Platforma pentru colectare/comparare teme de laborator

Proiect realizat de Grupa B2 in cadrul disciplinei IP. Profesor Lect. dr. Pistol Ionut

Pagina descriere proiect: https://fii-b2-2018.gitlab.io/cs-Assignment/

Repository proiect: https://gitlab.com/fii-b2-2018/ip-proiect.git

Tool de organizare: https://gitlab.com/fii-b2-2018/ip-proiect/boards

1. Ce trebuie făcut? Ce se dă? Ce se cere?

Descriere proiect:

1. temele pot fi comparate atat automat cat si vizual, pe platforma
2. genereaza rapoarte pe teme primite, autori teme, teme materie, etc.
3. permite exportarea temelor si a rapoartelor in formate editabile

Alte proiecte/platforme/companii care fac acelasi lucru:

1.1. Moodle
1.2. pbInfo
1.3. Blackboard
1.4. Site PSGBD & Site Retele

1.1 Moodle

Moodle este o platforma care le furnizeaza profesorilor si studentilor modalitati de a crea un mediu pesonalizat de invatare prin intermediul unui sistem integrat si sigur. Moodle livreaza un **set de functionalitati care sustin un mediu colaborativ si centrat pe actul de invatare**, astfel atentia fiind concentrata si pe profesor, dar si pe studentul care beneficiaza de respectivele materiale. De asemenea, Moodle este usor de utilizat din prisma interfetei simple cu utilizatorul, dar si a functionalitatilor drag-and-drop care faciliteaza si mai mult lucrul pe platforma.

In cadrul proiectului nostru vizam un set de functionalitati similare cu cele intalnite pe Moodle si anume: actiuni specifice fiecarui membru, indiferent de statut(profesor, student sau administrator), care includ crearea de conturi, crearea de noi cursuri pe diverse tematici, asignarea studentilor la un anumit curs initiat de un profesor, trimiterea de teme spre evaluare de catre studenti. De asemenea, o alta serie de functionalitati pe care dorim sa le implementam si sunt deja intalnite pe Moodle sunt urmatoarele: generarea de rapoarte privitoare la temele primite si autorii respectivelor teme, dar si exportarea temelor si rapoartelor in diverse formare editabile.

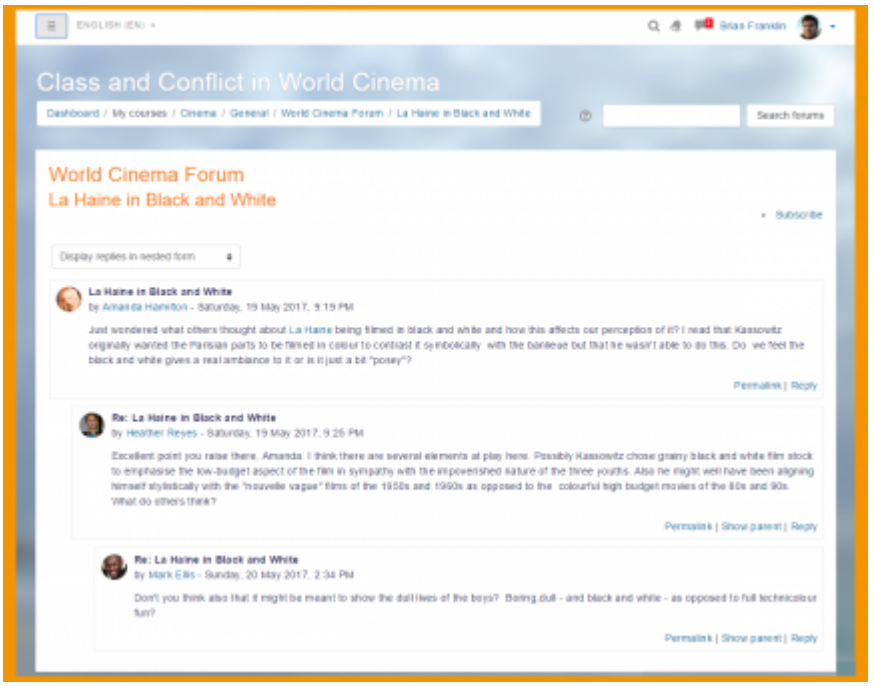
Pe langa ceea ce ofera Moodle la nivel de interactiune cu utilizatorii, ne propunem anumite functionalitati care inca nu sunt suportate pe aceasta platforma de invatare. Spre exemplu, temele care sunt trimise spre evaluare de catre studenti vor putea fi comparate intre ele si verificate contra plagiatului.

Moodle are urmatoarele tipuri de utilizatori:

- Administrator - are rolul de a mentine platforma actualizata, de a instala noi plugin-uri care se pliaza pe cerintele utilizatorilor, dar si de a intretine platforma in sensul unei bune functionari.
- Profesor - are rolul de a crea cursuri si de a furniza materiale de invatare studentilor. De asemenea, profesorul are rolul de a adauga studentii in cadrul unui curs si de a propune teme care se pliaza pe continutul livrat.
- Student - are rolul de participa activ in cadrul cursurilor initiate de catre profesori si de a trimite teme atunci cand sunt solicitate.

Metode de comunicare

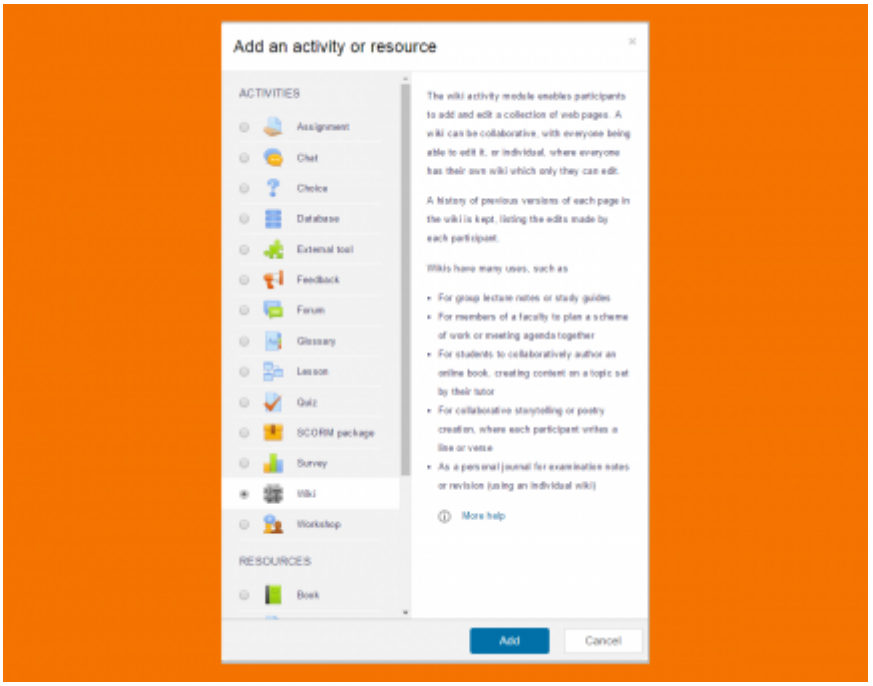
Nelipsit de la orice curs in Moodle, este un forum unde studentii si profesorii pot comunica. Acesta functionalitate este pilonul unui platforme colaborative care sustine procesul de invatare. Studentii mereu au intrebari asupra temelor, poate au idei indraznete, poate nu au citit cu atentie, poate au crezut ca tema e mai complexa decat este, poate enuntul este neclar, orice ar fi, posibilitatea de a avea forum-uri este bine venita, insa nu este obligatorie in proiectul nostru.



Moodle - Forum

Adaugarea activitatilor si resurselor

Intr-un curs, profesorul poate adauga diferite activitati si resurse. Activitatile sunt lucruri pe care elevul le poate face (de exemplu: teme, forum, intrebari, chestionare), iar resursele pot fi carti, video-uri, link-uri. Se vede cum Moodle a abstractizat ce contine un curs pentru a permite o flexiblitate foarte buna.



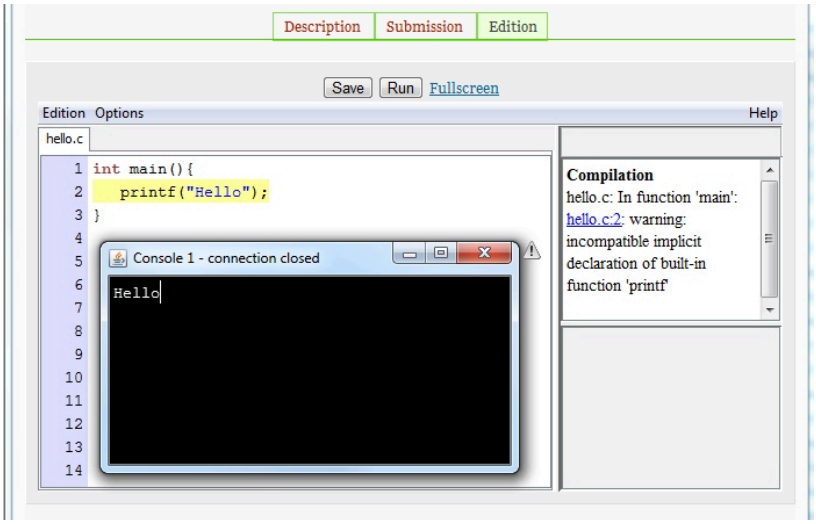
Moodle - View adaugare activitati

Moduri de desfasurare a cursului

Moodle permite crearea de cursuri cu activitati ce pot fi dirijate de către instructor, pot fi auto-paced sau amestecate. Aceasta functionalitate este **nice to have** si ne dorim sa o implementam in cadrul proiectului nostru.

Activitati specifice programarii

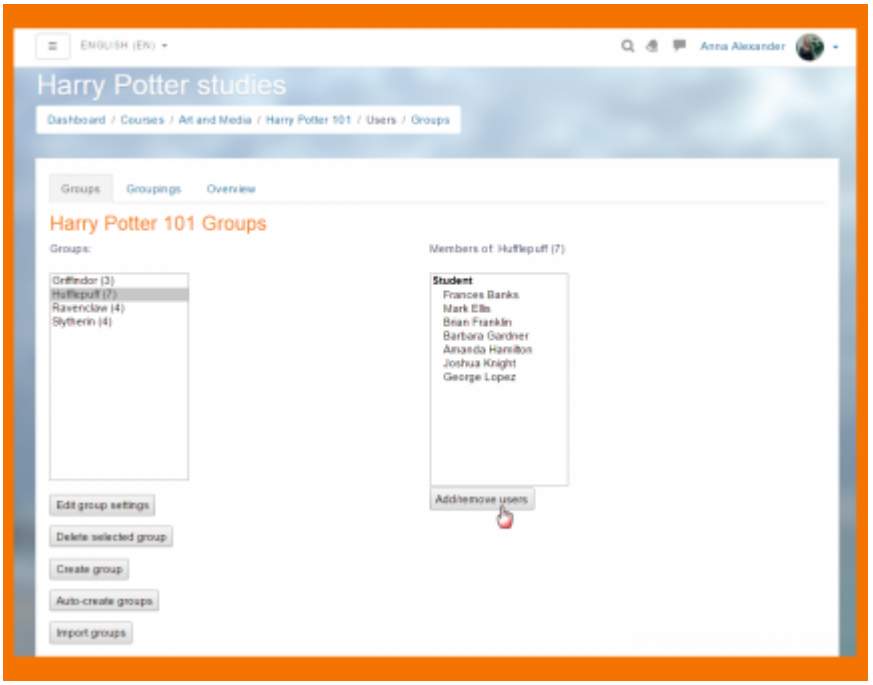
Moodle nu este optimizat pentru evaluarea temelor de programare, fiind necesara instalarea unor plugin-uri pentru acest lucru de genul VPL. Pluginul adauga un editor de cod si posibilitatea de rulare a multiple limbaje de programare (C, C++, Python, Java, etc.). Aceaste functionalitati trebuie sa existe si in aplicatia noastra.



Pluginul VPL

Lucrul in echipa

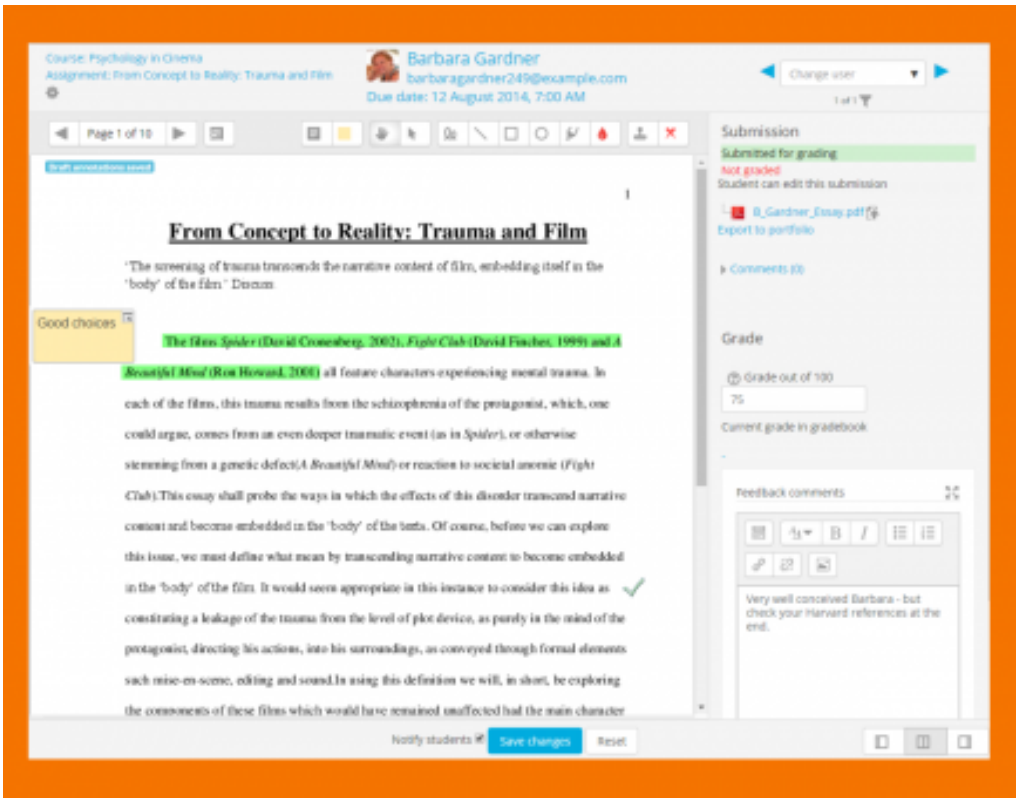
Moodle permite crearea de echipe in cadrul unui curs pentru rezolvarea temelor. Echipele sunt create de profesor, iar studentii sunt grupati tot de profesor. Acest proces va fi sigur anevoios pentru Romania, de aceea putem folosi plugin-ul Group Self Selection care permite studentilor sa-si creeze singuri echipe, cum doresc. In momentul in care o echipa vrea sa trimita o tema, e de ajuns ca un student sa o trimita, iar profesorul va putea sa-i evalueze (poate puen note si individual). Aceste functionalitati sunt necesare, deoarece temele pe echipe se regasesc des in Facultatea de Informatica.



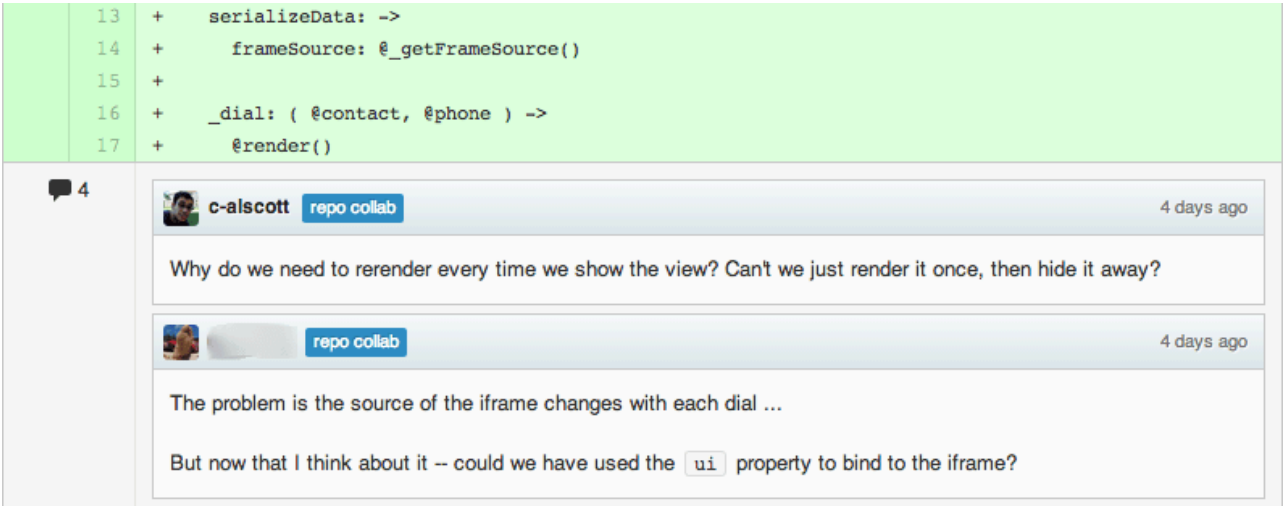
Moodle - modificarea componentei unei echipe

Evaluarea temelor si oferirea de feedback

Un aspect interesant este ca pentru evaluarea unei teme trimise sub forma de PDF este folosit proiectul open source GhostScript care aduce multa valoare procesului de evaluare, oferind posibilitatea unui feedback complex. Modul in care poate fi oferit feedback ne aduce aminte de Github (sau Gitlab / Bitbucket) unde putem comenta in cadrul unui commit pe orice linie. Aceasta functionalitate este aproape necesara pentru a asigura un proces de invatare-evaluare eficient si productiv.



Moodle - Feedback inline direct in browser (per line, notite, Comentarii, check)



Github - Feedback per line

Notarea

Profesorul evalueaza fiecare activitate in parte, iar acestea se pot vedea in catalog. Notele sunt vizibile la fiecare tema in parte, la final existand posibilitatea de a le exporta si de a genera rapoarte pe baza acestora. Studentul isi vede doar notele lui.

	Attendance			GD13 T3 Detailed Design Docs					
First name / Surname	Absences	Lates	Category total	DDD: Three Ideas	DDD: Concept Document	DDD: 1st Draft Design Document	DDD: Final Design Document	Professionalism	Course total
Range	0–7	0–7	0–10	0–30	0–50	0–50	0–75	0–20	0–100
[redacted]	-	-	10.0	23.0	38.0	38.0	62.0	20.0	85.9
[redacted]	2.0	1	0.0	14.0	22.0	35.0	53.0	20.0	66.8
[redacted]	1.0	1	2.5	23.0	35.0	41.0	59.0	20.0	77.7
[redacted]	1.0	-	5.0	17.0	31.0	34.0	43.0	10.0	57.5
[redacted]	1.0	-	5.0	21.0	36.0	39.0	58.0	20.0	78.6
[redacted]	-	-	10.0	20.0	41.0	33.0	68.0	20.0	86.9
[redacted]	-	-	10.0	21.0	39.0	29.0	55.0	20.0	78.3
[redacted]	-	-	10.0	18.0	41.0	39.0	56.0	Feedback 5/5 Game Overview	6
[redacted]	-	-	10.0	20.0	33.0	41.0	53.0	10.0	71.3
[redacted]	1.0	1	2.5	16.0	16.0	8.0	0.0	10.0	20.0
[redacted]	-	-	10.0	18.0	40.0	41.0	59.0	10.0	74.9
[redacted]	1.0	-	5.0	21.0	0.0	33.0	55.0	10.0	61.0
[redacted]	-	-	10.0	21.0	35.0	24.0	50.0	20.0	73.3
[redacted]	-	-	10.0	25.0	40.0	47.0	69.0	20.0	93.8
[redacted]	1.0	-	5.0	22.0	0.0	14.0	0.0	20.0	34.3
[redacted]	-	-	10.0	20.0	37.0	44.0	69.0	20.0	91.4
[redacted]	1.0	-	5.0	15.0	29.0	0.0	0.0	10.0	20.4
[redacted]	-	-	10.0	17.0	21.0	33.0	52.0	20.0	75.9
[redacted]	1.0	1	2.5	23.0	35.0	30.0	39.0	10.0	52.6
[redacted]	-	-	10.0	20.0	43.0	42.0	56.0	20.0	84.3
[redacted]	-	-	10.0	23.0	41.0	45.0	69.0	20.0	92.7
[redacted]	1.0	-	5.0	13.0	16.0	15.0	22.0	20.0	46.5
[redacted]	2.0	-	0.0	24.0	40.0	44.0	55.0	20.0	74.9

Moodle - Catalog

Rapoarte

Profesorii și studentii pot urmări progresul și finalizarea activitatilor la nivelul cursului. In cazul cursurilor mari unde sunt multe teme, acest mod de vizualizare este foarte benefic. Aceasta functionalitate nu este o functionalitate de baza pentru proiectul nostru.

Moodle - Situatie activitati / student

1.2 pbInfo

Pbinfo reprezinta o platforma de invatara dedicata elevilor si pasionatilor de algoritmica. Pbinfo ofera mai multe servicii care se pliaza pe programa invatamantului liceal: crearea de clase de catre profesori si asignarea temelor cu anumite deadline-uri si cerinte custom, posibilitatea de a comunica prin deschiderea unei conversatii cu toti participantii dintr-o clasa referitor la o anumita tema etc.
Punctul forte al aplicatiei este reprezentat de multitudinea de probleme, fiecare problema respectand urmatorul format : cerinta, grad de dificultate, categorie, input/output, restrictii, teste. Sistemul pune accent pe invatarea individuala(eventual, indrumata de un profesor coordonator) existand posibilitatea de **autoevaluare** real-time a solutiilor trimise si evaluate de catre compilatorul intern al platformei.Alte functionalitati intersante permit monitorizarea activitatilor unui elev de catre profesor cat si vizualizarea statisticilor periodice ce privesc evolutia individuala a acestuia, facandu-se comparatie cu temele/problemele rezolvate in semestrul curent.

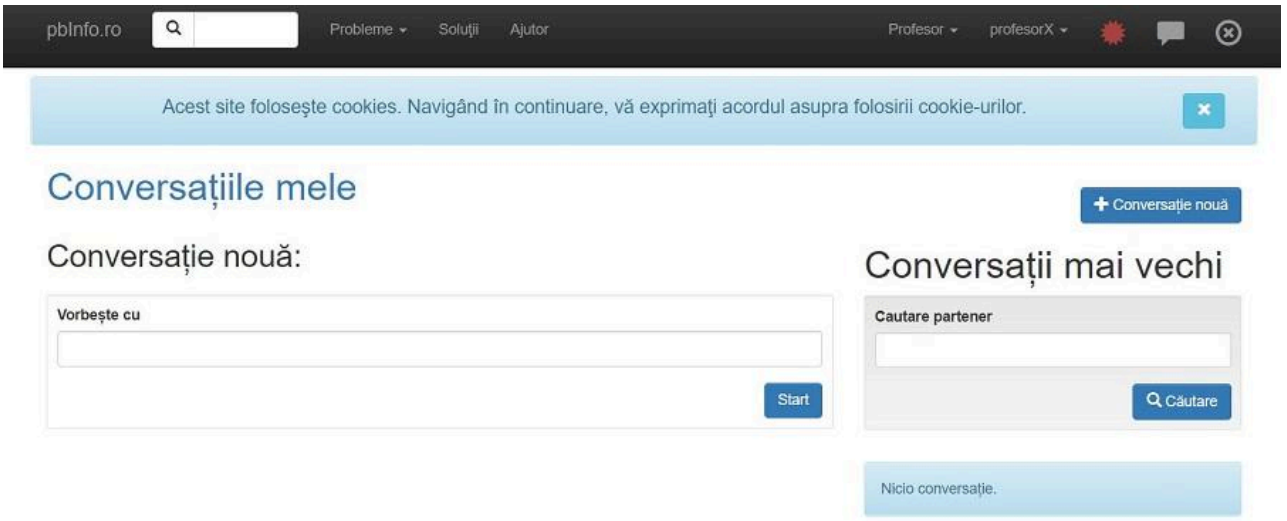
In cadrul proiectului nostru vizam un set de functionalitati similare cu cele intalnite pe pbinfo si anume: actiuni specifice fiecarui membru, indiferent de statut(profesor sau student), care includ crearea de conturi, crearea de noi clase pe diverse tematici, asignarea elevilor la o anumita clasa initiata de un profesor, trimiterea de teme spre evaluare de catre elevi,deschiderea de conversatii cu clasa pe o anumita tema. De asemenea, o alta serie de functionalitati pe care dorim sa le implementam si sunt deja intalnite pe pbinfo sunt urmatoarele: generarea de statistici privitoare la temele primite si autorii respectivelor teme, dar si exportarea temelor si rapoartelor in diverse formate editabile. O alta functionalitate pe care aspiram sa o implementam este permiterea profesorului de a seta perechi de fisiere input, output pentru fiecare tema, facilizandu-se automatizarea evaluarii respectivei teme. Astfel, un evaluator intern va compila codul sursa al unui utilizator, va compara output-ul generat de acesta cu cel *corect* si va oferi punctaje in functie de pragurile stabilite in prealabil de autorul temei pentru fiecare test. Desigur, evaluatorul ar trebui sa aiba si anumite limitari specifice cum ar fi: nepermiterea rularii de cod malitios, depunctarea pentru warning-uri generate pe parcursul executiei.

Pbinfo are doua tipuri de utilizatori:

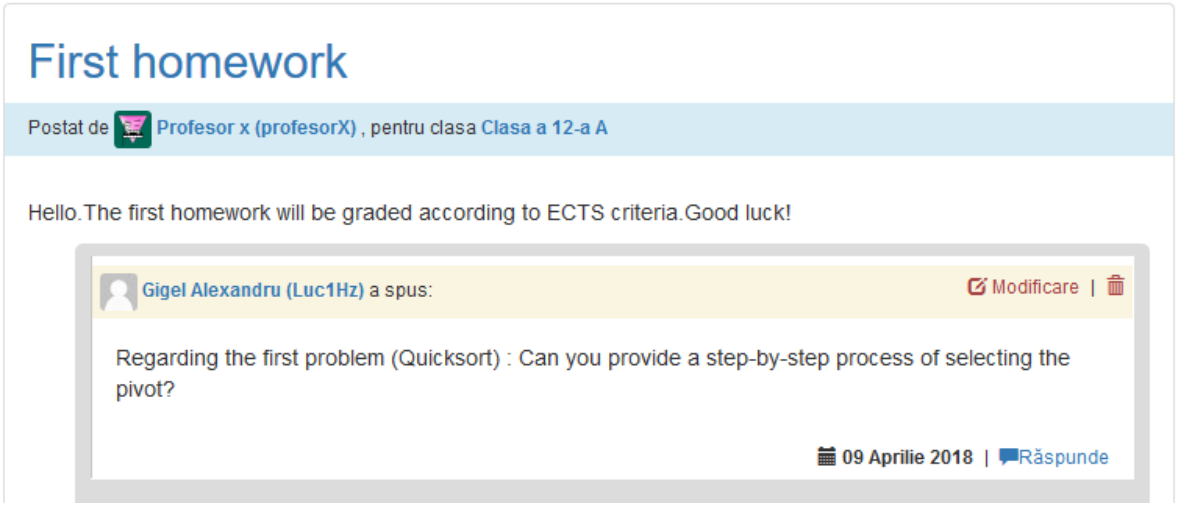
- Elev - poate adera la o clasa, rezolva teme specifice aceleia si participa la conversatii public prin intermediul postarilor din cadrul unei clase cat si privat prin modulul “Direct message”; se poate antrena individual rezolvand problemele de pe platforma
- Profesor - poate crea clase la care se pot inscrie elevi pe baza unui id al clasei si a unei parole setata de acesta; propune teme la care seteaza o lista probleme de pe platforma spre a fi rezolvate cat si data limita pana la care aceasta este valabila; propune solutii si materiale pentru o anumita problema; propune examene ; vizualizeaza solutiile elevilor.

Modul de comunicare

In cadrul pbInfo utilizatorii pot comunica prin intermediul unui chat ce le ofera de asemenea si istoricul conversatiilor.



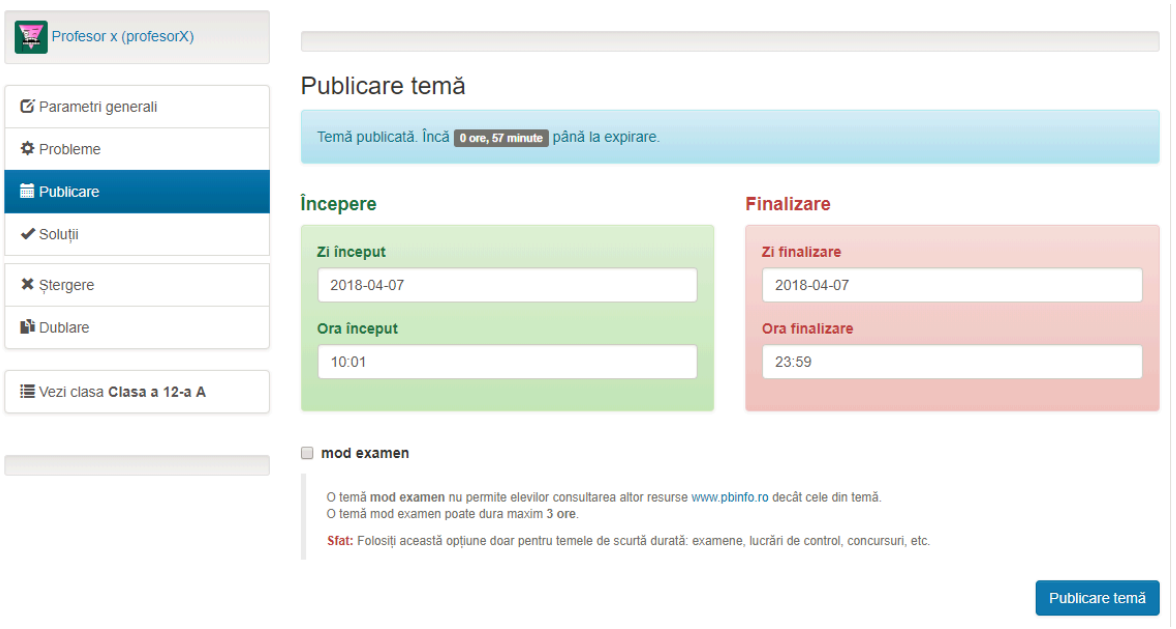
De asemenea, in cadrul fiecarei teme postate exista sectiunea de discutii si anunturi dedicata elevilor din clasa respectiva.Porfesorul si elevii pot posta diferite anunturi si intrebari.

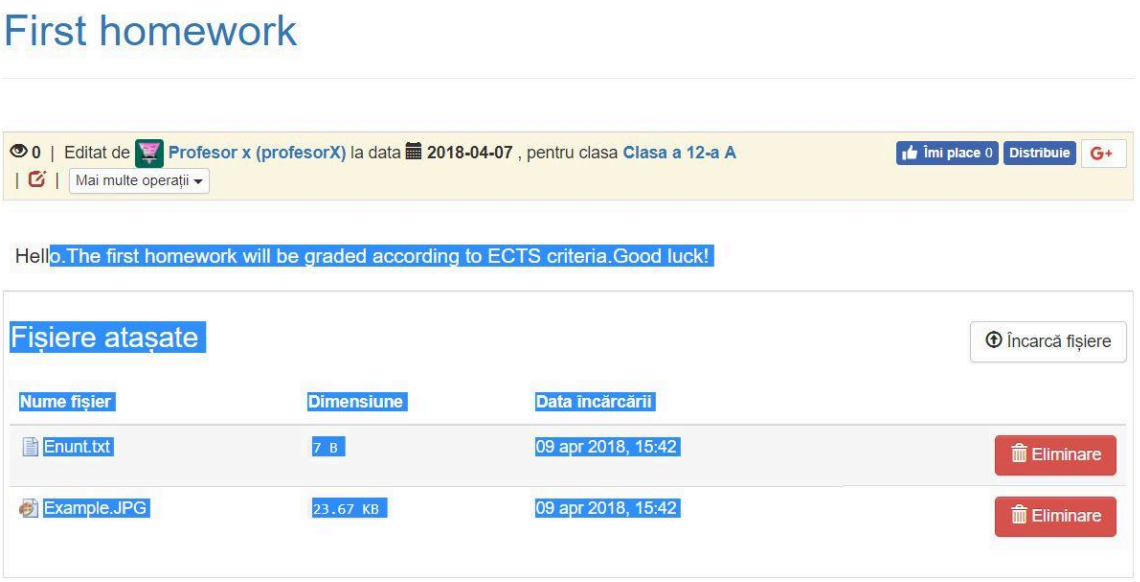


Acest lucru faciliteaza lucrul in echipa, astfel utilizatorii pot impartasi cunostinte cu ceilalti colegi.

Adaugarea materialelor

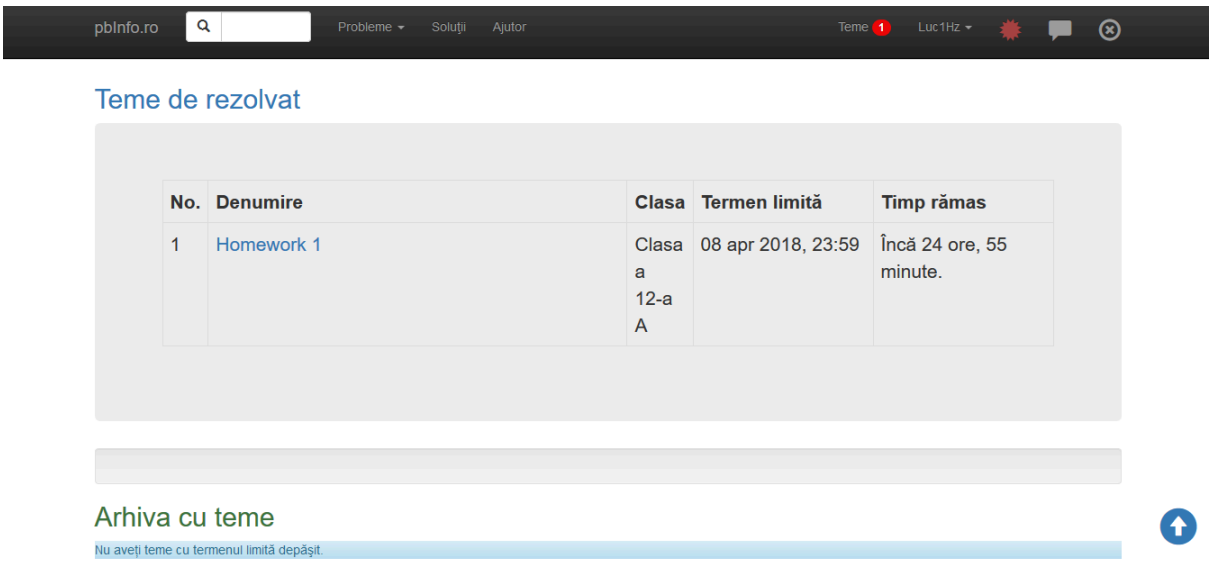
In cadrul clasei profesorul poate propune teme insotite de diferite materiale ce ajuta la rezolvarea acestora (de exemplu: carti,imagini, link-uri). Prin aceasta structurare fiecare utilizator beneficiaza de un acces mai facil la resurse.





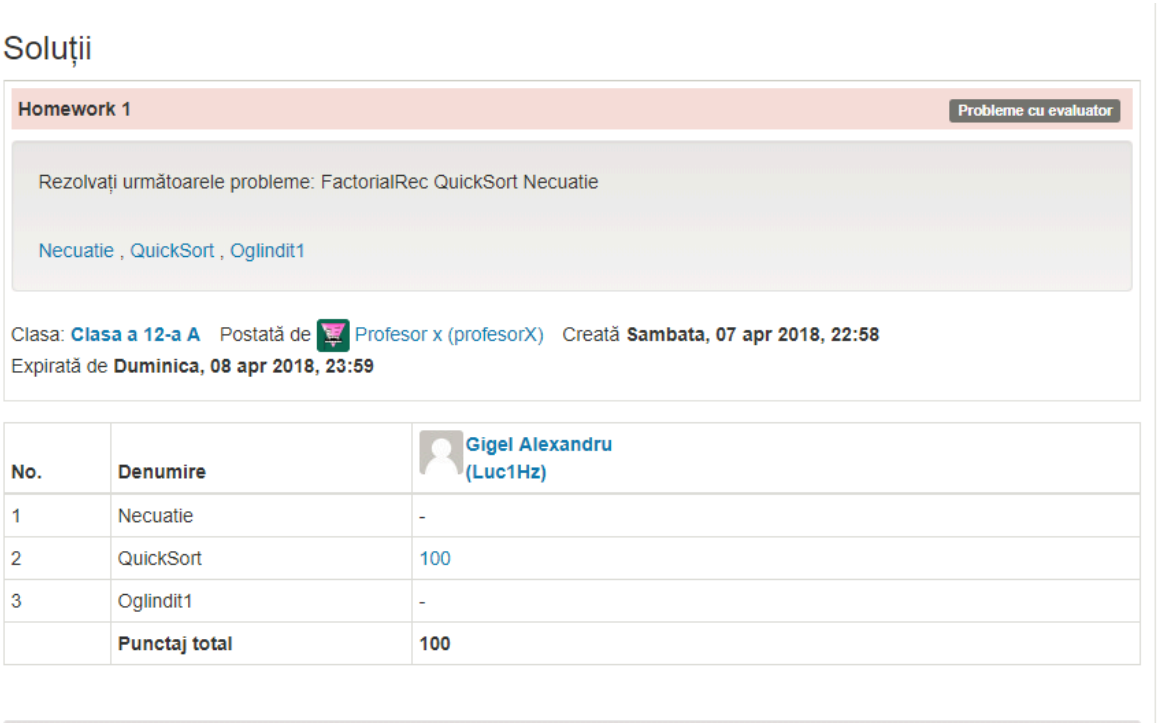
Gestionarea temelor curente de catre elev

Odata postata o tema, elevul primeste o notificare cu informatii generale despre aceasta si o legatura catre tema respectiva.



Evaluarea de catre profesor

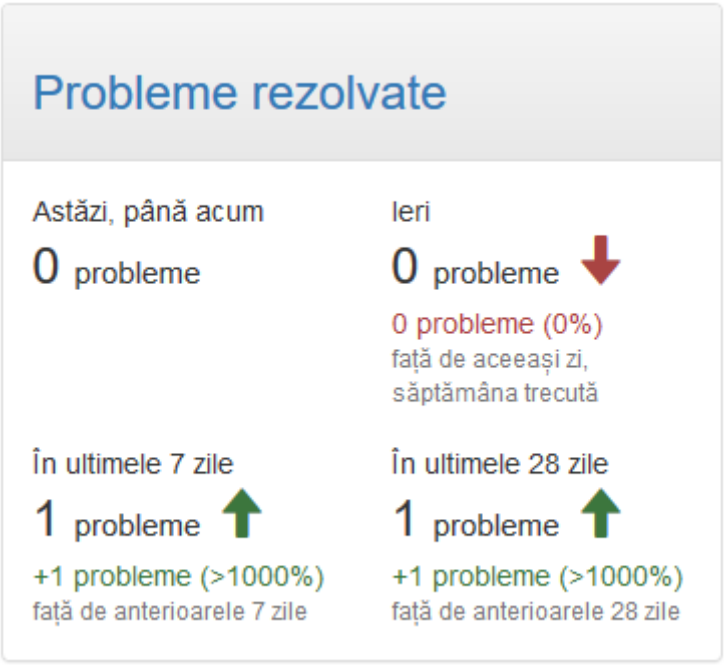
Profesorul poate vedea punctajul fiecarei elev si fisierele sursa atasate de acesta(si pentru temele care impun upload-ul si altor tipuri de fisiere).



Pregatirea individuala

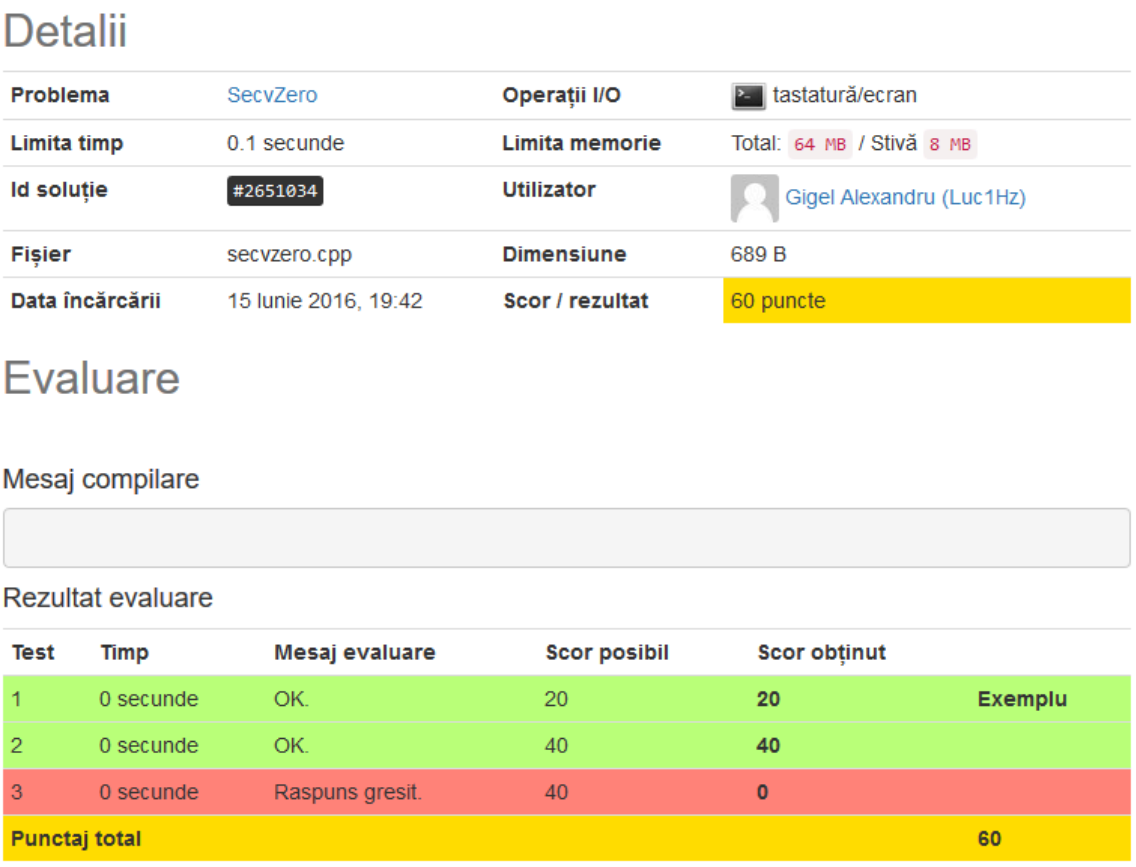
Pe pagina principala a utilizatorului sunt disponibile statistici legate de progresul acestuia.

In aplicatia noastra finala putem integra asemenea statistici care fac referire insa la progresul global(pe facultate) al tuturor studentilor care rezolva acea tema.



PbInfo - Statistici

De asemenea,pbinfo dispune de o gama larga de probleme pentru toate nivelurile de pregatire ale elevilor.
Un utilizator poate trimite codul sursa care rezolva problema data si se poate autoevalua.
In cadrul aplicatiei noastre, putem face referire la teme si exercitii de antrenament pentru fiecare tema propusa.Aceasta functionalitate ar fi **nite to have**.



De asemenea, existenta unui evaluator instant pentru a automatiza procesul de notare a temelelor este util, insa exista mai multe dificultati pe care le vom intampina la implementarea acestuia.Cum noi dorim ca aplicatia noastra finala sa se adreseze materiilor din facultate la momentul actual si unele teme sunt de asa natura incat nu pot fi

evaluate decat sub supravegherea directa a unui profesor(Ex: proiectul la retele - din pricina fork-urilor si thread-urilor evaluatorul ar putea semnala executabilele ca fiind malitioase, temele grafice la laboratoarele de grafica pe calculator si programare avansata) putem observa ca, pentru proiectul nostru, un astfel de evaluator ar fi **nice to have** insa vom analiza pe parcurs evolutia raportului dintre necesitatea unui astfel de tool si dificultatea integrarii acestuia.

1.3 Blackboard

Blackboard este o corporatie internationala care ofera solutii (platite) pentru gestionarea sistemului de invatamant atat pentru scoli (K-12) cat si pentru grupuri private, invatamant superior, ba chiar si pentru sesiuni de training organizate de diverse companii. Blackboard ofera astfel o gama larga de aplicatii software pentru organizarea si customizarea actului de invatare in orice tip de mediu.
Sistemul Blackboard se imparate in doua parti:

- Aplicatia server
- Aplicatia client

Aplicatia server

Aceasta aplicatie este customizabila in functie de nevoile personale ale clientului si impreuna cu functionalitati de baza precum:

- HD Video Conference streaming;
- OER (Open Educational Repository)
- Conținut digital îmbogățit si materiale video interactive
- Obiecte specifice invatatului cross-platform
- Integrare cu alte sisteme (Eg: Google, GoogleDrive, OneDrive, DropBox, etc)
- Sisteme de ‘tracking’, analiza si monitorizare online a activitatii participantilor
- Solutii SaaS si Cloud pentru o mai usoara gestionare

Aplicatia client

Aceasta aplicatie permite studentilor si tuturor celor care sunt inscrisi in sistem sa acceseze mediul de lucru Blackboard customizat de catre client (Eg. Scoala) pentru:

- Urmarirea propriei evolutii
- Verificarea taskurilor deja primite sau viitoare
- Uploadarea temelor
- Verificarea propriilor note si punctaje
- Accesarea materialului didactic
- Lucru impreuna cu alti colegi sau chiar personalul didactic

Tehnologii folosite

- Aplicatia Server:
    - FrontEnd: HTML5 CSS3
    - BackEnd: JavaScript si un DBMS Relational (SQL/Oracle)
- Aplicatia Client:
    - Disponibila pentru aproximativ toate platformele : Windows, Android, Apple
    - Conectarea se poate face si intr-un mod direct via browsere (HTTP si/sau HTTPS) la o adresa web dedicata, a institutiei/companiei ce a achizitionat produsul server
    - FrontEnd/BackEnd: Java/C#

Din cauza ca nu exista un account de demo si respectiv sistemul este unul pentru care trebuie platit, nu s-au putut analiza mai in amanunt modulele ce il compun.

1.4 Site PSGBD & Site Retele

Ambele site-uri au integrarea cu serverul de **LDAP** al facultatii si au drept scop verificarea plagiarismului. Ele accepta doar uploadarea cu fisiere zip care trebuie numite intr-un anumit fel, de exemplu pentru site-ul de PSGBD formatul este: nume_prenume_grupa_nr.tema.zip iar pentru site-ul de Retele trebuie doar denumire.proiect.zip. Sunt foarte multe probleme deoarece unii studenti nu inteleg sau uita modul in care trebuie trimise temele. Ambele

site-uri sunt facute in PHP iar pe front-end nu e nimic spectaculos. Site-ul de la Retele este facut cu versiunea de HTML, XHTML 1.0 Transitional, in timp ce site-ul de la PSGBD este facut cu HTML5.

LDAP(Lightweight Directory Access Protocol) este un protocol folosit pentru interogarea și modificarea serviciilor de directoare prin intermediul TCP/IP. Un director este un set de obiecte cu atribute organizate într-o structura ierarhică. Un exemplu simplu este cartea de telefoane, care conține o listă cu nume organizată alfabetic, unde fiecare nume are asociată o adresă și un număr de telefon. In cazul nostru serverul LDAP al facultatii este contine datele de autentificare atribuite fiecarui nou student la inceperea stagiului de studii de licenta, iar dupa finalizarea studiilor, acesta va fi sters pentru eficientizarea utilizarii memoriei. Acest cont ii va furniza studentului diferite servicii precum lucrulul pe server-ul facultatii, upload-ul de teme/proiecte, accesul la materiale didactice si comunicarea cu profesorii si secretariatul facultatii prin serviciul de WebMail. Mai pe scurt, toti studentii actuali ai facultatii sunt stocati ca intr-o carte telefonica pe acest server, iar cand un cadru didactic doreste accesul la datele din aceasta carte si folosirea lor pentru evaluare/distribuire materiale/colectarea temelor si a proiectelor, nu ramane decat sa se conecteze la acesta.
Inaintea fiecarui upload de teme/proiect studentului i se necesita autentificarea pe plaforma materiilor respective prin utilizarea numelui de utilizator si parolei retinute in LDAP pentru fiecare student. In acest fel i se atribuie studetului cu numele X si prenumele Y proiectul/tema cu numele A/B/C/D/E upload-at la data Z, prezentat la data T si evaluata cu nota V.
Iar in urma upload-ului tema/proiectul va fi verificat pentru evitarea plagiarismului. Aceasta verificare se poate face aplicand un agloritm care poate verifica caracter cu caracter a temelor intre ele sau prin verificarea parserului, fiind evaluate si comparate instructiunile si structurile dintr-un proiect cu instructiunile si structurile din celelalte proiecte.

2. Module

2.1 Front End

Descriere

Scopul echipei este acela de a oferi clientilor o interfata web usor de folosit, responsive, sugestiva.

Echipa

- Stiufliuc Gabriel (reprezentant)
- Motas David
- Jitaru Madalina
- Macovei Rares Alexandru
- Arcana Delia Beatrice
- Asandului Antonia Ilinca
- Ciobanu Grigore
- Dascalu Marilena
- Stefan Moraru
- Bizu Dan Alexandru
- Arhire Ionut
- Alexandru Vasile
- Gabriel Tincu

Concluzii trase pe baza analizei competiei

Din analiza facuta, ne-am dat seama ca avem nevoie de urmatoarele tehnologii:

- ReactJS + Redux
- LessCSS
- LocalStorage
- FetchAPI (eventual Axios)

Aplicatia va fi un clasic SPA, in care vom expune strictul necesar pentru a putea utiliza aplicatia propusa.
Am decis ca toate componetele create sa fie reutilizabile si sa ne folosim de REST API-ul creat pe partea de BackEnd prin anumite servicii create cu FetchAPI.
Am decis sa folosim Redux, un State Management foarte utilizat si popular la momentul actual, pentru a “stoca” si a refolosi de informatiile necesare in orice moment al aplicatiei.

2.2 Back End

Descriere

Scopul acestei echipe este de a configura si dezvolta aplicatiile care se vor afla pe server.

Echipa

- Cezar Andrici (reprezentant)
- Andreea Buterchi
- Madalina Racovita
- George Bugeag

- Miruna Vasilescu
- Diana Bolocan
- Marcu Palii
- Catalin Bujor
- Lucian Nestian
- Petru Cobzaru
- Ionut Dima

Concluzii trase pe baza analizei competiei

Din analiza facuta, ne-am dat seama ca probabil vom avea nevoie de urmatoarele componente (pe langa aplicatia web de pe server):

- Evaluarea surselor (C, C++, Java, Python, JavaScript, etc.)
- Verificarea de plagiarism
- Verificarea de cod malitios
- Logarea cu LDAP & OAuth 2
- Object Storage

Am decis ca vom dezvolta una sau mai multe aplicatii web in Node JS, ce vor expune un REST API ce va fi folosit de interfata web. Am ales Node JS din 2 motive:

1. Sunt colegi care au folosit NodeJS pe backend (la celelalte limbaje de programare posibile nu este nici unul)
2. Deoarece si colegii de la echipa de front end folosesc JavaScript, inseamna ca ne putem ajuta intre noi daca avem probleme cu limbajul de programare.

Am ales sa expunem un REST API pentru ca dezvoltarea Front End-ului si cea a Back End-ului sa nu depinda una de alta, aceasta practica fiind un standard in ziua de astazi.

Ca si framework vom folosi Express deoarece e lightweight, simplu de folosit, si sunt in echipa colegi care au lucrat cu el, care ne pot ajuta in caz ca suntem blocati.

Din proiectele analizate, Moodle este singurul Open Source, si are o structura creata in timp, in functie de nevoile pe care le-au avut. Noi vom incepe cu o arhitectura MVC (pentru ca asta am invatat).

2.3 Quality Assurance

Descriere

Echipa de QA va avea ca responsabilitate principala asigurarea indeplinirii tuturor sarcinilor de dezvoltare, punand accent pe definirea clara a criteriilor de calitate care va fi asigurata prin planificarea si executarea testelor, si urmarirea potentialelor probleme. Echipa se va axa pe automatizare, dar si pe testarea manuala, acolo unde automatizarea nu este fiabila. Standardele tuturor proceselor de calitate vor fi din ce in ce mai ridicate pe masura ce proiectul se va maturiza.

Echipa

- Cezar Cobuz (reprezentant)
- Maftei Claudiu Ioan
- Berea Mihai
- Mutu Gheorghita

Concluzii trase pe baza analizei competitiei

In urma analizei competitiei, am constat ca trebuie sa aveam in vedere urmatoarele aspecte:

- strategia noastra va fi bazata pe preventia greselilor sau a defectelor, astfel vom economisi timp
- automatizarea testelor va fi o prioritate
- platforma trebuie sa suporte mai multe tipuri de programe
- testele vor fi foarte complexe pentru a surprinde o gama cat mai ridicata de eventualele probleme/erori
- cerintele de calitate vor fi comunicate clar catre fiecare modul
- orice problema constatata va fi semnalata, in cel mai scurt timp, catre echipa responsabila si catre echipa de administrare pentru o solutionare eficienta
- vom avea in vedere si posibilele probleme de securitate aparute in proiectarea platformei

Avem in vedere testarea fiecarui use case ca punct de plecare si cel putin un test pentru fiecare modul. Scopul nostru este de a ne asigura de faptul ca produsul final corespunde cerintelor clientului si orice edge case ce ar putea produce un rezultat ambiguu este luat in calcul si tratat, incluzand si incercari neautorizate de a obtine acces/control asupra diferitelor date/module. Vom mentine un feedback constant pentru fiecare componenta ce va fi adusa in stadiu de testare, astfel construind o aplicatie stabila pe toate directiile.

Scenarii de Utilizare

1. Un nou utilizator se inregistreaza. Se va completa un form dupa care, in functie de constrangerile asupra username-ului, emailului, parolei, va fi adaugat un nou user in baza de date trimitandu-se la scurt timp un email cu datele necesare la adresa specificata. Daca constrangerile asupra datelor necesare nu sunt indeplinite, actiunile de mai sus nu se executa iar utilizatorul va fi avertizat vizual de modificarile necesare. (NOT ANYMORE)
2. Un utilizator, oarecare, al site-ului incearca sa se logheze. In functie de datele introduse, se va face o interogare catre baza de date, ce va fi procesata si va returna urmatoarele rezultate:
    - user inexistent -> avertizare vizuala catre guest
    - parola gresita -> avertizare vizuala catre guest
    - email inexistent -> avertizare vizuala catre guest
    - user si parola existente -> acces permis in aplicatie
    - interfata userului se va incarca.
3. Salvarea unei teme. Userul isi salveaza codul din mediul de lucru. Se va face o interogare catre baza de date si se va returna rezultatul acelei interogari sub forma unui mesaj catre utilizator, prin care i se va comunica rezultatul actiunii.
4. Compararea a doua teme. Daca oricare dintre teme nu exista in baza de date utilizatorul va fi atentionat. In cazul in care ambele teme sunt gasite, se vor face 2 interogari si se va folosi o librarie ce va stabili gradul de similitudine intre cele doua. Diferentele dintre teme vor fi comunicate vizual userului.
5. Utilizatorul se delogheaza. Cookie-urile se sterg urmand ca flag-ul din baza de date ce stabileste starea utilizatorului sa fie setat pe false. Utilizatorul este directionat catre pagina principala (home).
6. Utilizatorul intentioneaza sa trimita tema unui profesor pe email/cont. Apare un camp-text in care utilizatorul introduce datele necesare. O interogare pe baza de date va stabili daca acel email/cont este asociat aplicatiei si in functie de raspuns, el va primi confirmarea/infirmarea faptului ca tema a fost trimisa.
7. Introducerea unei noi materii -> Dupa logarea cu succes a unui profesor care este existent in baza de date el va putea verifica daca materiile la care preda se afla in platforma iar in caz contrar el va fi capabil de a introduce una noua. In cazul in care nu a facut aceasta verificare , in momentul in care va ajunge pe pagina unde poate insera o noua materie va primi un pop-up care ii va aduce la cunoastere faptul ca poate examina materiile aprobate de catre administrator, adica cele care sunt in uz pe platforma. De asemenea in cadrul introducerii datelor materiei, care urmeaza sa fie evaluata de catre un administrator dupa completarea acesteia, va fi verificat numele si abrevierea materiei respective pentru a fii respinsa in cazul in care exista deja una in baza de date pentru evitarea dublarii unei materii si al altor conflicte ce pot aparea pe partea de back-end. La final dupa trimiterea informatiilor reprezentand o noua posibila materie , administratorul va verifica eligibilitatea ei si o va putea taproba sau respinge fapte care vor duce la introducerea respectiv anularea materiei pe platforma.

8. Evaluarea finala a unei note -> Dupa logarea cu succes a unui profesor care este existent in baza de date el va avea drepturi pentru a reevalua notele puse in cadrul studentilor la materiile lor in cazul in care dumnealui decide ca au aparut anumite erori la nivel de platforma sau din alte motive. Profesorul va putea vedea continutul materialelor incarcate de elevi si evaluate de platforma iar in urma unei analizari concrete se va stabili daca nota va ramane aceeasi sau daca necesita o schimbare. In cazul in care eroarea s-a produs la nivelul platformei , ar trebui ca administratorul sa fie de asemenea informat, prin intermediul email-ului/contului, pentru remedierea platformei si imbunatatirii acesteia sau evaluarea cazului de exceptiei daca este unul in cauza.

2.4 Administrare

Descriere

Scopul acestei echipe este de a eficientiza organizarea proiectului pentru a asigura finalitatea acestuia.

Echipa

- Cezar Andrici (reprezentant Back End)
- Cezar Cobuz (reprezentant QA)
- Stefan Moraru (reprezentant Front End)

Concluzii trase pe baza analizei competiei

Celelalte proiecte fiind de dimensiuni mari, folosesc Scrum ca si metodologie de lucru, insa noi avand doar 7 saptamani la dispozitie, Kanban e mai potrivit. Vom folosi Issues de pe GitLab impreuna cu Boards pentru organizarea task-urilor. Astfel vom avea posibilitatea monitorizarii activitatii tuturor membrilor.

Am ales GitLab in loc de GitHub datorita urmatoarelor functionalitati:

- putem fi mai multi administratori pe repository
- putem limita cine face push pe branch-ul master

Faza 2

Diagrame:

- de clase (Front End)
- de clase (Back End)
- use-case (Se vor ocupa cei din QA)

Dar a mentionat ca vrea CAT MAI MULTE diagrame si ca ar fi bine sa avem si diagrame:

- de activitati
- de secventa
- Scenarii de utilizare
- Solutii tehnice concrete
- Sa ne apucam de scris cod
- Sa demonstram ca am inteles exact ce trebuie sa facem, in ce limbaj va fi implementarea.

Functionalitati:

- temele pot fi comparate atat automat cat si vizual, pe platforma
- genereaza rapoarte pe teme primite, autori teme, teme materie, etc.

Actors

- Application
- LDAP FII
- User
    - Guest
    - Admin
    - Student
    - Professor
- Course
- Assignment
- Homework - ce trimite un Student la Assignment
- Archive - Homeworks from past at the same Assignment
- Plagiarism Checker
- Anti-Malware
- Grade - assignment-student-grade
- Grade Catalog - per course
- Report
- Annotation
- Student Groups
- Notification

Definitions

- Enrollments - the association between an User and a Course
- Memberships - the association between a Student and a Group
- ArchiveAccess

Baza de date

TABELE

- User (id, email, token, role, createdAt, updatedAt)
  Roles:
    - Guest
    - Admin
    - Student
    - Professor

- Course (id, nameCourse)
- Session (id, userId, createdAt, updatedAt)
- Assignment (id, courseId, name, description, archiveId, dueDate, createdAt, updatedAt)
    - Ar trebui sa trecem daca e pe echipe sau individual
    - Setari pe echipe (marimea echipelor)
    - Mapare(numeSetare, valoare)
- File (id, path, createdAt)
- Homework (id, assignmentId, senderId, senderType, fileId, createdAt, updatedAt)
    - Polymorphism - sender
    - senderType = user | studentGroup
- Archive (id, name, createdAt, updatedAt)
- PlagiarismReport (id, fieId1, fieId2, report, createdAt)
- Anti-Malware (id, fieId1, report, createdAt)
- Grade (id, userId, homeworkId, grade, createdAt, updatedAt)
- Annotation (id, userId, fileId, line, body, createdAt, updatedAt)
- StudentGroup (id, name, createdAt, updatedAt)
- Notification (id, userId, description, createdAt, updatedAt)
- Enrollments (id, courseId, userId, role, createdAt, updatedAt)
- Memberships (id, userId, studentGroupId, role, createdAt, updatedAt)
- ArchiveAccess (id, archiveId, userId, createdAt, updatedAt)

User Stories

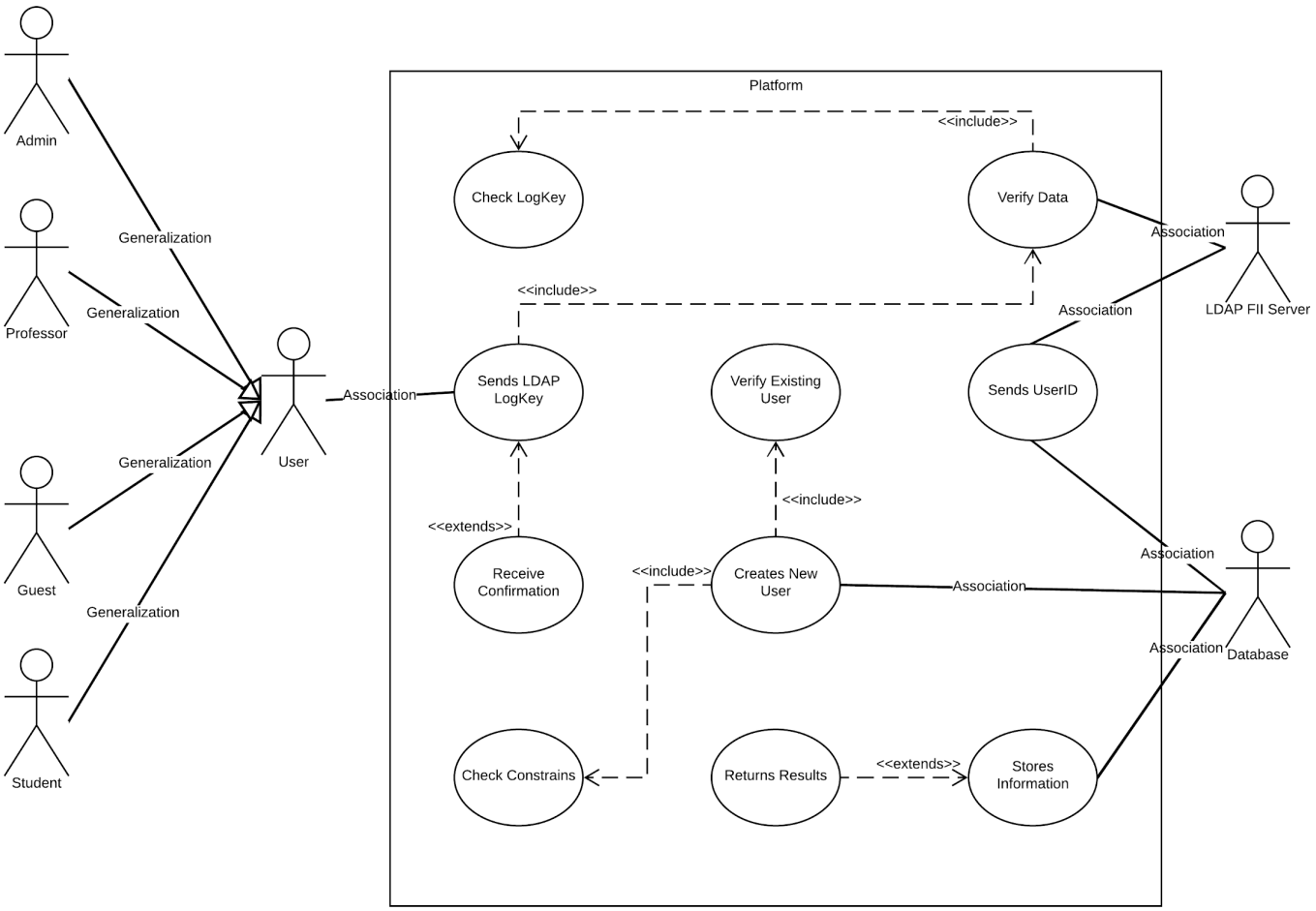
Legenda:
Functionalitati Obligatorii [ce trebuie implementate]
Functionalitati Extra [ce vor fi implementate]
Functionalitati Extra [ce ar putea fi implementate]

As a ...	**I want to ...**	**Because ...**
Guest	Login with LDAP FII	
Professor	CRUD Courses	
Professor	Assign other professors to a course	
Professor	CRUD Courses' Members	
Professor	CRUD Archives	
Professor	CRUD Assignment (settings: how many Homework, set deadline, opening, size limit, choose type group/individual)	
Plagiarism-Checker	Check Homework for plagiarism (in archives if necessary)	
Professor	Compare two pieces of Homework visually	
Professor	Grade Homework	
Professor	Export grade catalog to CSV/XLSX	
Professor	See/Download system's reports	

Professor	Annotate lines of code/file (from Homework)	
Professor	Add a forum to a course	
Student	Post in a forum	
Student	See My Courses	
Student	See Assignment	
Student	Send Homework to Assignment (with details (eg. how much each has contributed))	
Student	See grade for Homework	
Student	CRUD groups for Assignment / Join Group	
Student	Receive a mail when I was graded	
Admin	Manage Roles	
Application	Generate reports on students activity if there is a .git folder in the Homework	
Application	Compile Homeworks and check if they are working correctly	
Anti-Malware Checker	Check Homework for malware	
Application	Generates reports for received homework, authors, subjects	
Application	Have some kind of trello functionalities	

Diagrame Use Case

1. Login with LDAP FII



User (Can be Admin, Professor, Student and Guest):

- Sends a login key to the API of the external LDAP server
- Receives a login confirmation (be it positive or negative) from the LDAP server

LDAP Server:

- Verifies the data/LogKey sent by the User
- Sends a login response/request (Sends UserID) to the Application

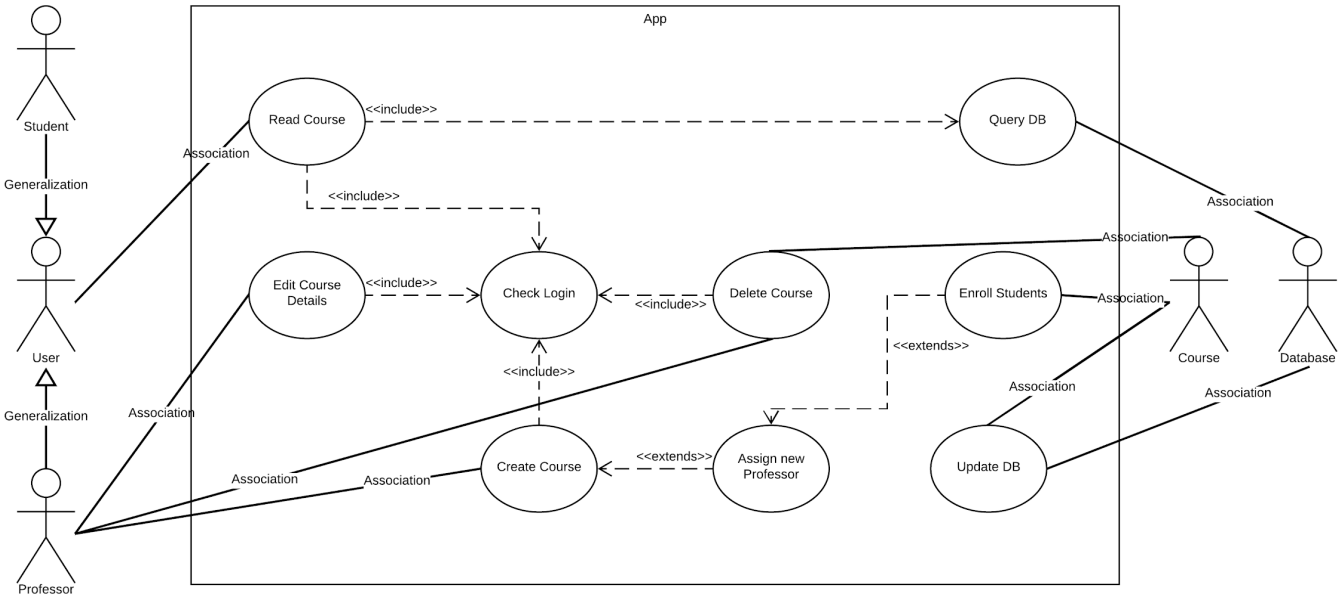
Application:

- Receives the login requests from the LDAP server
- Queries the database to compare the UserID
- If the UserID exists then it sends a confirmation and the acceptance of the login to the User
- If the UserID doesn't exists in the Database then it creates a new account and sends the needed Information to the User

Database:

- Stores the information/data sent by the Application
- Offers an answer (in the form of information/data) for the Application 'query'

2. Professor - CRUD courses / members, assign other professors to a course



Professor:

- Can create a course (this action is performed after checking his login and rights)
- After creating a course they can Assign Professors to the Course(this action is performed after checking their login and rights)
- After assigning professor(s) to the course he can Enroll Students to the course(this action is performed after checking their login and rights)
- Can edit a course's details: assign new professor(s), enroll new students (this actions are performed after checking their login and rights)
- Can delete a course (this action is performed after checking their login and rights)

Student:

- Can read/access/view a course
- Can be added to a course by a professor

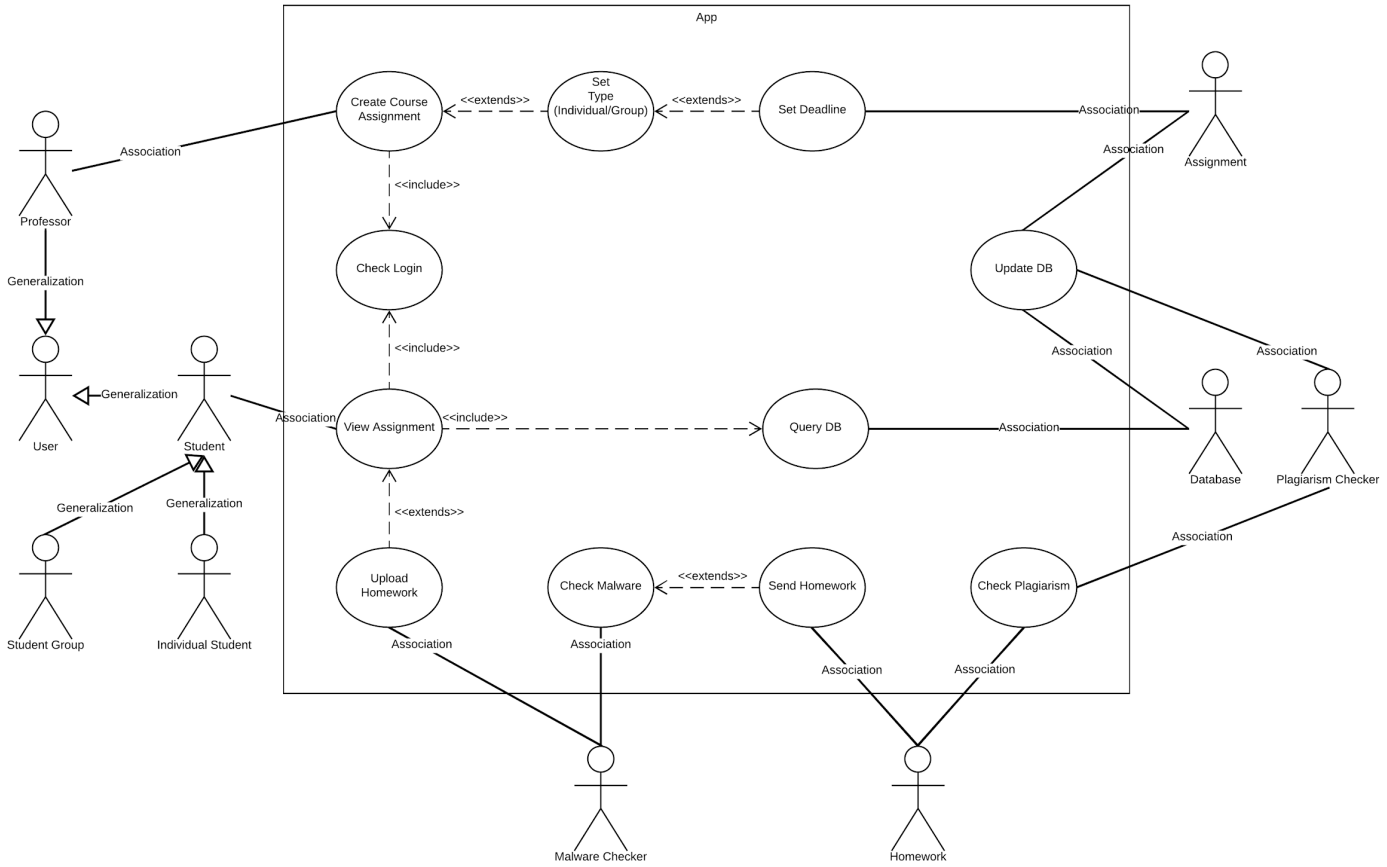
Course (System/Controller):

- It is called after completing a series of actions (create course-> assign new professor -> enroll students | edit course details | delete course)
- It stores and updates all the relevant data about course in the database

Database:

- Stores information/data in the DB
- Returns results (data/information) to User queries

3.Professor CRUD Assignment + Student View, Upload, Send Homework



Professor:

- Can create an assignment for a course (this action is performed after checking their login and rights)
- After creating an assignment they can set its type (Individual User or Group of Users) (this action is performed after checking their login and rights)
- After setting the assignment type they can set a deadline for the assignment (this action is performed after checking their login and rights)
- Can view assignment (this action is performed after checking their login and rights)

Assignment(System/Controller):

- It is called after completing the creation of an assignment for a course
- It is a system/controller that stores the relevant data of an assignment in the database

Student (Can be an individual Student or a group of Students):

- Can read/access/view an assignment (this action is performed after checking their login and rights)
- After accessing an assignment they can upload a homework (this action is performed after checking their login and rights)

Malware Checker(System/Controller):

- It is called after a Student uploads a homework
- It will check the uploaded homework for various malware and security risks; it will prompt an error in case of detection; if no detection is found (the uploaded homework is safe) it will send it forward, calling the Homework system/controller

Homework(System/Controller):

- It is called after the homework is checked and confirmed to be safe by the Malware Checker
- It will perform a Plagiarism check of the homework and then call the Plagiarism Checker system/caller

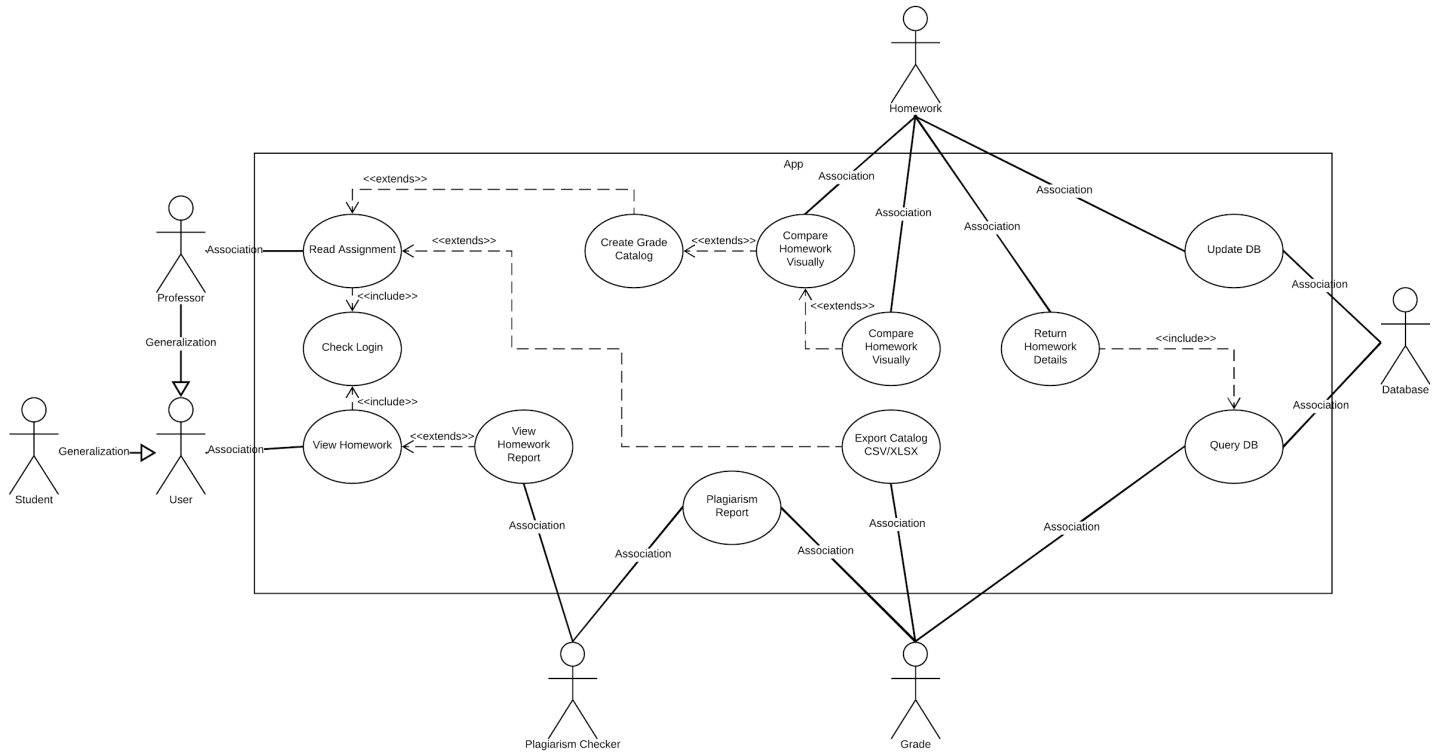
Plagiarism Checker(System/Controller):

- It is called after the plagiarism checking of a homework
- It will store data relevant to the plagiarism check of the homework in the database

Database:

- Stores information/data in the database
- Returns results to the user queries

4. Professor - Compare Homeworks, Grade Homework, View/Download Grades



Professor:

- Reads(accesses/views) the assignment
- After accessing the assignment he can create a grade catalog for the assignment
- After creating the grade catalog he can compare the uploaded homework both visually and functionally through the framework
- Can associate a grade to the homework that they view
- Has access to a functionality that allows them to export a catalog in CSV/XLSX format
- Can view homework details that are of interest (in a highlighted manner)

Student:

- Can view their previously uploaded assignments of homework
- Can view the report of their current plagiarism rate

Plagiarism Checker(System/Controller):

- It is called after a student's request to view a homework report
- It will generate a report of the plagiarism rate

Grade(System/Controller):

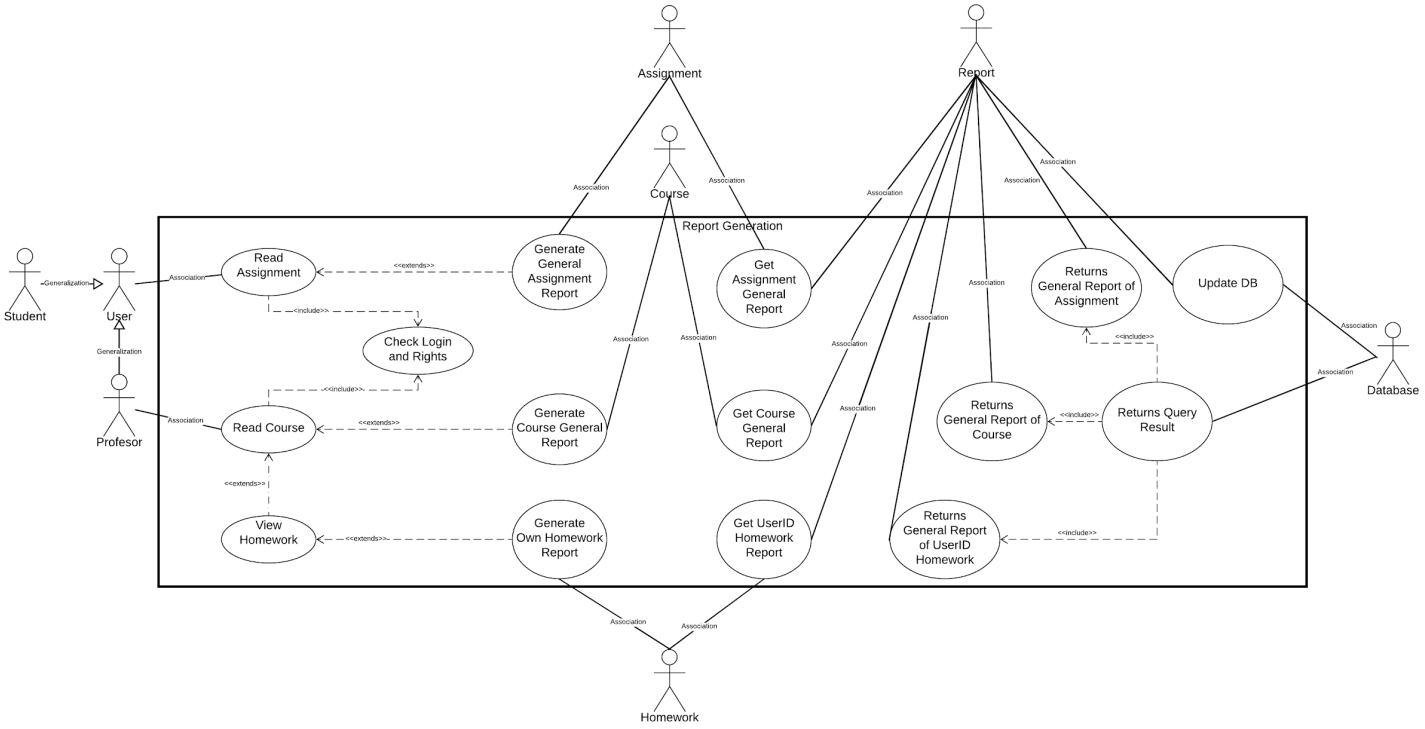
- It is a system controller that is called after the plagiarism report is completed
- It returns query results from the database (relevant report info about the homework, grades, etc)

Grade Catalog(System/Controller):

- Contains an organised representation of al the grades that are currently assigned

- It can be exported in CSV/XLSX format

5. Reports generation for received homework, authors, subjects:



Professor:

- Can read the course and assignments ;
- Generates a general assignment report .

Student:

- Can read the course ;
- Can view their own homework ;
- Generates own homework report.

Assignment:

- It is a system/controller that gets the relevant data of an assignment and sends it to the report system/controller.

Report:

- Returns general information about assignments , courses and UserId's homework.

Course:

- It is a system/controller that gets the relevant data of a course and sends it to the report system/controller.

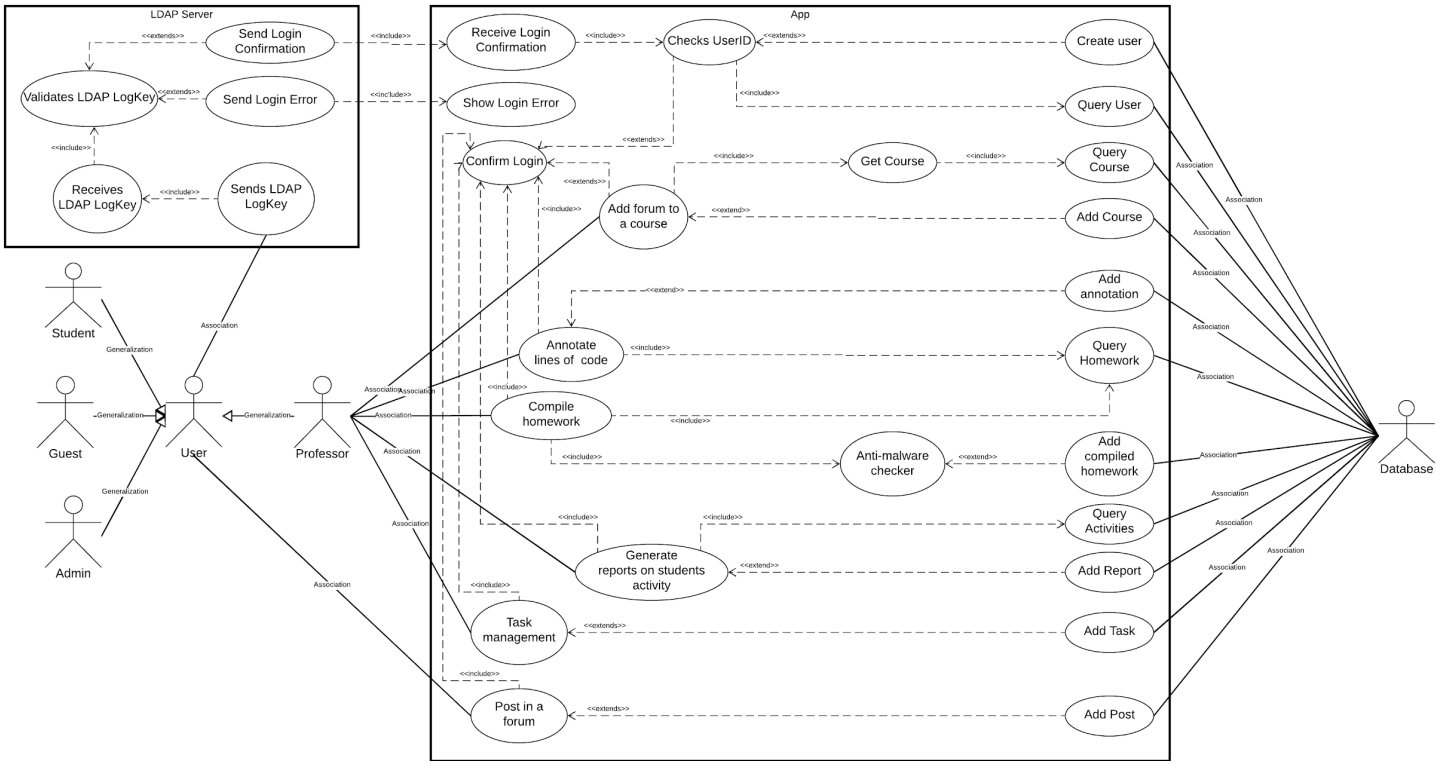
Homework:

- It is a system/controller that gets the relevant data of a homework and sends it to the report system/controller.

Database:

- Stores information/data in the database;
- Returns results to the user queries.

6. User - Extra



User (Guest, Student, Professor, Admin):

- Sends the LDAP LogKey to the LDAP Server
- Can receive a Login Error from the LDAP Server
- Receives the Login confirmation (may create an account if not already created, case in which the UserID assigned is the one from LDAP)
- Can add annotations to code lines (if they have the rights; meaning is a logged in Professor)
- Can add a forum for a course (if they have the rights; meaning is a logged Professor)
- Can makes a post within a forum (if they have the rights; meaning is a logged in Professor)
- Automatically generates activity reports (if the User is a Student)
- Can compile a homework assignment (if they have the rights; meaning is either a logged in Professor or logged in Student)
- Can do task management for a homework assignment (if they have the rights; meaning is either a logged in Professor or logged in Student)

LDAP Server:

- Receives LDAP LogKeys
- Sends Login Errors in case of wrong credentials
- Sends Login Requests to the Application

Database:

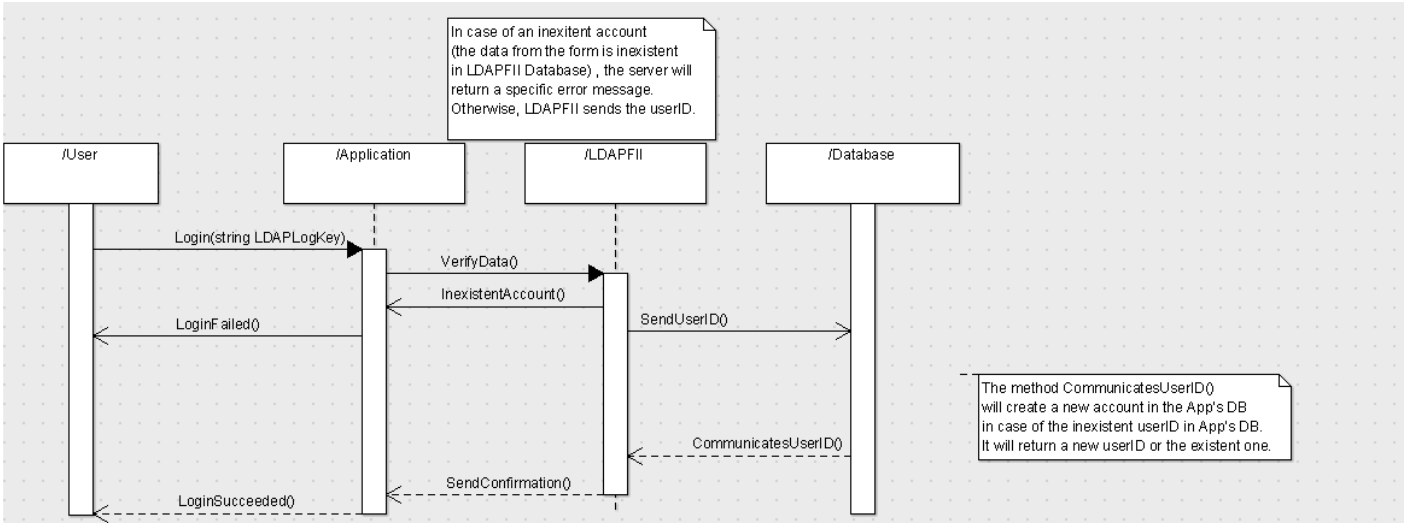
- Stores data
- Returns query results

Application:

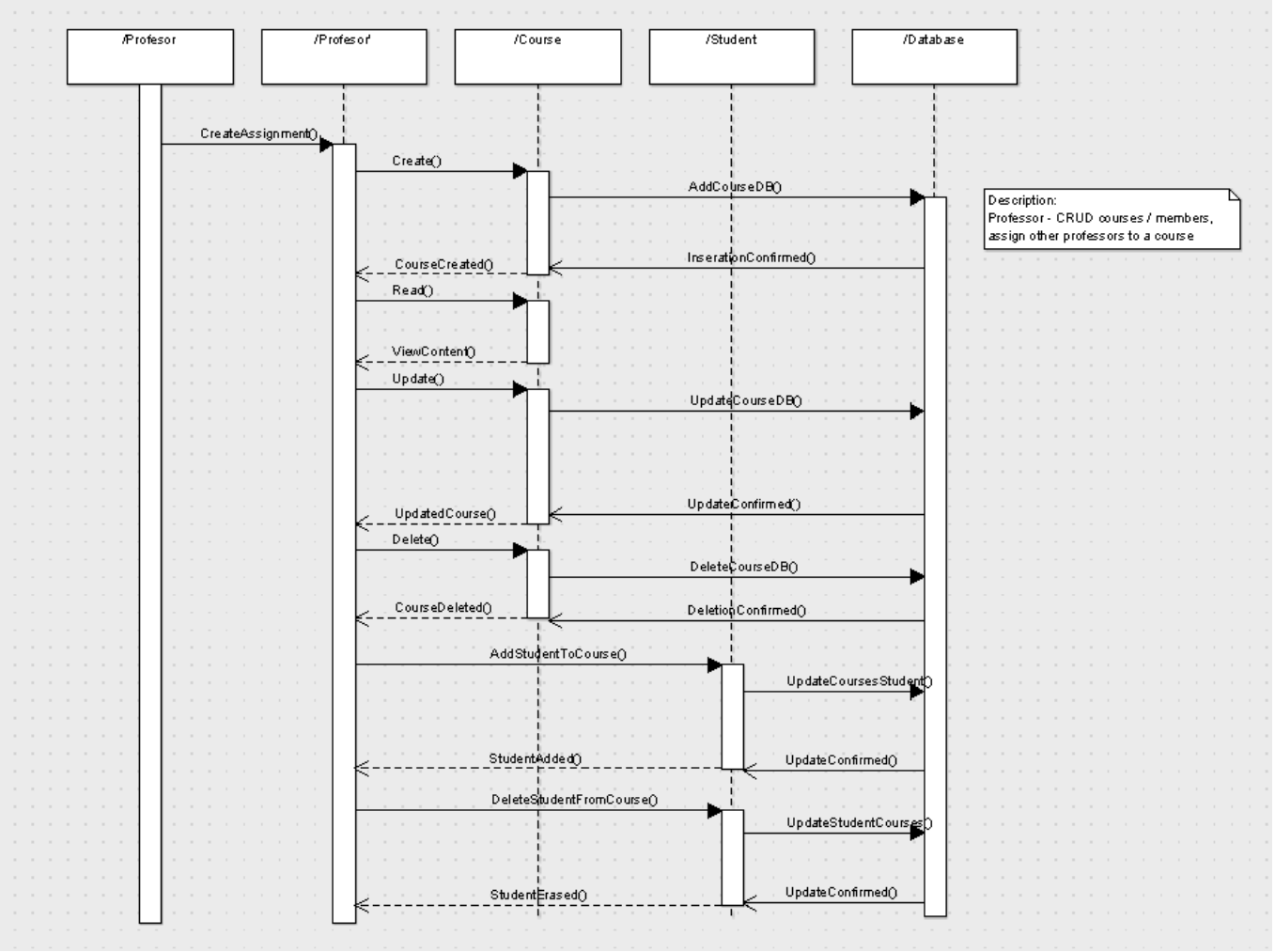
- Receives Login Requests from the LDAP Server
- Verifies whether or not the UserID exists within the LDAP Database
- Sends a confirmation to the User (may create an account if not already created, case in which the UserID assigned is the one from LDAP)

Diagrame secventiale

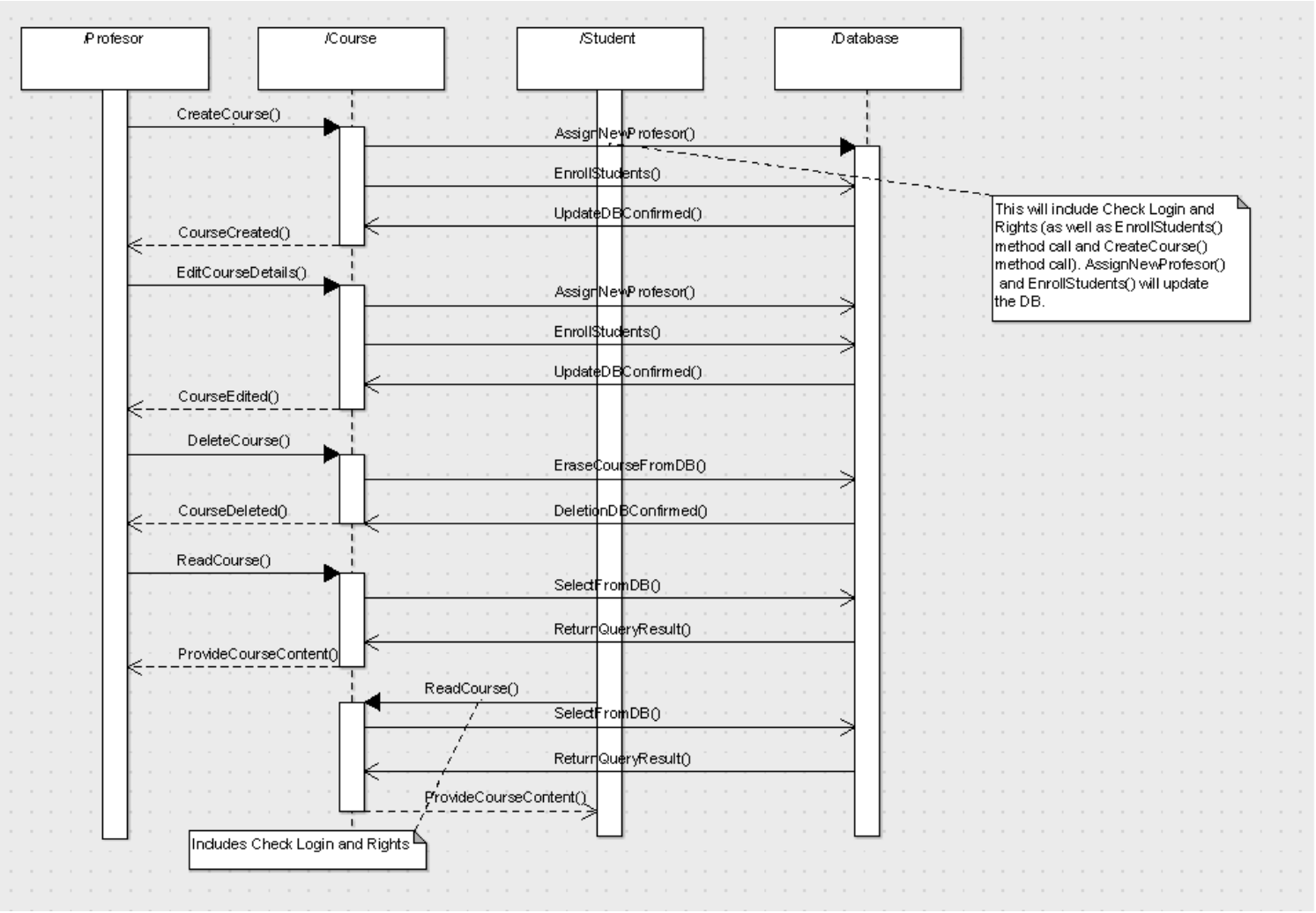
1. Login with LDAP FII



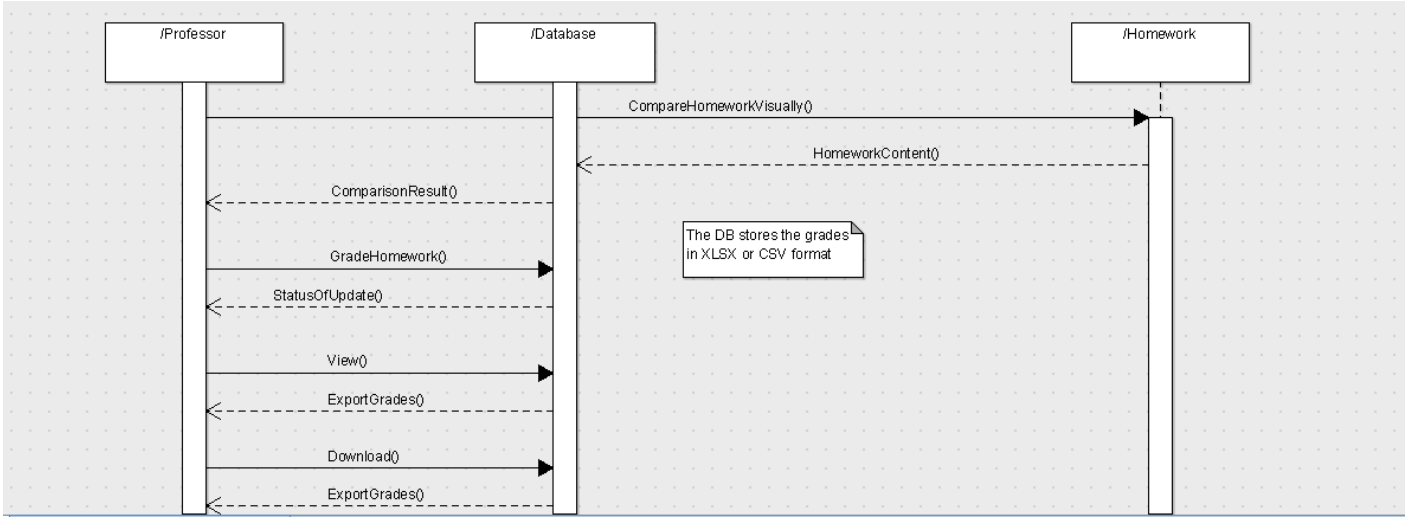
2. INITIAL Sequence Diagram for : Professor - CRUD courses / members, assign other professors to a course



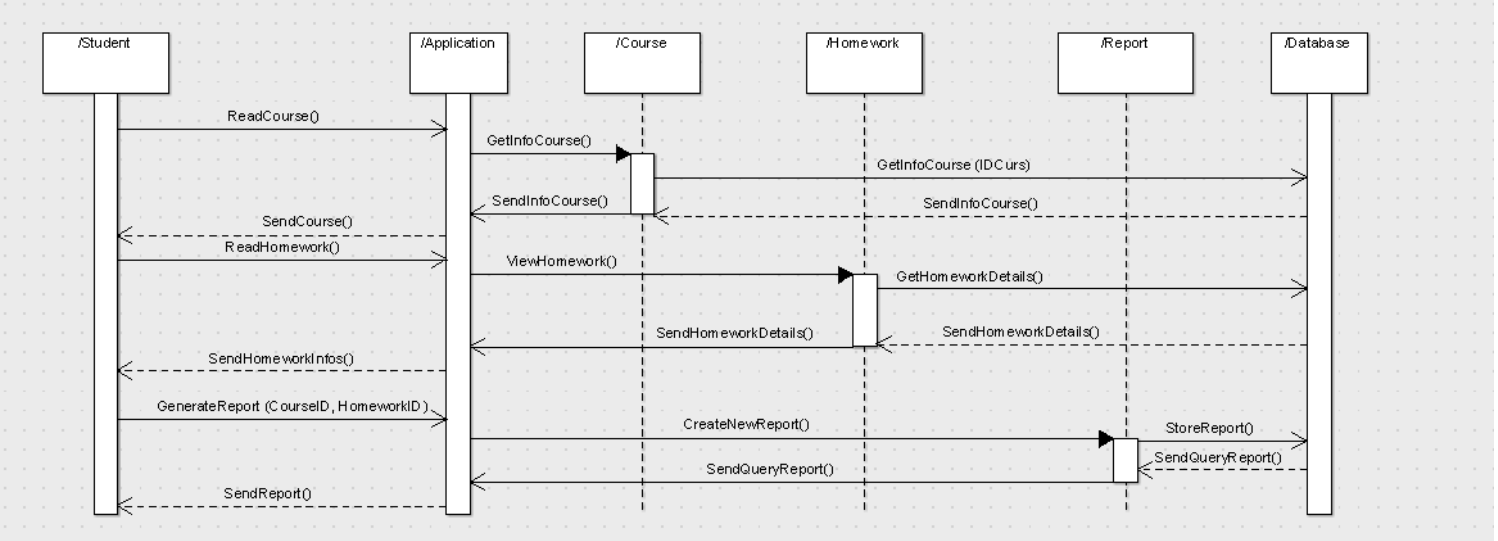
3. Updated Sequence Diagram for version 2 - Professor - CRUD courses / members, assign other professors to a course



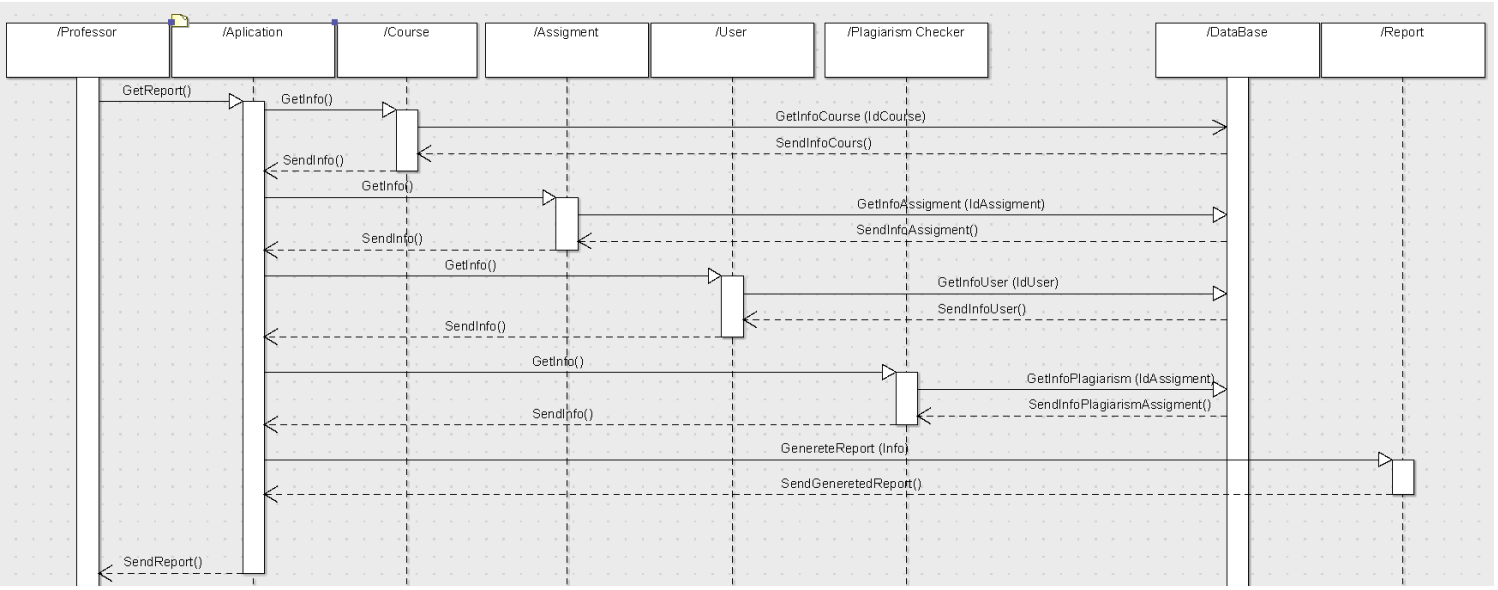
4. Professor - Compare Homeworks, Grade Homework, View/Download Grades



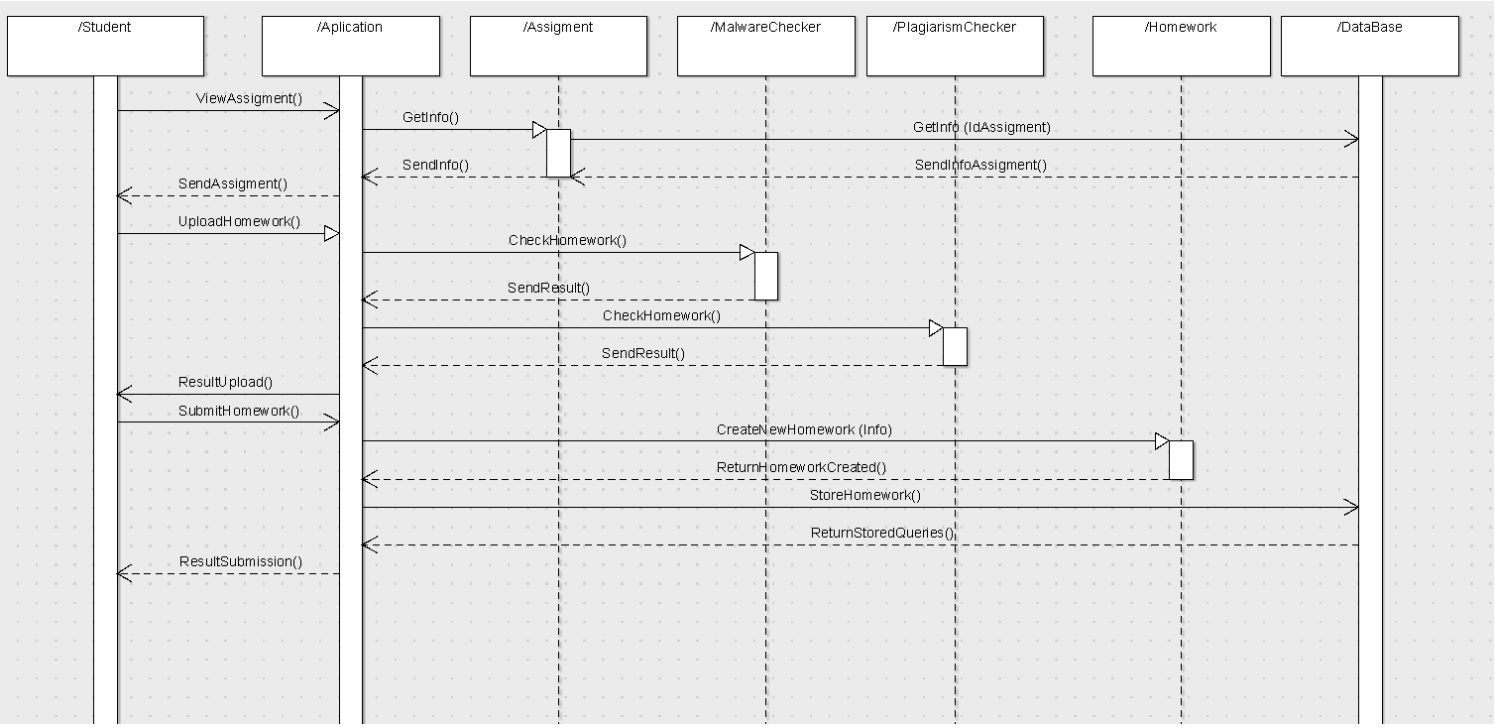
Reports generation for received homework, authors, subjects new version



5. Reports generation for received homework, authors, subjects:

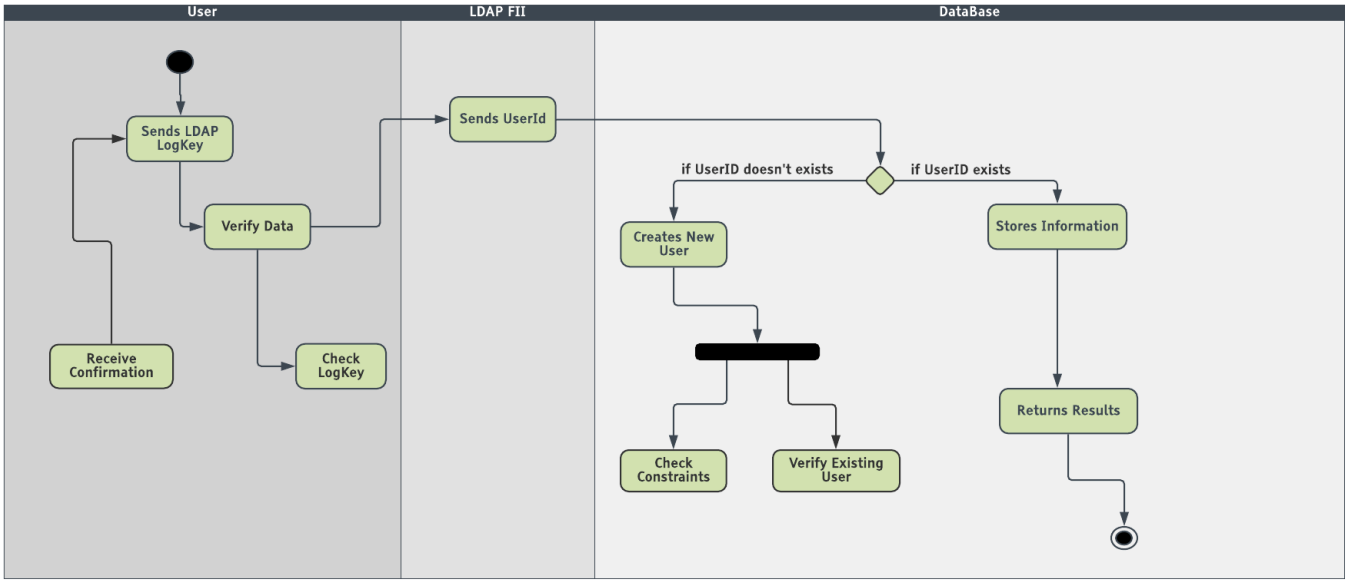


Student view,upload,submit homework

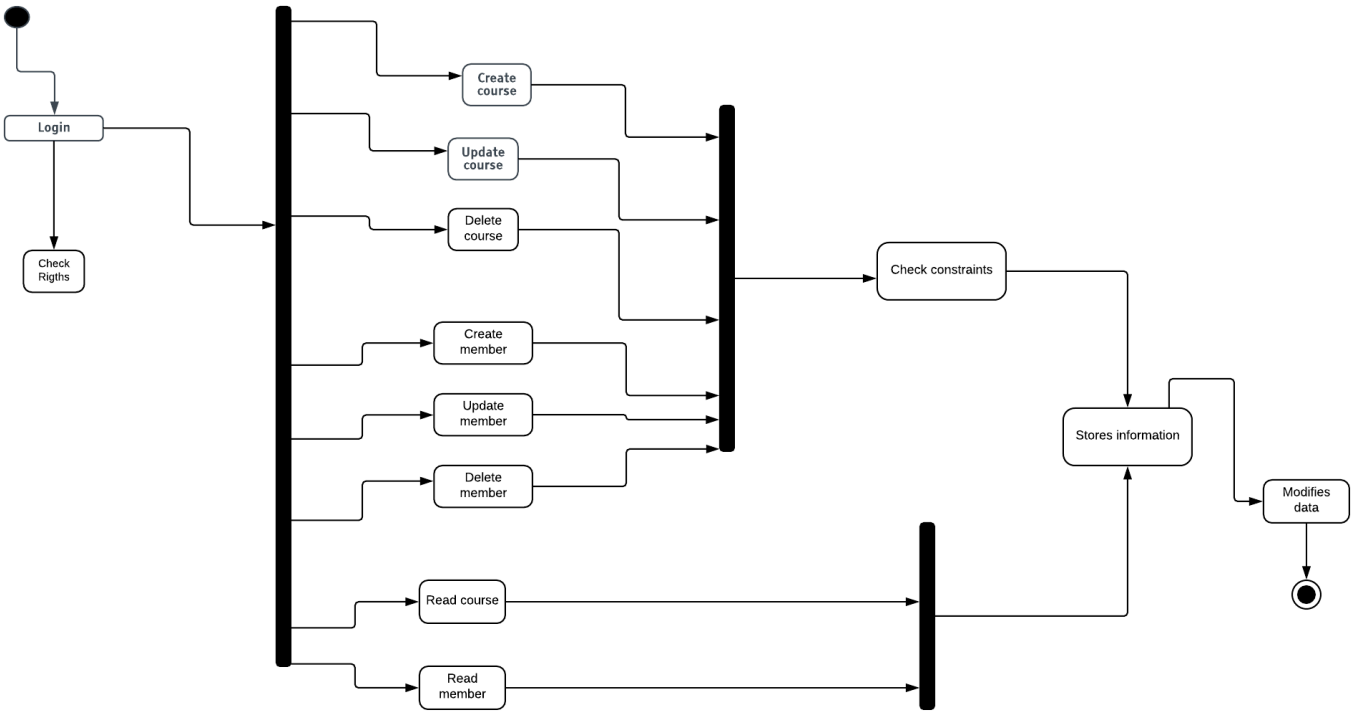


Diagrame de activitate

1. Login with LDAP FII Activity diagram



2. . Professor - CRUD courses / members, assign other professors to a course
Varianta 1



Varianta 2

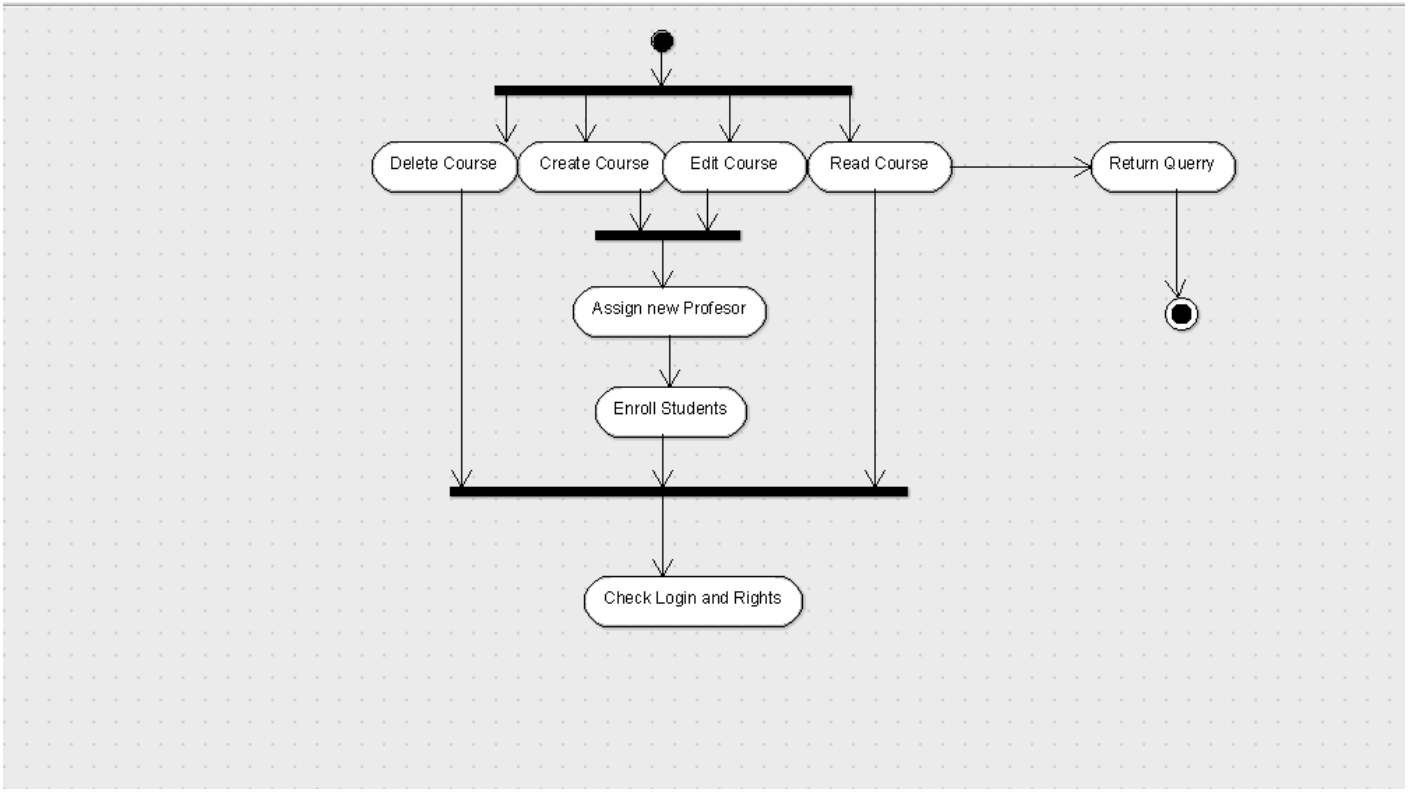
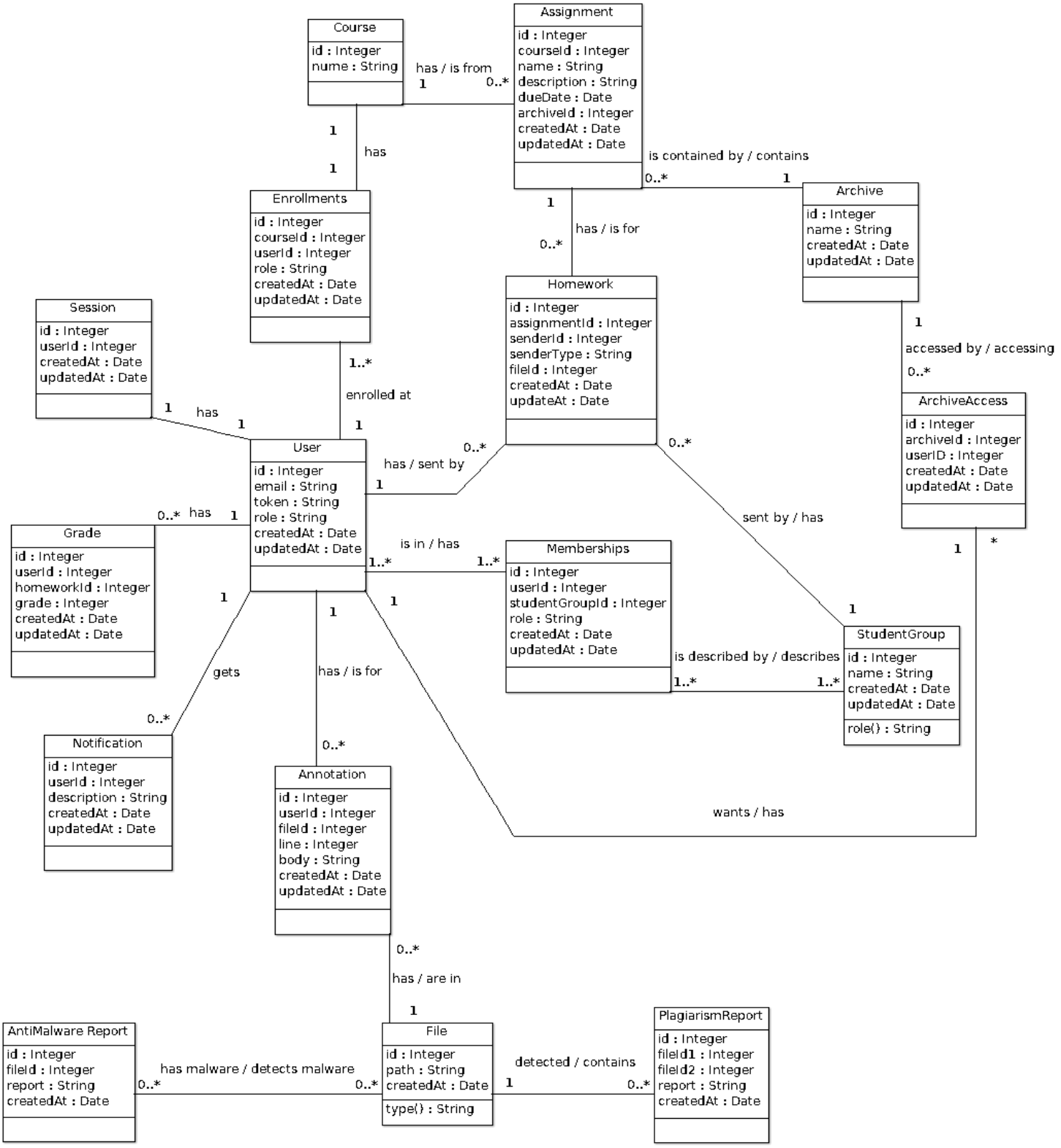

Delete Course
Create Course
Edit Course
Read Course
Return Querry
Assign new Profesor
Enroll Students
Check Login and Rights

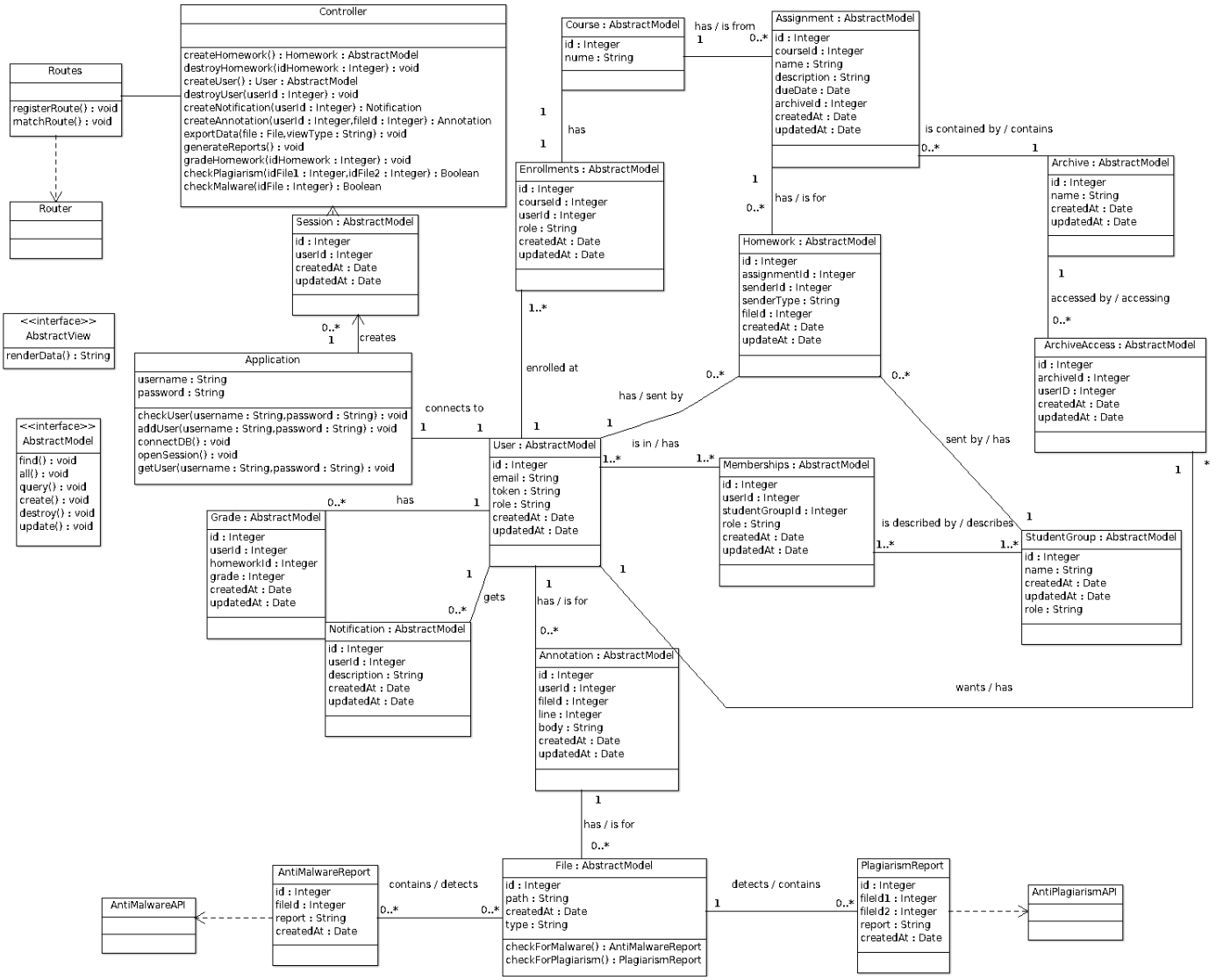
Diagrama bazei de date

deleteArchiveAccess

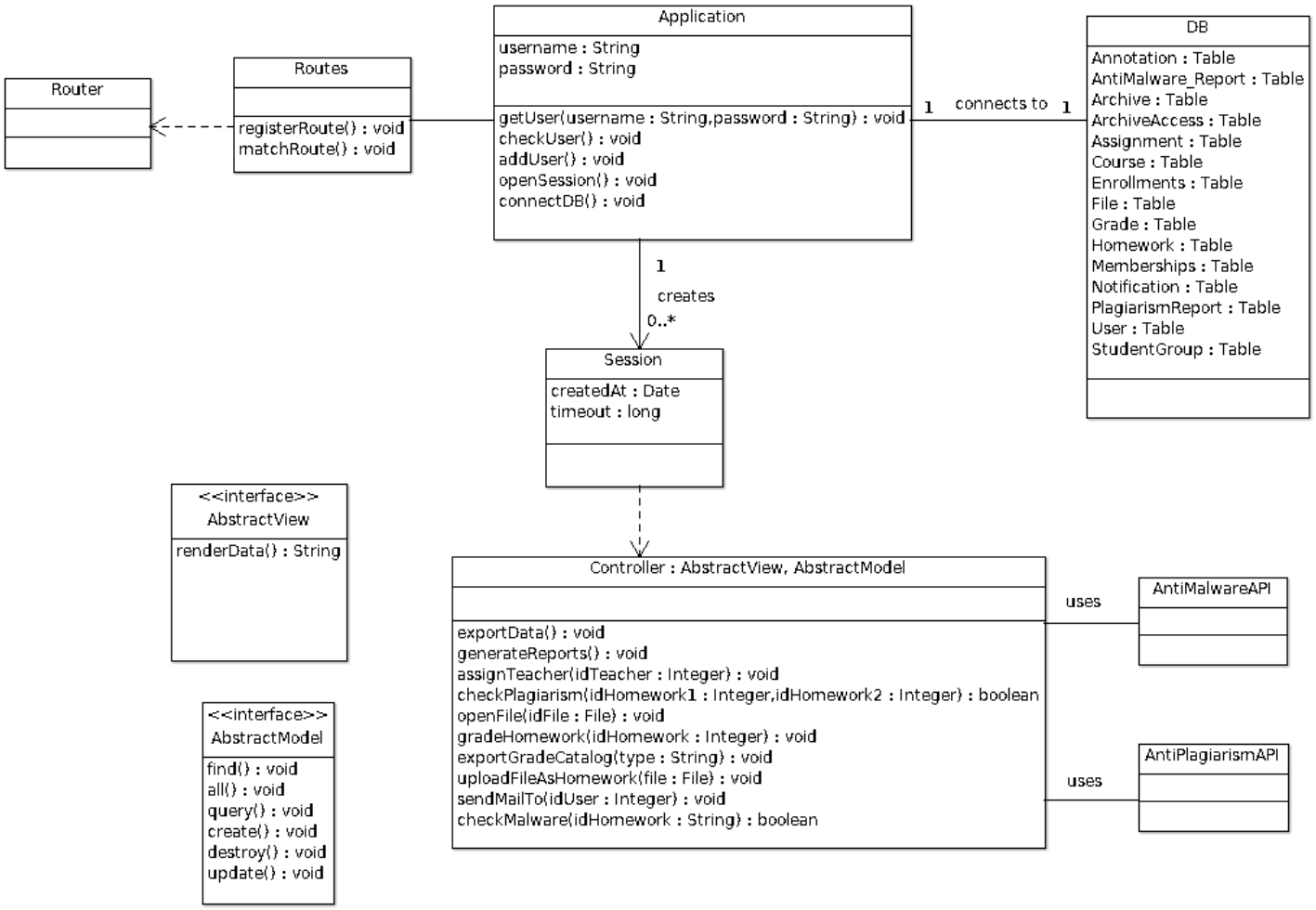


Diagramele de clase

Varianta 1



Varianta 2


Application
username : String
password : String
getUser(username : String,password : String) : void
checkUser() : void
addUser() : void
openSession() : void
connectDB() : void
Router
Routes
registerRoute() : void
matchRoute() : void
1
connects to
1
DB
Annotation : Table
AntiMalware_Report : Table
Archive : Table
ArchiveAccess : Table
Assignment : Table
Course : Table
Enrollments : Table
File : Table
Grade : Table
Homework : Table
Memberships : Table
Notification : Table
PlagiarismReport : Table
User : Table
StudentGroup : Table
1
creates
0..*
Session
createdAt : Date
timeout : long
<<interface>>
AbstractView
renderData() : String
<<interface>>
AbstractModel
find() : void
all() : void
query() : void
create() : void
destroy() : void
update() : void
Controller : AbstractView, AbstractModel
exportData() : void
generateReports() : void
assignTeacher(idTeacher : Integer) : void
checkPlagiarism(idHomework1 : Integer,idHomework2 : Integer) : boolean
openFile(idFile : File) : void
gradeHomework(idHomework : Integer) : void
exportGradeCatalog(type : String) : void
uploadFileAsHomework(file : File) : void
sendMailTo(idUser : Integer) : void
checkMalware(idHomework : String) : boolean
uses
AntiMalwareAPI
uses
AntiPlagiarismAPI