

UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI
FACULTATEA DE INFORMATICĂ



LUCRARE DE LICENȚĂ

**„Course Manager” - Aplicație
web de management
destinată învățământului
universitar**

Propusă de

Chirica Cosmin

Sesiunea: Iulie, 2017

Coordonator științific
Asist. Dr. Vasile Alaiba

UNIVERSITATEA „ALEXANDRU IOAN CUZA” IAŞI

FACULTATEA DE INFORMATICĂ

„Course Manager” - Aplicație web de management destinată învățământului universitar

Chirica Cosmin

Sesiunea: *iulie, 2017*

DECLARAȚIE PRIVIND ORIGINALITATE ȘI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că lucrarea de licență cu titlul „Course Manager - Aplicație web de management destinată învățământului universitar” este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referința precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
- codul sursă, imagini etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
- rezumarea ideilor altor autori precizează referința precisă la textul original.

Absolvent *Chirica Cosmin*

(semnătura în original)

DECLARAȚIE DE CONSUMĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „Course Manager - Aplicație web de management destinată învățământului universitar”, codul sursă al programelor și celealte conținuturi (grafice, multimedia, date de test etc.) care însotesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iași să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Absolvent *Chirica Cosmin*

(semnătura în original)

Cuprins

1	Introducere	6
1.1	Motivație.....	6
1.2	Descriere	6
1.3	Domenii de utilizare	7
2	Fundamente teoretice	8
2.1	Web/Aplicație web	8
2.2	Mediul de dezvoltare	8
2.2.1	Stiva LAMP.....	8
2.2.2	Composer.....	10
2.2.3	Git și GitHub	10
2.3	Limbajul de programare PHP	11
2.3.1	Ce este PHP?	11
2.3.2	Scurt istoric.....	11
2.3.3	Sintaxă și particularități	12
2.3.4	Framework-uri de PHP.....	13
2.4	Framework-ul Symfony2.....	13
2.4.1	Ce este Symfony?	13
2.4.2	Structura unei aplicații Symfony	14
2.4.3	Structura directoarelor	14
2.4.4	Bundle-uri.....	14
2.4.5	Componente.....	16
2.4.6	Particularități	16
2.4.7	Motive pentru care am ales Symfony	19
2.5	Doctrine ORM	20
3	Analiză și proiectare	21
3.1	Cerințele proiectului	21
3.2	Scenarii de utilizare	23
3.2.1	Logarea utilizatorului	23
3.2.2	Activarea contului utilizatorului.....	24
3.2.3	Adăugarea unui curs la cursuri favorite / Subscribe la un curs	25
3.2.4	Crearea unui curs.....	25
3.2.5	Crearea unui modul de tip seminar.....	26

3.2.6	Crearea unei teme	27
3.3	Analiza cerințelor	29
3.4	Proiectarea modelului aplicației	32
3.5	Proiectarea controlerelor	34
3.6	Proiectarea design-ului aplicației.....	35
4	Implementare.....	36
4.1	Implementarea funcționalităților legate de utilizator.....	36
4.2	Implementarea sistemului ce oferă permisiuni în cadrul aplicației	39
4.3	Implementarea sistemului de notificări	41
4.4	Implementarea view-urilor și al design-ului.....	44
5	Manual de utilizare	45
5.1	Logarea utilizatorului/Editarea profilului	45
5.2	Crearea unui curs	46
5.3	Crearea unui seminar	48
6	Posibilități de îmbunătățire.....	50
	Concluzii.....	51
	Bibliografie.....	52

1 Introducere

1.1 Motivație

În cadrul acestei lucrări propun dezvoltarea unei aplicații web ce urmărește să rezolve o problemă cu care se confruntă în primul rând studenții facultăților și anume felul în care fiecare profesor al facultății are un site personal unde face postări legate de cursul predat.

Cu toții știm că această metoda îngreunează modul de comunicare dintre studenți și profesori deoarece în fiecare semestru trebuie salvate link-uri către site-urile profesorilor, ce automat implică și verificarea periodică a acestora pentru a observa dacă au fost făcute noi postări sau noi anunțuri în cadrul unui anumit curs.

Consider că este importantă realizarea unei aplicații capabile să ofere o clasificare a tuturor cursurilor din cadrul unei facultăți printr-o interfață simplă și ușor de folosit. O asemenea aplicație ar oferi posibilitatea unui utilizator de a accesa mult mai rapid cursul dorit, ar oferi posibilitatea unui utilizator de a fi notificat atunci când există noi postări în cadrul cursurilor dorite și totodată posibilitatea de accesa orice informație a oricărui curs mult mai ușor doar prin intermediul unei căutări cu numele cursului.

Un alt plus pentru o asemenea aplicație ar fi ca aceasta să fie *responsive* deoarece majoritatea studenților utilizează dispozitivele mobile în marea majoritate a timpului și posibilitatea de accesare a platformei ar fi mult mai facilă și mai rapidă.

1.2 Descriere

Proiectul constă într-o aplicație web responsive (ce va putea fi accesată de pe orice dispozitiv ce are conexiune la internet, indiferent de dimensiunile ecranului) ce ofera utilizatorilor o experiență cât mai plăcută și mai ușoară de a comunica prin intermediul unei platforme de management a cursurilor din cadrul învățământului univeristar. Aplicația are rolul de a ajuta un utilizator să găsească cursul dorit cât mai ușor și de a fi la curent cu toate noutățile din cadrul acestuia. Aplicația va găzdui trei tipuri de utilizatori: profesor, profesor asociat și student. Un profesor are capacitatea de a adăuga cursurile predate de acesta iar platforma îi oferă posibilitatea de a administra aceste cursuri destul de ușor. Profesorul are posibilitatea de a contribui cu module asociate acestui curs, unde modulele sunt de 2 tipuri: cursurile susținute de acesta săptămânal și seminarii în cazul în care acesta susține și ore de seminar. În cadrul unui modul de curs/seminar, utilizatorul care are dreptul de a crea acest modul are și posibilitatea de a adăuga unul sau mai multe fișiere iar toți utilizatorii interesați de un modul din cadrul cursului pot contribui cu comentarii atunci când sunt neclarități.

După cum știm în viața reală, în cadrul unui curs colaborează mai mulți profesori. Aplicația oferă autorului cursului posibilitatea de a asocia cursului predat de acesta noi profesori ce vor avea dreptul de a adăuga module de tip seminar. La toate aceste lucruri legate de un curs mai intervine și posibilitatea de a adăuga o temă aferentă unui curs sau a unui modul.

Utilizatorul de tip student are posibilitatea de a face subscribe la anumite cursuri și de a urmări activitatea desfășurată în cadrul acestora pe parcursul întregului an universitar.

Obiectivele principale ale aplicației:

- posibilitatea cofirmării contului creat automat la înscrierea în cadrul facultății;
- logarea utilizatorului;
- completarea profilului și adăugarea de date;
- vizualizarea cursurilor în funcție de an și semestru;
- posibilitatea de a adăuga un curs a unui utilizator de tip profesor;
- posibilitatea asocierii dintre profesori și cursuri;
- posibilitatea de a contribui cu noi module utilizatorilor de tip profesor asociat;
- administrarea ușoară și eficientă a fișierelor din cadrul unui curs;

1.3 Domenii de utilizare

Această aplicație este destinată învățământului universitar oferind posibilitatea unei facultăți de a-și administra cursurile cât și posibilitatea de a oferi o mai bună comunicare între profesori și studenți. Conform celor descrise mai înainte, aplicația este adresată studenților și profesorilor din cadrul oricărei facultăți.

2 Fundamente teoretice

În cadrul acestui capitol vom prezenta aspectele teoretice utilizate în cadrul acestei lucrări cât și ce tehnologii au fost utilizate pentru realizarea acesteia.

2.1 Web/Aplicație web

Pentru început vor fi prezentate câteva detalii sumare despre ce reprezintă web-ul având ca și puncte de reper principalele caracteristici ale acestuia. De asemenea vom oferi și o definiție pentru termenul de web.

Web-ul este cunoscut ca și WWW(World Wide Web) și reprezintă un serviciu al internetului. Elementele cheie ce stau la baza acestuia sunt hypertext-ul, URI-urile și protocolul HTTP.

Hypertext-ul reprezintă un tip de document electronic și conține legături către alte documente de tip hypertext.

URI-ul (Uniform Resource Identifier) este un sir de caractere ce identifică în mod unic o resursă aflată pe web. Un URI este formatat după o anumită schemă. Identificarea resursei se poate face prin două metode. Fie pe baza numelui(URN- Uniform Resource Name), fie pe baza locației (URL - Uniform Resource Locator).

Protocolul HTTP(HyperText Transfer Protocol) este utilizat pentru accesul la conținutul web și este bazat pe modelul client-server.

Prin aplicație web trebuie să înțelegem o aplicație alcătuită dintr-o serie de pagini web interconectate având conținut generat dinamic de pe partea de sever. Această aplicație are rolul de a oferi utilizatorilor o funcționalitate specifică iar în cazul nostru, aplicație este dedicată segmentului universitar având ca și scop de a îmbunătăți metoda de comunicare dintre profesori și studenți.

2.2 Mediul de dezvoltare

Pentru a obține aplicația propusă încă din primele pagini ale acestui document, am utilizat o serie de tehnologii și tool-uri aferente domeniului web. În cele ce urmează vom descrie în detaliu ce tehnologii am utilizat în cadrul aplicației Course Manager, precizând rolul și motivul alegerii făcute.

2.2.1 Stiva LAMP

LAMP este un acronim pentru un set de componente software open-source, utilizate pentru a găzdui și rula aplicații web. Cele 4 componente majore, care fac parte din stiva LAMP, necesare rularii unei aplicații web, sunt: un sistem de operare, un server web, un server de baze

de date și un limbaj de programare pentru a genera conținut în mod dinamic pe partea de server. În majoritatea cazurilor, cele 4 componente sunt reprezentate de Linux (sistem de operare), Apache (server web), MySQL (server de baze de date) și PHP (limbaj de scripting). Acronimul vine de la inițialele numelor aplicațiilor folosite inițial, însă de-a lungul timpului au apărut și alte posibilități de înlocuire a componentelor. Serverul web, Apache, poate fi înlocuit de către alte server open-source precum Nginx sau lighttpd, server-ul de baze de date MySQL poate fi înlocuit cu ușurință de către MongoDB sau PosgresSQL iar pentru limbajul de scripting există înlocuitorii Python, Ruby sau Node.js .

În continuare vom da câteva detalii despre fiecare componentă din stiva LAMP.

Linux este un sistem de operare de tip pentru computere, servere, dispozitive mobile și dispozitive integrate, asemănătoare Unixului, de tip open source și la care lucrează o intreagă comunitate. Acest sistem de operare poate fi utilizat de către toate platformele de calculatoare importante, inclusiv dispozitive cu procesoare x86, ARM și SPARC, ceea ce îl face unul dintre cele mai acceptate sisteme de operare.

Sistemul de operare Linux urmează un design modular ceea ce duce la multe variații și distribuții ale acestuia. Un bootloader este responsabil pentru pornirea kernel-ului Linux. Kernel-ul se află în centrul sistemului Linux, gestionează accesul la rețea, procese sau aplicații de planificare, gestionează dispozitive periferice de bază și supraveghează serviciile sistemului de fișiere. Printre cele mai cunoscute distribuții dezvoltate de comunitate sunt Debian, Slackware sau Gentoo iar dintre cele comerciale Fedora (realizat de Red Hat), openSUSE(de la SUSE) și Ubuntu (de la Canonical).

Apache este un server web disponibil gratuit, distribuit sub licență open source. Versiunea 2.0 rulează pe majoritatea sistemelor de operare bazate pe UNIX (cum ar fi Linux, Solaris, Digital UNIX și AIX), pe alte sisteme derivate de la UNIX / POSIX (precum Rhapsody, BeOS și BS2000 / OSD), pe AmigaOS Windows 2000. Potrivit unui sondaj Netcraft (www.netcraft.com), 60% din toate site-urile web folosesc apache (62% inclusiv derivatele Apache), ceea ce face ca apache să fie utilizat pe scară mai largă decât toate celelalte servere web combinate.

MySQL este un sistem de gestionare a bazelor de date relaționale open source (RDBMS) bazat pe limbajul structurat de interogări (SQL). MySQL rulează pe aproape toate platformele, inclusiv Linux, UNIX și Windows. Deși poate fi utilizat într-o gamă largă de aplicații, MySQL este cel mai adesea asociat cu aplicații bazate pe web.

PHP este un limbaj de scripting și un interpretor disponibil gratuit și utilizat în principal pe serverele web Linux. PHP, derivat inițial din **Personal Home Page Tools**, reprezintă acum PHP: Hypertext Preprocessor, pe care secțiunea de FAQ(Frequently Asked Question) îl descrie ca un "acronym recursiv".

2.2.2 Composer

Composer este un manager de dependințe (dependency manager) pentru librăriile software scrise în limbajul de programare PHP. Funcționalitate de bază a unui package manager este de standardiza felul în care sunt organizate și instalate librăriile PHP. În plus, acesta poate determina dacă o librărie depinde de o altă librărie și are capacitatea de a instala toate aceste dependințe în cadrul proiectului. În cadrul unui proiect ce utilizează framework-ul Symfony, toate aceste librării se găsesc sub directorul `./vendor`. Principalul repository ce indexează toate aceste librării poartă numele de Packagist și este o platformă web online ce permite adăugarea, căutarea și organizarea librăriilor PHP. Composer poate face management la toate aceste pachete prin intermediul unui fișier `composer.json` ce se află în directorul rădăcină iar în interiorul acestuia se află un JSON ce conține toate librăriile împreună cu versiunile instalate. Trebuie precizat faptul că Symfony2 se bazează pe Composer pentru a administra librăriile utilizate în cadrul framework-ului.

Comenzi uzuale:

- `composer install` – instalează dependințele conform fișierului `composer.lock`
- `composer update` - actualizează toate dependințele pentru toate librăriile
- `composer update <vendor/package>` - instalează o nouă librărie, unde `<vendor/package>` reprezintă numele și versiunea luate după Packagist

Utilizarea acestuia în cadrul proiectului Course Manager va fi detaliată în capitolul despre implementare.

2.2.3 Git și GitHub

Git este un sistem de control al versiunilor (**VCS** - Version Control System) și un administrator pentru codul sursă (**SCM** - Source Code Management) al proiectelor software, fiind de tip distribuit și open-source. Fiind un sistem de control al versiunii, acesta are rolul de a ține evidența modificărilor asupra unor seturi de fișiere dintr-un proiect software, de-a lungul perioadei de dezvoltare, astfel încât putem reveni și revedea o anumită versiune dacă dorim. Printre avantajele unui astfel de sistem ar fi că pot lucra mai mulți programatori pe același proiect fără a se încurca între ei și în caz că s-ar produce o greșeală de programare sau s-ar pierde fișiere, se poate reveni întotdeauna la versiunea precedentă. Git poate fi utilizat doar local, pe o singură mașină, având toate funcționalitățile, dar și distribuit, putând fi folosit de o echipă de programatori prin intermediul repository-urilor aflate pe un server extern (remote server).

Acesta stochează modificările (adăugări și ștergeri) în commit-uri care sunt salvate local la început și pot fi apoi încărcate pe server. Sistemul mai oferă suport pentru branch-uri (ramuri) și merge-uire (îmbinare), astfel că un programator poate lucra independent de echipă la o anumită funcționalitate și apoi poate îmbina totul în codul principal. Un exemplu de serviciu de hosting (remote server) pentru Git este GitHub ce va fi descris în paragraful următor.

GitHub este o platformă web utilizată de proiectele software ce utilizează Git ca și sistem de versionare. Aceasta are ca și îmbunătățiri un sistem de issue-uri (probleme ce trebuie rezolvate pentru a îmbunătăți buna funcționare a aplicației), pull request-uri (cereri realizate de către developeri pentru a adăuga modificările realizate de către aceștia în branch-ul principal cu numele de master), comentarii, wiki-uri și un sistem foarte optim de vizualizare a diferențelor dintre branch-ul care se dorește a fi adăugat la proiect și codul ce se află pe branch-ul de master. Un proiect este găzduit de către GitHub într-un repository ce poate fi public (vizibil de către toți utilizatorii platformei) sau privat (vizibil doar de către owner-ul proiectului sau de către toate persoanele ce se află în interiorul unei organizații).

2.3 Limbajul de programare PHP



2.3.1 Ce este PHP?

PHP este un limbaj de scripting pe partea de server, ce a fost conceput pentru a crea pagini web cu conținut dinamic, însă se poate folosi și pentru a crea aplicații desktop, prin intermediul extensiei separate (PHP-GTK). Aceasta este considerat ca primă opțiune, pentru stiva LAMP în fața celorlalte limbaje de programare, Perl sau Python. PHP poate fi instalat pe toate sistemele de operare și platformele majore (Linux, Mac OS X, Solaris OS, Windows, etc.) și reprezintă un exemplu de software liber, emis sub licență sa proprietară (PHP license). Astfel, acesta este instalat pe o mare parte dintre serverele web actuale.

Numele PHP provine din limba engleză și este un acronim recursiv: „**P**hp: **H**ypertext **P**reprocessor”, având această semnificație de la versiunea a 3-a, în timpul primelor 2 versiuni, PHP fiind acronim pentru „Personal Home Page”.

2.3.2 Scurt istoric

Acesta a fost creat inițial de către **Rasmus Lerdorf**, un programator danez, care este de asemenea autorul primelor două versiuni și a participat apoi în dezvoltarea versiunilor ulterioare. Acesta a început dezvoltarea PHP în anul 1994, când a scris o serie de scripturi **CGI**(Common Gateway Interface) în Perl, pentru a-l ajuta să-i genereze CV-ul (curriculum vitae) online și să monitorizeze traficul de pe pagina sa personală. Apoi a îmbunătățit aceste scripturi, rescriindu-le în C pentru performanță mai bună, extinzându-le cu suport pentru formulare și a numit această versiune „**P**ersonal **H**ome **P**age/**F**orms **I**nterpreter” sau PHP/FI.

Un moment important în istoria PHP a fost în anul 1997, când **Zeef Suraski** și **Andi Gutmans** au rescris parserul de PHP și au pus bazele viitoarei versiuni PHP 3, versiune de la care au și schimbat numele în acronimul recursiv „**PHP: Hypertext Preprocessor**”.

În iunie 1998 a fost lansat PHP 3.0 iar mai apoi au început să rescrie nucleul PHP și să creeze Zend Engine în anul 1999 care urma să reprezinte baza PHP 4 ce a fost lansat în anul 2000.

În 2004 a fost lansat PHP 5 care se bazează pe Zend Engine 2.0 și a introdus noi funcționalități, cum ar fi: suport pentru POO (programare orientată-obiect), extensia PDO (PHP Data Objects) - care reprezintă o interfață pentru a accesa diferite baze de date, plus alte îmbunătățiri la nivel de performanță.

În anul 2005 se dorea începerea lucrului la versiunea 6, însă a fost amânată și majoritatea funcționalităților importante au fost adăugate în cadrul versiunilor PHP 5.3 (namespace-uri, closures etc), PHP 5.4 (traits, îmbunătățiri de performanță) etc.

Versiunea curentă de PHP este 7.1.5 și aduce îmbunătățiri pentru performanță, manipularea erorilor, suport pentru sisteme windows pe 64-bit, declarațiile de tip exacte, adăgarea claselor anonime și multe altele.

2.3.3 Sintaxă și particularități

PHP are o sintaxă foarte asemănătoare cu limbajul de programare C, adăugând în plus multe funcționalități necesare dezvoltării aplicațiilor web. Principalele instrucțiuni de control sunt cele de decizie (**if**, **switch**) și cele repetitive (cu număr cunoscut de pași - **for**, cu număr necunoscut de pași - **while**, **do while**). Codul PHP poate fi înglobat în cadrul documentelor HTML sau poate exista în fișiere separate, care conțin numai cod PHP și au de obicei extensia „.php”. Pentru a fi executat de interpretor, codul trebuie să fie inclus între tag-urile PHP, care cel mai adesea pot fi de forma: „<?php ... ?>”, fiind disponibile și alte tipuri de taguri în funcție de modul cum a fost configurat PHP. Un alt element de sintaxă, specific PHP-ului este faptul că variabilele trebuie prefixate cu simbolul „\$”, iar tipul variabilei nu trebuie specificat. PHP este un limbaj „weak typed” sau „dinamically typed”, adică tipul variabilelor se verifică la runtime. PHP oferă de asemenea și numeroase funcții predefinite pentru orice tip de procesare a datelor dorită: prelucrare a tipurilor de date, matematice și de conversie, de manipulare a sirurilor de caractere, de prelucrare a array-urilor (tablourilor), timp și dată, de access la resurse și lucrul cu fișiere, pentru manipulare a URL-urilor, de manipulare a bazelor de date, pentru accesarea resurselor (XML, PDF, JPEG), specifice sistemului de operare, etc.

De asemenea oferă și suport pentru interacțiune web, funcții pentru procesarea *cookie-urilor* și a *sesiunilor*, dar și o serie de array-uri disponibile oriunde în cadrul codului PHP (numite „Superglobals arrays” - tablouri super-globale), cum ar fi - `$_GET`, `$_POST`, `$_REQUEST`, `$_SESSION`, etc.

2.3.4 Framework-uri de PHP

Pentru a ușura munca programatorului dar și pentru a îmbunătăți eficiența și calitatea produselor, de-a lungul timpului au apărut o multitudine de framework-uri bazate pe PHP. Dintre cele mai cunoscute și folosite în ziua de astăzi sunt: Laravel, Symfony, Zend, CodeIgniter, CakePHP etc. În cadrul proiectului am folosit framework-ul Symfony2 iar în secțiunea ce urmează vom face o introducere în acest framework de PHP.

2.4 Framework-ul Symfony2



2.4.1 Ce este Symfony?

Symfony este un framework urmărind modelul MVC (model-view-controller) realizat în limbajul de programare PHP. Acest framework este open-source, apărut în varianta inițială cu mai mult de 10 ani în urmă (octombrie 2005) și a devenit unul dintre cele mai populare cadre de lucru PHP datorită multiplelor facilități și a bunei documentații.

Acest framework este bazat pe un set de componente decuplate și ce sunt la rândul lor reutilizabile în alte aplicații de top cum ar fi Drupal, phpBB, eZ Publish.

În spatele acestui framework există compania **SensioLabs**. Inițial creat pentru propriile nevoi, acest framework este în continuare instrumentul zilnic folosit pentru a dezvolta proiectele clienților de către echipele de dezvoltare ale firmei. Pe lângă firma SensioLabs ce se ocupă de menținerea și îmbunătățirea acestui framework, există o întreagă comunitate activă ce se ajută reciproc și numeroase companii investesc în îmbunătățirea lui.

Symfony este un framework de aplicații web de tip open-source, realizat în limbajul de programare PHP. Aceasta reprezintă un set de componente software prefabricate, ce pot fi rapid integrate în orice proiect și o serie de metodologii de dezvoltare, ce asigură stabilitatea, menținerea și decuplarea componentelor aplicațiilor, dezvoltate cu acesta.

Pentru numerotarea versiunilor de Symfony se folosește sistemul numit „Semantic versioning” de forma: MAJOR.MINOR.PATCH. În prezent, Symfony a ajuns la a treia versiune majoră și anume 3.0. Conform celor de la SensioLabs, în acest moment există 3 versiuni recomandate și anume: Symfony **2.8.20** (recomandat pentru proiectele ce au nevoie de

suport pentru o foarte mare perioadă de timp), Symfony **3.2.8** (versiunea curentă, stabilă și care are suport până la sfârșitul lui Ianuarie 2018) și Symfony **3.3.0-RC1** (utilizat pentru proiectele de test ce vor folosi următoarele versiuni ale acestui framework).

2.4.2 Structura unei aplicații Symfony

Vom începe prin a prezenta structura directoarelor unui proiect Symfony, iar apoi vom detalia modul cum este organizat intern framework-ul în compoziție și bundle-uri.

2.4.3 Structura directoarelor

Structura de directoare a unei aplicații Symfony este foarte flexibilă, însă structura recomandată pentru un proiect și pe care o respectăm și noi în cadrul aplicației este:

- **app/** - director ce conține fișiere de configurare, template-uri și traduceri;
- **bin/** - director ce conține fișierele executabile;
- **src/** - director ce conține codul PHP al proiectului;
- **test/** - director ce conține testele automate cum ar fi unit teste;
- **var/** - director ce conține fișiere generate automat cum ar fi fișierul de cache, loguri etc.
- **vendor/** - director ce găzduiește framework-ul Symfony cât și alte librării și bundle-uri third-party;
- **web/** - directorul rădăcină al proiectului;

Directorul **web** conține toate resursele publice necesare afișării unei pagini web cum ar fi imagini, fișiere de tip CSS sau fișiere de tip JavaScript. De asemenea, în cadrul acestui folder se mai găsesc și fișierele de configurare denumite „front controller” ce au rolul de a face management la fiecare request realizat către aplicație.

Mai trebuie precizat faptul că în cadrul directorului **app/**, care conține fișierele de configurare, se mai găseste și o clasa denumită AppKernel ce joacă rol de punct principal de acces al aplicației. Tot în această clasă sunt înregistrate și modulele din care este formată aplicația. Aceste module poartă numele de **bundle-uri** despre care vom da mai multe detalii în subcapitolele următoare.

2.4.4 Bundle-uri

Această secțiune introduce una dintre cele mai mari și mai puternice caracteristici ale acestui framework și anume sistemul de pachete.

Un bundle(pachet) este ca un fel de plugin din cadrul altor aplicații software. De ce se numește bundle și nu plugin? Acest lucru se datorează simplului fapt că în Symfony absolut

totul este un bundle, de la core-ul framework-ului până la codul scris de noi ca și programatori în cadrul aplicației. Tot codul scris pentru aplicație este organizat în unul sau mai multe bundle-uri. În Symfony, un bundle este un set structurat de fișiere de tip PHP, CSS, JavaScript, imagini etc. și care pot fi ușor partajate cu alți dezvoltatori.

Un bundle din Symfony merge pe principiu „first-class citizens”. Acest lucru înseamnă că fiecare funcționalitate este realizată prin intermediul unei clase. Un avantaj major al acestui principiu este faptul că se oferă flexibilitatea de a utiliza funcții din bundle-uri third-party fie de a distribui propriile bundle-uri. Oferă posibilitatea de a alege și activa caracteristicile dorite în cadrul aplicației în funcție de necesități.

Atunci cand se realizează un proiect nou Symfony, framework-ul generează automat un bundle denumit AppBundle. Framework-ul Symfony la nivel intern este compus din mai multe bundle-uri și anume:

- *FrameworkBundle* – are rolul de a lega toate componentele ce vor fi descrise în subcapitolul următor;
- *SecurityBundle* – oferă suport pentru securizarea aplicației;
- *TwigBundle* – este un template engine utilizat în implementarea interfeței aplicației;
- *MonologBundle* – menține logurile aplicației;
- *SwiftmailerBundle* – facilitează manipularea email-urilor în cadrul aplicației;
- *DoctrineBundle* – oferă suport pentru lucrul cu bazele de date;
- *SensioFrameworkExtraBundle* – implementează pattern-ul MVC (Model-View-Controller)

Mai trebuie precizat faptul că un bundle trebuie să respecte o structură de foldere precum:

- *Command* – conține fișisere PHP ce joacă rolul unor comenzi ce pot fi executate de la linia de comanda (CLI)
- *Controller* – conține toate Controllere-le aplicației. Fișierele pot fi organizate un subdirectoare;
- *DataFixtures* – în acest folder regăsim clase ce populează baza de date cu informații inițiale doar atunci cand este necesar;
- *Entity* – modelul este reprezentat sub forma unor clase/entități ce descriu fiecare tabelă a bazei de date;
- *Form* – folder ce conține toate formularele din cadrul aplicației;
- *Repository* – conține toate repositories/depozitele prin intermediul cărora putem interoga baza de date și extrage informațiile dorite;
- *Security* – conține clase ce se ocupă cu securitatea aplicației;
- *Services* – locul în care gasim funcționalitățile aplicației împărțite pe servicii;
- *Tests* – folder ce conține testele automate scrise de obicei în PHPUnit¹;

¹ PHPUnit: Framework specific limbajului de programare PHP și folosite la teste de tip unit

Un bundle trebuie să respecte structura descrisă mai sus cu posibilitatea de a adăuga noi foldere în funcție de necesități.

2.4.5 Componente

O componentă Symfony reprezintă o librărie software care conține cod scris în PHP, ce implementează anumite funcționalități necesare dezvoltării aplicațiilor web. Aceste componente reprezintă fundația framework-ului Symfony2 oferind libertatea de a implementa arhitectura aplicației după nevoile specifice cerințelor proiectului și de a utilizat doar componentele de care este strictă nevoie.

Dintre componentele mai importante putem enumera:

- **Form** – asigură creare, procesare și reutilizare a formularelor HTML printr-un procedeu simplificat
- **HttpFoundation** - interfață orientată-obiect de a lucra cu protocolul HTTP
- **HttpKernel** – unelte ce stau la baza un framework web, bazat pe HTTP
- **DependencyInjection** – standardează și centralizează modalitatea prin care sunt create obiectele
- **EventDispatcher** – facilitează comunicare între module prin intermediul evenimentelor și listener-elor
- **Routing** - mapează un request HTTP la un set de variabile preconfigurate
- **Security** - asigură un sistem de securitate asupra resurselor, prin autentificare și ACL² (Liste de Control al Accesului)
 - **Templating** - furnizează unelte necesare dezvoltării unui sistem de template-uri
 - **Yaml** - parsează fișiere (de configurare) YAML și le convertește în array-uri PHP

2.4.6 Particularități

Principala caracteristică a framework-ului Symfony2 este că arhitectura acestuia este decuplată, proiectele fiind aclătuite din bundle-uri. Conform documentației oficiale, este recomandat ca arhitectura aplicației să fie gândită și împărțită pe mai multe servicii. Acest lucru ne spune că Symfony2 nu utilizează pattern-ul MVC sub forma clasică, ci sub o formă puțin modificată deoarece este bazat pe servicii și fiind un framework de tip cerere-răspuns. Prin urmare, toată logica din cadrul unui controller trebuie împărțită în mai multe servicii, în primul rând pentru că acel cod poate fi refolosit și în alte părți ale aplicației. Acest lucru va transforma un controller într-o funcție mapată la o anumită rută, având ca și scop primirea unei cereri. Mai

² ACL: Access Control List

departe, acesta trimite către servicii datele necesare pentru procesare urmând ca apoi să returneze datele dorite în cadrul unui view pentru a fi afișate utilizatorului final.

Un alt plus al acestui framework este acela că dispune de o comunitate foarte mare ce pune la dispoziția dezvoltatorilor o multitudine de librării open-source. Mai mult, chiar în distribuția de bază Symfony vom întâlni librării open-source precum Doctrine ORM și Twig.

Doctrine are rolul de a mapa înregistrările din baza de date la obiecte PHP. Acest lucru se realizează prin intermediul entităților ce conțin proprietăți, iar fiecare proprietate are asociată o metodă de *set* și o metodă de *get* pentru a putea manipula informațiile din cadrul acestora. Maparea este realizată prin intermediul metadatelor și modalitatea utilizată în cadrul proiectului este de a folosi adnotări PHP. Aceste adnotări descriu relația dintre proprietățile clasei și coloanele din fiecare tabelă, ajută la descrierea relațiilor dintre tabele și totodată sunt și foarte ușor de vizualizat, manipulat.

Mai multe exemple de posibilele adnotări Doctrine, se pot vedea în următoarele capturi de ecran:

```
/**
 * Course
 *
 * @ORM\Table(name="course")
 * @ORM\Entity(repositoryClass="AppBundle\Repository\CourseRepository")
 * @UniqueEntity(fields="title", message="unique.title")
 * @UniqueEntity(fields="abbreviation", message="unique.abbreviation")
 */
class Course
```

Figură 1: Adnotări specifice unei clase ce definește o tabelă din baza de date

```

* @ORM\ManyToMany(targetEntity="User", inversedBy="labs")
* @ORM\JoinTable(
*     name="course_assistant",
*     joinColumns={
*         @ORM\JoinColumn(name="course_id")
*     },
*     inverseJoinColumns={
*         @ORM\JoinColumn(name="assistant_id")
*     }
* )
*/
private $assistants;

```

Figură 2: Adnotări specifice unei proprietăți ce definiește o coloană din tabelă

Twig este un produs realizat tot de către SensioLabs. Acesta este un template engine open-source pentru PHP și are rolul de a afișa în cadrul template-urilor (fișiere ce descriu informațiile vizualizate de către utilizator) datele trimise de la nivelul de model al aplicației, date ce vor fi vizualizate de către utilizatorii finali ai aplicației.

Exemplu de sintaxă Twig:

```

{% if not courses|length > 0 %}
    <div class="row">
        {% if constant('AppBundle\\Entity\\User::ROLE_PROFESSOR') in app.user.roles %}
            <h3>{{ 'heading.professor_no_courses'|trans }}</h3>
            <small>{{ 'message.professor_courses'|trans }}</small>
        {% elseif constant('AppBundle\\Entity\\User::ROLE_ASSOCIATE') in app.user.roles %}
            <h3>{{ 'heading.assciate_no_courses'|trans }}</h3>
            <small>{{ 'message.associate_courses'|trans }}</small>
        {% else %}
            <h3>{{ 'heading.not_subscribed'|trans }}</h3>
            <small>{{ 'message.subscribe_to_course'|trans }}</small>
        {% endif %}
    </div>
{% else %}

```

Figură 3: Exemplu de cod specific Twig

2.4.7 Motive pentru care am ales Symfony

Motivul pentru care am ales sa utilizez acest framework a fost acela de a avea la final o aplicație foarte bine structurată și tot odată să ofere o performanță și o experiență plăcută utilizatorului final.

Symfony se poate număra printre cele mai cunoscute și cele mai bune framework-uri de web deoarece oferă o **performanță ridicată**, oferă o **documentație foarte bine structurată și explicată** iar prin intermediul acestuia aplicația finală poate beneficia de orice funcționalitate implementată într-un mod decuplat. În plus, în cadrul proiectului putem beneficia de bundle-uri third-party care micșorează volumul de lucru iar prin intermediul unui bundle din cadrul proiectului putem ajuta și noi la rândul nostru alți dezvoltatori.

2.5 Doctrine ORM



Doctrine este un proiect ce are la bază mai multe librării de tip PHP și este axat în primul rând pe stocarea bazei de date cât și pe maparea acesteia în cadrul unor obiecte de tip PHP. Proiectele principale pe care se bazează Doctrine sunt **ORM(Object Relational Mapper)** și **DBAL(Database Abstraction Layer)**. De asemenea Doctrine beneficiază de concepte precum **Hibernate ORM** și a reușit să adapteze toate aceste concepte pentru a fi introduse în PHP.

Un ORM are rolul de a face legătura dintre tipurile de date puse la dispoziție de către un limbaj orientat-obiect și tipurile de date ale unei baze de date relaționale. Un beneficiu adus de către ORM este acela că dezvoltatorii aplicației nu mai sunt nevoiți să interacționeze cu comenzi de SQL. În schimb, aceștia vor lucra cu obiecte specifice limbajului de programare, unde o clasă va semnifica o intrare dintr-o tabelă din baza de date. Un alt beneficiu adus de către Doctrine ORM este acela că aduce o nouă metodă de interogare a bazei de date ce poartă denumirea de **DQL(Doctrine Query Language)**. Aceasta permite utilizarea obiectelor și a câmpurilor acestora în interogări, ne mai fiind nevoie de a lucra direct cu tabele și coloanele acestora.

DBAL sau **Database Abstraction Layer** este un API (Application Programming Interface) și are rolul de a crea o interfață comună între o aplicație și toate sistemele de gestiune a bazelor de date precum MySQL, MongoDB, etc.

Așa cum am menționat în subcapitolul anterior Doctrine este preinstalat și integrat în cadrul framework-ului Symfony. În cadrul proiectului, Doctrine ORM dispune de două directoare și anume `/Entity` și `/Repository`. Directorul `/Entity` conține clase PHP denumite și entități și au rolul de a mări obiectele la tabele din baza de date. Posibilitățile mapare sunt mai multe și anume: anotări PHP, YAML sau XML. Proiectul CourseManager utilizează anotările PHP în cadrul entităților. Directorul `/Repository` conține clase repository ce joacă rol de deposit iar prin intermediul acestora se realizează operațiile peste baza de date utilizând DQL.

3 Analiză și proiectare

Pe parcursul acestui capitol vom analiza cerințele și ceea ce dorim să obținem în cadrul acestui proiect, vom descrie diverse scenarii de utilizare pentru a vedea aplicația mai bine din punctul de vedere al utilizatorului și vom identifica principalele module ale aplicației. De asemenea vom proiecta în detaliu modulele aplicației în funcție de cerințele acesteia.

3.1 Cerințele proiectului

Sintetizând cele menționate în primul capitol unde am descrie în mare aplicația, putem spune că în cadrul acestui proiect trebuie să realizăm o platformă web al cărei scop este de a ajuta utilizatorul final, și anume profesorii și studenții prezenți în cadrul învățământului universitar, de a comunica și a împărtăși toate informațiile și resursele necesare în cadrul cursurilor dintr-o facultate. Utilizatorul are posibilitatea de a face management mult mai ușor la cursurile predate, urmările sau în cadrul cărora contribuie ca și profesor asociat. Pe lângă platformă oferită utilizatorilor finali, mai există și o interfață dedicată adminilor sistemului ce are ca și scop principal de a face management asupra conturilor utilizatorilor la care se mai adaugă mici funcționalități ce vor fi descrise în paginile ce urmează.

Plecând de la cerințele de bază enunțate mai sus, acestea pot fi extinse și grupate conform următoarei scheme:

- **funcționalități legate de contul utilizatorului:**
 - logarea în cadrul contului;
 - recuperarea contului în cazul în care utilizatorul a uitat parola;
 - resetarea parolei;
 - editarea profilului vizibil în cadrul platformei;
- **funcționalitățile unui cont de tip student:**
 - posibilitatea de a căuta toate cursurile din cadrul facultății;
 - posibilitatea de a face subscribe către un curs dorit ce implică automat primirea de notificări atunci când un anunț sau un modul de tip curs a fost adăugat;
 - posibilitatea de a viziona activitatea profesorilor asociați unui curs;
 - posibilitatea de a vizualiza modulele de tip curs sau seminar, temele și opțiunea de a lăsa un comentariu în cadrul acestor pagini;

- **funcționalitățile unui cont de tip profesor:**
 - un cont de tip profesor moștenește funcționalitățile unui cont de tip student cu excepția faptului că acesta nu poate face subscribe la un anumit curs;
 - posibilitate de a adăuga sau a elimina un profesor asociat în cadrul cursului predat de acesta;
 - posibilitate de a adăuga noi module de tip curs sau seminar;
 - posibilitatea de a adăuga o nouă temă fie în cadrul cursul, fie pentru un anumit seminar;
 - posibilitatea de a încărca resurse în cadrul fiecărui modul, fie în cadrul cursului;
 - atunci când un profesor este adăugat ca și profesor asociat în cadrul unui curs, acesta are posibilitatea de a adăuga noi module de tip seminar sau teme aferente seminarelor predate de acesta;
 - la toate funcționalitățile de mai sus ce implică acțiunea de a adăuga ceva nou evident ca autorul are și posibilitatea de editare sau ștergere;
- **funcționalitățile unui cont de tip profesor asociat:**
 - diferența majoră dintre un cont de tip profesor și un cont de tip profesor asociat este funcția acestuia din cadrul facultății. Prim urmăre, acesta are aceleași funcționalități ca și un cont de tip profesor cu mențiunea că acesta nu are posibilitatea de a face modificări asupra cursurilor și asupra datelor ce nu au fost create de acesta;
- **funcționalitățile unui cont de tip admin:**
 - un admin are accesibile în cadrul interfeței dedicată strict adminilor;
 - unui admin îi revine obligația de a crea conturi pentru studenți și profesori întrucât interfața dedicată acestora nu oferă această posibilitate; acest lucru se poate realiza printr-un formular în urma căruia va fi generat un nou cont;
 - un admin mai are posibilitate de a adăuga/edita cursuri, module de tip curs/seminar și anunțuri

Acum că am detaliat cerințele aplicației, avem o imagine mai clară despre ceea ce trebuie să obținem la final și ne va fi de folos în procesul de proiectare. În continuare vom descrie scenariile de utilizare pentru a vedea exact modul în care utilizatorul final va interacționa cu aplicația noastră.

3.2 Scenarii de utilizare

În cele ce urmează vom elabora mai multe scenarii de utilizare a aplicației ce ne vor exemplifica felul în care va funcționa aplicația cât și felul în care aceasta va fi folosită de către un utilizator.

Pentru descriere aceste scenarii ne vom folosi de diagramele UML. **UML**(Unified Modeling Language) este un limbaj de modelare ce este folosit pentru a dezvolta produse software, mai exact ilustrează felul în care este proiectată aplicația. Diagramele UML pot fi clasificate în mai multe tipuri precum diagrame de structură, diagrame comportamentale și diagrame de interacțiuni. În cadrul acestui subcapitol vom utiliza diagramele de tip comportamental numite și diagrame **Use Case**. Acestea prezintă funcționalitățile sistemului din punctul de vedere al utilizatorului final și sunt alcătuite din: actori (entități externe cu care sistemul interacționează), use case-uri (funcționalități ale sistemului) și relații (dependeștele dintre componente).

În continuare vom prezenta mai multe scenarii care descriu modul în care aplicația poate fi folosită de către un utilizator și felul în care aceștia vor interacționa.

3.2.1 Logarea utilizatorului

1. Obiectiv/Context

1. Utilizatorul aplicației accesează aplicația prin intermediul unui browser web și se află pe pagina de login având rolul de utilizator anonim deoarece acesta nu a intrat în contul său.
2. Pentru început, aplicația afișează un formular de login alcătuit din două câmpuri: adresa de email și parola. Pentru a se autentifica, acesta este nevoie să completeze cele două câmpuri.
3. Pe lângă formularul de login mai apare și opțiunea de recuperare parolă pentru cazul în care utilizatorul și-a uitat parola, fie opțiunea de activare cont utilizată de noi utilizatori ai platformei.

2. Scenariu/Pași

1. Utilizatorul introduce datele cerute în câmpurile din formularul de login și anume email și parolă.
2. Dacă datele introduse de utilizator sunt invalide se va afișa un mesaj de eroare referitor la câmpurile completate incorrect și va fi rugat să introducă din nou aceste date.

3. Diagrama Use Case



Figură 4: Diagramă UML pentru logarea utilizatorului

3.2.2 Activarea contului utilizatorului

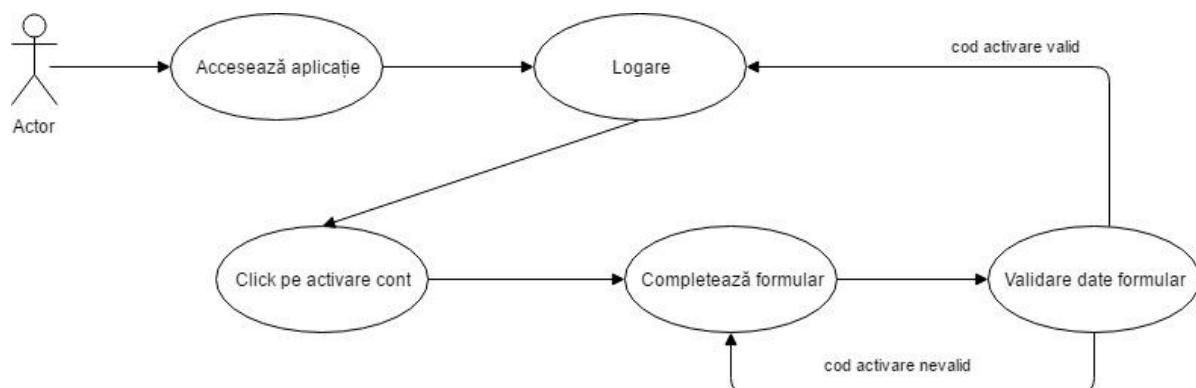
1. Obiectiv/Context

1. Un utilizator a accesat aplicația web și dorește să își activeze contul creat de către administratorul aplicației. Această metodă este o alternativă a accesării URL-ului trimis prin email către utilizator.

2. Scenariu/Pași

1. Utilizatorul va accesa pagina web prin intermediul link-ului afișat pe pagina de login a aplicației.
2. În cadrul paginii va fi afișat un formular cu un singur câmp utilizat pentru a introduce codul unic dedicat utilizatorului.
3. În cazul în care codul introdus nu este corect se va afișa un mesaj de eroare urmând ca utilizatorul să reintroducă codul.
4. După ce utilizatorul își activează contul, acesta va fi redirectat către pagina de login unde se va putea autentifica cu datele personale de autentificare.

3. Diagrama Use Case



Figură 5: Diagramă UML pentru activarea contului

3.2.3 Adăugarea unui curs la cursuri favorite / Subscribe la un curs

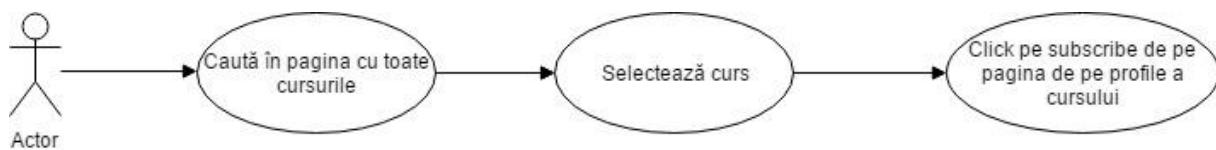
1. Obiectiv/Context

1. Un utilizator este autentificat în cadrul aplicației și are posibilitatea de a căuta un curs din cadrul facultății pentru a face subscribe/adăuga la favorite un curs al unui anumit profesor.
2. Acest scenariu este valabil doar pentru utilizatorii de tip student.

2. Scenariu/Pași

1. Utilizatorul se află pe pagina dedicată cursurilor.
2. Din cadrul acestei pagini acesta poate accesa sub-meniul Explore courses unde va găsi o listă cu toate cursurile din cadrul facultății.
3. Aici are posibilitate de filtrare a cursurilor după autor sau după perioada în care se desfășoară cursul.
4. Atunci când a găsit cursul dorit, utilizatorul selectează numele acestuia și va fi trimis către pagina cursului. Pe pagina cursului acesta are posibilitate de a face subscribe la acest curs ce implică faptul că va primi notificări legate de noile module de tip curs cât și notificări legate de anunțurile acestuia.

3. Diagram Use Case



Figură 6: Diagramă UML pentru subscribe la un curs

3.2.4 Crearea unui curs

1. Obiectiv/Context

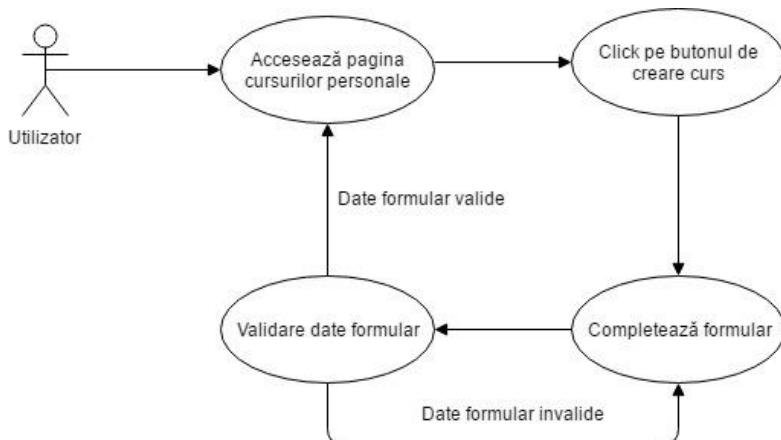
1. Un utilizator cu rolul de profesor este autentificat în cadrul aplicației și dorește să creeze un nou curs.

2. Scenariu/Pași

1. Utilizatorul cu rolul de profesor se află pe pagina unde se află listate toate cursurile personale.
2. Acesta are în cadrul acestei pagini un buton poziționat în dreptul filtrelor dedicate cursurilor prin intermediul căruia poate crea un curs nou.

3. Accesează pagina de creare curs nou prin selectarea acelui buton și va fi trimis pe o pagină ce conține un formular cu câmpurile necesare creării unui curs nou.
4. Utilizatorul trebuie să completeze câmpurile din formular, unele dintre ele fiind obligatorii. În cazul în care acesta omite să completeze ceva, va fi redirecționat pe aceeași pagină iar erorile vor fi afișate în dreptul fiecărui câmp.
5. După ce a creat noul curs, acesta va fi redirecționat către pagina cu toate cursurile personale având posibilitatea de a accesa fiecare curs.

3. Diagrama Use Case



Figură 7: Diagramă UML pentru crearea unui curs

3.2.5 Crearea unui modul de tip seminar

1. Obiectiv/Context

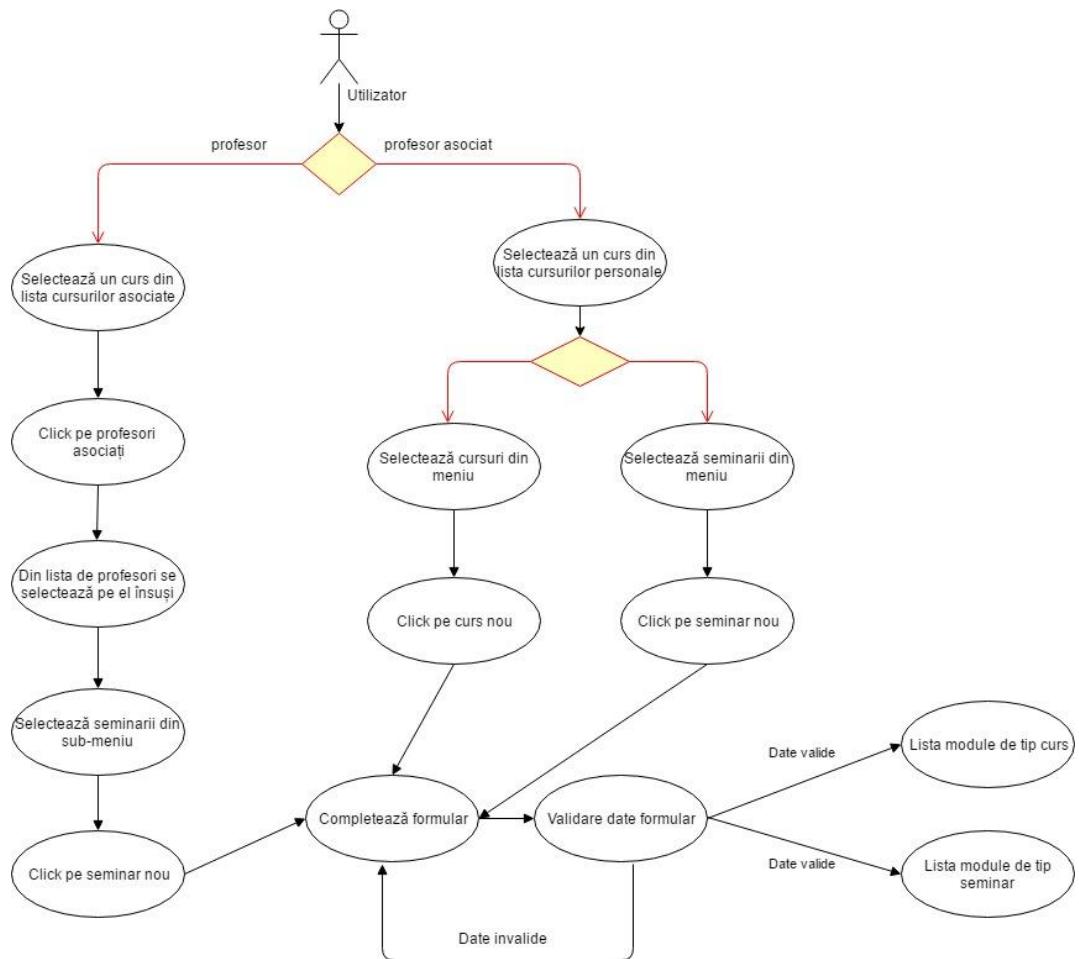
1. Un utilizator cu rolul de profesor sau de profesor asociate dorește să adauge un modul de tip curs sau seminar în cadrul unui curs.

2. Scenariu/Pași

1. Utilizatorul se află pe orice pagină a aplicației web. Acesta trebuie să acceseze meniul din partea stânga sus și apoi secțiunea Courses.
2. Următorul pas diferă în funcție de rol. În cazul în care utilizatorul are rolul de profesor, acesta trebuie să acceseze meniul de Courses pentru a adăuga un modul de tip curs, fie meniul de Seminars pentru a adăuga un modul de tip seminar. În cazul în care utilizatorul are rolul de profesor asociat, primul pas este de a accesa meniul de Associate professors. Urmează apoi ca acesta să se aleagă pe sine însuși din lista de profesori asociați urmând să apară un sub-meniu alcătuit din Profile, Seminars, Homework, Announcements. Conform acestui sub-meniu, un profesor asociat are posibilitatea de a adăuga doar un modul de tip seminar.

3. Următorul pas este identic în funcție de opțiunea aleasă și anume de a completa un formular cu datele necesare pentru a crea un nou modul de tip seminar sau curs.
4. Dacă câmpurile obligatorii ale formularelor nu sunt completate, utilizatorul va fi redirectionat către aceeași pagină unde vor fi afișate erorile fiecărui camp în parte.
5. Dacă un modul de tip curs/seminar a fost realizat cu succes, utilizatorul va fi redirectionat către lista de module aferente profesorului din cadrul cursului.

3. Diagrama Use Case



Figură 8: Diagramă UML pentru crearea unui modul de tip seminar

3.2.6 Crearea unei teme

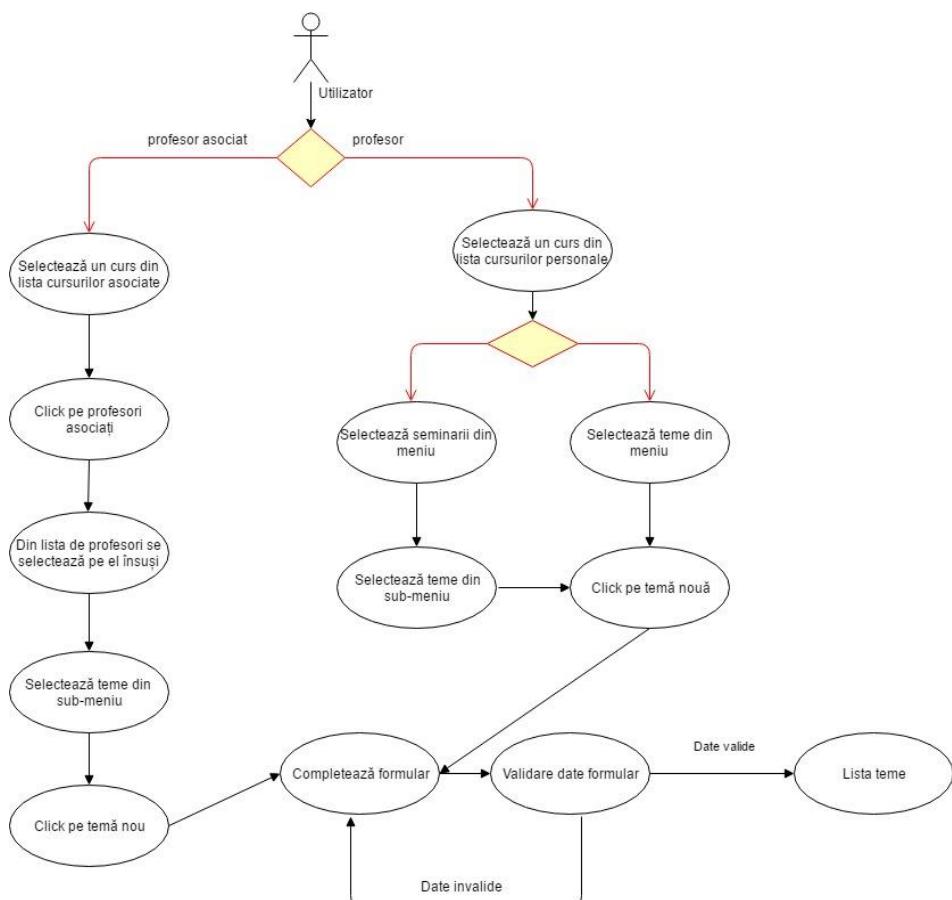
1. Obiectiv/Context

1. Un utilizator cu rolul de profesor sau de profesor asociat dorește să adauge o nouă temă în cadrul cursului sau în cadrul unui seminar.

2. Scenariu/Pași

1. Un utilizator ce are posibilitatea de a adăuga o temă și anume un profesor sau un profesor asociat trebuie să selecteze un curs precum a fost descris în scenariul anterior.
2. Următorul pas este în funcție de rolul utilizatorului. Dacă utilizatorul este autorul cursului, acesta are posibilitatea de a adăuga teme aferente cursului, fie teme în cadrul seminariilor în cazul în care acesta ține și seminarii. Dacă utilizatorul este doar un profesor asociat aceluiași curs, acesta are posibilitatea de a adăuga teme doar aferente seminariilor predate de acesta.
3. În funcție de rol și de ce temă a ales utilizatorul să creeze, acesta va fi redirecționat către o pagină cu un formular pentru a crea o temă.
4. Formularul trebuie completat iar în cazul în care utilizatorul a omis un câmp va fi redirecționat către aceeași pagină în care vor fi afișate erorile corespunzătoare acestuia.
5. După ce tema a fost creată, utilizatorul este redirecționat către lista cu temele personale în funcție de rol și de tipul temei.

3. Diagrama Use Case



Figură 9: Diagramă UML pentru crearea unei teme

3.3 Analiza cerințelor

În acest subcapitol vom descrie modulele principale ale aplicației și felul în care vom grupa funcționalitățile acesteia în cadrul fiecărui modul.

Precum am menționat într-un capitol anterior, aplicația noastră este scrisă în PHP utilizând un framework destul de puternic ce poartă numele de Symfony2. Codul sursă va fi structurat conform documentației oficiale Symfony ceea ce înseamnă că aplicația va fi alcătuită din unul sau mai multe *bundle-uri*. Un bundle este de fapt un director alcătuit la rândul său din mai multe directoare și fișiere de tip PHP, HTML, CSS, Javascript etc. Scopul unui bundle este de a implementa un modul al aplicației și asigură o funcționalitate a aplicației.

Pentru denumirea unui bundle am respectat standardele Symfony astfel încât numele este format dintr-un prefix, care în cazul nostru va purta denumirea de *App*. Aceste director se va afla în directorul */src* al directorului rădăcină al proiectului iar numele acestuia va fi postfixat cu *Bundle*.

Pornind de la cerințele și funcționalitățile aplicației descrise în paginile anterioare, aplicația va fi alcătuită dintr-un singuri bundle Symfony și anume:

- **AppBundle**
 - acesta este localizat la calea : *src/AppBundle*;
 - acest bundle va conține funcționalitățile interfeței dedicate utilizatorilor finali : profesori și studenți. În cadrul acestui bundle vom defini modele aplicației ce vor fi folosite și de *bundle-ul* dedicat adminilor;
 - pe lângă funcționalitățile dedicate utilizatorilor finali, aplicație conține și o interfață dedicată unui utilizator de tip admin. Cele două interfețe sunt dependente una de celalătă deoarece folosesc aceeași bază de date, acesta fiind și motivul amplasării acestora în cadrul același bundle;

Pe lângă acest bundle creat de noi în cadrul aplicației vom mai utiliza și alte *bundle-uri* open source dezvoltate de comunitatea Symfony. Acestea au rolul de a implementa anumite funcționalități des întâlnite în cadrul proiectelor ușurându-le munca programatorilor și oferindu-le acestora ocazia de a se focaliza strict asupra funcționalităților principale ale aplicației dezvoltate de către ei.

Bundle-urile open source folosite în cadrul proiectului sunt:

- **FOSUserBundle**
 - asigură funcționalitățile de bază pentru managementul utilizatorilor și anume înregistrare, autentificare, profil și integrare cu email
 - resurse:
<https://github.com/FriendsOfSymfony/FOSUserBundle>
- **SncRedisBundle**
 - acest bundle integrează Predis și phppredis în cadrul unui proiect de tip Symfony
 - Predis este un client Redis pentru PHP. Redis este un proiect open-source fiind o capacitate de stocare in-memory. Acesta este folosit ca și o bază de date de tip key-value.
 - extensia phppredis aduce un API(Application Programming Interface) ce facilitează comunicarea cu Redis.
 - resurse:
<https://github.com/snc/SncRedisBundle>
<https://github.com/nrk/predis>
<https://github.com/phppredis/phppredis>
<https://redis.io/>
- **DoctrineMigrationsBundle**
 - acest bundle integrează librăria Doctrine2 Migrations în cadrul unei aplicații de tip Symfony
 - migrările sunt folosite pentru a versiona schimbările din baza de date oferindu-i acesteia posibilitate de a reveni la o anumită stare
 - resurse:
<https://github.com/doctrine/DoctrineMigrationsBundle>
<http://www.doctrine-project.org/projects/migrations.html>
- **AsseticBundle**
 - acest bundle oferă posibilitate de a adăuga librăria Assetic în cadrul proiectului
 - oferă posibilitatea de a manipula într-un mod cât mai benefic toate resursele de tip CSS, Javascript și imaginile
 - resurse:
<https://github.com/symfony/assetic-bundle>
https://symfony.com/doc/2.8/assetic/asset_management.html

- **JMSerializerBundle**
 - oferă posibilitatea de a serializa informațiile cerute sub format JSON, XML sau YAML
 - resurse:

<http://jmsyst.com/bundles/JMSSerializerBundle>

- **FOSJsRoutingBundle**
 - acest bundle oferă posibilitatea de expune toate rutele în fișierle de tip JavaScript. Mai exact, toate rutele declarate în format-ul utilizat de către Symfony pot fi folosite și în fișierele JavaScript.
 - resurse:

<https://github.com/FriendsOfSymfony/FOSJsRoutingBundle>

- **VichUploaderBundle**
 - acest bundle facilitează metoda prin care se pot adăuga fișiserele în baza de date
 - resurse:

<https://github.com/dustin10/VichUploaderBundle>

- **IvoryCKEditorBundle**
 - acest bundle oferă posibilitatea de a integra în cadrul proiectului editorul de text CKeditor
 - resurse:

<https://github.com/egeloen/IvoryCKEditorBundle>

<http://ckeditor.com/>

- **GosWebSocketBundle**
 - acest bundle aduce funcționalitatea WebScoket-urilor în cadrul proiectului
 - acest bundle introduce funcționalitățile de tip server cât și de tip client oferind o integrare cât mai benefică
 - resurse:

<https://github.com/GeniusesOfSymfony/WebSocketBundle>

<https://en.wikipedia.org/wiki/WebSocket>

- **GosPubSubRouterBundle**
 - acest bundle are rolul de a aduce logica unui PubSub în cadrul proiectului rezolvând eventualele probleme ce intervin la implementarea lui
 - resurse:

<https://github.com/GeniusesOfSymfony/PubSubRouterBundle>

https://en.wikipedia.org/wiki/Publish%20%80%93subscribe_pattern

- **DoctrineCacheBundle**
 - acest bundle are rol de a face management la cache-ul pentru doctrine
 - resurse:

<https://github.com/doctrine/DoctrineCacheBundle>
- **KnpPaginatorBundle**
 - acest bundle are rolul de introduce paginare în cadrul proiectului
 - resurse:

<https://github.com/KnpLabs/KnpPaginatorBundle>

Acest bundle-uri ce au fost descrise mai sus au fost adăugate în cadrul proiectului prin intermediu utilitarului Composer. În capitolul dedicat implementării vom descrie felul în care au fost instalate, configurate cât și modul cum au fost folosite în cadrul proiectului.

3.4 Proiectarea modelului aplicației

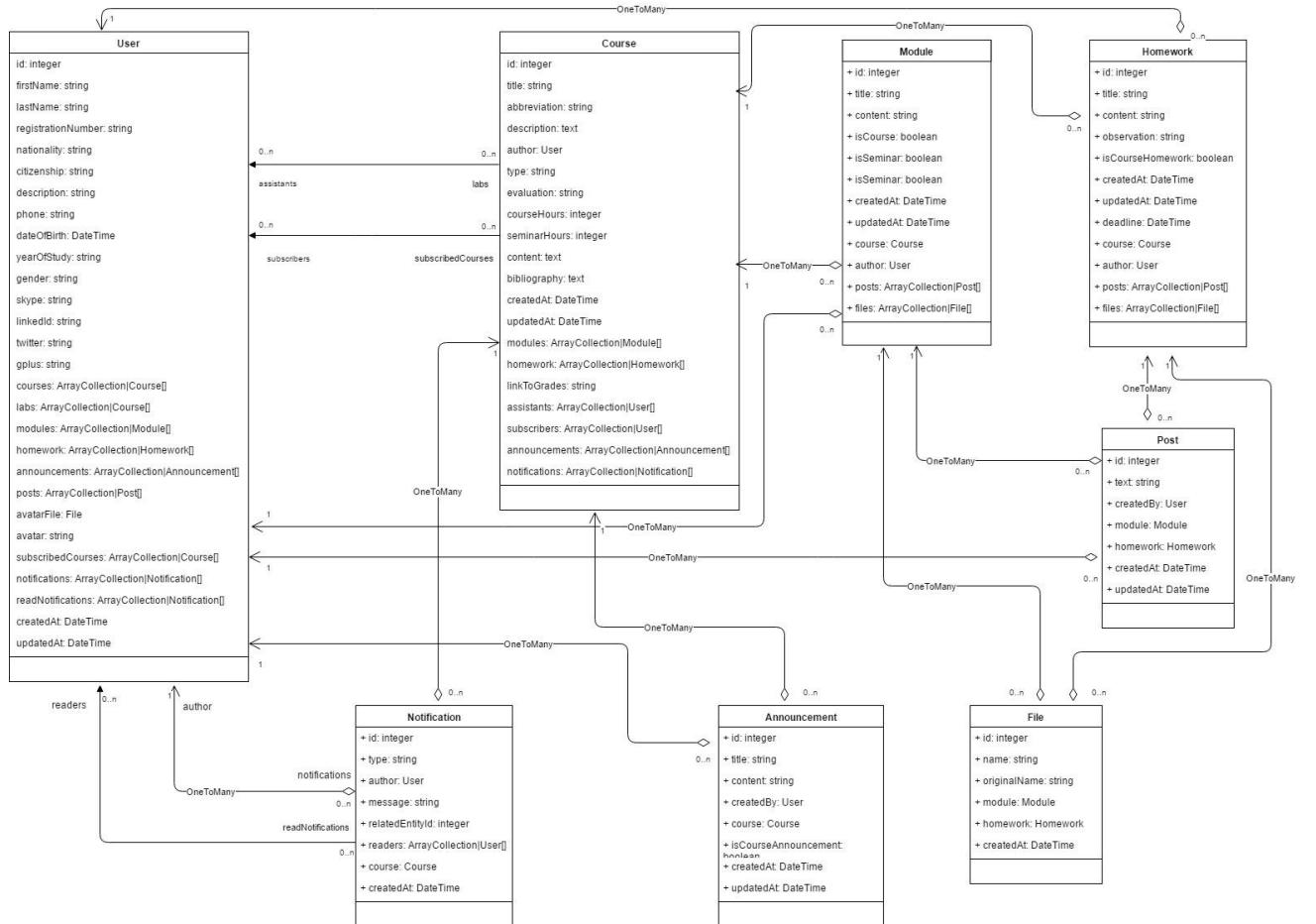
În acest subcapitol vom prezenta cum am proiectat modelul aplicației, mai exact mai exact cum am structurat informațiile cu care va lucra informația și efectiv ce operații vor avea loc asupra datelor introduse în aplicație. Pentru început vom prezenta entitățile din Doctrine ORM și felul în care acestea vor fi mapate la baza de date MySQL.

Entitățile se vor afla în cadrul folderului /Entity aflat în bundle-ul AppBundle. Acest folder este alcătuit la rândul sau din mai multe fișiere cu extensia .php iar fiecare fișier conține o clasa PHP. O asemenea clasă PHP poartă denumirea de entitate și este alcătuită din mai multe câmpuri și mai multe metode de tip getter și setter. Maparea acestei entități la baza de date se realizează prin intermediul metadatelor(date despre date). În cadrul proiectului noi am ales să folosim anotările PHP, însă mai există și posibilitatea de a utiliza fișiere de tip XML sau YAML.

Precum am precizat și mai sus, entitățile proiectului nostru sunt situate în AppBundle. Acestea sunt:

- Announcement
- Course
- File
- Homework
- Module
- Notification
- Post
- User

Următoarea diagramă de clase ne oferă o imagine asupra bazei de date a aplicației urmând ca mai apoi să descriem pe scurt scopul fiecărei entități cât și relațiile dintre acestea.



Figură 10: Diagramă de clase ce descrie schema bazei de date

Deoarece aplicația noastră este focusată asupra cursurilor din cadrul unei instituții de învățământ, vom începe prin a descrie pe scurt entitatea principală a acestui proiect.

După cum se poate observa am utilizat două relații de tip ManyToMany între entitățile *Course* și *User*. Prima dintre ele este între Labs și Assistants deoarece un curs poate beneficia de mai mult profesori asistenți iar aceștia la rândul lor pot fi asistenți la mai multe cursuri în același timp. A doua relație de același tip este între Subscribers și SubscribedCourses ce se ocupă cu funcționalitatea de subscribe din cadrul aplicației.

Putem continua cu relațiile de tip OneToMany între entitatea *Course* și entitățile *Module*, *Homework*, *Announcement*. Deoarece un obiect de tip *Course* poate conține câte o colecție din toate entitățile enumerate anterior, aceste relații sunt realizate prin intermediul câmpurilor Modules - Course, Homework - Course, Announcements - Course. Trebuie precizat faptul că fiecare instanță a unei entități de tip *Module*, *Homework*, *Announcement* este realizată

de către un utilizator ceea ce implică o relație de tip OneToMany între entitatea *User* și entitățile precizate anterior.

În cadrul diagramei ce descrie baza de date mai putem observa de asemenea entitate *Post*. O postare poate fi realizată de către un utilizator în cadrul unei teme, fie în cadrul unui modul de tip seminar sau curs. Relațiile de tip OneToMany dintre Posts și Module, Posts și Homework, Posts și CreatedBy ajută la implementarea funcționalităților descrise.

Pe lângă entitățile prezente descrise în modelul aplicației mai sunt și fișierele de tip repository al căror scop este de a realiza operații asupra bazei de date. Aceste fișiere conțin câte o clasă de tip PHP și se regăsesc în folderul *Repository* din cadrul unui bundle.

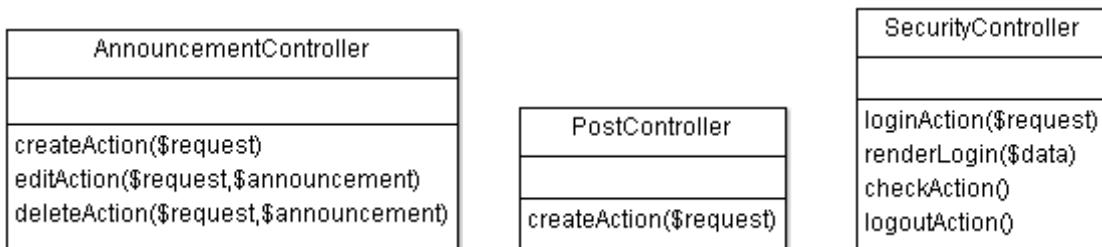
3.5 Proiectarea controlere-lor

O aplicație ce utilizează framework-ul web Symfony automat implementează modelul arhitectural **MVC**(Model-View-Controller).

Un controller are rolul de a actualiza modelul în funcție de datele primite de la utilizator. În Symfony, un controller este reprezentat printr-un fișier PHP ce conține o clasă populată de mai multe metode postfixate cu termenul Action.

Mai precis, atunci când un utilizator accesează un anumit URL, browser-ul va accesa o rută din proiect și va trimite un request HTTP către server. Intern, Symfony va analiza acel request și îl va trimite către metoda asociată acelei rute. În interiorul acestei metode se realizează toată logica dorită pentru o anumită pagină a proiectului urmând ca mai apoi să fie trimis un răspuns către browser-ul web.

În diagramele de clasă ce urmează sunt descrise toate controlerelor utilizate în cadrul proiectului oferind o imagine mai clară asupra metodelor acestora.



<p>CourseController</p> <hr/> <pre>listAction() exploreAction() associateAction() createAction(\$request) editAction(\$request,\$course) showAction(\$course) deleteAction(\$request,\$course) addAssociateProfessorAction(\$request,\$course) listAssociateProfessorAction(\$course) removeAssociateAction(\$request,\$course) associateProfessorsAction(\$course) courseModulesAction(\$course) seminarModulesAction(\$course) subscribeAction(\$course) unsubscribeAction(\$course) userProfileAction(\$course,\$userId) userSeminarsAction(\$course,\$userId) listHomeworkAction(\$course) showHomeworkAction(\$request,\$course,\$homeworkId) userHomeworkAction(\$course,\$userId) listAnnouncementsAction(\$course) userAnnounelementsAction(\$course,\$userId)</pre>	<p>ModuleController</p> <hr/> <pre>createCourseAction(\$request) createSeminarAction(\$request) editCourseAction(\$request,\$module) editSeminarAction(\$request,\$module) showAction(\$request,\$module) showSeminarAction(\$request,\$module) deleteCourseAction(\$request,\$module) deleteSeminarAction(\$request,\$module) downloadAttachmentAction(\$module) deleteUploadedFileAction(\$file) downloadFileAction(\$file)</pre>
<p>HomeworkController</p> <hr/> <pre>createAction(\$request) editAction(\$request,\$homework) showSeminarAction(\$request,\$homework) deleteAction(\$request,\$homework) deleteUploadedFileAction(\$file) downloadFileAction(\$file)</pre>	<p>UserController</p> <hr/> <pre>associateProfessorComponentAction(\$user) editAction(\$request,\$user) showAction(\$user) listProfessorsAction() notificationsAction(\$request) notificationsNumberAction() seenNotificationsAction(\$notifications)</pre>

3.6 Proiectarea design-ului aplicației

Design-ul aplicației va fi unul responsive ceea ce înseamnă că aplicația își va schimba interfața în funcție de rezoluția ecranului utilizatorului.

Pentru partea de front-end vom utiliza framework-urile Bootstrap și Materializecss la care se va adăuga HTML, CSS, JavaScript și biblioteca jQuery acolo unde este necesar.

Per total aplicația va avea un design simplist dar totodată și atractiv, având ca și scop principal de a ușura interacțiunea utilizatorului cu această aplicație și de a servi toate funcționalitățile prezentate la capitolele anterioare.

4 Implementare

În cadrul acestui capitol vom prezenta modul în care am implementat funcționalitățile aplicației și mai exact felul în care am ajuns la produsul final plecând de la componentele și modelele descrise în capitolele anterioare. În cele ce urmează vom vedea concret cum am trecut de la analiza cerințelor, modelare și proiectare la implementare. Vom analiza soluțiile găsite, problemele peste care am dat și vom exemplifica felul în care am rezolvat problemele prin exemple de cod.

Toate funcționalitățile aplicației sunt implementate în cadrul bundle-ului AppBundle și sunt divizate în funcție de interfață în subfolderele Controller/Admin, Controller/Main.

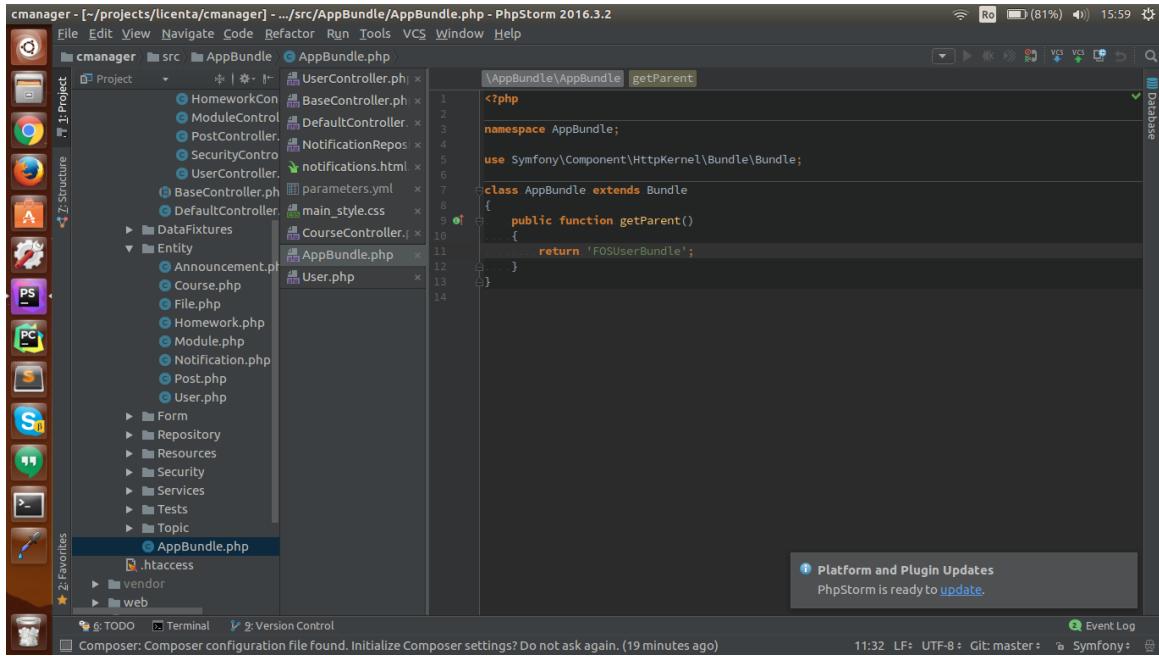
4.1 Implementarea funcționalităților legate de utilizator

Principalele funcționalități ale unui utilizator sunt de a se autentifica sau a-și activa contul, urmând ca mai apoi să poată interacționa în cadrul aplicației cu alți utilizatori prin intermediul cursurilor.

Înainte de a implementa toate aceste funcționalități am căutat librării open-source ce ne puteau ajuta în acest sens. Știm cu toții că în zilele noastre librăriile third-party au un rol foarte important întrucât simplifică procesul de dezvoltare al unei aplicații.

Am profitat în acest sens de marea comunitate din jurul framework-ului Symfony și am găsit pe Packagist un bundle ce poartă numele de **FOSUserBundle**. Scopul acestui bundle este de implementa funcționalitățile de bază legate de un utilizator cum ar fi: logare, autentificare, resetare parolă etc.

Pentru a putea folosi FOSUserBundle trebuie să suprascriem funcționalitățile oferite de acesta pentru a satisface nevoile personale în cadrul proiectului. Conform documentației oficiale, un prim pas în realizarea acestui lucru este de a crea o relație de tip "*parent-child*" între AppBundle și FOSUserBundle. Acest lucru se realizează prin suprascrierea metodei *getParent()* din cadrul clasei *AppBundle* conform figurei cu numărul 11.



Figură 11: Clasă ce setează ca părinte clasa FOSUserBundle

Următorul pas în utilizarea acestui bundle în cadrul proiectului este de a suprascrie funcționalitățile din FOSUserBundle. Prințipiu după care s-a realizat acest lucru a fost de înlocui conținutul funcțiilor ce ne interesează cu propria logică ținând cont totuși și de indicațiile celor ce au realizat FOSUserBundle.

Am început acest proces cu implementarea entităților Doctrine. Aceste entități sunt fișiere PHP ce definesc câte o clasă cu proprietăți și metode, aici purtând numele de getteri și setteri. Răspunsul la întrebare *"De ce entități?"* este că acestea ne oferă posibilitate de a crea un rând din baza de date cu un obiect. Astfel putem lucra mult mai ușor și mai optim cu informațiile din baza de date.

Maparea dintre o clasă PHP și o tabelă din baza de date se realizează prin intermediul adnotărilor specifice Doctrine: `@ORM\Table()`, `@ORM\Entity()`, `@ORM\Column()`, `@ORM\OneToMany`, `@ORM\ManyToOne()` etc.

Entitate ce se ocupă cu manipularea utilizatorilor este *User.php* situată în directorul *Entity*. Conform figurei cu numărul 12, clasa User extinde clasa BaseUser din FOSUserBundle (ceea ce implică faptul că aceasta va conține proprietățile din clasa de bază) la care vom adăuga câmpurile necesare unui utilizator din sistemul nostru: *firstName*, *lastName*, *registrationNumber*, *nationality*, *citizenship*, *descriprion*, *phone*, *dateOfBirth*, *yearOfStudy* etc. Definiția completă a acestei clase se găsește la capitolul anterior dedicat modelelor proiectului.

The screenshot shows the PhpStorm IDE interface with the project 'cmanager' open. The left sidebar shows the project structure with 'src' and 'AppBundle' selected. The main editor window displays the 'User.php' file under the 'AppBundle\Entity' package. The code defines a User entity with various properties like gender constants, roles, and database columns. The code is annotated with PHPDoc comments and annotations from the Doctrine ORM.

```


/*
 * User
 *
 * @ORM\Table(name="fos_user")
 * @ORM\Entity(repositoryClass="AppBundle\Repository\UserRepository")
 * @Vich\Uploadable
 * @UniqueEntity(fields="username", message="unique.username")
 * @UniqueEntity(fields="email", message="unique.email")
 */
class User extends BaseUser
{
    const GENDER_MALE = 'gender.male';
    const GENDER_FEMALE = 'gender.female';

    const ROLE_ADMIN = 'ROLE_ADMIN';
    const ROLE_STUDENT = 'ROLE_STUDENT';
    const ROLE_PROFESSOR = 'ROLE_PROFESSOR';
    const ROLE_ASSOCIATE = 'ROLE_ASSOCIATE';

    const GRAVATAR_BASE_URL = 'https://www.gravatar.com/avatar/';

    /**
     * @var int
     *
     * @ORM\Column(type="integer")
     * @ORM\Id
     * @ORM\GeneratedValue(strategy="AUTO")
     */
    protected $id;

    /**
     * @var string
     *
     * @ORM\Column(name="first_name", type="string", nullable=false)
     */
    private $firstName;
}


```

Figură 12: Clasa User. Exemplu de entitate

Controller-ul ce conține logica necesară pentru ca un utilizator să acceseze platforma se regăsește în fișierul SecurityController aflat în directorul Controller/Main. Acest fișier conține acțiunile: *loginAction()*, *renderLogin()*, *checkAction()*, *logoutAction()*.

Mai trebuie precizat faptul că aplicație oferă posibilitatea unui viitor dezvoltator al acestei aplicații de a crea un utilizator foarte simplu prin intermediu unei comenzi. În cadrul unui proiect Symfony, comenziile sunt declarate în folderul ce poartă denumirea de Command. Pentru fiecare comandă utilizată de la consolă se va crea un fișier asemantor celui din figura numărul 13. În cazul nostru, comanda valabilă de a crea un utilizator este *app/console app:user-create* fiind urmată de mai multi parametrii obligatorii sau nu. Odată rulată, un utilizator nou având statusul de user activ va fi inserat în baza de date.

The screenshot shows the PhpStorm IDE interface with the project 'cmanager' open. The left sidebar shows the project structure with 'src' and 'AppBundle' selected. The main editor window displays the 'UserCreateCommand.php' file under the 'AppBundle\Command' package. The code defines a UserCreateCommand class that extends ContainerAwareCommand. It includes methods for configuration and execution, handling input arguments for email, password, username, first name, last name, and role.

```


<?php
namespace AppBundle\Command;

use ...

/**
 * Create a new user object (firstname, lastname and role are optionally).
 *
 * Command usage: app:user-create email@email.com password username firstname lastname ROLE_ADMIN
 */
class UserCreateCommand extends ContainerAwareCommand
{
    protected function configure()
    {
        $this
            ->setName('app:user-create')
            ->addArgument('email', InputArgument::REQUIRED, 'The email of the new user')
            ->addArgument('password', InputArgument::REQUIRED, 'The password of the new user')
            ->addArgument('username', InputArgument::REQUIRED, 'The username of the new user')
            ->addOption('first_name', null, InputArgument::OPTIONAL, 'The first name of the new user')
            ->addOption('last_name', null, InputArgument::OPTIONAL, 'The last name of the new user')
            ->addOption('phone', null, InputArgument::OPTIONAL, 'The phone of the new user', '0123456789')
            ->addOption('role', null, InputOption::VALUE_TS_ARRAY | InputOption::VALUE_OPTIONAL,
    }
}

protected function execute(InputInterface $input, OutputInterface $output)
{
    $email = $input->getArgument('email');
    $password = $input->getArgument('password');
    $username = $input->getArgument('username');
    $firstname = $input->getOption('first_name');
    $lastName = $input->getOption('last_name');
}


```

Figură 13: Fișierul asociat comenzi de app:user-create

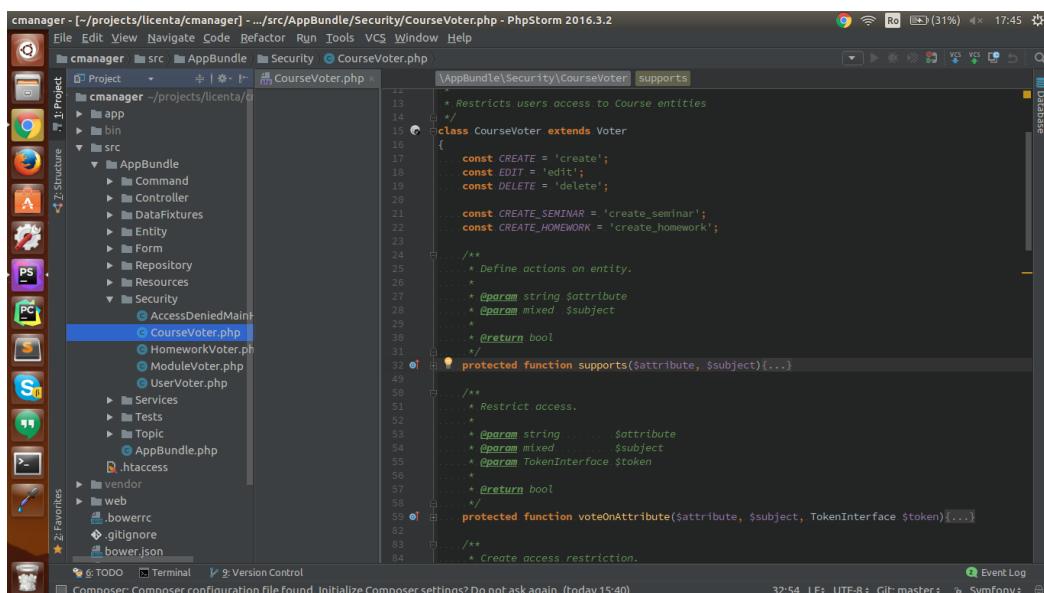
4.2 Implementarea sistemului ce oferă permisiuni în cadrul aplicației

În capitolele anterioare am discutat despre arhitectura aplicației și am precizat faptul ca pot fi mai multe tipuri de utilizatori. În cadrul aplicației vom avea utilizatori cu rol de student, profesor, profesor asociat și admin. Acest lucru implică faptul că fiecare utilizator are alte drepturi în cadrul aplicației cum ar fi: un profesor poate crea un curs, pe când un student are dreptul doar de a face subscribe și de vizualiza ce postări sunt făcute în cadrul acelui curs.

Pentru a rezolva această problemă am apelat la metoda descrisă de cei de la Symfony și anume de a utiliza *Voters*. Un *Voter* este un fișier ce conține o clasa PHP al cărei rol este de a verifica permisiunile unui anumit utilizator pe anumite pagini. Un proiect poate conține un număr infinit de *Voters* deoarece Symfony are o metodă destul de simplă pentru a verifica dacă un utilizator are acces sau nu la pagina dorită.

Absolut toți *Voterii* sunt apelați de către Symfony atunci când se foloșește metoda *isGranted()* pentru a verifica dacă utilizatorul este autorizat să acceseze acea resursă, fie când în interiorul unui Controller este utilizată metoda *denyAccessUnlessGranted()*. În final, Symfony strângă toate răspunsurile de la toți *Voterii* și decide dacă utilizatorul are sau nu acces la acea resursă în funcție de implementările făcute în cadrul *Voterilor*.

Pentru a crea un *Voter* trebuie să se extindă clasa *Voter* situată în directorul de Security al vendorului principal, Symfony. Acest lucru ne obligă să suprascriem metodele *supports()*, *voteOnAttribute()* descrise în clasa abstractă *Voter*. Din cadrul proiectului nostru vom lua ca și exemplu *CourseVoter*. Putem observa în figura cu numărul 14 structura fișierului și faptul că am respectat cele spuse mai sus.

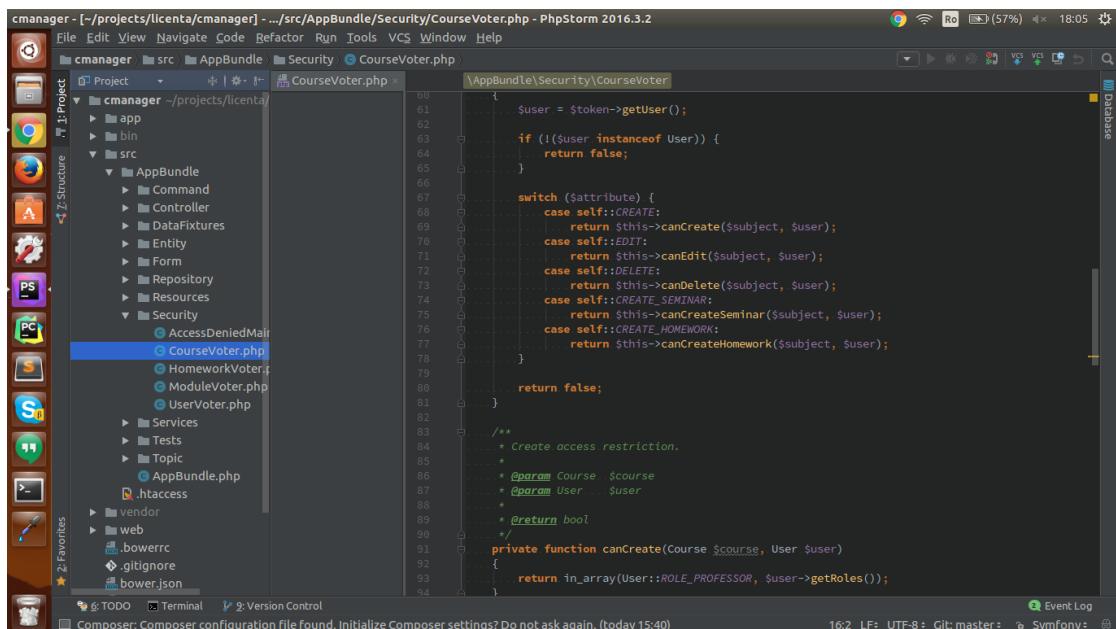


```
CourseVoter.php
1 * Restricts users access to Course entities
2 */
3 class CourseVoter extends Voter
4 {
5     const CREATE = 'create';
6     const EDIT = 'edit';
7     const DELETE = 'delete';
8
9     const CREATE_SEMINAR = 'create_seminar';
10    const CREATE_HOMEWORK = 'create_homework';
11
12    /**
13     * Define actions on entity.
14     */
15    protected function supports($attribute, $subject)
16    {
17        /**
18         * @param string $attribute
19         * @param mixed $subject
20         */
21        if ($attribute === 'is_granted') {
22            /**
23             * @return bool
24             */
25            return true;
26        }
27        /**
28         * @param string $attribute
29         * @param mixed $subject
30         */
31        if ($attribute === 'is_equal') {
32            /**
33             * @param string $attribute
34             * @param mixed $subject
35             * @param TokenInterface $token
36             */
37            return $subject->getSeminar();
38        }
39    }
40
41    /**
42     * Restrict access.
43     *
44     * @param string $attribute
45     * @param mixed $subject
46     * @param TokenInterface $token
47     */
48    protected function voteOnAttribute($attribute, $subject, TokenInterface $token)
49    {
50        /**
51         * @return bool
52         */
53        if ($attribute === 'is_granted') {
54            /**
55             * @param string $attribute
56             * @param mixed $subject
57             * @param TokenInterface $token
58             */
59            if ($subject->getSeminar() === $token->getSeminar()) {
60                return true;
61            }
62        }
63    }
64}
```

Figură 14: Fișierul *CourseVoter* ce implementează sistemul de *Voter*

Pentru început am declarat constantele *CREATE*, *EDIT*, *DELETE*, *CREATE_SEMINAR*, *CREATE_HOMEWORK* ce definesc acțiunile ce pot fi făcute de un utilizator în cadrul unui curs. În momentul în care se apelează metoda *denyAccessUnlessGranted()* din cadrul unui controller prima data va fi acționată metoda *supports()* din cadrul Voter-ului. Aceasta primește ca și parametru acțiunea definită prin intermediul constantelor și un obiect asupra căruia trebuie să se facă acea acțiune. Această funcție va returna *true/false* în funcție de parametrii transmiși și de acțiunile permise în cadrul funcției *supports()*. Dacă răspunsul returnat va fi *true* atunci va fi apelată metoda *voteOnAttribute()* din cadrul aceluiași Voter. În caz contrar se va trece la următorul Voter definit de către programator.

Rolul metodei *voteOnAttribute* este de a oferi acces sau nu asupra fiecărei funcționalități în funcție de utilizatorul autentificat și obiectul asupra căruia se doresc modificările. Funcționalitatea din cadrul acestei funcții poate fi simplificată prin utilizarea unui *switch* ce va trata fiecare caz în parte, urmând ca fiecare verificare să fie realizată în câte o funcție auxiliară în cadrul aceluiași Voter.



```

cmanager - [~/projects/llicenta/cmanager] - .../src/AppBundle/Security/CourseVoter.php - PhpStorm 2016.3.2
File Edit View Navigate Code Refactor Run Tools VCS Window Help
cmanager -/projects/llicenta/ Project CourseVoter.php
  cmanager -/projects/llicenta/
    app
    bin
    src
      AppBundle
        Command
        Controller
        DataFixtures
        Entity
        Form
        Repository
        Resources
        Security
          AccessDeniedMapper.php
          CourseVoter.php
          HomeworkVoter.php
          ModuleVoter.php
          UserVoter.php
        Services
        Tests
        Topic
        AppBundle.php
        htaccess
      vendor
      web
        .bowerrc
        .gitignore
        bower.json
  Favorites
  Event Log
  TODO Terminal Version Control
  Composer: Composer configuration file found. Initialize Composer settings? Do not ask again. (today 15:40)
  16:2 LF: UTF-8 Git: master Symfony: 

```

```

60
61     $user = $token->getUser();
62
63     if (!($user instanceof User)) {
64         return false;
65     }
66
67     switch ($attribute) {
68         case self::CREATE:
69             return $this->canCreate($subject, $user);
70         case self::EDIT:
71             return $this->canEdit($subject, $user);
72         case self::DELETE:
73             return $this->canDelete($subject, $user);
74         case self::CREATE_SEMINAR:
75             return $this->canCreateSeminar($subject, $user);
76         case self::CREATE_HOMEWORK:
77             return $this->canCreateHomework($subject, $user);
78     }
79
80     return false;
81 }
82
83 /**
84 * Create access restriction.
85 *
86 * @param Course $course
87 * @param User $user
88 *
89 * @return bool
90 */
91 private function canCreate(Course $course, User $user)
92 {
93     return in_array(User::ROLE_PROFESSOR, $user->getRoles());
94 }

```

Figură 15: Clasa CourseVoter

Aplicația CourseManager dispune de 4 Voteri ce verifică accesul utilizatorului pe diferite pagini în funcție de rolul acestuia. Acești Voteri sunt : *CourseVoter*, *HomeworkVoter*, *ModuleVoter*, *UserVoter*.

4.3 Implementarea sistemului de notificări

Aplicația conține și un sistem de notificări ce urmează a fi descris în acest subcapitol. În această versiune a aplicației putem spune că avem un sistem de notifică rudimentar și mai poate suferi îmbunătățiri în versiunile ce urmează.

Sistemul de notificări al aplicației constă în două tipuri de notificări și anume: notificare pentru postarea unui nou modul de tip curs în cadrul unui curs, notificare pentru un anunț general în cadrul cursului realizată de către autorul cursului la care a fost făcut subscribe.

Pentru implementarea s-au folosit **WebSockets** și un bundle realizat de către comunitatea Symfony pentru a ușura munca cu WebSockets ce poartă denumirea de **Gos Web Socket Bundle**. Un WebSocket este un protocol de comunicare ce oferă o comunicare de tip *full-duplex* într-un canal peste o conexiune de tip TCP.

În cadrul aplicației ne-am folosit de Gos Web Socket Bundle ce oferă mai multe facilități printre care și posibilitatea de a crea un **Topic Handler**. Un Topic este defapt reprezentarea pe partea de server a unui canal de tip **PubSub**. Pentru utilizarea, trebuie înregistrat un topic cu prefixul *chat* pentru a suporta PubSub. Precum se poate vedea și în *Figura 16*, am creat o clasă cu numele **CourseNotificationTopic** situată în directorul Topic. Această clasa implementează interfețele **TopicInterface**, **PushableTopicInterface** fapt ce implică că trebuie declarate și implementate următoare metode în cadrul clasei:

- **onSubscribe(ConnectionInterface \$connection, Topic \$topic, WampRequest \$request)** – metoda apelată atunci când un utilizator face subscribe la topic;
- **onUnsubscribe(ConnectionInterface \$connection, Topic \$topic, WampRequest \$request)** – metoda apelată atunci când un utilizator face unsubscribe la topic;
- **onPublish(ConnectionInterface \$connection, Topic \$topic, WampRequest \$request, \$event, array \$exclude, array \$eligible)** – metodă apelată atunci când un utilizator face publish pe topic;
- **getName()** – metoda ce returnează deumirea în cadrul serviciul de rutare;

În cadrul metodelor de mai sus, parametri acestora au următoarele seminificații:

- **ConnectionInterface \$connection** – semnifică conexiunea realizată de către un client;
- **TopicInterface \$topic** – semnifică obiectul de tip Topic; de asemenea acesta conține o listă cu toți utilizatorii ce au făcut subscribe la acest topic;
- **WampRequest** – semnifică requestul realizat prin intermediul la WebSocket;

```

<?php
namespace AppBundle\Topic;
use ...
class CourseNotificationTopic implements TopicInterface, PushableTopicInterface
{
    const CREATE_COURSE_MODULE = 'create_course_module';
    const NEW_COURSE_ANNOUNCEMENT = 'new_course_announcement';

    protected $clientManipulator;

    /**
     * UserNotificationTopic constructor.
     *
     * @param ClientManipulatorInterface $clientManipulator
     */
    public function __construct(ClientManipulatorInterface $clientManipulator)
    {
        $this->clientManipulator = $clientManipulator;
    }

    /**
     * This will receive any Subscription requests for this topic.
     *
     * @param ConnectionInterface $connection
     * @param Topic $topic
     * @param WampRequest $request
     */
    public function onSubscribe(ConnectionInterface $connection, Topic $topic, WampRequest $request)
    {
        $user = $this->clientManipulator->getClient($connection);
        dump($user);
    }
}

```

Figură 16 Captură de ecran cu clasa *CourseNotificationTopic*

După declararea clasei cu numele *CourseNotificationTopic* urmează specificarea adresei de rutare în cadrul serviciului utilizat de către framework-ul Symfony în fișierul cu numele *routing.yml* aflat în subdirectorul *Resources/config/pubsub*.

```

course_notification_topic:
    channel: course/notification/channel/room/{id}
    handler:
        callback: 'course.notification.topic' #related to the getName() of your topic

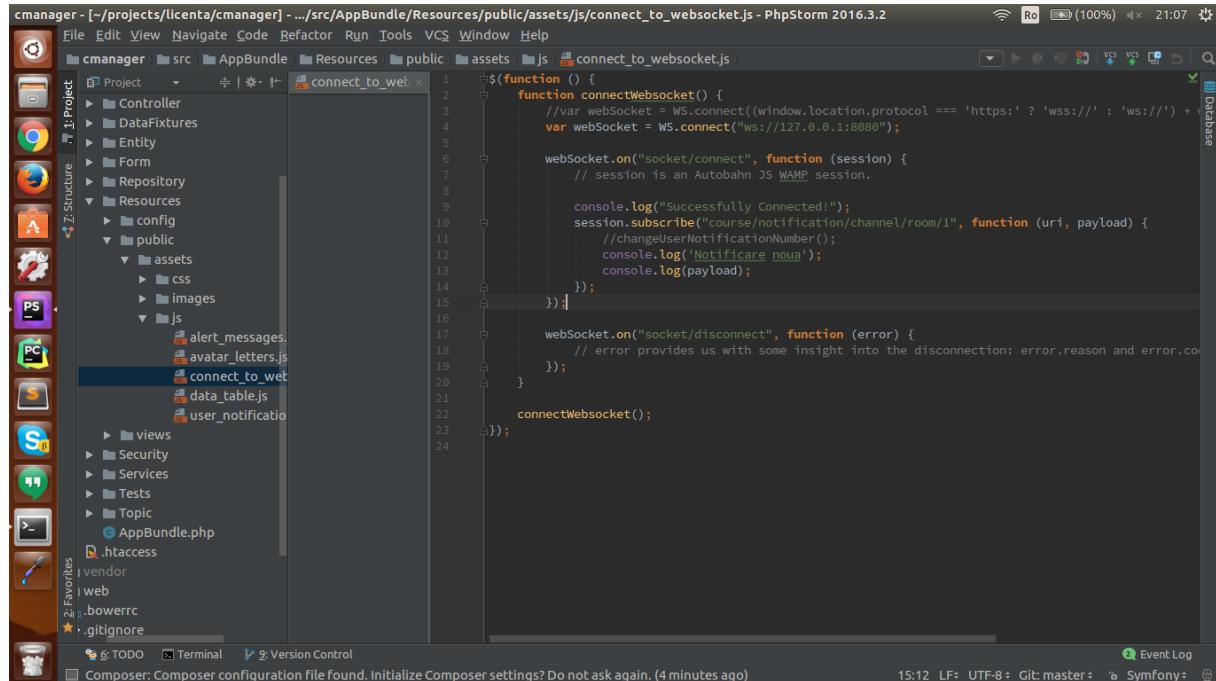
```

Figură 17 Captură de ecran cu fișierul de rutare dedicat topicului de notificări

Următorul pas în realizarea sistemului de notificări reprezintă setările pe partea de client. Pentru a include toate fișierele de tip *javascript* din bundle-ul utilizat am adăugat următorul apel de funcție: {{ ws_client() }}.

Odată ce am inclus toate fișierele dependente, putem folosi *gos_web_socket_client.js* pentru a interacționa serverul de web socket. În cazul acestui client JavaScript, ne-am folosit de două funcții ce apar și în *Figură 18* și anume:

- *websocket.on("socket/connect", function (session){})* – metodă folosită la conexiunea socket-ului; funcția de callback va permite conectarea la canalul de notificări realizat în cadrul proiectului;
- *websocket.on("socket/disconnect", function (error) {})* – metodă folosită la deconectarea socket-ului;



The screenshot shows the PhpStorm IDE interface. The title bar reads "cmanager - [-/projects/licenta/cmanager] - .../src/AppBundle/Resources/public/assets/js/connect_to_websocket.js - PhpStorm 2016.3.2". The code editor displays the following JavaScript code:

```

function () {
    var webSocket = WS.connect((window.location.protocol === 'https:' ? 'wss://' : 'ws://') + '127.0.0.1:8080');

    webSocket.on("socket/connect", function (session) {
        // session is an Autobahn JS WAMP session.
        console.log("Successfully Connected!");
        session.subscribe("course/notification/channel/room/1", function (uri, payload) {
            //changeUserNotificationNumber();
            console.log('Notificare noua');
            console.log(payload);
        });
    });

    webSocket.on("socket/disconnect", function (error) {
        // error provides us with some insight into the disconnection: error.reason and error.co
    });
}

connectWebSocket();

```

The code is for a file named "connect_to_websocket.js" located in the "public/assets/js" directory of the "cmanager" project. The code sets up a connection to a WebSocket server at "ws://127.0.0.1:8080". It handles the "socket/connect" event to subscribe to the "course/notification/channel/room/1" topic and log received notifications. It also handles the "socket/disconnect" event.

Figură 18 Captură de ecran a clientului JavaScript de conectare la serverul WebSocket

Singurul pas care mai trebuie făcut pentru a trimite o notificare este de a prelua serviciul *gos_web_socket.zmq.pusher* în cadrul unui controller și de a trimite o notificare de tip push pe topicul declarat în cadrul proiectului. Toți utilizatorii ce au făcut subscribe la un anumit curs vor primi notificările legate de acesta în timp real.

```

$pusher = $this->container->get('gos_web_socket.zmq.pusher');

$pusher->push(
    ['type' => CourseNotificationTopic::CREATE_COURSE_MODULE],
    'course_notification_topic',
    ['id' => $course->getId()]
);

```

Figură 19 Captură de ecran cu metoda de trimitere a unei notificări

4.4 Implementarea view-urilor și al design-ului

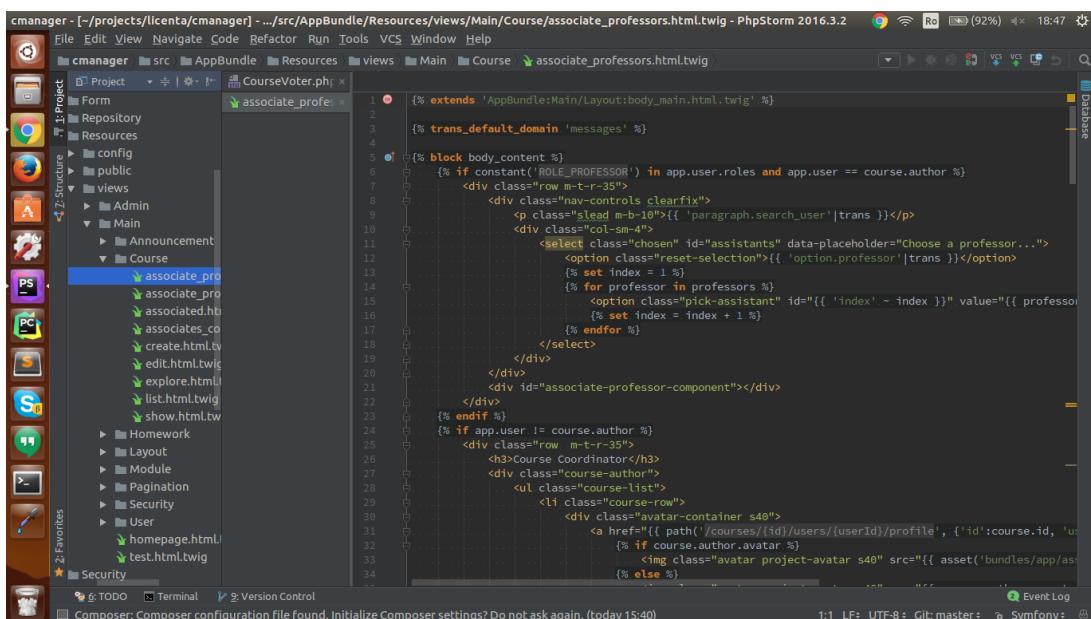
Pentru implementarea view-urilor, am utilizat HTML, CSS și JavaScript. Pe langă acestea am profitat de un template engine pentru PHP ce poartă numele de Twig la care se adaugă și front-end framework-urile Bootstrap și Materializecss.

Fisierele de tip view se află în cadrul bundle-ului sub directorul *Resources/view*. Acestea sunt grupate la rândul lor în alte foldere în funcție de controller-ul din care fac parte.

Twig a adus un plus în cadrul implementării interfeței deoarece acesta pune la dispoziția programatorului posibilitatea de a moșteni un template din alt template cât și organizarea codului pe blocuri.

În cadrul folderului view vom observa două foldere și anume *Admin*, *Main*. În fiecare folder există o ierarhie de foldere și fișiere printre care se regăsește căte un fișier ce poartă denumirea de *base.html.twig*. Acest fișier reprezintă template-ul de bază și conține diverse blocuri oferind posibilitatea fișierelor ce extind acest *base.html.twig* să suprascrie blocurile în funcție de funcționalitățile dorite.

Un fișier cu extensia *.twig* este conform celui din figura numărul 21.



```
% extends 'AppBundle:Main/Layout:body_main.html.twig'
%
% trans_default_domain 'messages'
%
{# block body_content #}
{# if constant('ROLE_PROFESSOR') in app.user.roles and app.user == course.author #}
    <div class="row m-t-r-35">
        <div class="nav-controls clearfix">
            <p class="lead m-b-10">{{ 'paragraph.search_user'|trans }}</p>
            <div class="col-sm-4">
                <select class="chosen" id="assistants" data-placeholder="Choose a professor...">
                    <option class="reset-selection" value="">{{ 'option.professor'|trans }}</option>
                    {# set index = 1 #}
                    {# for professor in professors #}
                        <option class="pick-assistant" id="{{ 'index' ~ index }}" value="{{ professor }}>{{ professor }}</option>
                    {# endfor #}
                </select>
            </div>
        </div>
        <div id="associate-professor-component"></div>
    </div>
{# endif #}
{# if app.user != course.author #}
    <div class="row m-t-r-35">
        <h3>Course Coordinator</h3>
        <div class="course-author">
            <ul class="course-list">
                <li class="course-row">
                    <div class="avatar-container s40">
                        <a href="{{ path('/courses/{id}/users/{userId}/profile', {'id':course.id, 'userId':course.author}) }}>
                            {# if course.author.avatar #}
                                
                            {# else #}
                                
                            {# endif #}
                        </a>
                    </div>
                </li>
            </ul>
        </div>
    </div>
{# endif #}
```

Figură 21: Exemplu de fișier cu extensia *.twig*

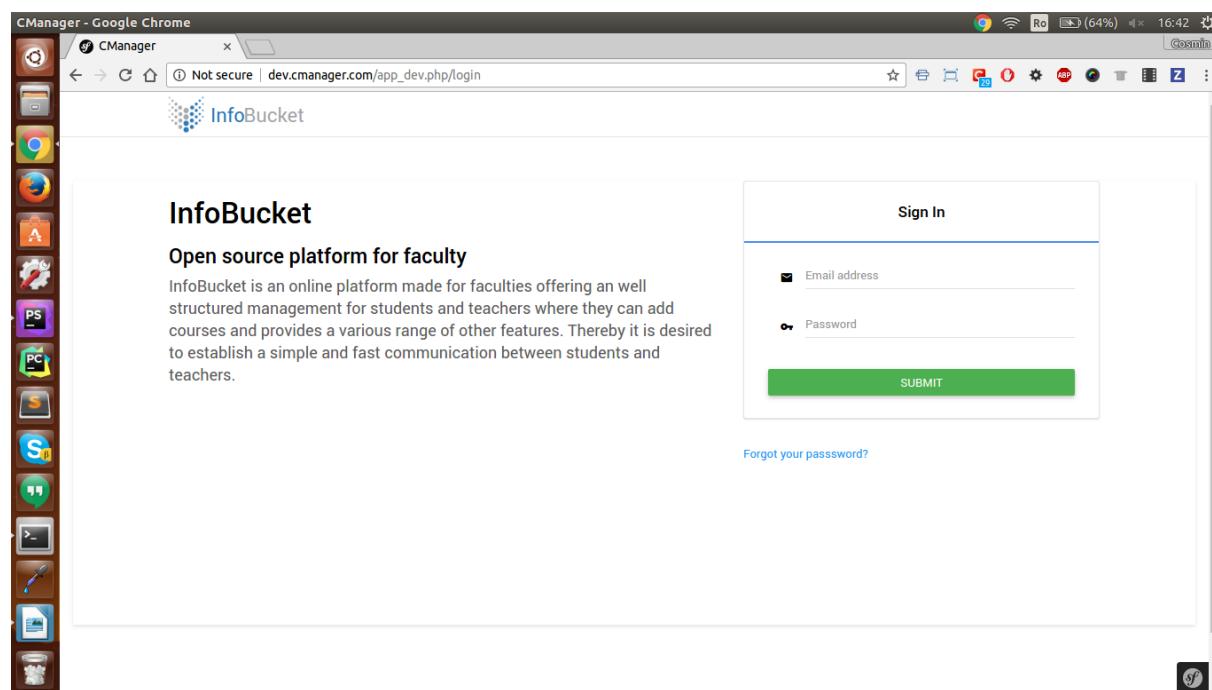
5 Manual de utilizare

În cadrul acestui capitol vom prezenta aplicația sub forma unui manual de utilizare. Vom explica diversele funcționalități ale aplicației iar pentru a ușura înțelegerea aplicației vom adăuga și capturi de ecran.

5.1 Logarea utilizatorului/Editarea profilului

Atunci când un nou utilizator al aplicației va accesa platforma dezvoltată, acesta va vizualiza pagina principală a acesteia ce conține un formular de login cât și o scurta descriere a aplicației. Menționăm faptul că dacă un utilizator a fost autentificat în cadrul aplicației acesta va avea acces la diversele pagini ale aplicației fără a mai trece prin pasul de login întrucât este salvat în sesiune. În caz contrar, acesta va fi automat redirecționat către pagina de login a aplicației.

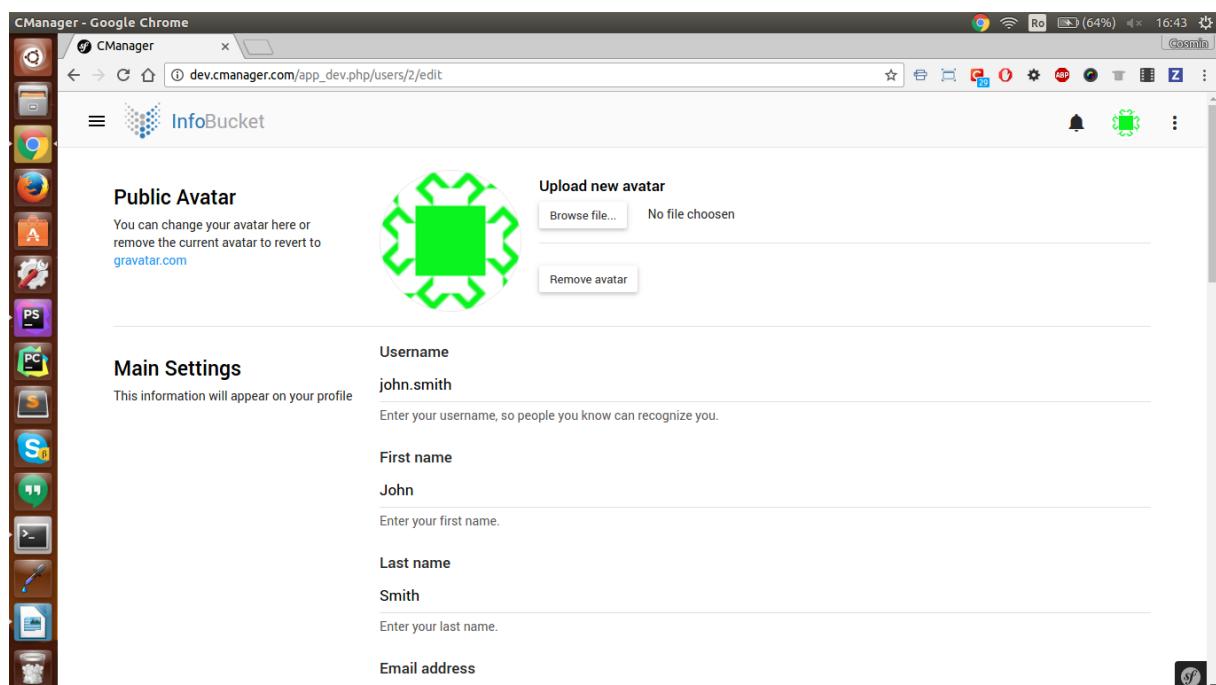
Încă de la început trebuie precizat faptul că un utilizator nu are posibilitatea de a-și crea singur un nou cont. Fiecare cont este generat automat de către un admin ce are dreptul de a accesa interfața dedicată administrării aplicației. Un admin are posibilitatea de a crea unul sau mai multe conturi prin completarea formularului de creare cont a unui utilizator. Crearea unui cont implică automat trimitera unui email către fiecare utilizator pentru confirmarea și activarea contului.



Figură 22 Captură de ecran cu pagina de autentificare

După autentificarea cu succes a utilizatorului, acesta va fi redirecționat către pagina dedicată cursurilor la care utilizatorul a facut subscribe.

Deoarece contul a fost creat de către un admin, utilizatorul este sfătuit să își editeze profilul cu toate informațiile dorite deoarece ceilalți utilizatori ai platformei vor avea posibilitatea de a vizualiza profilul acestuia. Acest lucru se poate realiza foarte ușor printr-un click pe poza de profil ce este setată inițial prin intermediul platformei **gravatar.com**, fie prin accesarea meniului imediat următor și selectarea link-ului de **Edit**. Pagina de editarea a profilului oferă posibilitatea utilizatorului de a-și seta informații precum: *Firt name, Last name, Skype account, Gplus Account, LinkedIn Account* etc.

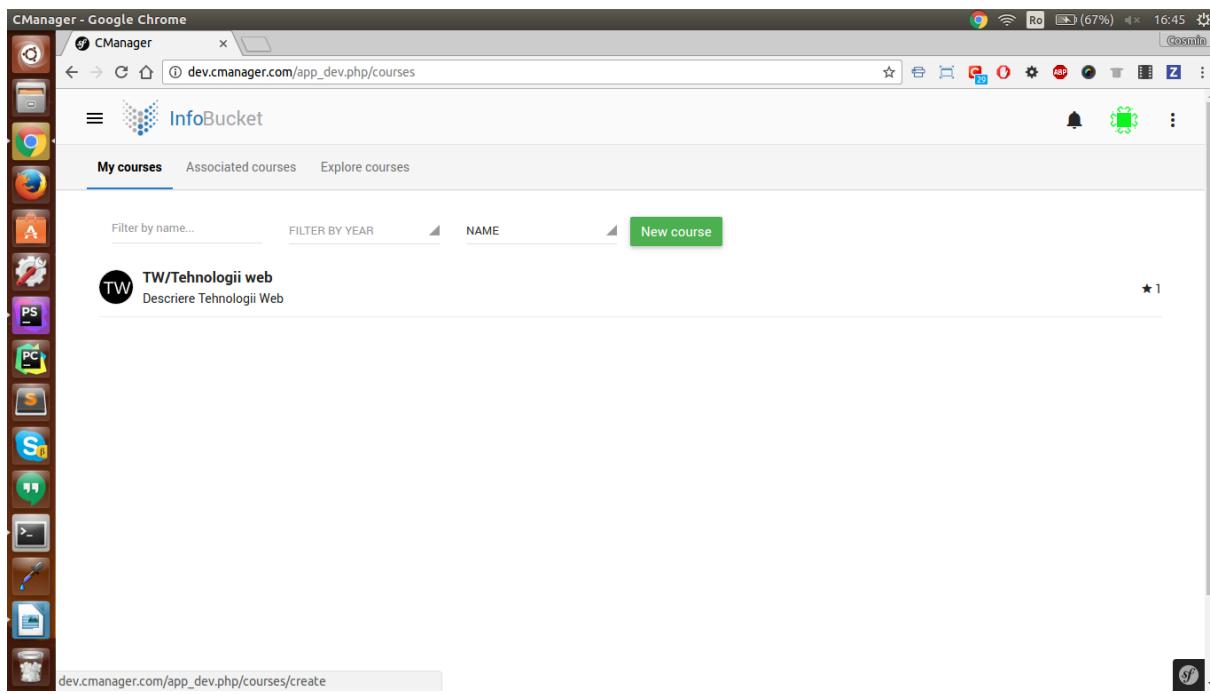


Figură 23 Captură de ecran cu pagina de editarea a profilului unui utilizator

5.2 Crearea unui curs

Pentru a crea un curs în cadrul aplicației, utilizatorul va trebui să fie autentificat în cadrul acestuia și să dețină rolul de profesor. Rolul unui utilizator nu poate fi schimbat de către acesta în cadrul paginii de editare profil, acestea fiind create/modificate de către un admin în interfața de administrare.

După ce utilizatorul este autentificat în cadrul aplicației, acesta va fi redirecționat către pagina **My courses** ce conține toate cursurile adăugate de către utilizator. În cazul în care acesta se află pe o altă pagină, secțiunea cursurilor poate fi accesată prin intermediul meniului din stânga sus și accesarea link-ului **Courses**.



Figură 24 Captură de ecran cu pagina cursurilor unui profesor

Următorul pas este de a crea noul curs ce va fi introdus în cadrul aplicației, lucru ce poate fi realizat prin apăsarea butonului **New course**. După ce cursul a fost creat cu succes, acesta va fi disponibil pe pagina de **Explore courses**. În cadrul acestei pagini un utilizator are posibilitatea de căuta un anumit curs și apoi de a face subscribe asupra acestuia. Acest lucru implică faptul că utilizatorul va primi notificări atunci când un profesor postează în cadrul cursului sau atunci când se fac anunțuri cu scop general în cadrul cursului.

Course title	
Create your course by filling out the inputs from the form	
Enter title, so people know the full title of the course.	
Course abbreviation	
Enter course abbreviation.	
Course description	
Description	
Enter course description.	
Type	Evaluation
MANDATORY	EXAM
Course Hours	Seminar Hours

Figură 25 Captură de ecran cu pagina de creare curs

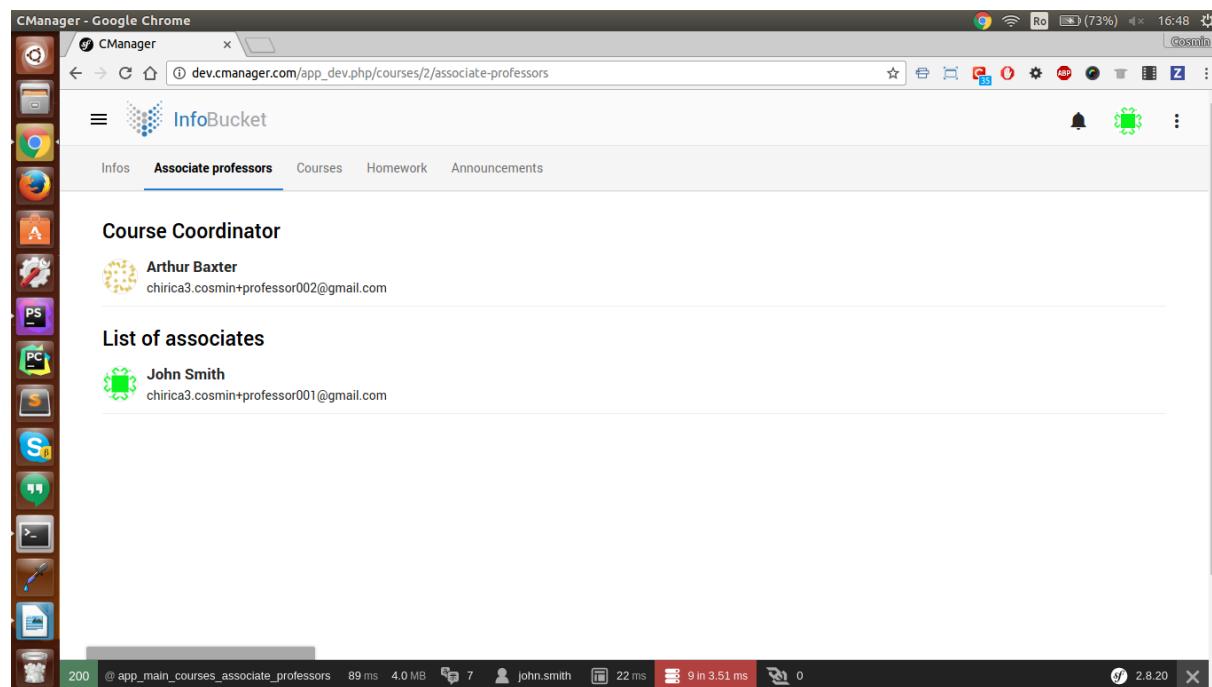
5.3 Crearea unui seminar

Am ales să prezentăm pașii necesari de adăugare a unui seminar în cadrul unui curs deoarece celelalte opțiuni precum adăgare unei teme, a unui anunț se realizează asemănător.

Un seminar poate fi adăugat atât de către un utilizator cu rolul de profesor cât și de un utilizator cu rolul de profesor asociat. Un profesor asociat poate adăuga un seminar în cadrul unui curs doar dacă este adăugat în cadrul cursului ca și asociat ce are dreptul de a preda.

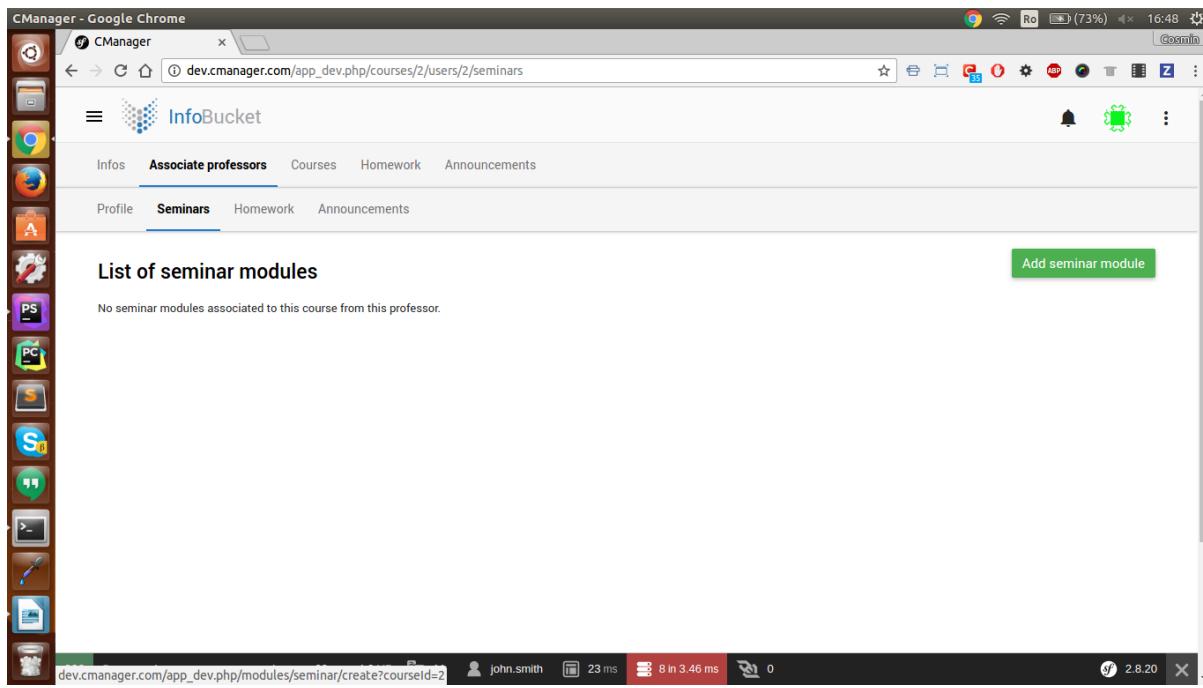
Pașii ce trebuie urmați de către un utilizator cu rolul de *profesor* sunt puțini mai simpli de descris. Utilizatorul trebuie să se autentifice în cadrul aplicației și va fi redirectat către pagina cursurilor personale. Următorul pas este de a selecta cursul dorit fiind redirecționat către pagina cursului. Pagina cursului oferă utilizatorului un nou meniu unde se află și secțiunea de **Seminars**. În continuare acesta trebuie să acceseze acestă pagină și să adauge un nou seminar.

În cazul unui utilizator cu rolul de *profesor asociat* mai intervin câțiva pași intermediari. Odată autenticat în cadrul aplicației, acesta trebuie să selecteze un curs din lista de cursuri. În acest fel acesta va accesa pagina cursului iar din secțiunea **Associate professors** va trebui să se selecteze pe el însuși din lista de profesori asociați, conform Figurii 26.

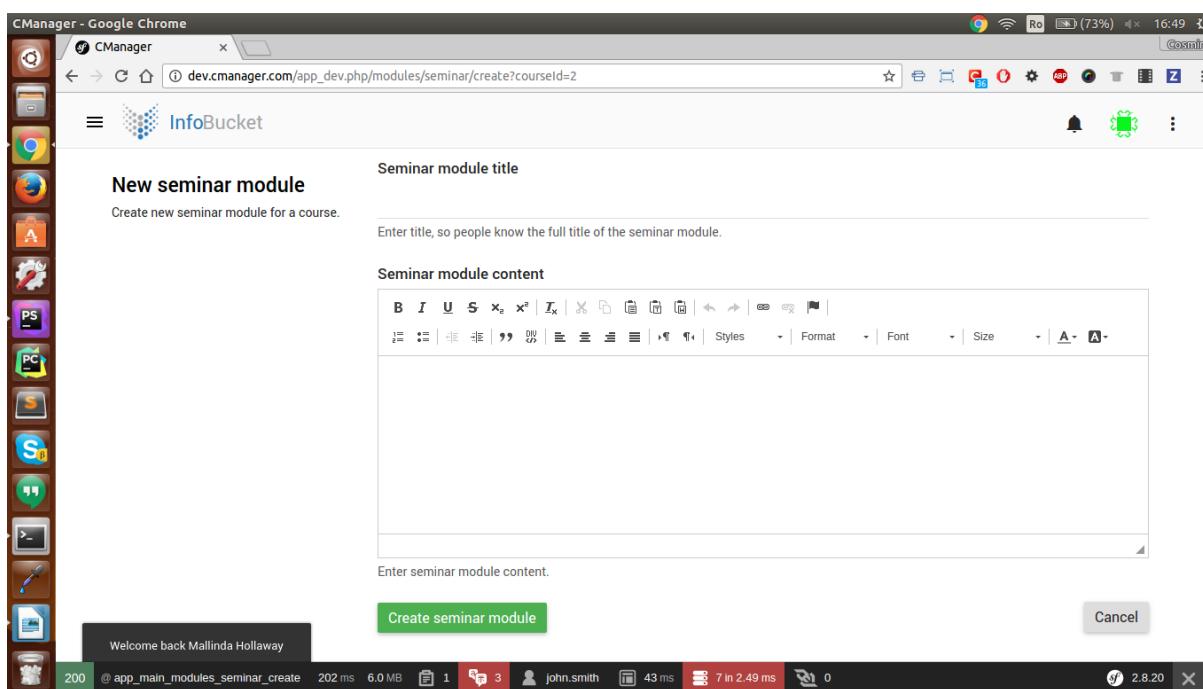


Figură 26 Captură de ecran cu lista profesorilor asociați unui curs

În urma selecției făcute la pasul anterior, un nou meniu va apărea în cadrul aplicației. Acest meniu oferă posibilitatea unui profesor asociat să adauge seminarii, teme sau anunțuri în cadrul cursului. În cazul în care persoana autentificată este chiar profesorul selectat, acesta are opțiunea de **Add seminar module** ce îi permite acestuia să adauge un nou seminar urmând a fi redirectat către pagina de creare a unui seminar după click pe acest buton.



Figură 27 Captură de ecran ce surprinde meniul asociat unui profesor asociat



Figură 28 Captură de ecran cu pagina de creare a unui nou seminar

6 Posibilități de îmbunătățire

Prin intermediul acestei aplicații am urmărit în special a crea un tool ce oferă posibilitatea de a ușura felul în care studenții au acces la un set de informații din cadrul unui curs cât și de a facilita comunicarea acestora cu profesorii universitari.

Cu toate acestea, după o analiză atentă asupra aplicației putem preciza că există multiple modalități de îmbunătățire a aplicației.

O prima posibilitate de a îmbunătăți aplicație este de a crea posibilitatea de a adăuga o extensie a aplicației prin intermediul căreia pot fi verificate cunoștințele studenților. O astfel de opțiune introdusă în cadrul aplicației ar aduce un bonus imens deoarece toate testele teoretice ar putea fi automatizate prin intermediul platformei online.

Ne mai putem gândi de asemenea la perfecționare sistemului de notificări oferind posibilitatea de a ofer unui utilizator mai multe tipuri de notificări. Un exemplu în acest sens ar fi ca atunci cand un utilizator lasă un comentariu în cadrul unui curs/seminar/teme ce ai lăsat și tu un comentariu, să se trimită o notificare de tip push ce scoate în evidență că există o nouă activitate în cadrul postării respective.

O ultimă îmbunătățire ar mai putea fi de a adăuga în cadrul proiectului un framework de JavaScript cum ar fi *ReactJS* sau *VueJS*. Acest lucru ar aduce un mare plus pe partea de *user experience* deoarece toată interfața ar putea oferi o fluiditatea mult mai mare aplicației. Acest lucru ar implica crearea unui *API(Application Programming Interface)* care să pună la dispoziția framework-ului de front-end diverse endpoint-uri pentru a lucra cu datele din aplicație.

Concluzii

Ca și o notă de final putem spune că am realizat o aplicație web ce se ridică la standardele actuale utilizând cele mai noi tool-uri în materie de aplicații web. În plus, prin prezentul document oferim o documentație detaliată asupra prodului final ce surprinde toate etapele necesare dezvoltării unei aplicații software precum: *analiza cerințelor, proiectarea arhitecturală, proiectarea detaliată, implementarea codului, verificarea aplicației* cât și un *manual de utilizare* al aplicației.

În cadrul proiectului am folosit tehnologii moderne de dezvoltare a proiectelor web precum stiva LAMP, atenția sporită fiind asupra framework-ului PHP *Symfony2* ce vine la pachet cu un sistem de management a baze de date ce poartă numele de *DoctrineORM*. Trebuie precizat și faptul ca această aplicație este una responsive/fluidă întrucât aceasta poate fi folosită cu ușurință de pe orice dispozitiv mobil sau tabletă conectate la internet.

Prin intermediul acestei lucrări scrise cât și a proiectului practic, consider că am surpins și am pus în practică destul de bine majoritatea noțiunilor învățate și prezentate în cadrul facultății.

Bibliografie

1. **PHP Docs**
[Online] <http://php.net/docs.php>
2. **Symfony Docs**
[Online] <https://symfony.com/doc/current/index.html>
3. **Symfony2 book**
[Online] https://symfony.com/pdf/Symfony_book_2.8.pdf
4. **Symfony2 cookbook**
[Online] https://symfony.com/pdf/Symfony_cookbook_2.8.pdf
5. **Doctrine Docs**
[Online] <http://docs.doctrine-project.org/projects/doctrine-orm/en/latest/>
6. **Composer Docs**
[Online] <https://getcomposer.org/doc/>
7. **Twig Docs**
[Online] <https://twig.sensiolabs.org/doc/2.x/>
8. **Twitter Bootstrap**
[Online] <http://getbootstrap.com/>
9. **Materialize framework**
[Online] <http://materializecss.com/>
10. **Publish-subscribe pattern**
[Online] https://en.wikipedia.org/wiki/Publish%20-%20subscribe_pattern
11. **WebSocket**
[Online] <https://www.html5rocks.com/en/tutorials/websockets/basics/>
12. **Socket.io Docs**
[Online] <https://socket.io/docs/>
13. **MVC pattern**
[Online] <https://medium.freecodecamp.com/model-view-controller-mvc-explained-through-ordering-drinks-at-the-bar-efcba6255053>
[Online] <https://realpython.com/blog/python/the-model-view-controller-mvc-paradigm-summarized-with-legos/>