligació de dissenyar les modificacions que es volen realitzar en funció d'aquesta arquitectura.

Back-end Tots els gestors de contingut incorporen un back-end d'administració des d'on es configura l'aplicació, es gestionen els usuaris, continguts, extensions, plantilles, etc. Aquest facilita el treball dels administradors de la plataforma i estalvia una gran part de temps de desenvolupament comparat amb un framework on aquest s'hauria de desenvolupar des de zero.

Temes i editors de contingut Per tal que els usuaris puguin crear continguts de forma fàcil i ràpida, els CMS incorporen temes (plantilles) ja construïts, que donen estructura visual al contingut del site. A més incorporen editors de contingut anomenats *WYSIWYG* (*What You See Is What You Get*); aquests permeten editar text amb format enriquit aplicant els estils de text definits al disseny de la plantilla.

Plugins Les grans comunitats d'usuaris que hi ha al voltant dels gestors de contingut fa que es disposi de multitut de plugins ja desenvolupats per a realitzar tasques diferents a les que incorpora l'aplicació base. D'aquesta forma es poden afegir diverses funcionalitats sense haver de dedicar temps al desenvolupament de les mateixes.

2.1.2.2 Tipus de Gestors de Continguts

Al mercat es poden trobar molts Gestors de Continguts diferents, una forma adient de classificar-los és en funció del propòsit per al qual serveixen.

Fòrums PHPBB, SMF, vBulletin, ...

Wikis PHPWiki, MediaWiki, MoinMoin, ...

eCommerce Magento, Prestashop, osCommerce, ...

Websites Wordpress, Drupal, Joomla!, ...

eLearning Moodle, Dokeos, Sakai,...

2.1.3 *Learning Management Systems (LMS)*

Un *Learning Management System* és un tipus concret de Gestor de Continguts enfocat a l'ensenyament. Són utilitzants tant en formació on-line com a element conductor de l'ensenyament i en formació presencial com una eina que acompanyi la formació als centres, i es fan servir tant en entorns laborals com eduactius. L'estructura i funcionalitats són molt similars a la d'un CMS però centrat en la gestió

dels usuaris, la creació de cursos i materials de formació i la gestió de l'accés dels usuaris a aquests.

2.1.3.1 Característiques

Gestió d'usuaris Dins del sistema i assignació d'usuaris a cursos.

Assignació de rols Als usuaris per donar-los més o menys permisos dins d'un curs.

Creació i gestió de materials Permeten pujar materials en diferents formats o crear-ne de nous.

Calendari De tasques, fites i events del curs.

Evaluació Mitjançant testos o exàmens.

Valoració De la participació al curs, agrupant i ponderant la participació en les tasques, testos i activitats.

Fòrums De discussió on els estudiants i professors poden participar.

2.1.3.2 Carències

Si bé els LMS han evolucionat molt des de la seva aparició, la gran majoria de les plataformes Open Source disponibles al mercat tenen una mancança molt important: es centren exclusivament en el desenvolupament d'un curs i no cobreixen les demés necessitats que tenen els centres educatius a l'hora de gestionar la participació d'un alumne en el curs o plà d'estudis. Algunes d'aquestes necessitats que no cobreixen els LMS són les següents.

Agrupació De cursos en forma de plans d'estudi, com es fa en un centre amb educació presencial.

Login unificat Amb la resta d'eines on-line del centre.

Matriculació Si bé els LMS permeten afegir usuaris a cursos, aquest procés no inclou la gestió del pagament i en molts centres suposa un pas a gestionar manualment per al personal del centre.

Pràctiques En molts cursos es realitza part del programa d'un curs en subgrups d'alumnes, com pot ser el cas de pràctiques o seminaris. La majoria de LMS no ofereix una funcionalitat per aquests casos.

Evolució de l'estudiant Seguiment de l'evolució de l'estudiant al llarg de la seva participació al curs o cursos d'un plà d'estudis, poden veure la seva evolució i donant accés a cursos en funció de diferents requisits com pot ser els cursos superats prèviament.

Promoció Del centre, dels seus plans d'estudi, mètodes educatius, professorat, campus etc.

Capítol 3

Planificació i Recursos

En aquest apartat es llisten els recursos necessaris per a realitzar aquest Projecte.

3.1 Hardware

L'infraestructura necessària per a realitzar les tasques de desenvolupament del projecte i la seva posterior implantació és la següent.

Servidor Una instància virtual a AWS (*Amazon Web Services*). Processador Intel Xeon E5-2676@2.4 GHz amb 8 GB de RAM.

Emmagatzemament Disc SSD de 8 GB per guardar el codi que s'ha d'executar constantment i un disc magnètic de 512 GB per les demés dades.

Desenvolupament Portàtil Apple MacBook Pro amb OS X 10.10.4. Aquesta és la maquina utilitzada per a la realització del projecte, tot i que un ordinador de gamma mitja seria suficient.

3.2 Software

Programari utilitzat per al desenvolupament de l'aplicació i programari necessari per al correcte funcionament de la platafoma web. Exceptuant el sistema operatiu del servidor d'Amazon, tot el programari utilitzat és Open Source i gratuït.

Sistema operatiu Amazon Linux AMI.

Servidor HTTP Apache 2.4.10.

Base de Dades MySQL 5.5.42.

Intèrpret PHP 5.5.21.

Framework Symfony 2.3.27 LTS.

LMS Moodle 2.8.3.

Editor VIM 7.3.

UML StarUML

3.3 Planificació

Un cop vist l'estudi sobre els recursos necessaris s'ha realitzat una planificació inicial del treball en funció del temps disponible i l'estimació del temps que es creu que serà necessari per la realització de les diferents parts.

En el Diagrama de Gantt de la figura 3.1 es pot veure la planificació inicial que es va fer al principi del projecte.

La planificació inicial no s'ha complert, afectant al desenvolupament de diferents mòduls de l'aplicació, com pot ser el *tracking* de l'alumne o les notificacions. Aquest enraderiment ha sigut causat principalment per un motiu: el canvi en la manera d'afrontar el projecte on s'ha donat més importància al disseny del software, dissenyant i escrivint interfícies per al mòdul de pagament i al d'integració amb LMS que no pas a la implementació de moltes funcionalitats concretes. Aquest canvi s'ha realitzat amb l'objectiu de dotar de més extensibilitat i flexibilitat a l'apliació, permetent que s'adapti a diferents plataformes de pagament i diferents LMSs. Això ha provocat que la implementació dels altres mòduls s'hagi allargat més del previst.

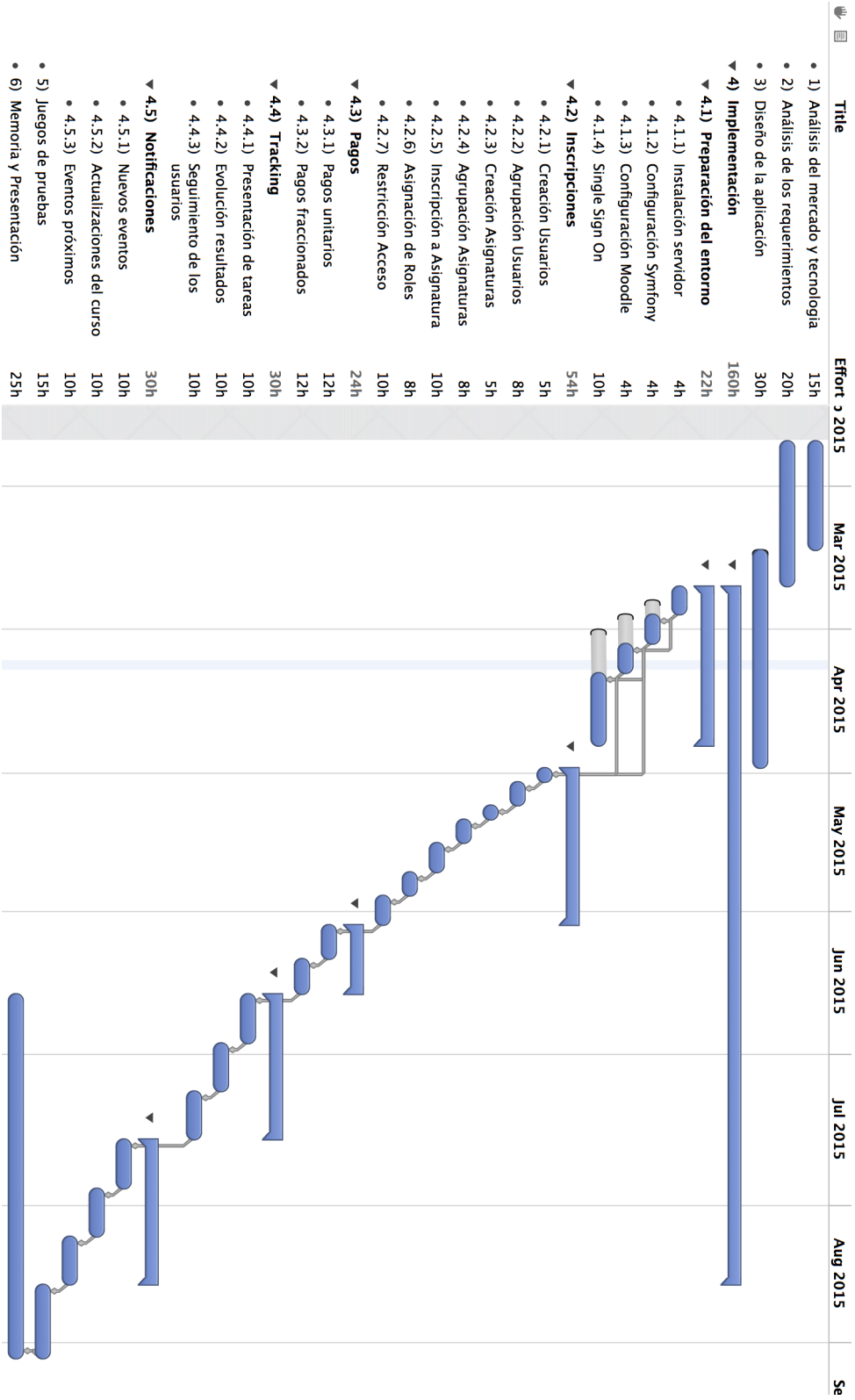

Figura 3.1: Planificació inicial

3.4 Estimació de costos

En aquest apartat es vol fer una estimació del cost que suposaria realitzar aquest projecte per una empresa ja existent, aquesta suposició implica que no es conta el cost de donar d'alta l'empresa ni el lloguer de la oficina, entre d'altres.

D'una banda es calculen els costs que suposaria pagar a un analista que s'encarregui del disseny de l'arquitectura de l'aplicació i a un desenvolupador que l'implementi. Els preus / hora són aproximats i intenten reflectir un salari competitiu a Barcelona incloent en el preu els impostos que l'empresa paga per el treballador.

També s'inclouen els costos del lloguer anual del servidor a Amazon Web Services descrit a l'apartat anterior (pagant tot l'any per avançat) [4], el cost dels ordinadors necessaris per als treballadors (un ordinador / portàtil genèric de gamma mitja amb un monitor extra) i el cost del software, que en aquest cas és zero, ja que tot el que s'ha llistat anteriorment és gratuït.

Analista Programador 65h * 30€ / hora = 1.950€

Desenvolupador 160h * 16€ / hora = 2.560€

Material 2 * 1.000€ + 2 * 150€ = 2.300€

Servidor 1.300€

Software 0€

Total = 8.110€

Capítol 4

Disseny de l'aplicació

En aquest capítol es descriu l'estructura de l'aplicació web, components utilitzats, estructura de la Base de Dades i arquitectura del software, alhora que es valoren les diferents tries que s'han fet a l'hora de desenvolupar-la.

4.1 Estructura de l'aplicació

L'aplicació s'ha desenvolupat utilitzant PHP i MySQL a la part del servidor i HTML5, CSS, Javascript i jQuery per al frontal. Les raons darrera d'aquestes tries és que són les tecnologies més utilitzades en el desenvolupament d'aplicacions web [5] i són utilitzades per moltes de les pàgines més visitades actualment, a més de ser tecnologies gratuïtes i en constant evolució.

A l'hora de triar un framework o CMS s'ha triat utilitzar el framework Symfony2, la raò principal apart de que és un framework robust i molt utilitzat al mercat és perquè ja es té una mica d'experiència a l'hora de treballar amb ell, cosa que redueix molt el temps d'adaptació al ja conèixer les seves peculiaritats. A més utilitzar un framework en comptes d'un Gestor de Continguts proporciona més flexibilitat, un dels objectius principals del projecte.

Els components principals que s'han desenvolupat per a l'aplicació es poden separar en:

Interfícies Les interfícies defineixen els mètodes que s'han d'implementar quan es vol incorporar una nova 'llibreria d'integració'. L'aplicació permet, mitjançant

els paràmetres de configuració, definir quina llibreria és la que s'instanciarà. D'aquesta manera es dona flexibilitat a l'aplicació, ja que s'aconsegueix que no depengui d'una implementació concreta ni d'un software determinat.

Mòduls Una sèrie de mòduls diferenciats (usuaris, cursos, comandes, ...). Aquests mòduls proporcionen diferents funcionalitats, emmagatzemen les dades a les corresponents taules de la Base de Dades i realitzen, quan és necessari, crides a les interfícies o, més ben dit, als objectes que les implementen.

A la figura 4.1 es pot veure com les diferents parts de l'aplicació s'interrelacionen.

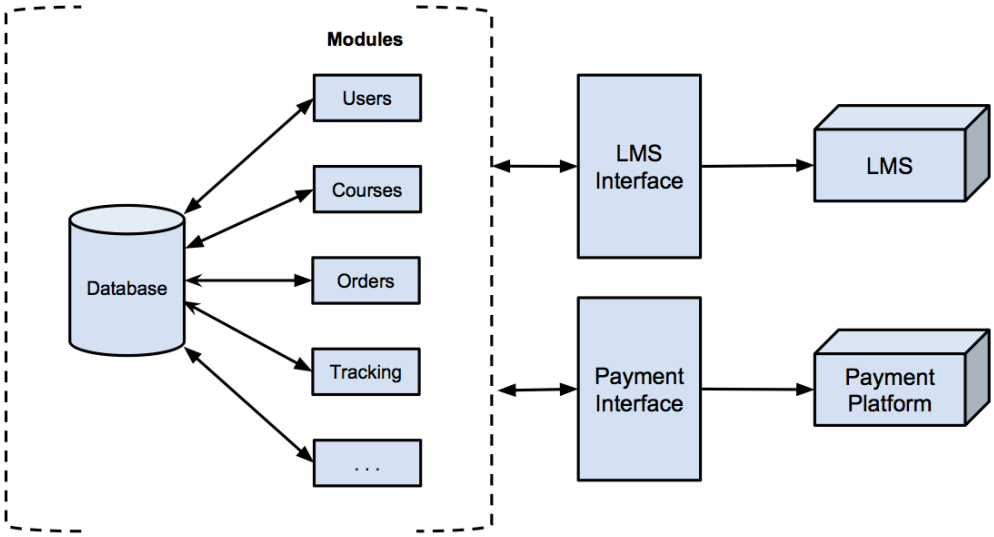

Figura 4.1: Estructura de l'aplicació

4.2 Estructura de la Base de Dades

La figura 4.2 mostra el diagrama relacional de la Base de Dades. En aquest es defineixen les diferents entitats de l'aplicació i les relacions existents entre elles.

Només s'especifiquen els camps necessaris per tal de tenir una idea general de com és aquesta estructura.

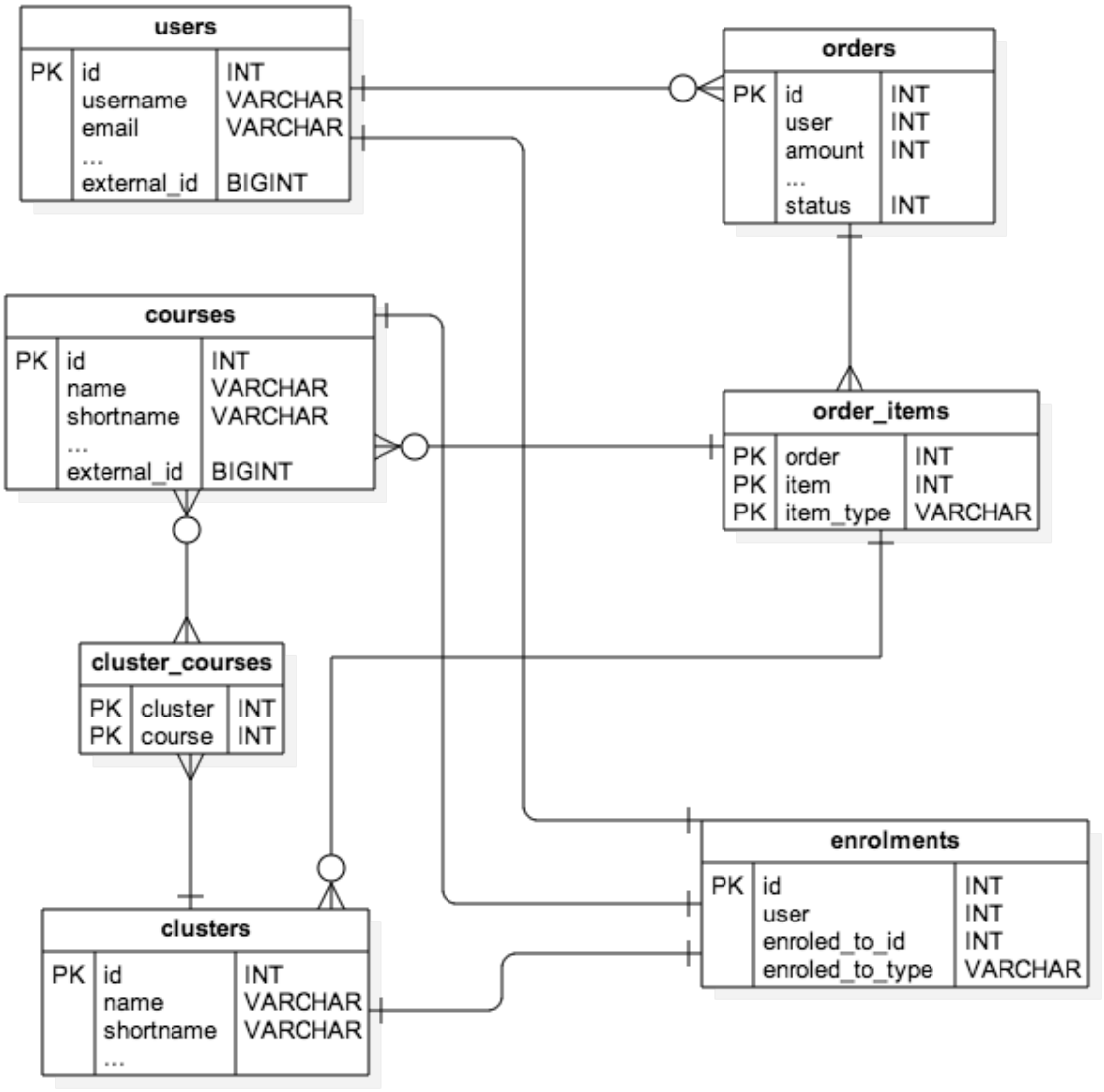

Figura 4.2: Estructura de la Base de Dades

4.3 Arquitectura

L'objectiu principal de l'aplicació és que tingui flexibilitat per adaptar-se a les diferents infraestructures de centres educatius ja existents, de manera que es pugui incorporar sense haver de canviar necessàriament la plataforma LMS que s'utilitzi. Això s'aconsegueix mitjançant l'ús d'interfícies aprofitant una de les propietats que proporcionen: el polimorfisme [6].

Per a mostrar aquest propietat i donar un exemple de la flexibilitat de la que es vol dotar a l'aplicació en el projecte es defineixen dues interfícies, la de pagament i la del LMS. La primera dóna la possibilitat de processar els pagaments amb la plataforma que sigui més útil per al centre, com per exemple Stripe o PayPal, sense que la tria d'una o altra afecti al funcionament de l'aplicació. La segona permet integrar l'aplicació amb el *Learning Management System* que s'utilitzi al centre, de manera el desenvolupament d'aquest projecte no estigui lligat a un LMS concret ni impliqui modificar tota la infraestructura ja existent.

En la figura 4.3 es detalla el diagrama de classes de l'aplicació, on es poden observar les dues interfícies comentades anteriorment així com la seva relació amb la resta de l'aplicació

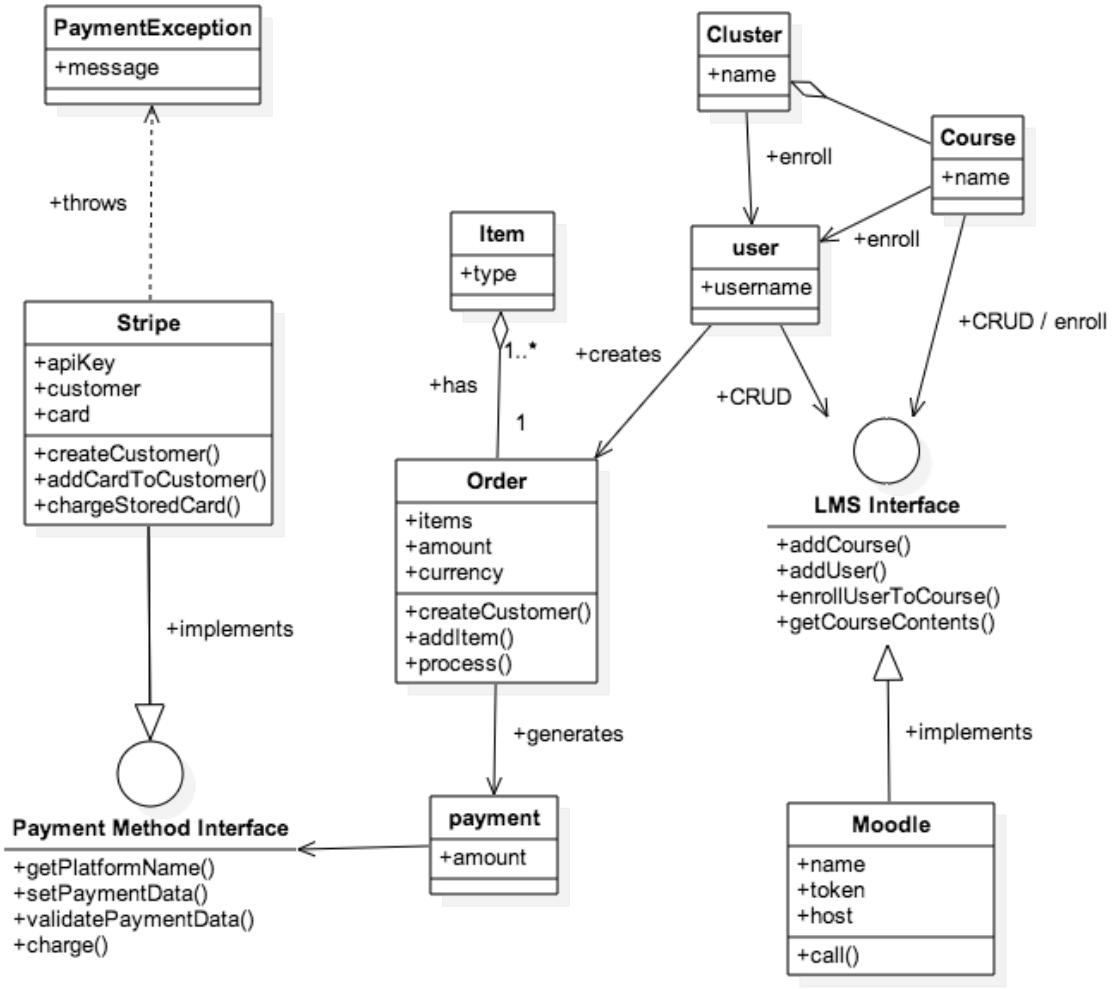

Figura 4.3: Diagrama de Classes

4.4 Interfície Web

A l'hora de dissenyar una aplicació web, s'intenta que la interfície resulti senzilla d'utilitzar als usuaris i alhora sigui capaç d'oferir una gran quantitat d'informació de la forma més gràfica possible.

L'interfície web d'aquest projecte consta d'una part pública on qualsevol usuari pot accedir a veure el llistat de cursos públics disponibles, una part per usuaris registrats on veure i gestionar la participació en els cursos en que un està matriculat i una part dedicada als administradors del centre per a crear els usuaris i cursos així com gestionar l'accés d'els usuaris a aquests i definir els diferents preus per a cadascun. A la figura 4.4 es pot veure un diagrama de casos d'ús on s'especifiquen les funcionalitats disponibles per a aquest projecte.

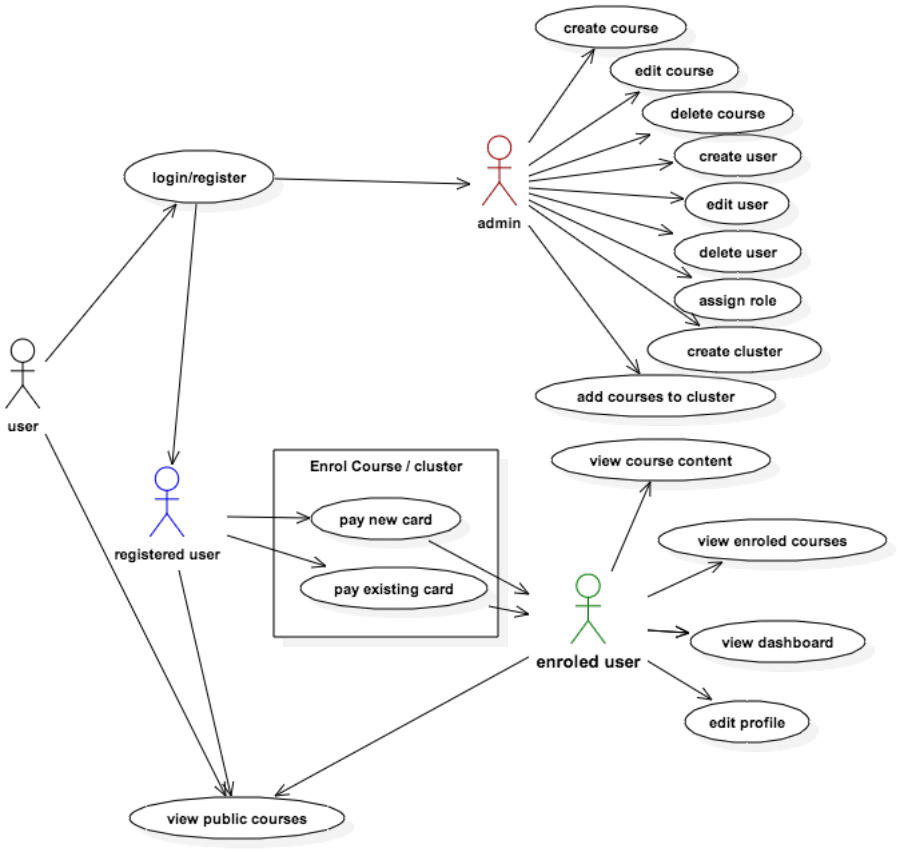

Figura 4.4: Diagrama Casos d'Ús

Per tal de poder desenvolupar una interfície que sigui el més usable possible s'ha adaptat una plantilla html Open Source anomenada Admin LTE [7], basada en *Bootstrap* i *jQuery*, que ofereix una sèrie d'eines i de components predefinits per tal

d'agilitzar el desenvolupament del *front-end* del site.

Aquesta plantilla incorporta estils per a la correcta visualització de botons, formularis, taules, llistats, gràfiques dinàmiques, etc. Així s'aconsegueix una bona experiència d'usuari mentre es perd el mínim de temps possible en el disseny de la interfície web, que no és l'objectiu principal d'aquest projecte. A la figura 4.5 s'observa l'estructura de l'interfície web del site i un exemple de contingut d'un curs extret del LMS.

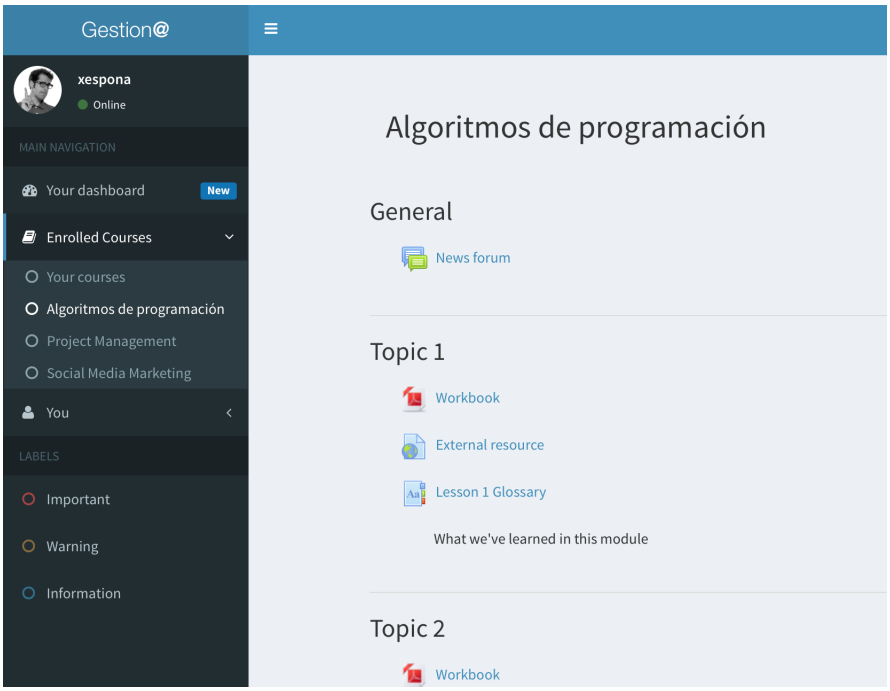

Figura 4.5: Interfície Web

Capítol 5

Implementació

En aquest capítol es detalla la implementació de les diferents parts que s'han realitzat per aquest projecte. Consta de cinc apartats, i es segueix un format similar a l'hora de descriure'ls per tal de seguir una estructura coherent. L'explicació de cada apart es divideix en cinc subapartats diferenciats:

Descripció de l'apartat.

Objectiu que es pretén aconseguir.

Implementació

Problemes que s'han trobat a l'hora de desenvolupar-lo.

Treball futur per ampliar o millorar l'apartat.

5.1 Base de l'aplicació

5.1.1 Descripció

Per al correcte desenvolupament del projecte primer s'ha de definir una estructura base de l'aplicació. Com el projecte està desenvolupat utilitant el framework Symfony2 v2.3 (d'ara en endavant senzillament Symfony) és important tenir en compte de quina forma es vol utilitzar, quines possiblitats i limitacions ofereix, i quines serán les diferents funcionalitats que se n'utilitzarán [8]. En aquest apartat es descriu de quina forma s'ha utilitzat Symfony per tal de construïr aquesta base.

5.1.2 Objectiu

El que es vol aconseguir es tenir una primera versió de l'aplicació web que, sense oferir cap funcionalitat concreta, defineixi l'estructura i el procediment a seguir a l'hora d'implementar el projecte. Així doncs es vol definir el *routing* del site, la forma d'interaccionar amb la Base de Dades, un procediment per tractar les peticions a l'aplicació (incloent la gestió dels formularis) i es vol aconseguir una plantilla HTML base a partir d'on anar afegint funcionalitat.

5.1.3 Implementació

5.1.3.1 Routing

L'estructura del routing de Symfony, quan s'utilitza el model basat en fitxers YAML, requereix que per cada ruta del sistema es defineixin: el *slug* de la ruta, el controlador i l'acció del controlador que l'executarà, els valors per defecte dels paràmetres dinàmics de la ruta i l'expressió regular que defineix el format dels paràmetres.

Com que l'aplicació consta d'un back-end i un front-end, es decideix que totes les rutes de la part del back-end incorporen el prefix */admin*. A partir d'aquí les que són relacionades amb la gestió d'usuaris es troben sota */users*, les relatives a cursos sota */courses* i per als grups de cursos (clústers) s'utilitza el prefix */clústers*. Les rutes relatives al front-end es separen en dos grups: les rutes utilitzades en la part de l'usuari (dashboard, perfil, ...) van sota */me* i les utiilitzades pels cursos cursos

(visualització, matriculació, ...) sota */course*. A la figura 5.1 es pot veure un petit exemple de les rutes definides per al projecte.

```
delete_cluster:
    path:   /admin/clusters/delete/{id}
    defaults: { _controller: CourseBundle:Course:adminDeleteCluster, id:0 }
    requirements:
        id: \d+

cluster_add_course:
    path: /admin/cluster/addCourse/{clusterId}
    defaults: { _controller: CourseBundle:Course:adminClusterAddCourse, clusterId:0 }
    requirements:
        clusterId: \d+

################### FRONT

view_course:
    path: /course/{shortname}
    defaults: { _controller:ProjectAppBundle:Course:view, shortname:'' }
    requirements:
        shortname: '[a-zA-Z0-9]+'

enroll:
    path: /course/{shortname}/enroll
    defaults: { _controller:ProjectAppBundle:Course:enroll, shortname:'' }
    requirements:
        shortname: '[a-zA-Z0-9]+'
```

Figura 5.1: Exemple de Routing

5.1.3.2 Base de Dades

Symfony incorpora un ORM (*Object Relational Mapper*) anomenat Doctrine, aquest ORM està totalment desacoplat del framework i la seva utilització és totalment opcional. En aquest projecte però, no s'utilitza la capa de més alt nivell de Doctrine sinó que es treballa directament amb la seva DBAL (*Database Abstraction Layer*). La DBAL és una capa d'abstracció que funciona damunt de la més coneguda PDO (*PHP Data Objects*) i ofereix una API flexible i intuitiva per a la comunicació amb les Bases de Dades relacionals més utilitzades.

D'aquesta manera es pot treballar amb una eina que facilita la creació i execució de consultes a la Base de Dades sense afegir una altra corba d'aprenentatge, ja que és molt senzilla d'utilitzar. A més també permet la utilització dels anomenats *Prepared Statements*, aquests aporten una millora en el rendiment ja que redueixen el overhead i també en la seguretat al escapar els paràmetres de les consultes per

prevenir la Injecció SQL. A la figura 5.2 hi ha un exemple de la senzillesa d'una query per actualitzar un curs, donat un array amb les dades de la taula de cursos.

```
$data = array_intersect_key($data, array(
    'shortname'     => 1,
    'name'          => 1,
    'description'   => 1,
    'amount'        => 1,
));
return (bool) db_update('courses', $data, array('id' => $course->getId()));
```

Figura 5.2: Consulta update amb DBAL

5.1.3.3 Gestió de peticions

En qualsevol aplicació basada en un patró MVC (*Model - Vista - Controlador*), les peticions als servidor són ateses pels controladors. Symfony no n'és una excepció. Es desenvolupa un Controlador Base que s'encarrega de realitzar les accions necessàries a tots els demés controladors, que l'extendran. Així s'evita repetir codi i també ajuda al manteniment de l'aplicació si mai es vol fer un canvi arreu d'aquesta.

Els diferents controladors de l'aplicació acaben tenint una estructura similar que es pot resumir en:

Control d'accés tant dels usuaris a les parts restringides com per a prevenir la utilització de paràmetres que, tot i ser del tipus correcte (d'això se n'encarrega el routing), no representen valors existents a la Base de Dades (per exemple un identificador de curs no existent).

Gestió dels formularis si n'hi ha. Bàsicament qualsevol acció relacionada amb un formulari, consta de validació i enregistrament de les dades.

Preparació de les dades que es necessiten per mostrar correctament les diferents parts de la pàgina que es vol renderitzar.

Renderització de la plantilla corresponent, on es passen les dades necessàries.

A la figura 5.3 es pot veure una versió simplificada del controlador que s'encarrega de gestionar les matriculacions a un curs.

```php
// Access Control
if (!$user instanceof User || !$course instanceof Course) {
    $session->getFlashBag()->add('error', 'An error ocurred');
    return $this->redirect($this->generateUrl('project_homepage'));
}

if ($cp->isEnrolled($course, $user)) {
    $session->getFlashBag()->add('error', 'You\'re already enrolled in this course!');
    return $this->redirect($this->generateUrl('user_courses', array('username' => $user->getUsername())));
}

// Handle form submission
if ($this->request->isMethod('POST')) {
    EnrollFormType::validate($form);
    if ($form->isValid()) {
        $success = EnrollFormType::submit($form);
        if ($success === TRUE) {
            $session->getFlashBag()->add('notice', 'Congratulations!');
            return $this->redirect($this->generateUrl('user_home', array('username' => $user->getUsername())));
        }
    }
    foreach ($form->getErrors() as $error) {
        $session->getFlashBag()->add('error', $error->getMessage());
    }
}

// Render template
return $this->render('ProjectAppBundle:Course:enroll-pay.html.twig', array(
    'form'      => $form->createView(),
    'course'    => $course,
    'user'      => $user,
));
```

Figura 5.3: Controlador simplificat

5.1.3.4 Gestió de formularis

Symfony permet definir una clase de tipus formulari on definir els diferents camps que el composen, el tipus i opcions de cada camp, les dades amb les que emplenar el camp en el cas de que el formulari estigui mostrant dades ja existents a la Base de Dades, l'etiqueta per al camp, o les restriccions que ha de tenir (que una xifra sigui de tipus enter per exemple). Tot i això moltes vegades s'han de definir validacions extraordinàries, com pot ser que un usuari no es pugui matricular a un curs si ja hi està matriculat o si matricular-se requereix haver superat algun altre curs prèviament.

5.1.3.5 Plantilla HTML

Per tal de separar totalment la visualització de les dades (Vista) del controlador, a Symfony s'utilitza el sistema de plantilles Twig. Apart de desacoplar la Vista de la gestió de les dades, incorpora funcionalitats molt interessants com pot ser l'herència de plantilles o la definició de blocs sobreescrivibles dintre d'aquestes.

En aquest projecte s'utilitza Twig per tal de tenir una plantilla base composta de diferents blocs comuns (essent els més importants capçalera, peu i barra lateral de l'aplicació, però també altres com el títol de la pàgina, els '*breadcrumbs*' o les notificacions d'error i d'èxit) i d'una part dinàmica on es mostra el contingut diferenciat de cada pàgina. A més es defineix un bloc genèric per a la renderització uniforme dels camps dels formularis que incorpora la renderització del camp en el seu estat original i la renderitziació del camp en el cas que hi hagi algun error amb les dades.

A la figura 5.4 es veu una part de la plantilla utilitzada per a la pàgina inicial de la part d'administració on s'aprecia com s'extén la plantilla base i es defineixen els continguts per als diferents blocs dinàmics a l'hora que també s'inclouen altres plantilles mes específiques.

```
{% extends 'UserBundle::base.html.twig' %}
{% set topText = 'Gestion<b>@</b>dmin' %}

{% block title %}The admin Page{% endblock %}

{% block contentheader %}
    <h1>
        Administration
        <small>it all starts here</small>
    </h1>
{% endblock %}

{% block content %}
    <section>
        {% include 'ProjectAppBundle:Admin:userTable.html.twig' %}
    </section>
    <section>
        {% include 'ProjectAppBundle:Admin:courseTable.html.twig' %}
    </section>
    <section>
        {% include 'ProjectAppBundle:Admin:clusterTable.html.twig' %}
    </section>
{% endblock %}
```

Figura 5.4: Plantilles i blocs a Twig

5.1.4 Problemes

5.1.4.1 Base de Dades

Symfony només dóna accés directe a la conexió a la DBAL des dels controladors. Així doncs per aconseguir tenir la conexió disponible en els Models i en les classes dels formularis es defineix una conexió i una sèrie de mètodes d'àmbit global a l'aplicació. D'aquesta forma es poden utilitzar les funcions de la Base de Dades des de qualsevol lloc. Veure figura 5.5.

```php
function db_insert($table, $data) {
    global $db;
    if (is_object($db)) {
        return $db->insert($table, $data);
    }
    return FALSE;
}
```

Figura 5.5: Exemple de funcions globals d'accés a la BD

5.1.4.2 Gestió de peticions

Si es vol que els models de l'aplicació puguin ser genèrics, extensibles i intercanviables sense ser depenents de l'arquitectura utilitzada, s'han de definir una sèrie de *providers* que s'encarreguen de tractar amb la Base de Dades i retornar els objectes dels diferents Models (figura 5.6).

```php
private function getBy($param, $value, $count = FALSE)
{
    $sql = '
        SELECT *
          FROM courses
         WHERE '. $param .' = ?
         LIMIT 1
    ';

    $stmt = db_executeQuery($sql, array($value));
    if ($count)
        $return = $stmt->rowCount();
    else
        $return = $stmt->fetchObject('Project\CourseBundle\Model\Course');

    return $return;
}
```

Figura 5.6: Course Provider

5.1.4.3 Gestió de formularis

Per tal de poder tenir una estructura unificada a l'hora de gestionar la validació (de condicions extraordinàries) i enregistrament de les dades dels formularis de l'aplicació a les clases de tipus formulari es defineixen dos mètodes estàtics (*validate* i *submit*). Tot i no ser una pràctica molt aconsellable des del punt de vista de la programació orientada a objectes l'utilització d'aquestes funcions permet obtenir una estructura lògica i uniforme al llarg de l'aplicació alhora que permeten un desenvolupament més àgil. A la figura 5.7 es pot veure un exemple de la funció de validació a l'hora de crear un usuari a l'aplicació.

```
static public function validate(&$form, $user, $up)
{
    $sentData = $form->getData();
    $errors = FALSE;

    if ($up->usernameExists($sentData['username'], $user->getId())) {
        $form->get('username')->addError(new FormError('This username already exists, try with another one'));
        $errors = TRUE;
    }

    return !$errors;
}
```

Figura 5.7: Validació per a la creació d'un nou usuari

5.1.5 Treball futur

Una de les millores que es pot realitzar al projecte en aquest apartat és la utilització dels Models com a Entitats (*Entities*) de Symfony. Aquestes permeten agilitzar la gestió de la interacció dels models amb la Base de Dades i també la creació de fomularis per a aquests. No s'ha realitzat en el projecte ja que tot i ser una característica important de Symfony també implica un temps d'aprenentatge elevat.

5.2 Mòdul Usuaris

5.2.1 Descripció

Com a qualsevol aplicació web és necessari disposar d'un mòdul per a gestionar els usuaris de la mateixa i donar una sèrie de funcionalitats tant en la part d'administració com al frontal de la web. En aquest apartat es descriu el mòdul d'usuaris que s'ha desenvolupat i les diferents funcionalitats que proporciona.

5.2.2 Objectiu

Es preten construïr un mòdul que permeti crear, editar i esborrar usuaris, donar als usuaris la possiblitat de registrar-se i loguinar-se a la plataforma així com d'editar les seves dades personals i gestionar la seva participació als cursos als que estan matriculats.

5.2.3 Implementació

L'implementació d'aquest mòdul es pot dividir en diferents apartats, diferenciant entre l'arquitectura interna del mòdul, les funcionalitats de la part d'administració i les de la part frontal de l'aplcació.

5.2.3.1 Arquitectura

Per tal de desacoplar la funcionalitat del modul de la forma en que es s'accedeix a les dades dels usuaris es crea un Model UserProvider, i un model User. L'últim defineix les propietats de la classe i conté els diferents *getters* i *setters* per tal d'accedir a aquestes. El Model UserProvider és especific per a aquesta aplicació i conté les diferents funcions per a interactuar amb les dades dels usuaris com inserció, edició, esborrat d'usuaris i també funcions per comprobar si un usuari es troba dins la Base de Dades, si la contrassenya de l'usuari és correcta o si ja hi ha un usuari amb aquest nom d'usuari a la plataforma, entre d'altres.

La idea darrera d'aquesta separació dels models és que la manera amb l'aplicació interactiua amb els usuaris no sigui depenent de si les dades dels usuaris es troben

emmagatzemades a la Base de Dades (com en aquest projecte) o de si, per exemple, s'accedeix a elles via una API externa de tipus REST.

5.2.3.2 Administració d'usuaris

La part d'administració d'usuaris des del back-end de l'aplicació proporciona als administradors de la plataforma les següents funcionalitats:

Creació d'usuaris Permet afegir usuaris a la plataforma.

Edició d'usuaris Per canviar les dades d'un usuari ja existent a la plataforma, així com els seus rols.

Esborrat d'usuaris Per donar de baixa els usuaris existents a la plataforma.

Llistat d'usuaris De la plataforma; des d'on es poden veure les dades dels usuaris que estan registrats i accedir a les funcionalitats anteriors.

5.2.3.3 Part pública

A la part pública d'usuaris és on es troben les funcionalitats relacionades amb el que un usuari pot realitzar amb la seva compta, com pot ser gestionar les seves dades i els cursos on està apuntat. Són les següents:

Login Permet a l'usuari accedir a la seva compta dins de l'aplicació. S'ha utilitzat el mòdul de login que incorpora Symfony i que es pot adaptar a qualsevol implementació de la clase User sempre que es defineixi un *Security Controller* i un *User Provider* que implementin les interfícies requerides per proporcionar les dades corresponents de l'usuari amb el format adient [9].

Dashboard És la pàgina inicial d'un usuari. Des d'aquí l'usuari te una visió general del seu estat dins la plataforma, el registre de la seva activitat recent, les noves notificacions que ha rebut o les properes activitats que l'incumbeixen. Per a dotar a aquesta pàgina d'un aspecte atractiu es volen utilitzar els components que proporciona l'ús de la plantilla HTML AdminLTE.

Cursos Permet a l'usuari veure un llistat de cursos on està matriculat, l'estat actual de la seva participació dins dels cursos així com accedir a cadascún d'aquests.

Perfil Des d'aquest apartat l'usuari pot editar les dades de la seva compta a la plataforma.

5.2.4 Problemes

A l'hora de desenvolupar el login de la plataforma amb el component de Symfony s'ha d'implementar la interfície UserProviderInterface, així com definir un Security Controller que permeti al framework saber quina és la classe que implementa la interfície. Entre d'altres funcions necessàries, s'ha d'incorporar a la classe les funcions que codifiquen el password de l'usuari segons l'algoritme escollit i generen un *password salt* per afegir seguretat al procés. Amb tot això, aquesta implementació no resulta trivial i serveix com a mostra de la dificultat a l'hora d'utilitzar les funcionalitats que proporciona un framework avançat com Symfony.

5.2.5 Treball futur

Una de les millores a afegir al mòdul d'usuaris és la de poder utilitzar un SSO (*Single Sign On*) [10], d'aquesta manera es dóna la possiblitat al centre de tenir un sistema de login centralitzat entre les diferents aplicacions web que disposi.

Tot i utilitzar el Model UserProvider per a desacoplar les funcionalitats del mòdul de la implementació concreta de la clase User, una millora a tenir en compte és definir una interfície que indiqui quins són els mètodes a implementar en el cas de voler utilitzar un UserProvider diferent, que no utilitzi la Base de Dades com a plataforma on emmagatzemar les dades dels usuaris.

5.3 Mòdul Cursos

5.3.1 Descripció

Aquest mòdul s'encarrega de donar d'altra nous cursos i grups de cursos (clústers) a l'aplicació. També és l'encarregat de gestionar la matriculació dels usuaris a aquests cursos així com de donar o denegar l'accés d'un usuari a un curs concret. En aquest apartat es descriu aquest mòdul i les diferents funcionalitats que proporciona.

5.3.2 Objectiu

L'objectiu d'aquest mòdul és afegir a l'aplicació les funcionaltiats necessàries per tal de gestionar (afegir, editar, esborrar) cursos i grups de cursos així com de permetre als usuaris matricular-se a aquests, veure'n el contingut i realitzar el seguiment de la seva participació.

5.3.3 Implementació

L'implementació d'aquest mòdul es divideix en diferents apartats, diferenciant entre l'arquitectura interna del mòdul, les funcionalitats de la part d'administració i les de la part pública de la plataforma.

5.3.3.1 Arquitectura

De la mateixa forma que amb el mòdul d'usuaris, es vol desacoplar la funcionalitat del mòdul de la forma en que es s'accedeix a les dades dels cursos i clústers. Per aquest motiu es crea un Model CourseProvider, i un model Course, que aproximadament funcionen com en el cas dels usuaris.

És a dir, un s'encarrega de definir les propietats d'un curs / clúster i gestionarles i l'altre és el que afegeix les funcions per a la interacció amb aquestes i les funcionalitats relatives a la matriculació d'usuaris en cursos, restriccions d'accés a un curs o la visualització dels seus continguts.

Creació Permet afegir cursos i grups de cursos a la plataforma.

Edició Per canviar les dades d'un curs o grup de cursos ja exsitent. Aquí també es on s'afegeixen cursos a un clúster i des d'on es definiran les restriccions d'accés a aquests.

Esborrat Per donar de baixa els cursos o grups de cursos existents a la plataforma.

Llistat De la plataforma; des d'on es poden veure les dades dels cursos / clústers existents i accedir a les funcionalitats anteriors.

5.3.3.2 Part pública

La part pública del mòdul de cursos defineix les funcions necessàries per tal de visualtizar un curs o clúster, aplicar les restriccions d'accés als cursos d'un clúster, veure les notificacions relatives al curs o matricular-s'hi. Són les següents:

Visualització Permet als usuaris no registrats veure la descripció general del curs. Un cop l'usuari està registrat al curs es mostren els continguts i les notificacions.

Matriculació Funcionalitat per tal de matricular-se a un curs o a un clúster. Permet realitzar el pagament de l'import definit, i ofereix a l'usuari de realitzar-lo amb un mètode de pagament prèviament guardat.

5.3.4 Problemes

No s'han trobat problemes específics del mòdul en sí, sinó relacionats amb com es definirà el tipus d'accés que un usuari té a un curs / clúster (temporal, indefinit), els rols de l'usuari dins d'un curs o com fer la gestió de les ordres de pagament per a un curs / clúster.

5.3.5 Treball futur

Es preten millorar la forma en que es dóna o restringeix l'accés d'un usuari a un curs, oferint la possibilitat que aquest accés sigui temporal. Un cop l'accés d'un usuari hagi caducat, informar-lo d'aquest fet i donar-li la possibilitat (si encara no ha acabat el curs satisfactòriament) d'ampliar aquest accés.

5.4 Interfícies

5.4.1 Descripció

La creació de les interfícies és probablement una de les parts més importants del projecte, ja que és la que permet dotar al software de la flexiblitat necessària per a adaptar-se a les diferents infraestructures existents als centres on es vulgui implementar l'aplicació. S'han dissenyat dues interfícies: la de Pagament (*PaymentInterface*) i la del LMS (*LMSInterface*), i s'ha realitzat una implementació de cadascuna per tal de mostrar el seu propòsit i funcionalitat.

En aquest apartat s'explica com s'utilitzen aquestes interfícies dins de l'aplicació, quina és la configuració que s'ha de fer a Symfony per tal d'utilitzar una o altra implementació de les interfícies i es veu algun mètode concret.

5.4.2 Objectiu

El que es vol aconseguir amb aquestes interfícies és dotar al projecte d'una sèrie de funcionalitats que no depenguin de, per exemple, quin sigui el LMS que s'utlitza en un centre. Així s'obté un software capaç d'abarcar un mercat mes ampli i de satisfer les necessitats concretes dels possibles clients, a més d'augmentar la flexibilitat i futura extensibilitat del mateix.

5.4.3 Implementació

5.4.3.1 Configuració

Les classes que implementen les interfícies es passen com a parámetres dels diferents providers de l'aplicació. Per exemple, la classe que implementa la interfície LMSInterface es passa com a parámetre tant a UserProvider com a CourseProvider, d'aquesta manera quan s'executen funcions que han de tenir un impacte sobre les dades d'aquests la classe es troba disponible per a realitzar la crida al mètode corresponent permet que aquest canvi sigui reflexat al LMS.

Tot i això, primer de tot s'ha de definir una forma de customitzar de forma senzilla quina és la classe que s'utilitza a l'aplicació per a cada interfície. Això

s'aconsegueix realitzant els passos següents:

1. Definint els providers que han d'utilitzar l'interfície com a serveis.
2. Injectant la classe al servei corresponent com a argument.
3. Definint aquesta injecció amb un paràmetre customitzable en comptes d'una classe concreta.

D'aquesta forma s'aconsegueix que només canviant un paràmetre, es pugui triar quina és la classe que s'injecta als providers i, per tant, quina és la implementació de la interfície que s'utilitza. A la figura 5.8 es pot veure un exemple concret on es defineix el model CourseProvider com a servei que rep un argument, i com aquest argument és un paràmetre customitzable.

```
services:
    projectcoursebundle.model.course_provider:
        class: Project\CourseBundle\Model\CourseProvider
        arguments: ["@projectlmsbundle.model.lms_model"]
```

Figura 5.8: Configuració d'un model com a servei

En el cas d'aquest projecte, s'utilitza Moodle [11] com a LMS, de manera que s'injecta aquesta classe i així es defineix en el paràmetre de la configuració (figura 5.9). Si es volgués canviar a un altre LMS s'hauria de crear una classe que també implementi la interfície LMSInterface i modificar el valor d'aquest pràmetre.

```
parameters:
    lms.model: Project\LMSBundle\Model\Moodle
```

Figura 5.9: Paràmetre customitzable

Ara es pot accedir al servei des dels controladors de Symfony i en el moment de fer la petició del servei, aquest s'instancia rebent com a paràmetre un objecte de la classe injectada, en aquest cas de la classe Moodle, que implementa la interfície LMSInterface.

S'ha aconseguit que la resta de l'aplicació pugui tenir una execució normal sense preocupar-se de quina és aquesta classe gràcies a la propietat del polimorfisme, doncs es pot dir que 'es confia' que l'objecte rebut implementarà els mètodes que es necessiten.

5.4.3.2 Utilització

Com s'acaba de comentar, ara les classes que reben aquest objecte injectat poden fer-ne ús sense saber quina classe concreta implementa, tal i com es pot veure a la figura 5.10, on la classe CourseProvider crida a la funció enrolUserToCourse de un objecte de qualsevol classe que implementi la interfície LMSInterface.

```
try {
    $response = $this->lms->enrollUserToCourse($course->getExternalId(), $user->getExternalId());
} catch (\Exception $e) {
    return $e->getMessage();
}
```

Figura 5.10: Exemple de polimorfisme

De la mateixa forma s'aconsegueix que, mitjançant la interfície PaymentInterface, la classe Order pugui processar un pagament utilitzant la plataforma que es prefereixi, sense haver de preocuparse del funcionament intern de la mateixa (figura 5.11).

```
if ($this->paymentMethod->charge($payment_id, $this->user, $this->amount, $this->currency, $savePaymentMethod=TRUE)) {
    $success = TRUE;
    $this->status = ORDER_CHARGED;
    $this->markPayment($payment_id, PAYMENT_CHARGED);
} else {
    $this->status = ORDER_FAILED;
    $this->markPayment($payment_id, PAYMENT_FAILED);
}
```

Figura 5.11: Processament d'un pagament

5.4.4 Problemes

Definir aquesta creació de serveis i injecció de paràmetres correctament dins de Symfony és una tasca costosa quan no s'hi està acostumat. A més tot i que la resolució és correcta encara existeix el problema que aquests serveis només son accessibles des dels controladors de l'aplicació i s'han de passar com a paràmetres a les altres classes on es volen fer servir com, per exemple, a la classe del formulari EnrolFormType on es creen les ordres de pagament.

Un altre problema, comú en l'ús d'interfícies es la correcta definició d'aquestes ja que és fàcil tendir a definir tots els mètodes que es necessiten per a una implementació

concreta de l'interfície, perdent part de la flexibilitat que aporten. Per això és important aconseguir dissenyar correctament l'aplicació des d'un inici.

5.4.5 Treball futur

Una millora important i que encara dotaria de més flexiblitat a l'aplicació és la creació d'interfícies per a les classes UserProvider i CourseProvider, de manera que es puguin acceptar implementacions ja existents d'aquestes classes sempre que compleixin amb els requisits de les interfícies.

5.5 Ordres

5.5.1 Descripció

Per al correcte funcionament de l'aplicació és necessari tenir un mòdul que gestioni les ordres de pagament dels usuaris, que en aquest cas estan lligades a la matriculació dels alumnes als cursos o clústers. Les ordres han de poder contenir ítems o productes (una matriculació d'un usuari a un curs) i generar un o més pagaments en funció de si són fraccionats o si un pagament no s'ha pogut processar correctament.

5.5.2 Objectiu

Obtenir un mòdul que sigui el suficientment genèric com per poder ser ampliat en el futur per acceptar nous tipus de productes (funcionalitats extres) o diferents tipus de pagaments, com per exemple subscripcions.

5.5.3 Implementació

Per aquest projecte s'ha optat per una aproximació no excessivament complexa a l'hora de generar i processar ordres de pagament, que sempre estaran relacionades amb la matriculació a un curs o un clúster. Les ordres, tenen un estat a la base de dades que indica si s'han pogut processar correctament o si han estat fallides i contenen una sèrie de ítems o productes que representen les matriculacions (per poder representar una compra que agrupi més d'una matriculació a un curs).

L'ordre defineix la moneda en que es pagarà i la suma dels preus dels productes de l'ordre serveix per calcular el preu total d'aquesta, que serà el que s'haurà de cobrar amb la plataforma de pagament corresponent. A l'hora de processar una ordre es segueix el següent procediment:

1. Guardar l'ordre a la base de dades, amb el codi d'estat pendent.
2. Guardar cadascun dels ítems de l'ordre a la base de dades.
3. Processar el pagament de l'ordre amb la plataforma de pagament corresponent.
4. Si qualsevol dels anteriors falla, s'actualitza l'ordre amb el codi d'estat fallada.

5. Si es pot completar el procés, s'actualitza l'ordre i el pagament d'aquesta amb el codi d'estat correcte.

Per aquest projecte i a partir de la interfície PaymentInterface, s'ha implementat la plataforma de pagament Stripe [12]. Aquesta plataforma, bastant nova en l'àmbit de l'estat Espanyol i encara no molt utilitzada, proporciona totes les funcionalitats necessàries per al desenvolupament d'una aplicació de qualsevol tamany a partir de la seva API de tipus REST.

Mitjançant Stripe, a part de la funcionalitat de realitzar un pagament, també s'ofereix la possiblitat a l'usuari d'emmagatzemar les tarjetes de crèdit prèviament utiltzades de forma segura, ja que no s'emmagatzema cap dada sensible al servidor de l'aplicació sinó que es la mateixa platafoma la que les gestiona.

5.5.4 Problemes

El principal problema és definir els ítems o productes de l'ordre de manera que en un futur serveixin per acceptar nous tipus de producte, com poden ser accés a la plataforma amb un model de subscripció o funcionalitats extra.

Un altre possible problema és la gestió dels impostos associats a l'ordre de pagament. D'acord amb la normativa actual aquests depenen del país de l'usuari que realitza la compra i per tant s'ha de poder adaptar-se a aquest requeriment.

5.5.5 Treball futur

El treball futur d'aquest mòdul comença a estar definit a partir dels dos problemes descrits a l'apartat anterior, ja que cap dels dos s'ha resolt en aquest projecte, degut a la complexitat dels mateixos.

Un altre millora a realitzar, tot i que està més relacionada amb la plataforma de pagament és la possibilitat de realitzar pagaments fraccionats i recurrents.

Capítol 6

Conclusions i treball futur

Tenint en compte els objectius objectius inicials del projecte, l'estat de l'art actual, les expectatives dels usuaris envers les aplicacions web i després de veure en el capítol anterior els mòduls que s'han implementat i el procés dut a terme per a fer-ho, aquest capítol pretén extreure algunes conclusions generals sobre el projecte.

A més resumeix el treball futur que es pot realitzar sobre el projecte per tal de millorar l'experiència de tots els usuaris de la plataforma i afegir les funcionalitats necessàries per tal d'obtenir una aplicació robusta i atractiva..

6.1 Conclusions

Construïr una aplicació complerta que intenti cobrir tot el pas d'un alumne per un centre educatiu suposa un repte tant a l'hora de dissenyar-la com d'implementar-la, sobretot tenint en compte els ambiciosos objectius que s'havien posat al principi i el temps limitat amb el que es compta a l'hora de realitzar un projecte final de carrera.

Es creu que ha resultat una bona idea el fet d'acabar enfocant aquest projecte a l'objectiu d'obtenir una arquitectura robusta per l'aplicació, aquesta decisió a permès a plataforma resultant ser més flexible per adaptar-se a les necessitats de diferents centres i també més extensible de cara al futur, encara que hagi suposat un cost a l'hora d'implementar tots els mòduls que es pretenien fer en un principi fent que alguns d'aquests no es poguessin implementar.

S'ha pogut comprovar els beneficis que aporta l'utilització d'un framework com

Symfony en termes de rapidesa i flexibilitat a l'hora de desenvolupar. Encara que a vegades suposi una dificultat entendre el correcte funcionament de les seves eines, si s'hagués d'implementar des de zero el temps de desenvolupament augmentaría moltíssim.

Com es pot veure en l'annex A, el rendiment de l'aplicació era acceptable, obtenint uns temps de resposta correctes per a un projecte final de carrera. Tot i això, aquests temps no es consideraven suficientment bons i es va provar de millorar-ne el rendiment afegint una optimització en forma de sistema de memòria cau a la gestió de sessions a PHP, anomenat Memcache. Amb aquesta optimització s'ha aconseguit que els temps de resposta del projecte siguin tant ràpids com els que els usuaris d'avui en dia esperen i estan acostumats a obtenir.

Així doncs s'ha aconseguit desenvolupat una aplicació utilitzant només components Open Source que cobreix unes determinades necessitats que cap altre LMS de codi obert del mercat ofereix.

6.2 Treball futur

L'objectiu d'aquest projecte és aconseguir desenvolupar una aplicació complerta que permeti agrupar totes les funcionalitats necessàries per a un centre educatiu, ja sigui presencial o online. Si bé el treball desenvolupat fins ara defineix una arquitectura bastant robusta encara hi ha moltes àrees a cobrir i altres en les que millorar, com s'ha comentat en el capítol 5. Així doncs, aquest apartat vol fer un recull de les diferents funcionalitats sobre les que treballar en un futur.

Usuaris Una millora molt interessant és definir una interfície que defineixi les funcions necessàries per a tractar amb usuaris, igualment com s'ha fet amb la interfície de pagament o la del LMS. D'aquesta forma la plataforma es podría acoplar millor a una arquitectura ja existent on es disposa de les dades dels usuaris i així evitar tant la duplicació de les dades com el manteniment d'aquestes en dos llocs diferents. Una altra possibilitat per aconseguir el mateix és desenvolupar un mòdul d'autenticació única (*Single Sign-On* que permeti registrar i autenticar els usuaris mitjançant un *Identity Provider* extern.

Agrupació d'Usuaris En grups que permetin el registre massiu així com el seguiment dels diferents usuaris dins el grup i la definició de rols dintre del grup. Aquesta funcionalitat es pot utlitzar, per exemple, per a la formació en l'àmbit professional de treballadors d'una empresa, així que també és una possiblitat de negoci.

Restriccions d'accés Als cursos d'un clúster (grup de cursos / plà d'estudis) en funció dels cursos o activitats superats prèviament.

Dashboard Creació d'una eina que permeti a un usuari accedir a l'evolució que està tenint en un curs o clúster, de forma visual. Aquesta eina ha de permetre realitzar el mateix a un professor d'un curs per als usuaris d'aquest o a un mànager d'un grup d'usuaris. Aquest dashboard aprofitaria els diferents mòduls 'visuals' que proporciona la plantillla HTML que s'ha utlitzat per al projecte.

Subcursos Permetre a un curs definir cursos depenents d'aquest on poder segmentar els usuaris del curs pare i assignar professors i horaris de forma automàtica, per exemple per a la realització de pràctiques o seminaris que requereixen un nombre d'usuaris més reduït. Aquests subcursos tindríen les mateixes funcionalitats i les evaluacions d'aquests es veurien reflexades al curs pare.

Events Afegir a l'aplicació un mòdul que consti d'una classe que estigui observant events (*EventListener*) i d'una sèrie d'events que es disparin en determinats moments en funció de l'acció realitzada per un usuari o de l'estat actual de la plataforma. Aquest mòdul afegiria encara més flexibilitat i extensibilitat a l'aplicaciò.

Notificacions Implementar un mòdul que notifiqui als usuaris dels events i dates límit pròximes així com d'objectius assolits i possibles avisos. Conectar l'aplicació amb un servei d'emails transaccionals com Mandrill o SendGrid facilitaria la comunicació d'aquestes notificacions als usuaris sense resultar molt costós d'implementar.

Pagament Ampliar la funcionalitat del mòdul d'Ordres de manera que es pugui oferir als usuaris moltes més opcions a l'hora de realitzar un pagament, com per

exemple pagaments fraccionats, subscripcions mensuals que donguin accés a un grup de cursos o pagaments en nom d'un o més usuaris d'un grup d'usuaris. Una altra millora relacionada amb els pagaments és la d'oferir la possibilitat de pagar un import en més d'una moneda (*currency*), això es pot realitzar amb dues aproximacions diferents: definir els diferents imports en la creació d'un curs / grup de cursos o disposar d'una API externa que s'encarregui de calcular el tipus de canvi de cada moneda respecte l'original.

Certificació Generació de certificats un cop l'alumne ha acabat un curs o un grup de cursos als que prèviament s'ha assignat un certificat.

Recompenses Afegir un sistema de recompenses *virtuals* que otorgui medalles als alumnes en funció de diferents criteris: si han obtingut qualificacions extraordinàries, si han completat més d'un tant per cent de la feina del curs, un cop han superat un nombre determinat de cursos, a mesura que compleixen antiguitat a la plataforma, etc. Aquest tipus de pràctiques (anomenades *Gamification*) estimulen als usuaris a millorar la seva participació i a ser més actius a l'aplicació.

Afiliació i *Whitelabeling* Oferir la possiblitat d'afegir afiliats a la plataforma. Aquests afiliats tindrien la possiblitat de revendre els cursos o grups de cursos a usuaris mitjançant, per exemple, la creació de tokens únics que donguin accés a una sèrie de cursos. Els Afiliats podrien generar tants tokens com volguessin i els usuaris que els utilitzen no tindríen constància d'estar utilitzant una plataforma que no és de l'afiliat ja que tots els textos i logos seríen canviats pels d'aquest. Es cobraría a l'afiliat en funció dels tokens generats o bé dels tokens que s'utilitzin.

CRM Definir una interfície que defineixi les funcions necessàries per tal de conectar l'aplicació a un CRM *Customer Relationship Management* i així poder oferir al departament de vendes una informació molt més complerta: pàgines que els usuaris visiten, evolució dins d'un curs, avisar quan l'usuari ha acabat un curs (satisfactòriament o no), etc.

Bibliografia

[1] Jeff Knupp. What is a Web Framework?. http://www.jeffknupp.com/blog/2014/03/03/what-is-a-web-framework 3 de Març de 2014.

[2] Kay Chan. Choose the right PHP framework. http://www.creativebloq.com/design/choose-right-php-framework-12122774 27 de Desembre de 2012.

[3] Wikipedia. Web content management systems. https://en.wikipedia.org/wiki/Web 31 d'Agost de 2015.

[4] Amazon Web Services. Billing Calculator. http://calculator.s3.amazonaws.com/index.html Última visita: 12 de Setembre de 2015.

[5] Built With. Programming Language Usage. http://trends.builtwith.com/framework/programming-language 19 de Març de 2014.

[6] Wikipedia. Polymorphism (Computer Science). https://en.wikipedia.org/wiki/Polymorphism_(computer_science 28 d'Agost de 2015.

[7] Almsaeed Studio. Admin LTE Control Panel Template. https://almsaeedstudio.com Última visita: 12 de Setembre de 2015.

[8] Symfony. Symfony Book (2.3). http://symfony.com/doc/2.3/book/index.html Última visita: 12 de Setembre de 2015.

[9] The Symfony Cookbook. How to Build a Traditional Login Form. http://symfony.com/doc/2.3/cookbook/security/form_login_setup.html Última visita: 12 de Setembre de 2015.

[10] Stephen Lawton. Secure Authentication With Single Sign-On Solutions. `http://www.tomsitpro.com/articles/single-sign-on-solutions,2-853.html` 6 de Gener de 2015.

[11] Moodle. `https://moodle.org` Última visita: 12 de Setembre de 2015.

[12] Stripe. Feature-packed payments. `https://stripe.com/us/features` Última visita: 12 de Setembre de 2015.

[13] Badgeville. Gamification. `https://badgeville.com/wiki/Gamification` Última visita: 12 de Setembre de 2015.

[14] Margaret Rose. Customer Relationship Management definition. `http://searchcrm.techtarget.com/definition/CRM` Novembre de 2014.

[15] DeveloperSide.NET Load Testing Apache with AB. Customer Relationship Management definition. `https://www.devside.net/wamp-server/load-testing-apache-with-ab-apache-bench` 19 d'Agost de 2014.

[16] Joe Dog Software. Siege. `https://www.joedog.org/siege-home/` Última visita: 12 de Setembre de 2015.

[17] Memcached. `http://memcached.org/` Última visita: 12 de Setembre de 2015.

Annex A

Rendiment

En aquest apèndix es vol veure quin és el rendiment de l'aplicació realitzant-li alguns tests de càrrega amb les eines Apache AB [15] i Siege [16]. Per fer els testos s'utilitzen dues pàgines en concret, una on es mostra el llistat de cursos als que un usuari te accés (*/me/courses/*) i l'altra és la pàgina d'un curs concret on l'usuari està matriculat */course/ap2016* i on es visualitza el contingut d'aquest curs, carregant-lo mitjançant la API de Moodle. S'han triat aquestes pàgines perquè son una bona representació del que serien les pàgines més visitades en l'aplicació real.

Per a realitzar aquests testos s'ha utilitzat el portàtil Apple Mac Book Pro amb OS X 10.10.4 amb un processador en comptes del servidor d'Amazon AWS. Els resultats a la màquina real serien millors, doncs el processador es menys potent, però donen una bona idea del rendiment de l'aplicació. Els testos s'han realitzat utilitzant casos que intentin simular la realitat del mercat.

A.0.1 Primera versió

El test següent està realitzat amb Apache AB i mostra el rendiment de l'aplicació al realitzar 600 peticions fetes per 30 usuaris concurrents. És a dir, cada un dels 30 usuaris fa 20 carregues seqüencials o, el que és el mateix, 30 peticions per segon durant 20 segons. Els resultats són els següents.

Llistat de cursos de l'usuari

Es veu com l'aplicació respon molt bé en aquest cas, amb un número de peticions

```
Server Software:        Apache/2.4.16
Server Hostname:        pfc.localhost.com
Server Port:            80

Document Path:          /me/courses
Document Length:        9227 bytes

Concurrency Level:      30
Time taken for tests:   19.512 seconds
Complete requests:      600
Failed requests:        0
Keep-Alive requests:    0
Total transferred:      5698200 bytes
HTML transferred:       5536200 bytes
Requests per second:    30.75 [#/sec] (mean)
Time per request:       975.597 [ms] (mean)
Time per request:       32.520 [ms] (mean, across all concurrent requests)
Transfer rate:          285.19 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       2
Processing:   274  955 145.8    955    1238
Waiting:      268  949 145.6    949    1232
Total:        275  955 145.7    956    1238

Percentage of the requests served within a certain time (ms)
  50%    956
  66%   1010
  75%   1056
  80%   1071
  90%   1156
  95%   1209
  98%   1218
  99%   1225
 100%   1238 (longest request)
owner@Bullet:~/Downloads/siege-3.1.1$ ab -n600 -c 30 -k -C PHPSESSID=ib2gh1hek66k8jbarat85b0rt4 http://pfc.localhost.com/me/courses
```

Figura A.1: Pàgina de cursos. Apache AB.

per segon molt elevat. La mitjana del temps de resposta són 955ms. S'ha de tenir en compte que aquesta és una pàgina amb poc contingut i consultes a la base dades però igualment el resultat és correcte.

Vista d'un curs

En canvi en aquesta pàgina els temps de resposta són molt més elevats, amb la mitjana al voltant dels 7s, un temps totalment inacceptable en una aplicació del món real. La baixada en el rendiment és deguda a que en aquesta petició s'està realitzant també una crida a la API de Webservices de Moodle per tal d'aconseguir els continguts del curs.

```
Server Software:        Apache/2.4.16
Server Hostname:        pfc.localhost.com
Server Port:            80

Document Path:          /course/ap2016
Document Length:        12753 bytes

Concurrency Level:      30
Time taken for tests:   142.043 seconds
Complete requests:      600
Failed requests:        0
Keep-Alive requests:    0
Total transferred:      7813800 bytes
HTML transferred:       7651800 bytes
Requests per second:    4.22 [#/sec] (mean)
Time per request:       7102.159 [ms] (mean)
Time per request:       236.739 [ms] (mean, across all concurrent requests)
Transfer rate:          53.72 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.3      0       5
Processing:  1378 6934 668.0   7048    8282
Waiting:     1371 6927 668.0   7041    8276
Total:       1379 6934 667.9   7048    8287

Percentage of the requests served within a certain time (ms)
  50%   7048
  66%   7070
  75%   7082
  80%   7092
  90%   7111
  95%   7121
  98%   7141
  99%   7144
 100%   8287 (longest request)
owner@Bullet:~/Downloads/siege-3.1.1$ ab -n600 -c 30 -k -C PHPSESSID=ib2gh1hek66k8jbarat85b0rt4 http://pfc.localhost.com/course/ap2016
```

Figura A.2: Vista d'un curs. Apache AB.

Utilitzant Siege per a introduïr un retard aleatori entre peticions

El test anterior tampoc reflexa l'estat real de l'aplicació doncs normalment les peticions que els usuaris realitzen estaràn més separades en el temps. Mitjançant l'ús de Siege es pot introduïr una demora entre les peticions de cada usuari simulat. En aquest test es modifica l'anterior fent que hi hagi un retard aleatori d'entre 1 i 10 segons per cada petició d'un usuari simulat.

```
Transactions:                 600 hits
Availability:              100.00 %
Elapsed time:              198.56 secs
Data transferred:            7.30 MB
Response time:               3.60 secs
Transaction rate:            3.02 trans/sec
Throughput:                  0.04 MB/sec
Concurrency:                10.87
Successful transactions:      600
Failed transactions:            0
Longest transaction:         6.76
Shortest transaction:        0.29

FILE: /Users/owner/siegelog/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
owner@Bullet:~/Downloads/siege-3.1.1$ siege --concurrent=30 -d10 --reps=20 --header="Cookie: PHPSESSID=0i0m8rjgqupkam02j1abfmikj7" http://pfc.localhost.com/course/ap2016
```

Figura A.3: Vista d'un curs amb retràs entre peticions. Siege.

En aquest test el temps de resposta s'ha reduït a la meitat al donar una mica més de temps entre peticions. 3.6 segons és un temps elevat però que podria ser acceptable en una aplicació molt complexa, tot i que aquest no és el cas de l'estat

actual de l'aplicació del projecte.

A.0.2 Segona versió

Per realitzar aquesta segona versió, s'introdueix l'ús de memcached [17] per a emmagatzemar les dades de les sessions del usuaris en memòria en comptes de al disc (l'opció per defecte que normalment realitza PHP). Es pren aquesta decisió després de comprovar que la gestió de les dades de la sessió ocupa més del 50% del temps de la petició. Per tal de fer-ho s'ha d'afegir l'extensió i configurar PHP, Symfony i Moodle adequadament.

Llistat de cursos de l'usuari

```
Server Software:        Apache/2.4.16
Server Hostname:        pfc.localhost.com
Server Port:            80

Document Path:          /me/courses
Document Length:        9227 bytes

Concurrency Level:      30
Time taken for tests:   8.583 seconds
Complete requests:      600
Failed requests:        0
Keep-Alive requests:    0
Total transferred:      5698200 bytes
HTML transferred:       5536200 bytes
Requests per second:    69.91 [#/sec] (mean)
Time per request:       429.140 [ms] (mean)
Time per request:       14.305 [ms] (mean, across all concurrent requests)
Transfer rate:          648.35 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:   145  425  75.6    425     764
Waiting:      133  414  74.4    414     735
Total:        146  425  75.6    425     764

Percentage of the requests served within a certain time (ms)
  50%    425
  66%    444
  75%    459
  80%    468
  90%    504
  95%    571
  98%    620
  99%    634
 100%    764 (longest request)
owner@Bullet:~/Downloads/siege-3.1.1$ ab -n600 -c 30 -k -C PHPSESSID=gftlbgu4som84k7obj0g95keg3 http://pfc.localhost.com/me/courses
```

Figura A.4: Pàgina de cursos. Apache AB. Memcached.

Aquí es pot veure clarament com utilitzant Memcache es redueix el temps de resposta pràcticament a la meitat i això tenint en compte que en aquesta primera versió del test ja s'havia aconseguit un temps de resposta molt bo, inferior a un segon. Els 425ms actuals són tot un èxit i es consideren més que adequats per la complexitat de la pàgina que s'està provant.

Vista d'un curs

```
Server Software:        Apache/2.4.16
Server Hostname:        pfc.localhost.com
Server Port:            80

Document Path:          /course/ap2016
Document Length:        12753 bytes

Concurrency Level:      30
Time taken for tests:   38.796 seconds
Complete requests:      600
Failed requests:        0
Keep-Alive requests:    0
Total transferred:      7813800 bytes
HTML transferred:       7651800 bytes
Requests per second:    15.47 [#/sec] (mean)
Time per request:       1939.795 [ms] (mean)
Time per request:       64.660 [ms] (mean, across all concurrent requests)
Transfer rate:          196.69 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    0   0.1      0       1
Processing:  1494 1925 128.4   1933    2307
Waiting:     1488 1916 128.2   1924    2299
Total:       1494 1925 128.4   1933    2308

Percentage of the requests served within a certain time (ms)
  50%   1933
  66%   1988
  75%   2022
  80%   2032
  90%   2081
  95%   2117
  98%   2170
  99%   2195
 100%   2308 (longest request)
owner@Bullet:~/Downloads/siege-3.1.1$ ab -n600 -c 30 -k -C PHPSESSID=gftlbgu4som84k7obj0g95keg3 http://pfc.localhost.com/course/ap2016
```

Figura A.5: Vista d'un curs. Apache AB. Memcached

El resultat aconseguit amb el segon test és de 1.9 segons de temps de resposta és prou bo, molt més si es té en compte que s'ha reduït en 5 segons respecte a la versió que no utilitzava l'optimització de Memcache i que també s'està executant Moodle per obtenir els continguts del curs.

Utilitzant Siege per a introduïr un retard aleatori entre peticions

```
Transactions:                 2000 hits
Availability:               100.00 %
Elapsed time:               157.94 secs
Data transferred:            24.32 MB
Response time:                1.89 secs
Transaction rate:            12.66 trans/sec
Throughput:                   0.15 MB/sec
Concurrency:                 23.92
Successful transactions:      2000
Failed transactions:             0
Longest transaction:          3.50
Shortest transaction:         0.28

FILE: /Users/owner/siegelog/siege.log
You can disable this annoying message by editing
the .siegerc file in your home directory; change
the directive 'show-logfile' to false.
owner@Bullet:~/Downloads/siege-3.1.1$ siege --concurrent=100 -d10 --reps=20 --header="Cookie: PHPSESSID=gftlbgu4som84k7obj0g95keg3" http://pfc.localhost.com/course/ap2016
```

Figura A.6: Vista d'un curs amb retràs entre peticions. Siege. Memcached

Finalment en aquest test s'obté un temps de resposta de l'aplicació de 0.72s per a la visualització del curs i els seus continguts, un temps que es pot considerar tot

un èxit en la millora del rendiment de la pàgina i que realment suposa tot un èxit per a l'aplicació.

Forçant la màquina

S'ha volgut provar fins a quin punt el sistema dóna un rendiment acceptable, i tal i com es veu en la figura següent, l'aplicació aguanta fins a 120 peticions per segon durant 20 segons per a la pàgina on es visualitza el contingut d'un curs. S'ha de dir que a partir d'aquí el temps ja puja de manera excessiva i a partir de 150 peticions el servidor comença a funcionar de manera anormal, tallant alguns dels threads. Obtenir aquests resultats és molt important ja que denota que l'aplicació es pot ampliar tranquilament ja que es té bastant marge si es pensa en el temps de respota.

Figura A.7: Forçant la màquina. Memcached

Índex de figures

Signat:

Xavier Espona Roig

RESUM

L'evolució de la formació acadèmica fa dificil trobar un centre educatiu que no disposi d'un (LMS) que serveixi per distribuir els materials i realitzar el seguiment a l'estudiant. Tot i això costa trobar una eina Open Source que cobreixi les diferents necessitats d'un centre educatiu de forma unificada: promoció, captació d'estudiants, venta dels cursos, etc. sense deixar de banda les funcionalitats d'una plataforma d'aprenentatge. Aquest projecte vol donar una breu visió de l'estat de l'art a la vegada que presenta una solució a aquests problemes utilitzant un software amb una arquitectura suficientment flexible com per adaptar-se a infraestructures ja existents.

RESUMEN

La evolución de la formación académica hace difícil encontrar un centro educativo que no disponga de un (LMS) que sirva para distribuir los materiales y realizar el seguimiento al estudiante. Sin embargo cuesta encontrar una herramienta Open Source que cubra las necesidades del centro educativo de forma unificada: promoción, captación de estudiantes, venta de cursos, etc. sin olvidar las funcionalidades de una plataforma de aprendizaje. Este proyecto da una visión del estado del arte a la vez que presenta una solución a estos problemas usando un software con una arquitectura lo suficientemente flexible como para adaptarse a infraestructuras ya existentes.

ABSTRACT

The evolution of the academic training makes it difficult to find an institution that does not have an (LMS) that serves to distribute the materials and track the student progress. Nonetheless it is hard to find an Open Source platform that covers and unifies the needs of an educative institution: promotion, student recruitment, course sales, etc. while providing all the usual features of an educational platform. This project aims to give a view of the current state-of-the-art and to provide a solution to this problems using a software with a flexible architecture that can adapt to existing infrastructures.