Universitat de Vic
Escola Politècnica Superior

Treball Final de Carrera

Plataforma “Cloud Computing" per la comunicació entre pares i professors (Saas)

Dani Pastor Bartes

Enginyeria Tècnica d’Informàtica de Gestió
Directora: Dolors Anton
Vic, Setembre de 2014

Index

Resum de Treball Final de Carrera
Enginyeria Tècnica d'Informàtica de Gestió

Títol: Plataforma "Cloud Computing" per la comunicació entre pares i professors (Saas)
Paraules clau: Saas, cloud computing, multitenancy, educació, pares, alumnes, escola.
Autor: Dani Pastor Bartes
Direcció: Dolors Anton
Avalador:
Data: Setembre de 2014

Resum

El projecte que es presenta a continuació és el resultat fruit de la detecció d'un problema de comunicació entre les escoles (sobretot de primària) i el pares dels alumnes que hi cursen els estudis, i la necessitat de trobar-hi una solució. Amb aquesta premissa, i tenint en compte que la tecnologia cada vegada ens ofereix més i millors eines per gestionar les nostres necessitats, es porta a terme la construcció d'un servei al núvol *(Saas)* capaç de cobrir aquestes necessitats i de fer-ho de la manera més senzilla i eficient possible. La plataforma **Aula** és un servei on tant professors com pares dels alumnes poden comunicar-se i consultar la informació referent als seus alumnes (en el cas del professors), i dels seus fills (en el cas dels pares/tutors). La solució adoptada ha de ser capaç de funcionar sobre qualsevol aparell (ordinador, tablet o mòbil) i en qualsevol lloc on hi hagi connexió a Internet. No s'ha de realitzar cap tipus d'instal·lació un cop el sistema estigui funcionant, i s'hi podran registrar tants centres com siguin necessaris, així com professionals i pares dels alumnes. També s'ha valorat el manteniment mínim del sistema i la seva escalabilitat, per poder fer front a diferents volums de dades. Així doncs, el projecte **Aula** es presenta com una solució per gestionar el dia a dia dels professors i alumnes, però que no pretén ser substitut de cap altre sistema de gestió que pugui tenir el centre.

Final Project Abstract
Technical Degree in Computer Management

Title: Cloud Computing platform for communication between parents and teachers (SaaS)
Keywords: Saas, cloud computing, multitenancy, education, parents, students, school.
Author: Dani Pastor Bartes
Management: Dolors Anton
Guarantor:
Date: September, 2014

Abstract

The project presented below is the result of a detection of the existing communication problem between schools (mainly primary schools) and parents and the need to find a solution. With this in mind, and considering that technology increasingly offers more and better tools to handle our needs, this project consists on building a cloud service *(SaaS)* which can satisfy these needs with the easiest and the most efficient possible way. Platform **Aula** is a service where teachers and parents can communicate and consult information related to students (for teachers), and their children (for parents/tutors). The solution adopted should be able to run on any device (computer, tablet or mobile phone) with an Internet connection. There is no need to install any program once the system is running, providing unlimited amount of centers, professionals and parents. A minimum system maintenance and scalability has also been considered to cope with different amount of data. Therefore, the **Aula** project is presented as a solution for managing the day to day tasks of teachers and students, but it is not intended as a substitute for any other management system already existing in the centre.

1. Introducció

En un món cada vegada més connectat, les tasques que necessitem al dia a dia s'han de gestionar de forma fàcil i eficaç. La comunicació entre les persones ha esdevingut des de sempre una eina completament necessària per dur a terme els nostres objectius. En els últims anys la tecnologia ha canviat radicalment la forma de comunicar-nos, i ho ha fet a una velocitat mai vista, adoptant en molt poc temps noves formes de passar-nos informació. L'ensenyament ha estat des de sempre un valor per part de la nostra societat, i la comunicació amb els seus docents ens ha permès dur a terme un seguiment mes acurat dels nostres fills. Aquest seguiment, però, a vegades és erràtic i confús, i molts cops no podem accedir a la informació que voldríem en el moment oportú ni en el lloc on voldríem. D'aquestes necessitats neix la plataforma *Aula*, un servei web on tant pares com professors poden mantenir un canal de comunicació privat, i així poder fer un seguiment més personalitzat de l'alumne. La plataforma està dissenyada per donar cobertura a tots els centres i pares que vulguin utilitzar-la, i està pensada perquè les tasques de manteniment siguin les mínimes necessàries, així com també de disposar d'un gran factor d'escalabilitat.

La plataforma *Aula* és una aplicació al núvol, on després de registrar-se, cada usuari tindrà a la seva disposició una eina per poder realitzar aquestes tasques de manera ràpida i senzilla, sense cap tipus d'instal·lació i disponible en qualsevol lloc i en qualsevol aparell que tingui connexió a internet. La finalitat principal del projecte és dotar a les dues parts (escola i pares) d'un sistema per compartir informació i poder fer un seguiment més periòdic de la feina dels alumnes a les escoles.

2. Objectius

L'objectiu principal del treball és muntar una plataforma al núvol que sigui multiusuari, que estigui preparada per suportar mult-idioma i que permeti als professionals del centre i als pares dels alumnes enviar-se comunicats per poder informar-se mútuament de les incidències del dia a dia. A més a més, tota aquesta informació ha de quedar registrada per poder-la consultar i ha de ser accessible des de qualsevol lloc on hi hagi connexió a internet així com també des de qualsevol aparell (ordinador, tauleta, mòbil…).

Una altre finalitat que s'ha perseguit en aquest projecte és la de no dependre de cap instal·lació per part dels usuaris i que l'aplicació es pugui utilitzar un cop fet el registre en el cas dels centres, i de prèvia autorització (assignació d'usuari) d'aquests en el cas del pares i professionals subscrits al centre, sense intervenció de cap tipus de servei tècnic.

3. Exposició del treball

3.1. Especificacions dels requeriments

La idea de la creació d'una plataforma *Saas* ("*Software com a servei")* per la comunicació entre pares i professionals d'un centre escolar, surt de la necessitat de tenir la informació del dia a dia dels alumnes disponible en qualsevol lloc i en qualsevol moment. El canal de comunicació diari entre els professionals (professors) i pares es a través dels seus fills, però molts cops la informació que es rep no és l'adequada o no és la que el professor volia fer arribar, o a vegades, la informació que rep el professor no és la que el pare de l'alumne volia transmetre. També cal dir, que en edats mes petites és necessari fer un seguiment dels deures que ha de fer l'alumne, com per exemple saber quin dia s'han d'entregar, per d'aquesta manera poder gestionar i planificar millor el temps fora de l'àmbit escolar.

Es detecta que la solució adoptada per la majoria de centres per poder realitzar aquestes tasques són la comunicació directe amb el pares (en edats més petites) i la utilització d'una "agenda" física, on els alumnes apunten totes les feines o comunicats que els professors o el centre volen fer arribar als pares. Els inconvenients són tots els derivats de tenir la informació en un suport físic. L'agenda a vegades s'oblida dins el centre quan s'acaben les hores escolars, es pot perdre, quan vols fer una consulta no la tens a mà (ex: estàs fora de casa, …) o la informació que ha escrit l'alumne no es la correcte. En el cas de la comunicació directe entre pares i professionals, aquests últims es troben repetint en molts casos el mateix discurs a cada pare, formant embussos alhora de sortir de l'escola i moltes vegades mals entesos entre les dues parts. En els comunicats mes generals, els que fa el centre, s'utilitza l'alumne per fer arribar una nota als pares, sense poder saber si la nota ha arribat finalment al seu destinatari.

La proposta que s'ofereix per sol-ventar aquests problemes és la de disposar d'un servei al núvol que permeti una comunicació ràpida, privada i segura entre els professionals del centre i els pares dels alumnes. En cap cas aquesta solució pretén reemplaçar les solucions adoptades fins ara pels centres, sinó que més aviat és una eina extra poder garantir totes aquestes tasques. D'aquesta manera desapareixen els problemes derivats del suport físic, i tant professionals com pares poden tenir la informació accessible des de qualsevol lloc i en qualsevol moment.

3.2. Estudi de la viabilitat i solució conceptual

Per poder construir aquesta solució es necessita el desenvolupament d'un software capaç de cobrir totes les necessitats detectades, que sigui multiusuari, que es pugui contractar a traves d'una pagina web i que estigui preparat per ser mult-idioma. Aquesta solució passar per un servei *SAAS,* i no ha de necessitar de cap tipus d'instal·lació per cap de les dues parts (centre i pares). El cost del software són les hores dedicades per el seu desenvolupament i el cost de les eines per desenvolupar aquest software és de 0€, doncs s'utilitzen eines *open-source* o amb llicència de proves completament funcionals. Aquestes eines són:

- PHP versió 5.5.3:
  Llenguatge utilitzat en el servidor.
- Framework Symfony versió 2.3:
  Framework basat en PHP que s'utilitza per programar l'aplicació.
- MySQL versió 5.5.33:
  Base de dades relacional utilitzada en el projecte.
- HTML, CSS i javascript:
  Llenguatges utilitzats en la part del client.
- JQuery 1.10.2:
  Llibreria de javascript utilitzada en la part del client.
- SublimeText 2 (Editor de text):
  Editor de text molt senzill i àgil, perfecte per programar amb rapidesa.
- Bootstrap:
  Llibreria CSS que s'utilitza sobretot pel seu potencial en el disseny responsive.
- OS X versió 10.9.3 (Mavericks):
  Sistema operatiu dels ordinadors Apple (MAC).
- MAMP versió 2.2 (Apache-MySql-PHP):
  Conjunt de software que permet tenir un servidor local amb PHP i la base de dades MySql en sistemes operatius OSX.
- Git (control de version):
  Es un sistema de control de versions que ja ve de serie dins el sistema operatiu OSX.
- SourceTree (client pel control de versions):
  Un client per poder gestionar de manera gràfica i de forma mes senzilla l'entorn Git.
- Bitbucket (servei allotjament pel control de versions):
  Aquí guardarem el nostre projecte per tenir una copia al núvol, a més a més de poder-lo compartir si fos necessari amb altres desenvolupadors.

Per l'allotjament del servei en producció s'utilitzarà un compte de *Amazon web service (AWS)*, que té un cost de 0€ el primer any, per poder realitzar les proves pertinents. El cost a partir d'aquest any depèn del tràfic de dades així com del tipus de màquina virtual aixecada. La decisió d'agafar aquest servei i no un altre, és la possibilitat de tenir un any gratuït i poder escalar fàcilment les necessitats del sistema. Tot i que és un producte mes car que un servidor tradicional, et permet abstraure't de molta feina del servidor i de

poder comptar en tot moment amb les eines necessàries per dotar de més o menys capacitat al sistema, amb els avantatges de tenir tot una serie de tasques molt ben resoltes (copies de seguretat, instal·lacions, capacitat, …). A més a més, pel tipus d'aplicació és necessari tenir un servei de 24x7, 365 dies l'any, i *Amazon* amb la seva estructura és l'opció que mes garanties ofereix, tenint en compte que també es necessari utilitzar un sistema de virtualització per poder aixecar més màquines, ram, disc… si es donés el cas, i *Amazon* té ara per ara el millor producte en aquest aspecte.

Altres opcions d'allotjament serien per exemple:

Dinahosting: Hosting professional des de 9,9€ al mes.
Arsys: Plan Empresarial des de 8€ al mes.

També s'utilitza un sistema de control de versions per poder desenvolupar tot el sistema amb seguretat i més eficàcia, i d'aquesta manera tenir la porta oberta de cara a que nous desenvolupadors puguin treballar conjuntament amb el projecte. El sistema de versions utilitzat és el *Git*, que ja ve instal·lat de serie en el sistema operatiu OSX. El servei d'allotjament al núvol de control de versions és *Bitbucket*, que és gratuït fins per a 5 usuaris a data d'avui, marge suficient per les necessitats del projecte.

4. Anàlisi del sistema

Analitzem el sistema per determinar quina és la informació que s'ha de registrar i quines seran les seves funcionalitats. Per fer-ho, presentem un diagrama de classes i la descripció d'esdeveniments. Aquí voldria aclarir el perquè s'ha utilitzat l'anglès com a idioma per treballar dins el codi del projecte.

Tots els noms donats a les classes, taules, funcions, variables… estan en anglès. Exposo aquí els avantatges que crec que això comporta:

- Noms sense ambigüitats: El no tenir accents, ni caràcters especials, t'oblides dels possibles problemes que aixo pot comportar a nivell de codi, i els noms són clars i sense confusions.

- internacionalització: Agradi o no, l'anglès és l'idioma per defecte en el món del desenvolupament web. El fet d'utilitzar un altre idioma et privarà que possibles col·laboradors puguin entendre més ràpidament el teu codi. Si el projecte s'obre a la comunitat *opensource*, o es distribueix en un altre país, o es ven a una altre empresa, és molt més fàcil que tot estigui en anglès, casi qualsevol desenvolupador entendrà a que es refereix la classe "user", però no tothom tindrà clar que vol dir la classe "usuari" o "usuario", per posar un exemple.

El perquè la memòria està redactada en català, a diferencia dels noms donats dins el programa, és perquè aquest és l'idioma en que millor m'expresso i entenc que traduir una memòria o un manual es relativament fàcil (fins i tot es poden utilitzar eines de traducció automàtica), sense que aixo afecti per res el funcionament de l'aplicació, però intentar canviar els noms de totes les classes i funcions d'un projecte es una tasca molt més complexa que en la majora dels casos ningú farà, i només incrementarà la possibilitat de cometre errors en el codi. Així doncs, utilitzar l'anglès com idioma per treballar dins el codi (tot i que en el meu cas he tingut que fer un petit esforç extra) crec que és la millor manera de tenir un codi el mes universal possible, i és un tema que a diferència de la memòria o manual, no es podrà canviar fàcilment més endavant.

4.1.Diagrama de classes

Treballarem amb les següents classes:

- Compte - Classe per registrar tot el referent al compte (multiusuari). És una classe que ens permet gestionar les rutes per saber en quin compte estem. No s'utilitza directament la classe *centre* perquè volia una classe per donar un significat mes general, no tant concret, podríem dir que és la persona o entitat que crea el centre. D'aquesta manera en possibles modificacions tindríem una classe que podria relacionar-se amb diferents centres.

- Centre - Classe que conté la informació que necessitem del centre. És purament informativa i serveix perquè els usuaris vinculats al centre puguin accedir a la informació del mateix.

- Notificacions - Classe per registrar les notificacions. Aquesta classe ens ha de permetre registrar les diferents notificacions del sistema.

- Membres - Classe per registrar la informació tant de professionals com d'alumnes. És una classe que serveix per referir-nos a qui enviem les notificacions i quines relacions hi ha entre estudiants i professionals.

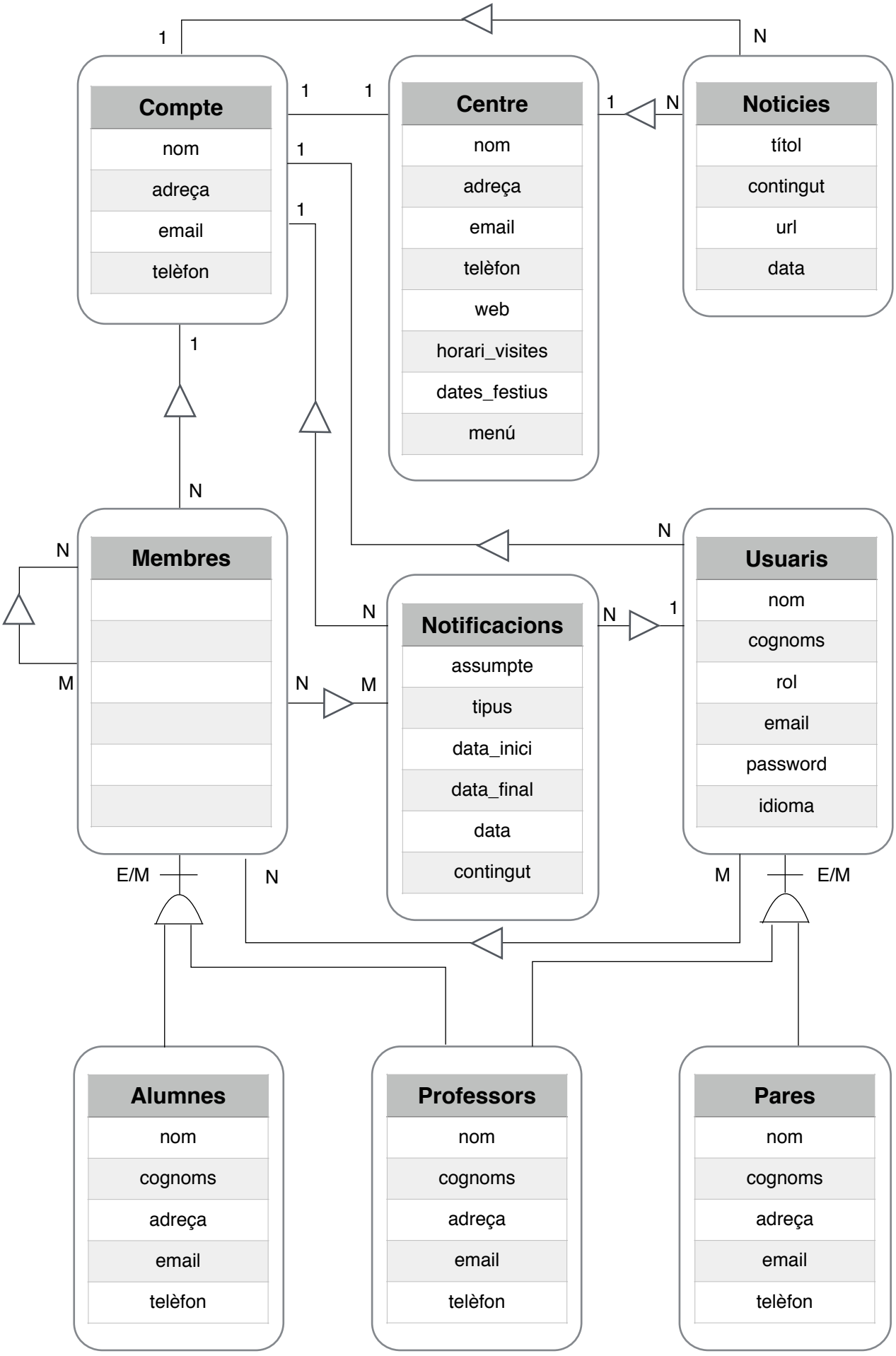
- Usuaris - Classe per gestionar tot lo referent als usuaris de la plataforma. Aquesta classe és l'encarregada de portar el tema de la validació d'usuaris i de donar accés als llocs segons el seu rol.

- Noticies - Aquí es registren les noticies del centre. Es una classe per anar enregistrant les noticies de cada centre.

- Professionals - Aquí es registren les dades dels professionals.

- Estudiants - Aquí es registren les dades dels estudiants.

Com he dit, al ser una plataforma multiusuari treballem amb la classe *compte*, que serà l'encarregada de gestionar els diferents comptes de l'aplicació. Tot seguit presento el següent diagrama de classes:


Compte
nom
adreça
email
telèfon
Centre
nom
adreça
email
telèfon
web
horari_visites
dates_festius
menú
Noticies
títol
contingut
url
data
Membres
Notificacions
assumpte
tipus
data_inici
data_final
data
contingut
Usuaris
nom
cognoms
rol
email
password
idioma
Alumnes
nom
cognoms
adreça
email
telèfon
Professors
nom
cognoms
adreça
email
telèfon
Pares
nom
cognoms
adreça
email
telèfon
1
N
E/M
M

4.2.Model funcional

Llisto aquí els principals esdeveniments de l'aplicació:

Nom: Registrar un centre.
Resposta: S'agafen les dades del formulari, es comproven i es crea el centre i l'usuari administrador.
Serveis: formulari.mostrar; dades.comprovar; centre.crear; usuari.crear;

Nom: Registrar un usuari.
Resposta: S'agafen les dades del formulari, es comproven i es crea l'usuari.
Serveis: dades.agafar; dades.comprovar; usuari.crear;

Nom: Accedir a l'aplicació.
Resposta: S'avalua l'usuari (mail i password), s'avalua el seu rol i es redirigeix a la ruta d'inici de l'aplicació que li correspon.
Serveis: usuari.validar; usuari.comprovar_rol; interficie.mostrar;

Nom: Donar d'alta un professional.
Resposta: S'agafen les dades del formulari, es comproven i es crea el professional.
Serveis: formulari.mostrar; dades.comprovar; professional.crear;

Nom: Donar d'alta un estudiant.
Resposta: S'agafen les dades del formulari, es comproven i es crea l'estudiant.
Serveis: formulari.mostrar; dades.comprovar; estudiant.crear;

Nom: Donar de baixa un professional.
Resposta: S'accedeix al professional, i es borra aquest.
Serveis: professional.mostrar; professional.borrar;

Nom: Donar de baixa un estudiant.
Resposta: S'accedeix al estudiant, i es borra aquest.
Serveis: estudiant.mostrar; estudiant.borrar;

Nom: Modificar un professional.
Resposta: Es mostra el formulari, es validen les dades i es modifica el professional.
Serveis: professional.mostrar; dades.validar; professional.modificiar;

Nom: Modificar un estudiant.
Resposta: Es mostra el formulari, es validen les dades i es modifica l'estudiant.
Serveis: estudiant.mostrar; dades.validar; estudiant.modificar;

Nom: Assignar un estudiant amb un usuari.
Resposta: S'accedeix al estudiant, s'accedeix al usuari i es crear una relació entre els dos.
Serveis: estudiant.mostrar; usuaris.mostrar; assignació.crear;

Nom: Relacionar un estudiant i un professional.
Resposta: S'accedeix al estudiant i al professional i es crear una relació entre els dos.
Serveis: estudiant.mostrar o professional.mostrar; relació.crear;

Nom: Entrar una noticia al centre.
Resposta: Es mostra el formulari, es validen les dades i es crear la noticia.
Serveis: formulari.mostrar; dades.validar; noticia.crear;

Nom: Editar una noticia del centre.
Resposta: Es mostra el formulari, es validen les dades i es modifica la noticia.
Serveis: formulari.mostrar; dades.validar; noticia.modificar;

Nom: Donar de baixa una noticia del centre.
Resposta: Es busca la noticia i es borra.
Serveis: noticia.busca; noticia.borrar;

Nom: Mostrar el centre.
Resposta: Es busca el centre i es mostra la seva informació.
Serveis: centre.buscar; centre.mostrar;

Nom: Mostrar els estudiants d'un centre.
Resposta: Es busca el centre, es busquen els seus estudiants i es mostren.
Serveis: centre.buscar; estudiants.buscar; estudiants.mostrar;

Nom: Mostrar els professionals d'un centre.
Resposta: Es busca el centre, es busquen els seus professionals i es mostren.
Serveis: centre.buscar; professionals.buscar; professionals.mostrar;

Nom: Enviar una notificació.
Resposta: Es mostra el formulari, es validen les dades i es crea una notificació.
Serveis: formulari.mostrar; dades.validar; notificació.crear;

Nom: Eliminar una notificació.
Resposta: Es busca la notificació i s'elimina.
Serveis: notificació.buscar; notificació.eliminar;

5. Disseny del sistema

Tot el resultat de l'anàlisi l'hem de lligar amb una tecnologia concreta, per fer-ho presentem el diagrama d'interfícies, el disseny de la B.D, i el disseny de l'aplicació, que lliga amb com s'estructura el codi.

5.1. Diagrama d'interfícies

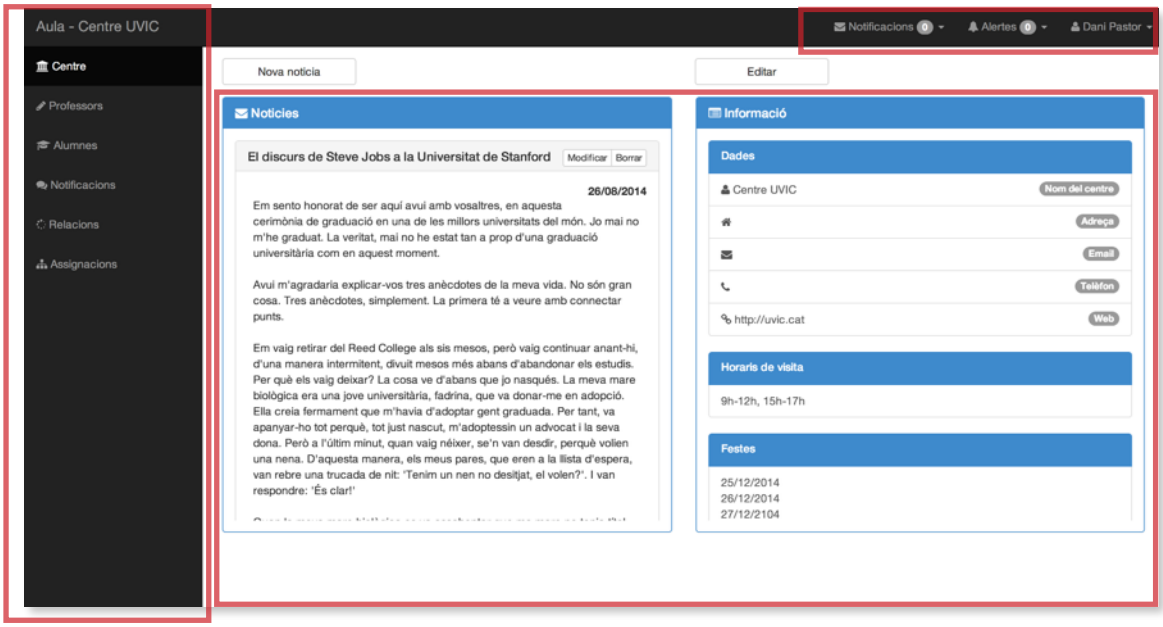
Per el desenvolupament de la part de l'interfície de la plataforma s'ha utilitzat, entre d'altres eines, la llibreria de CSS *Bootstrap*. Les raons principals per utilitzar aquesta llibreria és que ens permet tenir tota una sèrie d'elements comuns en el disseny perquè l'usuari a l'hora d'interactuà amb el sistema el senti més homogeni i còmode. L'altre aspecte important, és que ens permet construir una interfície "responsive", és a dir, que s'adapta als diferents formats de pantalla, i no només en quan a visualització, sinó també en el que respecte a l'usabilitat. Així doncs, s'aconsegueix que un sol desenvolupament d'interfície serveixi tant per l'ordinador, com per la tablet com per un telèfon mòbil.

S'ha utilitzat per norma l'herència de 3 capes per dissenyar l'interfície. Un *layout* principal o base engloba el menú superior i lateral de l'aplicació, que és comú en tots els apartats del sistema. Un segon layout és l'encarregat de definir el lloc dels submenús de cada apartat, i defineix l'estructura del contingut de l'apartat, que consta de dues columnes. El tercer layout s'utilitza per mostrar el detall de cada apartat i definir els seus elements. Tots els menús i opcions de cada secció de la plataforma estan col·locats en el mateix lloc per facilitar al màxim l'us de l'aplicació.

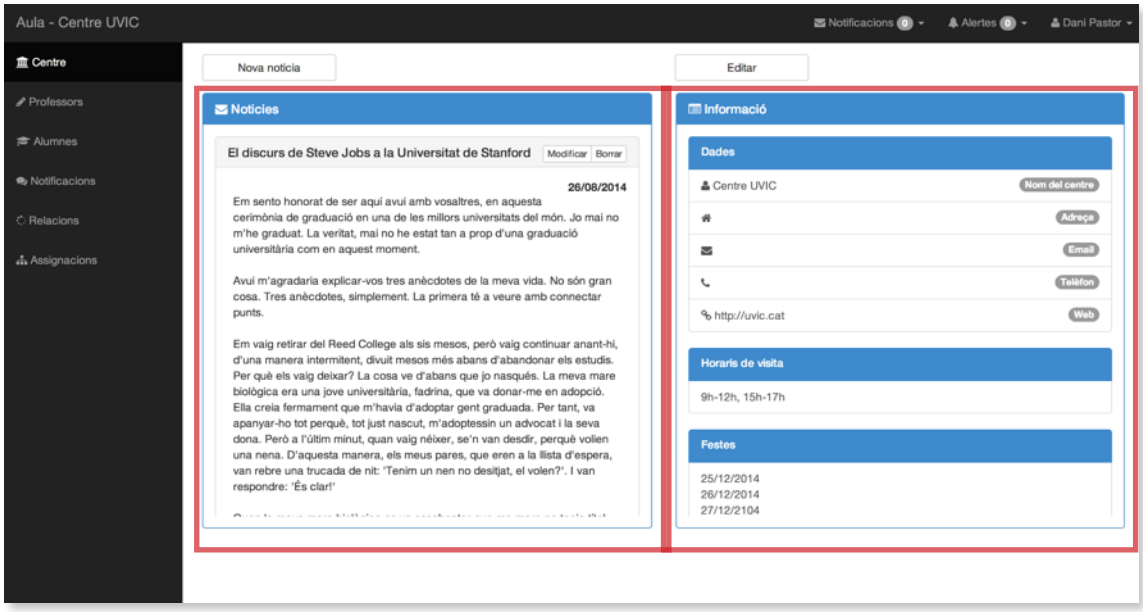
Per construir les vistes de l'aplicació s'ha utilitzat un motor de plantilles anomenat *twig*, que ve integrat en l'instal·lació de *symfony*. Aquest motor ens permet una lectura molt clara del codi de les vistes, així com de disposar de diferents instruccions per mostrar degudament la informació i poder gestionar-la segons convingui.

Tot seguit veurem tres pantalles definint alguns del apartats de cada *layout*:

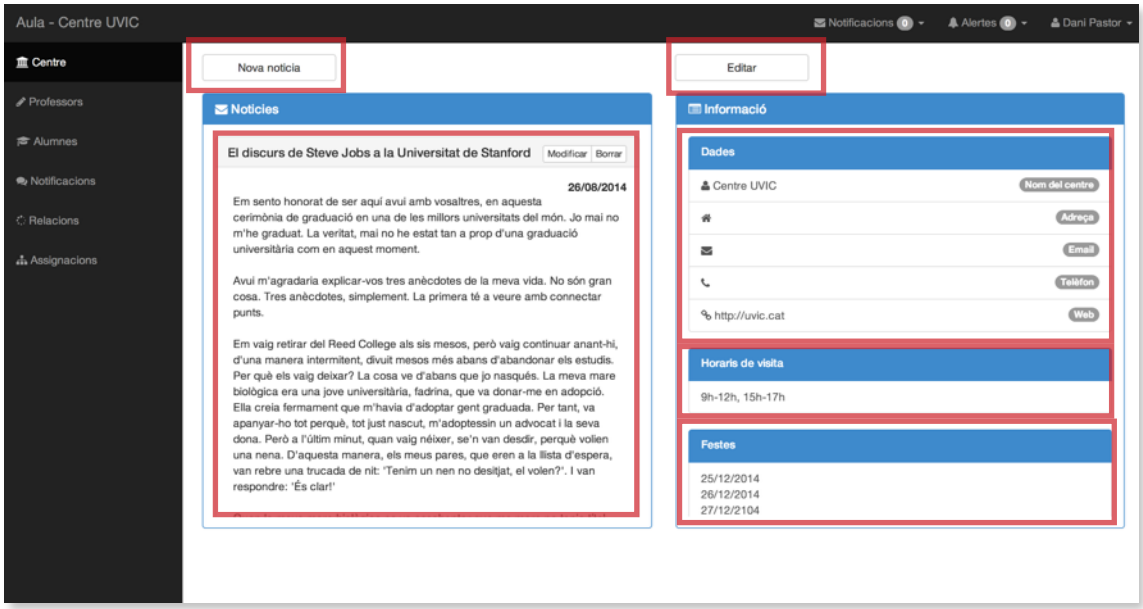
Seccions del *layout* base (base.html.twig):



Seccions del *layout* segon nivell (layout2.html.twig):

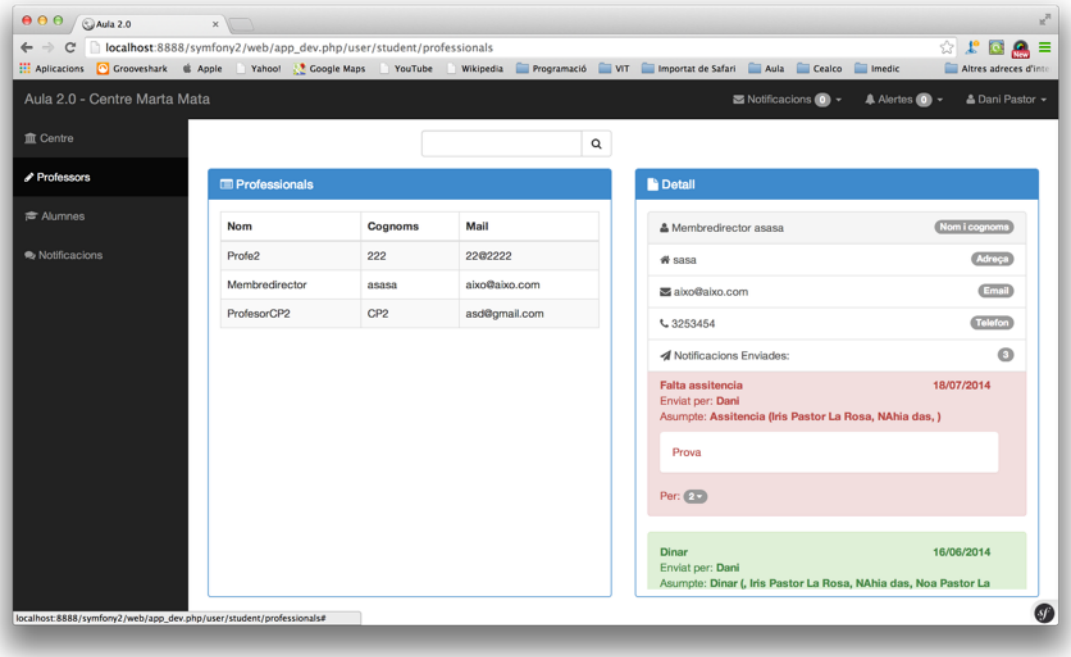


Exemple de *layout* final (vistes de cada apartat):

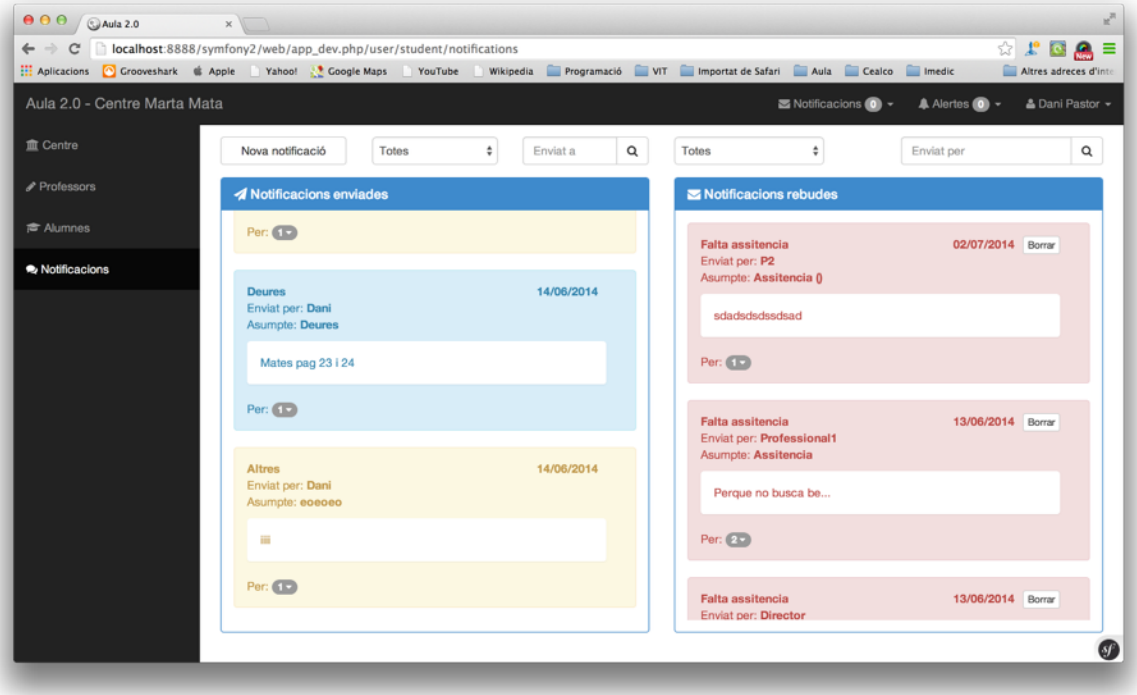


Per aconseguir una millor experiència d'usuari, s'ha utilitzat en la part del client *javascript* (juntament amb *Jquery*) per segons quins apartats, així com AJAX per carregar informació del servidor sense recarregar la pàgina. En tot moment s'ha intentat que l'experiència d'usuari fos com si es tractes d'una aplicació instal·lada en local. També s'han fet servir els icones de *fortawesome* en tota l'aplicació, i no s'ha utilitzat cap formulari modal (pop up) per així poder aconseguir la màxima experiència d'usuari en mòbils. L'entrada de dades es realitza a partir d'uns formularis que segueixen en tot moment els mateixos patrons de conducta, com per exemple posar el boto d'acceptar o de tornar enrere sempre en el mateix lloc.

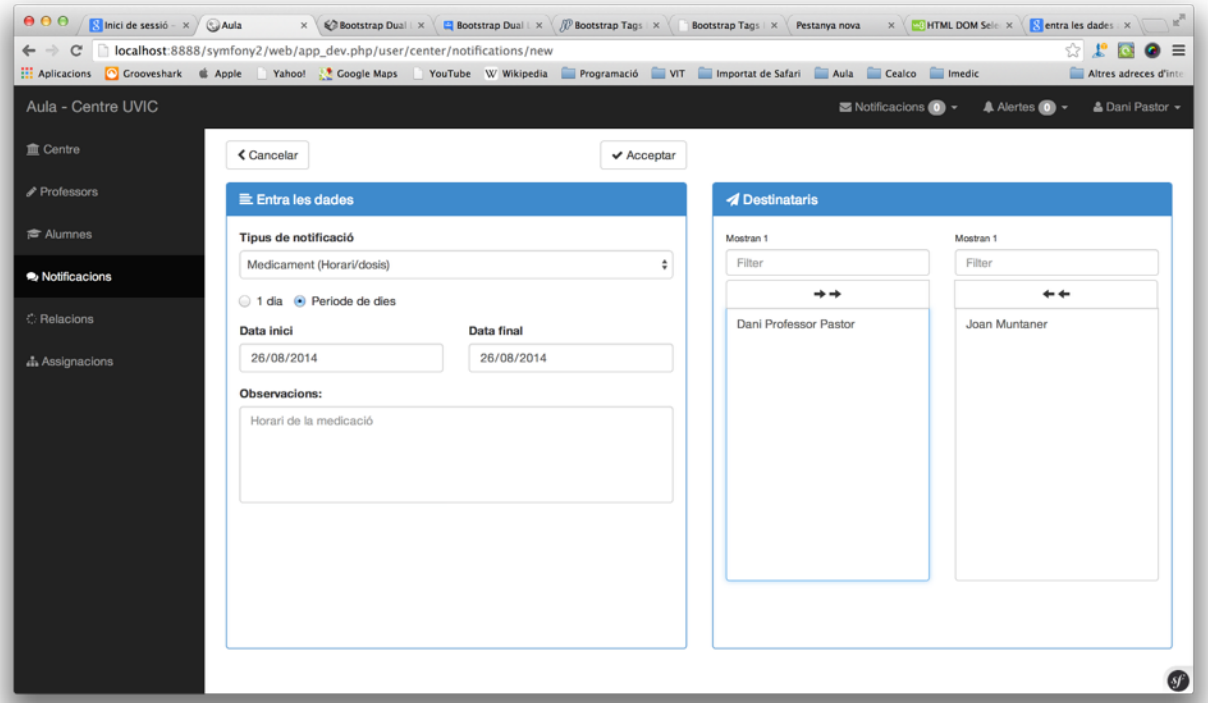
Vista de la pantalla dels professionals vista per un usuari (estudiant):



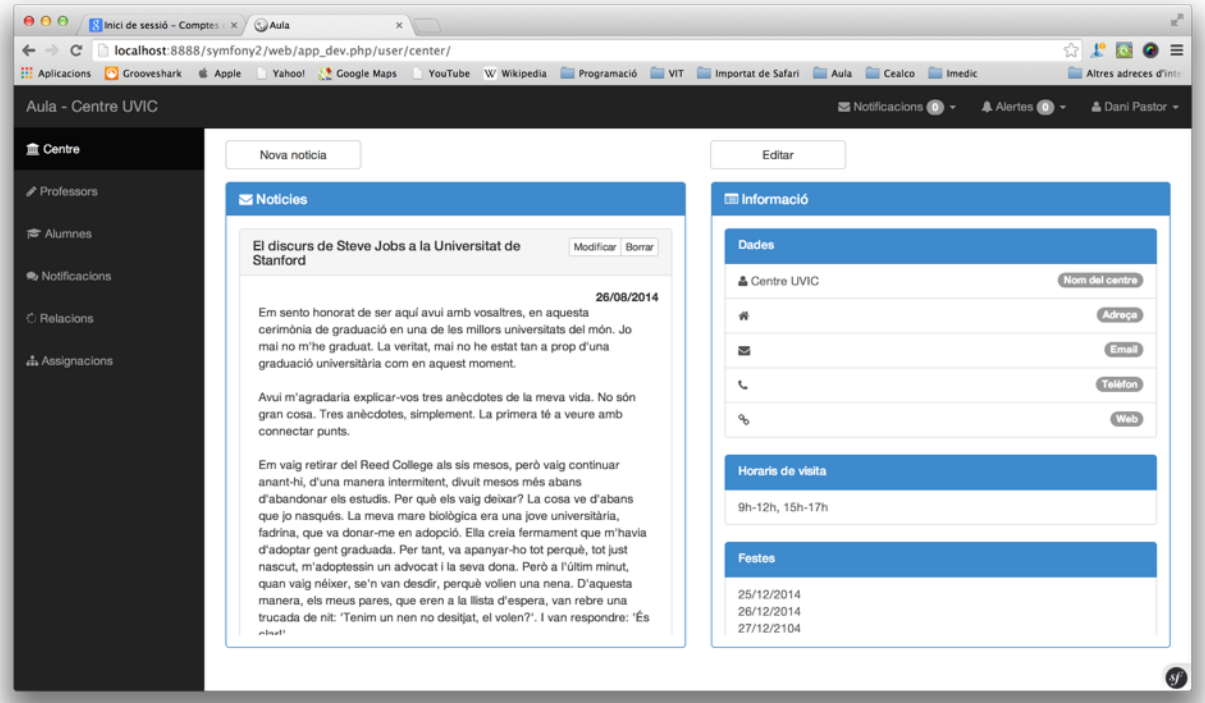
Vista de la pantalla de notificacions vista per un usuari (estudiant):



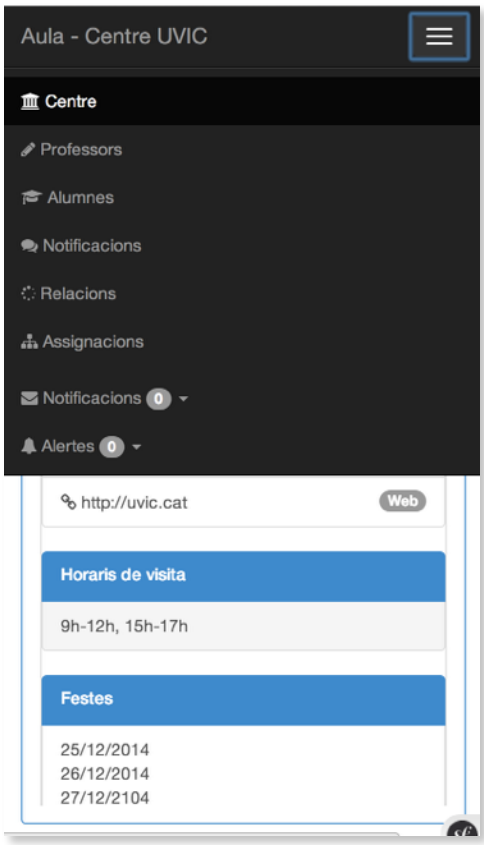
Exemple de la vista per entrar notificacions:

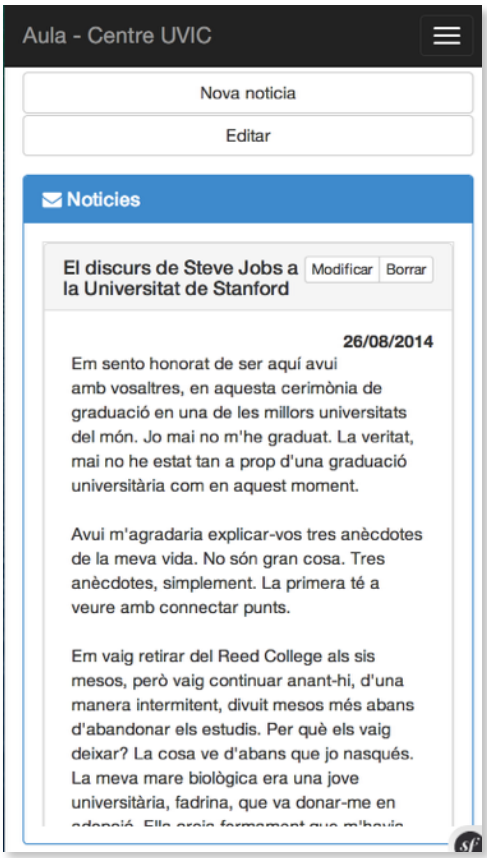


Exemple de la vista inicial amb les noticies del centre:



Exemple del menú en una pantalla format mòbil i d'una vista de noticies:





5.2.Disseny de la B.D

Com hem dit abans, treballem amb la base de dades relacional *MySql* a través d'una llibreria anomenada *doctrine*. Partint del diagrama de classes que abans hem presentat, fem una traducció de les entitats per presentar el model relacional. Les taules (ja amb nomenclatura amb anglès) seran les següents:

- au_account: Representa a la classe compte.
- au_center: Representa a la classe centre.
- au_member: Representa les classes professionals i estudiants.

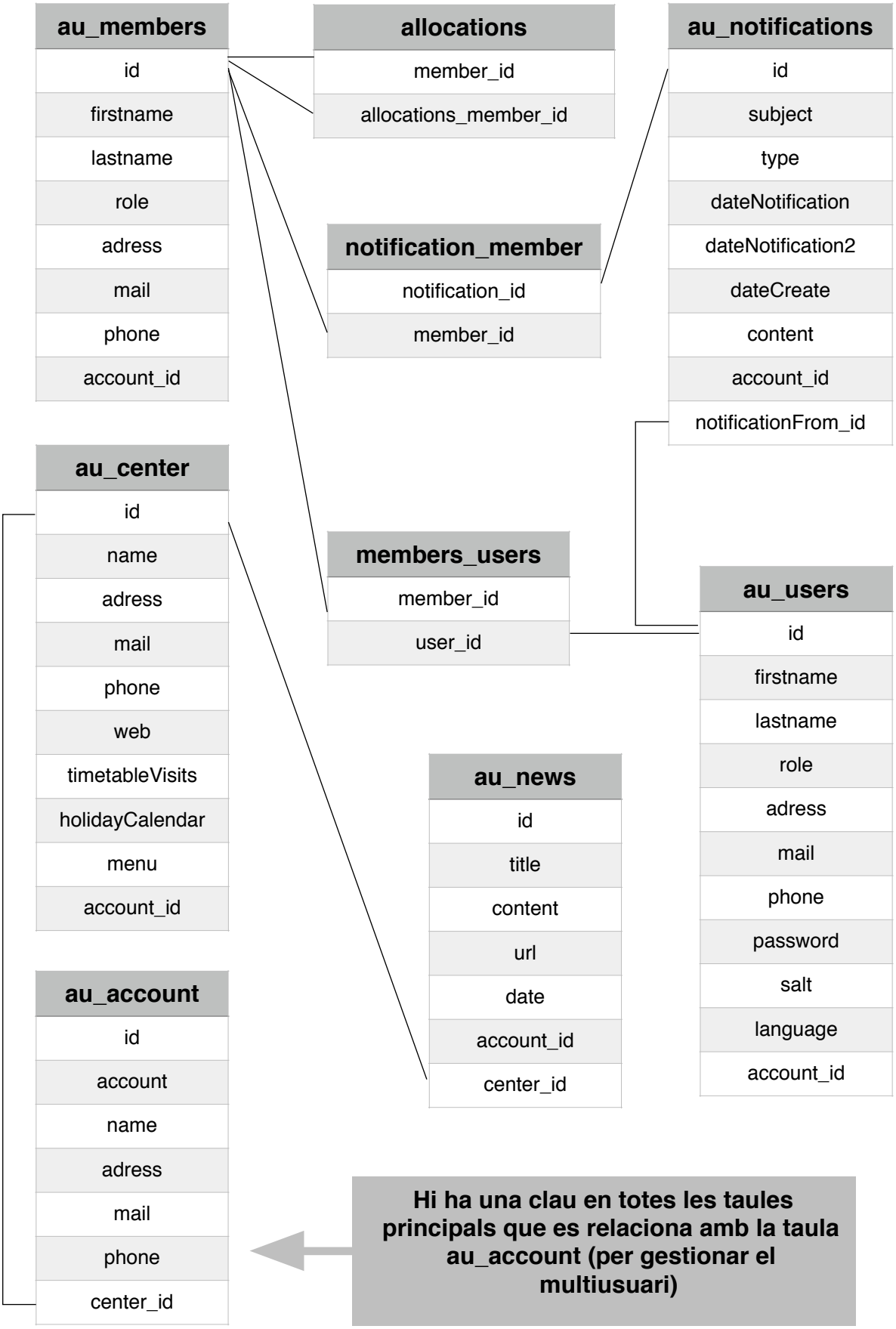
Les classes professionals i estudiants s'han unificat en una taula anomenada *au_members* i se li ha afegit el camp *role*, per identificar si es tracta d'un estudiant o d'un professional. D'aquesta manera ens es molt mes fàcil fer el tractament de les notificacions perquè ens permet tenir els destinataris en una sola taula, independentment de si són professionals o estudiants.

- au_news: Representa la classe noticies.
- au_notification: Representa la classe notificacions.
- au_user: Representa la classe usuaris.

Tot seguit detallo les taules que s'han creat per gestionar les relacions M:N.
En aquest projecte he utilitzat el sistema d'anotacions dins el codi de symfony (més endavant es mostraran exemples), així les relacions N:M entre les taules es creen automàticament. Un desavantatge d'aquest sistema és que de moment *doctrine* no permet la creació automàtica de camps extres en aquestes taules, per tant, si es necessita informació suplementaria en una relació, s'haurà de crear la taula a part amb el camp extra i amb les relacions 1:N i N:1 respectivament.

- allocations: Representa la relació (assignació) de membres i usuaris.
- members_users: Representa la relació entre usuaris.
- notification_member: Representa la relació entre notificacions i els seus destinataris (membres).

A continuació presentem la traducció dels diagrama de classes a les taules de la B.D MySql:


au_members
id
firstname
lastname
role
adress
mail
phone
account_id
allocations
member_id
allocations_member_id
au_notifications
id
subject
type
dateNotification
dateNotification2
dateCreate
content
account_id
notificationFrom_id
notification_member
notification_id
member_id
au_center
id
name
adress
mail
phone
web
timetableVisits
holidayCalendar
menu
account_id
members_users
member_id
user_id
au_users
id
firstname
lastname
role
adress
mail
phone
password
salt
language
account_id
au_news
id
title
content
url
date
account_id
center_id
au_account
id
account
name
adress
mail
phone
center_id
Hi ha una clau en totes les taules principals que es relaciona amb la taula au_account (per gestionar el multiusuari)

Les taules es creen en línia de comandes segons les anotacions que haguem posat en el codi de les classes dins del projecte. *Doctrine* agafa aquestes anotacions i crea les taules i les relacions pertinents. D'aquesta manera tenim un sistema ràpid i fiable per crear i poder modificar ràpidament l'estructura de les dades. Els *getters* i *setters* també es poden crear automàticament amb instruccions en línia de comandes. Ex:

"php app/console doctrine:schema:update —force"

Amb aquesta instrucció crearíem les entitats a la B.D i les seves relacions segons les anotacions que haguéssim posat dins el nostre codi. Ex:

```
"…..
namespace Aula\NotificationBundle\Entity;

use Doctrine\ORM\Mapping as ORM;

/**
 * @ORM\Entity(repositoryClass="Aula\NotificationBundle\Entity
\NotificationRepository")
 * @ORM\Table(name="AU_NOTIFICATION")
 */
class Notification
{
    /**
    * @ORM\Id
    * @ORM\Column(type="integer")
    * @ORM\GeneratedValue
    */
    protected $id;
    /** @ORM\ManyToOne(targetEntity="Aula\AccountBundle\Entity\Account") */
    protected $account;
    /** @ORM\ManyToOne(targetEntity="Aula\UserBundle\Entity\User") */
    protected $notificationFrom;
    /** @ORM\ManyToMany(targetEntity="Aula\MemberBundle\Entity\Member")
*/
    protected $to;
    /** @ORM\Column(type="string") */
```

```
    protected $subject;
    /** @ORM\Column(type="string") */
    protected $type;
    /** @ORM\Column(type="date") */
    protected $dateNotification;
    /** @ORM\Column(type="date") */
    protected $dateNotification2;
    /** @ORM\Column(type="date") */
    protected $dateCreate;
    /** @ORM\Column(type="string") */
    protected $content;

….”
```

Les anotacions comencen amb aquest símbol /** i acaben amb aquest */. Amb les anotacions podem passar molta informació de com volem els camps, des del seu tipus, si pot ser null, la llargda, … També a través de les anotacions podem crear camps en les entitats que ja mapegen una relació de la taula. Per exemple:

```
“….

    /**
     * @ORM\ManyToMany(targetEntity="Aula\UserBundle\Entity\User",
inversedBy="members")
     * @ORM\JoinTable(name="members_users",
     *      joinColumns={@ORM\JoinColumn(name="member_id",
referencedColumnName="id")},
     *      inverseJoinColumns={@ORM\JoinColumn(name="user_id",
referencedColumnName="id")}
     *      )
     */
    protected $users;
….”
```

Amb aquesta anotació dins la classe podem accedir al camp corresponent i ja tindrem els resultats de la relació de la consulta. En aquest exemple podem

veure com creem el camp *users,* que ens dona els usuaris relacionats amb un membre.

I amb aquesta instrucció crearíem els *getters* i *setter* bàsics per treballar amb els camps de cada classe:

"php app/console generate:doctrine:entities NotifiactionsBundle"

Ex:

"….

```
/**
     * Get id
     *
     * @return integer
     */
    public function getId()
    {
        return $this->id;
    }

    /**
     * Set account
 *
     * @param string $account
     * @return Notification
     */
    public function setAccount($account)
    {
        $this->account = $account;

        return $this;
    }

    /**
     * Get account
```

```
 *
 * @return string
 */
public function getAccount()
{

…."
```

Per crear el model relacional, com he dit, s'ha utilitzat la base de dades *MySql*, però la feina de relacionar-se amb la Base de dades es fa a mitjançant una llibreria *Doctrine*, que incorpora *Symfony*. La seva utilització és completament voluntària, però facilita molt el treball amb la B.D, doncs es crea una capa entre aquesta i l'aplicació que mapeja la informació per així poder treballar amb objectes dins del codi. L'estructura de la Base de dades i les relacions, es creen utilitzant el sistema d'anotacions dins les diferents classes de PHP. A més a més, el fet d'utilitzar aquesta llibreria, et permet poder migrar cap a una altre sistema de base de dades sense haver de tocar pràcticament cap línia de codi i només s'hauria d'indicar a *Doctrine* quina nova base de dades es vol utilitzar, prèvia comprovació que aquesta estigui suportada per la llibreria.

Tot seguit mostro un exemple de com s'instancia una classe a la B.D. Primer creem l'objecte, tot seguit a través dels *setters* creats anem omplint els camps amb els diferents valors, creem una instancia de *doctrine* i persistim la informació. Cal dir que la informació no queda guardada fins que no s'aplica el mètode *flush().* D'aquesta manera podem treballar amb tots els objectes que vulguem i al final de tot el proces, si tot es correcte, guardar-ho tot a la B.D.

```
"….
$notification = new notification();
$notification->setAccount($account);
…….

$em = $this->getDoctrine()->getManager();
$em->persist($notification);
$em->flush();
…."
```

5.3.Disseny de l'aplicació

Com hem dit abans, la plataforma utilitza l'arquitectura model-vista-controlador. Tot el disseny està estructurat perquè depenent del tipus d'usuari s'accedeixi a uns controladors o a uns altres. Per aconseguir això, s'han definit uns espais de noms i unes rutes individuals per cada rol d'usuari. Accedint a aquestes rutes s'accedeix al controlador corresponent i s'executa l'acció desitjada. Totes aquestes rutes per accedir a la part NO pública de l'aplicació, estan protegides per un *firewall* que ens permet tenir els diferents rols separats i poder escalar l'aplicació fàcilment. Els *Bundles*, (així és com *symfony* anomena les agrupacions que fas dins el codi), que he creat per treballar són els següents:

- AccountBundle
- CenterBundle
- StudentBundle
- ProfessionalBundle
- MemberBundle
- NotificationBundle
- UserBundle

S'han creat dos Bundles (*StudentBundle* i *ProfessionalBundle*) que no tenen cap entitat en el model relacional (és una abstracció de l'entitat members), però que faciliten la creació i estructuració de les rutes depenent del tipus d'usuari dins del codi. També s'ha decidit que l'entitat de noticies (*news*) estigui dins el *Bundle CenterBundle*.

Per generar aquestes entitats s'ha utilitzat la següent instrucció en la línia de comandes:

"php app/console doctrine:generate:entity"

Aquesta comanda ens permet generar les entitats de manera fàcil i sense cometre errors. Un cop executada ens demanarà una sèrie d'informació (nom del *bundle*, sistema per definir les entitats, camps de l'entitat…) i un cop finalitzada ja tindrem la nostra entitat creada. A partir d'aquí, com hem dit abans en l'apartat del disseny de la B.D, podem executar les instruccions per

crear les taules i les funcions bàsiques per llegir i escriure en els camps corresponents. Cada Bundle que creem tindrà un controlador ja escrit per defecte, però podem escriure els que nosaltres vulguem. Aquí s'ha optat per utilitzar un sol controlador per cada Bundle i implementar a dins les accions necessàries. Les rutes també estan estructurades per cada Bundle, per així poder tenir el codi el mes estructurat possible. D'aquesta manera tenim un sistema de rutes comú en el projecte, que hereta les rutes particulars de cada bundle, formant tot l'enruta't del sistema. Tot seguit mostrem un exemple de les rutes generals de l'aplicació:

"…

member:
resource: "@MemberBundle/Resources/config/routing.yml"
prefix: /

account:
resource: "@AccountBundle/Resources/config/routing.yml"
prefix: /

user:
resource: "@UserBundle/Resources/config/routing.yml"
prefix: /

center:
resource: "@CenterBundle/Resources/config/routing.yml"
prefix: /user/center

professional:
resource: "@ProfessionalBundle/Resources/config/routing.yml"
prefix: /user/professional

student:
resource: "@StudentBundle/Resources/config/routing.yml"
prefix: /user/student

notification:
resource: "@NotificationBundle/Resources/config/routing.yml"

prefix: /

home:
path: /home
defaults: {_controller:AccountBundle:Default:home }

home_centers:
path: /center/{accountname}
defaults: {_controller:CenterBundle:Default:web }

…"

Dins el codi mai es fa referència a cap url, sinó que s'utilitza el nom donat en les rutes, així podem canviar fàcilment una ruta en qualsevol moment i només ho hem de fer en un lloc concret. Com es pot veure, per cada Bundle creat s'importen les seves rutes. L'encarregat de protegir els diferents apartats de l'aplicació segons l'accés és el sistema de seguretat que incorpora symfony. En aquest arxiu definim els *firewalls* que volem i les rutes que hi poden accedir. Ex:

"…

access_control:
- { path: ^/user/login, roles: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/user/register, roles: IS_AUTHENTICATED_ANONYMOUSLY }
- { path: ^/user/center/, roles: ROLE_CENTER }*
- { path: ^/user/professionals/, roles: ROLE_PROFESSIONAL }*
- { path: ^/user/students/, roles: ROLE_STUDENT }*
- { path: ^/user/access, roles: [ROLE_CENTER, ROLE_PROFESSIONAL, ROLE_STUDENT] }

…"

Aquí estem definint els diferents rols que tenen els usuaris de l'aplicació així com l'accés a les seves rutes corresponents. Per exemple, tenim que tots els usuaris *ROLE_CENTER* parteixen de la ruta *user/center/**, per tant, en totes les rutes que tinguin aquest espai de noms, només hi podran accedir els

usuaris amb el rol *ROLE_CENTER*. Aquesta distribució inicial d'accessos segons el rol la fa l'acció *access* del controlador de *UserBundle:*

```
user_access:
   path:     /user/access
   defaults: { _controller: UserBundle:Default:access }
```

Per això podem veure que aquesta ruta es accessible per tots els tipus d'usuari. També podem veure com hi han rutes que s'hi pot accedir sense tenir cap role assignat, com per exemple la pagina de registre i de *login*. En aquest arxiu també es determina el tipus de seguretat per encriptar els password i es defineix la classe que s'utilitzarà per ser proveïdor d'usuaris.

Per aconseguir que l'usuari tingui la sensació que esta treballant en una aplicació en local, s'ha utilitzat Javascript, Jquery i AJAX. Alguns exemples:

```
"….

$(".detailtable").on('click', 'tr', function(){
      var id = $(this).closest('tr').attr('value');
      var url = '{{ path(role~'_detail_'~option, {'id':'paramid'}) }}';
      url = url.replace("paramid", id);
      $('#detail_{{option}}').load(url);
    });

…"
```

Aquesta funció fa que al seleccionar una fila d'una taula, es carregui mitjançant AJAX la informació de la selecció. Com es pot veure aquí, *var url = '{{ path(role~'_detail_'~option, {'id':'paramid'}) }}';*, s'utilitzen paràmetres passats a la vista per construir la ruta necessària que executarà el controlador amb l'acció corresponent, d'aquesta manera una sola funció serveix per totes les vistes i usuaris que necessitin aquesta acció. Aquest sistema es una manera de tenir molt poc codi javascript i fàcil de mantenir però comporta que s'hagin de definir algunes rutes que no es necessiten, o posar condicionals dins les vistes amb *twig* perquè no es llegeixin en segons quins casos (si la ruta no existeix el sistema crea una excepció).

Per consultar la informació necessària en cada moment utilitzem consultes SQL que s'executen sempre dins el controlador. Aquestes consultes estan dividides per cada Bundle en un repositori a part (arxiu), per així poder gestionar-les mes fàcilment (a més a més estan protegides contra l'injecció de codi SQL). A Totes les consultes principals se li passa un paràmetre amb *l'account (compte)* de l'usuari per poder filtrar la informació. A continuació poso un exemple d'una consulta i descric les diferents consultes que es realitzen a la B.D:

"…

```
public function findStudent($id,$account)
    {
        $em = $this->getEntityManager();
        $dql = 'SELECT m FROM MemberBundle:Member m WHERE m.id
= :id
        and m.role = :role and m.account = :account';
        $consulta = $em->createQuery($dql);
        $consulta -> setParameter('id',$id);
        $consulta -> setParameter('account',$account);
        $consulta -> setParameter('role','student');
        return $consulta->getOneOrNullResult();
    }
```

…"

Consultes del *AccountBundle* (*AccountRepository*):

- **findAccount($id)**: Retorna l'account segons el seu id.
- **findAccountName($accountName)**: Retorna el nom del compte.

Consultes del *CenterBundle* (*CenterRepository*):

- **findCenter($account)**: Retorna un centre segons el compte.
- **findCenters()**: Retorna tots els centres. (no utilitzat de moment).

Consultes del *CenterBundle* (*NewsRepository*):

- **findNews($account)**: Retorna les noticies d'un compte.
- **findNew($account,$id)**: Retorna la noticia amb l'id passat.

Consultes del *MemberBundle* (*MemberRepository*):

- **findMember($id,$account)**: Retorna un membre segons l'id.
- **findAllMembers($account)**: Retorna tots els membres d'un compte.
- **findStudent($id,$account)**: Retorna un estudiant segons el seu id.
- **findProfessional($id,$account)**: Retorna un estudiant segons el seu id.
- **findCenter($id,$account)**: Retorna un membre tipus center (no utilitzat de moment).
- **findStudents($account)**: Retorna tots els estudiants d'un compte.
- **findProfessionals($account)**: Retorna tots els professionals d'un compte.
- **searchMembers($search,$account)**: Retorna els membres filtrats segons el nom o cognom contingui la cadena $search o en cas de passar-hi un "*" retorna tots els membres.
- **searchMembersRole($search,$roleMember,$account)**: Retorna els membres filtrats segons el nom o cognom contingui la cadena $search, i segons el rol indicat. En cas de passar-hi un "*" retorna tots els membres segons el rol.

Consultes del NotificationBundle (NotificationRepository):

- **findNotification($id,$account)**: Retorna una notificació segons el seu id.
- **findAllNotification($account)**: Retorna totes les notificacions d'un compte.
- **findUserNotification($user, $account)**: Retorna les notificacions enviades per un usuari.
- **findNotificationsCenterDay($account)**: Retorna totes les notificacions d'un centre que siguin del dia.
- **findNotificationsDay($member, $account)**: Retorna totes les notificacions del dia enviades a un membre.
- **findNotificationsReceived($member, $account)**: Retorna totes les notificacions enviades a un membre.

- **findAlertsCenterDay($account)**: Retorna totes les alertes del dia d'un centre.
- **findAlertsDay($member, $account)**: Retorna totes les alertes del dia enviades a un membre.
- **searchNotificationsReceived($type,$search,$member, $account)**: Retorna totes les notificacions d'un tipus concret rebudes per un membre i que continguin la cadena $search en el seu nom o cognom.
- **searchNotificationsSend($type,$search,$user, $account)**: Retorna totes les notificacions d'un tipus concret enviades per un usuari i que continguin la cadena $search en el seu nom o cognom.
- **searchNotificationsSendUserToMember($type,$search,$member,$user, $account)**: Retorna totes les notificacions d'un tipus concret rebudes per un membre i enviades per un usuari concret i que continguin la cadena $search en el seu nom o cognom.

Consultes del *UserBundle* (*UserRepository*):

- **findUser($id,$account)**: Retorna un usuari d'un compte segons el seu id.
- **findUsername($mail)**: Retorna un usuari segons el seu mail (*username*).
- **findAllUsers($account)**: Retorna tots els usuaris d'un compte.
- **findMail($mail,$account)**: Retorna un usuari d'un compte segons el seu mail.
- **searchUsers($search,$account)**: Retorna un usuari d'un compte on el seu nom o cognom contingui la cadena $search.

Per entrar de dades per part de l'usuari s'ha optat per crear formularis com a serveis, per poder-los utilitzar en més d'una vista. Aquesta manera de treballar permet tenir mes control sobre les dades que es volen mostrar i recollir en un formulari. Poso aquí un exemple:

```
"....

class NotificationType extends AbstractType
{
   public function buildForm(FormBuilderInterface $builder, array $options)
   {
```

```
        $builder->add('to','entity', array(
                    'class' => 'MemberBundle:Member',
                    'property' => 'firstname',
                    'multiple' => true,
                    'attr'=> array('style'=>'display:none',)));
        $builder->add('subject');
        $builder->add('type', 'choice', array('choices' => array(
            'lack' => 'Falta assitencia',
            'medicament' => 'Medicament (Horari/dosis)',
            'exit' => 'Comunicar sortida',
            'visit' => 'Demenar hora de visita',
            'schoolwork' => 'Deures',
            'lunch' => 'Dinar',
            'extraschool' => 'Extra escolars',
            'other' => 'Altres')));

        $builder->add('dateNotification', 'date', array('widget' => 'single_text'));
        $builder->add('dateNotification2', 'date', array('widget' => 'single_text'));
        $builder->add('dateCreate');
        $builder->add('content', 'textarea');
        $builder->getForm();
    }

….”
```

Aquestes classes es creen en un arxiu a apart, aquí podem especificar cada camp exactament com el volem representar, i si es dones el cas fins i tot i podem injectar consultes per mostrar i limitar els valors que l’usuari pot entrar. Ex:

```
“….

class MemberStudentType extends AbstractType
{
    public function buildForm(FormBuilderInterface $builder, array $options)
    {
        $account = $options['account'];
        $search = "*";
```

```
        $builder->add('firstname');
        $builder->add('lastname');
        $builder->add('address');
        $builder->add('mail');
        $builder->add('phone');
        $builder->add('users',
            'entity',
             array(
                 'class'=>'Aula\UserBundle\Entity\User',
                 'query_builder' => function (\Aula\UserBundle\Entity
\UserRepository $repository) use ($account,$search)
                 {
                   if ($search == "*") {
                    return $repository->createQueryBuilder('s')
                         ->where('s.account = ?1')
                         ->setParameter(1, $account)
                         ->add('orderBy', 's.lastname ASC');}
                   if ($search != "*") {
                    return $repository->createQueryBuilder('s')
                         ->where('s.account = ?1 and (s.firstname like ?2 or
s.lastname like ?2)')
                         ->setParameter(1, $account)
                         ->setParameter(2, $search)
                         ->add('orderBy', 's.lastname ASC');}
                 },
                 'required' => false,
                 'expanded' => true,
                 'multiple' => true,
                )

            );
        $builder->getForm();
    }

….”
```

Aquí podem veure com utilitzem una consulta (query_builder) dins un camp per determinar quins usuaris poden estar relacionats amb un membre

(només els que pertanyen al mateix compte) segons una cerca per nom. Un cop fet això, dins el controlador que fa la crida al formulari, creem un formulari amb aquesta classe, i ja tindrem tots els camps disponibles per treballar dins la nostra vista. Ex:

"….

```
$form = $this->createForm('memberStudent', $student, array('account' =>
$account));
$form->handleRequest($request);
    if ($form->isValid()) {
```

…."

Finalment voldria dir que en el moment de crear-se un compte també es crea una pàgina web genèrica del centre. A través d'aquesta web es poden registrar els usuaris (pares o professionals) del centre. També hi ha una pàgina web general de login, i una general de la plataforma, que un cop dins permet accedir a un centre determinat d'una llista desplegable.

Per acabar detallo el sistema de rutes que permet controlar tots els esdeveniments de l'aplicació:

Esdeveniments de l'usuari administrador del centre (usuari tipus "center"):

Nom: center_index
Event: Apartat del centre corresponent a l'administrador
Resposta: CenterBundle:Default:index
Descripció: Mostra l'apartat corresponent al centre, amb la informació i les accions que pot fer l'administrador del compte (user=center).

Nom: center_professionals
Event: Apartat dels professionals corresponent a l'administrador
Resposta: CenterBundle:Default:professionals, id

Descripció: Mostra l'apartat corresponent als professionals amb la informació i les accions que pot fer l'administrador del compte (user=center). Es passa l'id del professional per poder mostrar el seu detall.

Nom: center_students
Event: Apartat dels estudiants corresponent a l'administrador
Resposta: CenterBundle:Default:students
Descripció: Mostra l'apartat corresponent als estudiants amb la informació i les accions que pot fer l'administrador del compte (user=center).

Nom: center_notifications
Event: Apartat de les notificacions corresponent a l'administrador
Resposta: CenterBundle:Default:notifications, id
Descripció: Mostra l'apartat corresponent a les notificacions amb la informació i les accions que pot fer l'administrador del compte (user=center). Es passa l'id de la notificació per poder mostrar el seu detall.

Nom: center_allocations
Event: Apartat de les assignacions corresponent a l'administrador
Resposta: CenterBundle:Default:allocations
Descripció: Mostra l'apartat corresponent a les assignacions amb la informació i les accions que pot fer l'administrador del compte (user=center).

Nom: center_relations
Event: Apartat de les relacions corresponent a l'administrador
Resposta: CenterBundle:Default:relations
Descripció: Mostra l'apartat corresponent a les relacions amb la informació i les accions que pot fer l'administrador del compte (user=center).

Nom: center_detail_student
Event: Detall de la informació de l'estudiant
Resposta: StudentBundle:Default:student, id
Descripció: Mostra el detall de l'estudiant seleccionat. Es passa l'id de d'estudiant.

Nom: center_detail_professional
Event: Detall de la informació del professional

Resposta: ProfessionalBundle:Default:professional, id
Descripció: Mostra el detall del professional seleccionat. Es passa l'id del professional.

Nom: center_detail_notification
Event: Mostrar un notificació
Resposta: NotificationBundle:Default:notification, id
Descripció: Mostra la notificació. Es passa l'id de la notificació.

Nom: center_detail_allocations
Event: Mostra les assignacions
Resposta: UserBundle:Default:allocations, id
Descripció: Mostra els membres assignats a un usuari. Es passa l'id de l'usuari.

Nom: center_detail_relations
Event: Mostra les relacions
Resposta: MemberBundle:Default:relations, id
Descripció: Mostra les relacions que te un membre. Es passa l'id del membre.

Nom: center_new_professional
Event: Entrar un nou professional
Resposta: ProfessionalBundle:Default:new
Descripció: Crea un nou objecte professional, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: center_new_student
Event: Entrar un nou estudiant
Resposta: StudentBundle:Default:new
Descripció: Crea un nou objecte estudiant, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: center_new_notification
Event: Entrar un nova notificació
Resposta: NotificationBundle:Default:new
Descripció: Crea un nou objecte notificació, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: center_edit_relations
Event: Modificar les relacions d'un membre
Resposta: MemberBundle:Default:edit, id
Descripció: Carrega l'objecte membre de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: center_edit_allocations
Event: Modificar les assignacions d'un usuari
Resposta: UserBundle:Default:edit, id
Descripció: Carrega l'objecte usuari de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: center_delete_professional
Event: Eliminar un professional
Resposta: ProfessionalBundle:Default:delete, id
Descripció: Busca i si existeix borra de la B.D l'objecte membre amb identificador id.

Nom: center_delete_student
Event: Borrar un professional
Resposta: StudentBundle:Default:delete, id
Descripció: Busca i si existeix borra de la B.D l'objecte membre amb identificador id.

Nom: center_delete_notification
Event: Eliminar una notificació
Resposta: NotificationBundle:Default:delete, id
Descripció: Busca i si existeix l'elimina de la B.D les relacions entre una notificació i els membres de l'usuari actual.

Nom: center_edit_professional
Event: Editar un professional
Resposta: ProfessionalBundle:Default:edit, id
Descripció: Carrega l'objecte professional de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: center_edit_student
Event: Editar un estudiant
Resposta: StudentBundle:Default:edit, id
Descripció: Carrega l'objecte estudiant de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: center_edit_student
Event: Editar un estudiant
Resposta: StudentBundle:Default:edit, id
Descripció: Carrega l'objecte estudiant de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: center_profile
Event: Editar el perfil de l'usuari
Resposta: CenterBundle:Default:profile, option
Descripció: Carrega l'objecte usuari de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: center_new_new
Event: Entrar una noticia del centre
Resposta: CenterBundle:Default:newNews
Descripció: Crea un nou objecte noticia, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: center_delete_new
Event: Borrar una noticia del centre
Resposta: CenterBundle:Default:deleteNews, id
Descripció: Busca i si existeix borra de la B.D la noticia amb l'identificador id.

Nom: center_edit_new
Event: Modificar una noticia del centre
Resposta: CenterBundle:Default:editNews, id

Descripció: Carrega l'objecte noticia de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: center_edit_center
Event: Editar les dades del centre
Resposta: CenterBundle:Default:edit
Descripció: Carrega l'objecte centre de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Esdeveniments de l'usuari professional (usuari tipus "professional"):

Nom: professional_center
Event: Apartat corresponent al centre
Resposta: ProfessionalBundle:Default:center
Descripció: Mostra l'apartat corresponent al centre, amb la informació i les accions que pot fer un usuari professional (user.type=professional).

Nom: professional_professionals
Event: Apartat corresponent als professionals
Resposta: ProfessionalBundle:Default:index, id
Descripció: Mostra l'apartat corresponent als professionals. Es passa l'id del professional per poder mostrar el seu detall.

Nom: professional_students
Event: Apartat corresponent als estudiants
Resposta: ProfessionalBundle:Default:students
Descripció: Mostrar l'apartat corresponent als estudiants.

Nom: professional_notifications
Event: Apartat de les notificacions
Resposta: ProfessionalBundle:Default:notifications
Descripció: Mostrar l'apartat corresponent a les notificacions.

Nom: professional_relations
Event: Apartat de les relacions entre membres

Resposta: ProfessionalBundle:Default:relations
Descripció: Mostrar l'apartat corresponent als membres que esta relacionat el professional

Nom: professional_detail_student
Event: Detall de la informació de l'estudiant
Resposta: StudentBundle:Default:student, id
Descripció: Mostra el detall de l'estudiant seleccionat. Es passa l'id de d'estudiant.

Nom: professional_detail_professional
Event: Detall de la informació del professional
Resposta: ProfessionalBundle:Default:professional, id
Descripció: Mostra el detall del professional seleccionat. Es passa l'id del professional.

Nom: professional_detail_notification
Event: Mostrar un notificació
Resposta: NotificationBundle:Default:notification, id
Descripció: Mostra la notificació. Es passa l'id de la notificació.

Nom: professional_detail_allocations
Event: Mostra les assignacions
Resposta: UserBundle:Default:allocations, id
Descripció: Mostra els membres assignats a un usuari. Es passa l'id de l'usuari.

Nom: professional_detail_relations
Event: Mostra les relacions
Resposta: MemberBundle:Default:relations, id
Descripció: Mostra les relacions que te un membre. Es passa l'id del membre.

Nom: professional_new_professional
Event: Entrar un nou professional
Resposta: ProfessionalBundle:Default:new
Descripció: Crea un nou objecte professional, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: professional_new_student
Event: Entrar un nou estudiant
Resposta: StudentBundle:Default:new
Descripció: Crea un nou objecte estudiant, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: professional_new_notification
Event: Entrar un nova notificació
Resposta: NotificationBundle:Default:new
Descripció: Crea un nou objecte notificació, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: professional_delete_notification
Event: Borrar un professional
Resposta: NotificationBundle:Default:delete, id
Descripció: Busca i si existeix borra de la B.D les relacions entre una notificació i els membres de l'usuari actual.

Nom: professional_edit_relations
Event: Modificar les relacions d'un membre
Resposta: MemberBundle:Default:edit, id
Descripció: Carrega l'objecte membre de la B.D, mostra el formulari per modificar les dades (relacions), i si tot es correcte actualitza el nou objecte a la B.D.

Nom: professional_edit_student
Event: Modificar un estudiant
Resposta: StudentBundle:Default:edit, id
Descripció: Carrega l'objecte usuari de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: professional_profile
Event: Editar el perfil de l'usuari
Resposta: ProfessionalBundle:Default:profile, option

Descripció: Carrega l'objecte usuari actual de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Esdeveniments de l'usuari professional (usuari tipus "professional"):

Nom: professional_center
Event: Apartat corresponent al centre
Resposta: ProfessionalBundle:Default:center
Descripció: Mostra l'apartat corresponent al centre, amb la informació i les accions que pot fer un usuari professional (user.type=professional).

Nom: professional_professionals
Event: Apartat corresponent als professionals
Resposta: ProfessionalBundle:Default:index, id
Descripció: Mostra l'apartat corresponent als professionals. Es passa l'id del professional per poder mostrar el seu detall.

Nom: professional_students
Event: Apartat corresponent als estudiants
Resposta: ProfessionalBundle:Default:students
Descripció: Mostrar l'apartat corresponent als estudiants.

Nom: professional_notifications
Event: Apartat de les notificacions
Resposta: ProfessionalBundle:Default:notifications
Descripció: Mostrar l'apartat corresponent a les notificacions.

Nom: professional_relations
Event: Apartat de les relacions entre membres
Resposta: ProfessionalBundle:Default:relations
Descripció: Mostrar l'apartat corresponent als membres que esta relacionat el professional

Nom: professional_detail_student
Event: Detall de la informació de l'estudiant
Resposta: StudentBundle:Default:student, id

Descripció: Mostra el detall de l'estudiant seleccionat. Es passa l'id de d'estudiant.

Nom: professional_detail_professional
Event: Detall de la informació del professional
Resposta: ProfessionalBundle:Default:professional, id
Descripció: Mostra el detall del professional seleccionat. Es passa l'id del professional.

Nom: professional_detail_notification
Event: Mostrar un notificació
Resposta: NotificationBundle:Default:notification, id
Descripció: Mostra la notificació. Es passa l'id de la notificació.

Nom: professional_detail_allocations
Event: Mostra les assignacions
Resposta: UserBundle:Default:allocations, id
Descripció: Mostra els membres assignats a un usuari. Es passa l'id de l'usuari.

Nom: professional_detail_relations
Event: Mostra les relacions
Resposta: MemberBundle:Default:relations, id
Descripció: Mostra les relacions que te un membre. Es passa l'id del membre.

Nom: professional_new_professional
Event: Entrar un nou professional
Resposta: ProfessionalBundle:Default:new
Descripció: Crea un nou objecte professional, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: professional_new_student
Event: Entrar un nou estudiant
Resposta: StudentBundle:Default:new
Descripció: Crea un nou objecte estudiant, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: professional_new_notification

Event: Entrar un nova notificació
Resposta: NotificationBundle:Default:new
Descripció: Crea un nou objecte notificació, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: professional_delete_notification
Event: Borrar un professional
Resposta: NotificationBundle:Default:delete, id
Descripció: Busca i si existeix borra de la B.D les relacions entre una notificació i els membres de l'usuari actual.

Nom: professional_edit_relations
Event: Modificar les relacions d'un membre
Resposta: MemberBundle:Default:edit, id
Descripció: Carrega l'objecte membre de la B.D, mostra el formulari per modificar les dades (relacions), i si tot es correcte actualitza el nou objecte a la B.D.

Nom: professional_edit_student
Event: Modificar un estudiant
Resposta: StudentBundle:Default:edit, id
Descripció: Carrega l'objecte usuari de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: professional_profile
Event: Editar el perfil de l'usuari
Resposta: ProfessionalBundle:Default:profile, option
Descripció: Carrega l'objecte usuari actual de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Esdeveniments de l'usuari estudiant (usuari tipus "student"):

Nom: student_center
Event: Apartat corresponent al centre

Resposta: StudentBundle:Default:center
Descripció: Mostra l'apartat corresponent al centre, amb la informació i les accions que pot fer un usuari estudiant (user.type=student).

Nom: student_professionals
Event: Apartat corresponent als professionals
Resposta: StudentBundle:Default:professionals
Descripció: Mostra l'apartat corresponent als professionals.

Nom: student_students
Event: Apartat corresponent als estudiants
Resposta: StudentBundle:Default:index
Descripció: Mostrar l'apartat corresponent als estudiants.

Nom: student_notifications
Event: Apartat de les notificacions
Resposta: StudentBundle:Default:notifications
Descripció: Mostrar l'apartat corresponent a les notificacions.

Nom: student_detail_student
Event: Detall de la informació de l'estudiant
Resposta: StudentBundle:Default:student, id
Descripció: Mostra el detall de l'estudiant seleccionat. Es passa l'id de d'estudiant.

Nom: student_detail_professional
Event: Detall de la informació del professional
Resposta: ProfessionalBundle:Default:professional, id
Descripció: Mostra el detall del professional seleccionat. Es passa l'id del professional.

Nom: student_detail_notification
Event: Mostrar un notificació
Resposta: NotificationBundle:Default:notification, id
Descripció: Mostra la notificació. Es passa l'id de la notificació.

Nom: student_new_professional
Event: Entrar un nou professional

Resposta: ProfessionalBundle:Default:new
Descripció: Crea un nou objecte professional, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: student_new_student
Event: Entrar un nou estudiant
Resposta: StudentBundle:Default:new
Descripció: Crea un nou objecte estudiant, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: student_new_notification
Event: Entrar un nova notificació
Resposta: NotificationBundle:Default:new
Descripció: Crea un nou objecte notificació, mostra el formulari per entrar les dades, les verifica i si tot es correcte registre el nou objecte a la B.D.

Nom: student_delete_student
Event: Borrar un estudiant
Resposta: StudentBundle:Default:delete, id
Descripció: Busca i si existeix borra de la B.D un estudiant.

Nom: student_delete_notification
Event: Borrar una notificació
Resposta: NotificationBundle:Default:delete, id
Descripció: Elimina la relació que existeix entre els membres d'un usuari i la notificació.

Nom: student_edit_student
Event: Modificar un estudiant
Resposta: StudentBundle:Default:edit, id
Descripció: Carrega l'objecte usuari de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Nom: student_profile
Event: Editar el perfil de l'usuari
Resposta: StudentBundle:Default:profile, option

Descripció: Carrega l'objecte usuari actual de la B.D, mostra el formulari per modificar les dades, les verifica i si tot es correcte actualitza el nou objecte a la B.D.

Altres esdeveniments comuns a tots els tipus d'usuaris i esdeveniments que pertanyen a la zona publica:

Nom: search_member
Event: Buscar membres (filtrar)
Resposta: MemberBundle:Default:search, roleMember, search
Descripció: Busca una part d'un text en el nom o cognom d'un tipus de membre concret.

Nom: edit_search_member
Event: Buscar els membres relacionats en l'edició d'aquest
Resposta: MemberBundle:Default:editsearch, roleMember, search
Descripció: Busca una part d'un text en el nom o cognom d'un tipus de membre concret, en el formulari per editar el membre i les seves relacions.

Nom: user_login
Event: Fer login per entrar a la zona privada
Resposta: UserBundle:Default:login
Descripció: Comprova si la contrasenya i usuari són correctes i en cas afirmatiu carrega (redirigeix) a user_access.

Nom: user_logout
Event: Desconnectar i sortir de la zona privada
Resposta: UserBundle:Default:logout
Descripció: Tanca la sessió i redirecciona a la pàgina del login.

Nom: user_register
Event: Registrar-se a la plataforma
Resposta: UserBundle:Default:register
Descripció: Crea l'usuari i omple la informació necessària a la B.D. Si es necessari crea el compte.

Nom: user_access
Event: Accedir a la plataforma
Resposta: UserBundle:Default:access
Descripció: Carrega la informació necessària segons l'usuari i redirigeix a la pàgina del centre.

Nom: search_user
Event: Buscar usuaris (filtrar)
Resposta: UserBundle:Default:search
Descripció: Filtra els usuaris assignats a un membre segons una cadena text buscada en el nom i cognom.

Nom: home_centers
Event: Accedir a la web del centre
Resposta: CenterBundle:Default:web
Descripció: mostrar la pagina web del centre, amb el formulari per poder registrar-se.

6. Conclusions

La conclusió principal que he tret de la creació del projecte és que qualsevol cosa, per simple que sigui, requereix esforç i dedicació. En un principi vaig plantejar el projecte amb una serie d'opcions que finalment han quedat descartades. En el transcurs de la creació d'aquesta aplicació, m'he adonat que és necessari centrar-se en les funcionalitats mes bàsiques i deixar per més endavant i per futures versions, altres funcionalitats que en principi podien semblar atractives, i buscar "el mínim producte viable", és a dir, aquella proposta que et permeti validar el producte ràpidament per poder saber si estàs en el camí correcte. Com a millores que crec que s'haurien d'afegir al projecte, serien la possibilitat de crear grups per enviar notificacions massives, així com la possibilitat de poder marcar les notificacions amb etiquetes.

El projecte m'ha servit per poder conèixer les bases d'un nou llenguatge i de noves tecnologies, aprendre a buscar informació de manera ràpida i eficaç i a utilitzar tots els recursos d'internet per resoldre els problemes que han anat sorgint al llarg del desenvolupament de l'aplicació, a més a més de tornar-me a introduir en el món de la programació molts anys després d'haver-ho deixat.

7. Agraïments

En el meu cas particular, el fet d'entregar el projecte després de molts anys d'haver acabat la carrera, m'ha suposat una feina extra que no hagués estat possible sense l'ajuda i l'empenta de moltes persones del meu voltant.

En primer lloc voldria donar la gràcies a la tutora del meu projecte, la Dolors, que després de tants anys m'ha donat les pautes necessàries per acabar aquest treball. Seguidament voldria agrair a la gent d'imedic, l'Isaac i en Xavier, la seva paciència i la seva generositat en donar-me l'oportunitat d'entrar en un projecte real com es imedicplus. També vull agrair a la Lina el seu cop de mà final amb les traduccions i de manera molt especial donar les gràcies amb en Jordi *(FacturaDirecta)*, les seves paraules i el seu coneixement han sigut claus en tot aquest procés.

En l'apartat més personal vull agrair als meus pares, Josep i Merçe, l'oportunitat que em van donar de poder estudiar el que volia, sempre els estaré agraïts, a la meva germana Marta per estar sempre al meu costat amb un somriure, i sobretot a la meva família: les meves filles, Iris, Nahia i Noa.. i en especial a la meva dona Raquel, que ha tingut d'aguantar-me en els moments mes difícils i sempre m'ha fet costat tot aquest temps, donam-he la confiança necessària per tirar endavant aquest projecte.

8. Webgrafía

Totes les fonts d'informació que s'han utilitzat per la creació del projecte són recursos d'internet, a part d'un *ebook*, i moltes consultes a google i fòrums.

- http://gitnacho.github.io/symfony-docs-es/
- http://librosweb.es/symfony_2_3/
- http://symfony.es
- http://symfony.com
- http://www.doctrine-project.org
- http://www.maestrosdelweb.com
- http://librojquery.com
- http://stackoverflow.com
- http://librosweb.es/ajax/
- http://getbootstrap.com
- http://librosweb.es/pro_git/
- ebook: Desarrollo web ágil con symfony2

“Design is not just what it looks like and feels like. Design is how it works.”
- Steve Jobs -