

Introducere

Motivație

Susținerea lucrării de licență reprezintă un pas important în viața fiecărui student aflat în an terminal, aceasta dându-i încă o șansă de a demonstra că se pot pune, cu succes, în practică toate cunoștințele acumulate pe parcursul întregii facultății. El poate să realizeze și să arate, de asemenea, faptul că și-a asumat toate aceste cunoștințe în mod conștiincios și reușita asocierii lor într-un mod cât mai productiv poate să îi deschidă noi orizonturi.

Pentru a ajunge cu bine la momentul în care lucrarea de licență trebuie prezentată în fața comisiei de evaluatori, un student trebuie să se pregătească intens pe tot parcursul anilor de studiu prin gândire, analiză, asociere și mai ales lucru practic. La început de an terminal, când se consideră că studentul are cunoștințe suficiente, începe efectiv perioada dezvoltării lucrării de licență. Acesta este un parcurs anevoios, cu multe ore de așteptare pe holurile facultății, cu multe email-uri trimise și, mai ales, cu multe întrebări la care studentul nu cunoaște răspunsurile.

În mod convențional, procesul de elaborare a lucrării de licență și, în cazul în care este necesar, a unei componente practice care să stea la baza acestei lucrări se realizează pe parcursul mai multor luni sub supravegherea unui coordonator, urmărind îndeaproape sfaturile acestuia. În acest sens au loc întâlniri periodice între cele două părți implicate în proces. Aceste perioade pot varia în funcție de student sau coordonator și se stabilesc într-un interval din zi în care ambii participanți sunt prezenți în facultate și au la dispoziție un interval de timp pentru a discuta eventualele nelămuriri sau următorii pași ce trebuie urmați.

De cele mai multe ori, profesorul coordonator nu dispune de suficient timp în cadrul perioadei în care este la facultate pentru a oferi sprijin suficient studenților pe care îi îndrumă, iar studenții, din cauza examenelor, testelor sau a locului de muncă, nu reușesc mereu să ajungă cu toate nelămuririle către cadrul didactic aferent lor și, astfel informație prețioasă este pierdută și nevalorificată la adevăratul ei potențial.

Datorită tuturor acestor motive, am ales să realizez o aplicație web care să faciliteze întreg procesul de supraveghere a dezvoltării acestei lucrări pentru profesorii coordonatori și procesul de a fi coordonat al studentului.

Obiective generale

Având ca principal mediu de utilizare cel universitar, studenții își pot crea un cont fiind restricționați de apartenența lor la o facultate. Deci aceștia folosesc email-ul facultății curente pentru a se putea înregistra și mai apoi autentifica. De crearea conturilor coordonatorilor se ocupă o persoană desemnată pentru acest lucru, rolul acesteia fiind de administrator.

Un student are la dispoziție o listă care va conține numele coordonatorilor disponibili pentru o anumită sesiune de licență și poate aplica la unul dintre acei profesori urmând ca cererea lor să fie acceptată sau respinsă de către coordonator. În același timp, dacă un profesor are o idee prededefinită o poate face publică astfel încât studenții interesați de subiecte/domenii să poată lua la cunoștință aceste aspecte.

Aplicarea unui student la un îndrumător presupune, de asemenea, scrierea unui paragraf în care să ofere câteva detalii despre o posibilă idee pentru viitoarea temă sau poate cere o sugestie în caz că nu are încă ceva clar în minte.

Managementul cererii unui student de către un coordonator ține strict de acesta. El are posibilitatea să își păstreze criteriile de departajare pe care le-a utilizat până în momentul de față, acceptând sau respingând cererile studenților după cum consideră de cuviință.

Un coordnator își poate alocă un număr maxim de studenți pe care să îi coordoneze, sub o limită stabilită în prealabil de către reprezentanții facultății, pentru fiecare sesiune de prezentări în funcție de planurile proprii iar după cel acel număr va fi depășit niciun alt student nu va mai putea aplica pentru îndrumarea lui.

Dacă cererea unui student este acceptată, el nu mai poate aplica pentru un alt coordonator iar în caz contrar posibilitatea aplicării există până când acesta își găsește îndrumătorul, procesul de selectare a unui îndrumător continuând în același mod: aplicare cu o primă idee despre temă sau cererea unui sfat și mai apoi acceptare/respingere din partea coordonatorului.

În perioada dintre momentul aplicării, din partea studentului, și momentul acceptării/respingerii, din partea cadrului didactic, studentul este redirecționat către aceeași pagină în care sunt listate opțiunile domnilor profesori, dar opțiunea de a aplica din nou este blocată. În cazul în care la un termen hotărât de către reprezentanții facultății există studenți care nu au un profesor coordonator asociat, atunci respectivii studenți se vor adresa secretariatului și își vor

prezenta motivele pentru care doresc să fie coordonați de un cadru didactic dar nu au reușit să se înscrie în timp util, și vor rămâne să susțină lucrarea de licență într-o sesiune ulterioară.

După acest pas începe efectiv interacțiunea coordonator-student. Fiecare student poate accesa pagina coordonatorului său unde acesta postează periodic anunțuri, termene limită și orice alt tip de conținut dorește să îl aducă la cunoștință studenților săi. Studenții pot, de asemenea, posta întrebări/nelămuriri pe această pagină, aceasta fiind publică pentru toți studenții înscriși la respectivul coordnator.

Dacă cei doi o să își dorească totuși o întâlnire față în față vor trebui doar să se puna de acord cu data și ora întâlnirii, aceste date fiind completate într-un loc special rezervat care este prestabilit de coordonator în funcție de orele sale disponibile.

Când studentul începe munca propriu-zisă, prima versiune a lucrării sale va fi încărcată într-un sistem de versionare și mai apoi un link către acest sistem va fi pus la dispoziție, urmând ca profesorul coordonator să o verifice și să acorde sfaturi de care studentul să țină cont mai departe. Pe parcursul evoluției lucrării, studentul nu trebuie decât să se asigure că toate modificările lui sunt actualizate în sistemul de versionare folosit.

Ceilalți studenți înscriși la același profesor coordonator pot, de asemenea, să vadă lucrările tuturor colegilor și eventual să își spună părerea pentru a îl ajuta pe studentul posesor al respectivei lucrări să o relizeze cât mai bine cu putință.

Atât coordonatorul cât și studentul au acces la o serie de grafice realizate pe baza sistemului de versionare folosit de către student care furnizează informații de tipul: de câte ori un student și-a actualizat lucrarea raportat la săptămână, ultimele actualizări ale lucrării sau proporția limbajelor de programare folosite. Aceste informații sunt obținute din sistemul de versionare folosit de către student cu ajutorul GitHub API¹ (API ce expune informații referitoare la codul sursă pe baza acțiunilor utilizatorilor).

Fiecare student și coordonator are acces la o pagină special construită pentru a le furniza acestora informații despre perioada susținerii lucrării de licență, perioadele în care se desfășoară

¹ GitHub API: <https://developer.github.com/v3/>

înscriserile, documentele necesare și sfaturi utile pentru a putea fi siguri că totul va merge bine pe parcursul prezentării lucrării.

Pe lângă rolurile de student și coordonator, un alt rol secundar există în cadrul aplicației – cel de administrator. Acesta are responsabilitatea de a adăuga noi profesori coordonatori în sistem, de a șterge din coordonatorii existenți dacă aceștia devin indisponibili sau de a modifica datele deja înregistrate în cazul în care este nevoie.

Scurtă descriere a soluției

Aplicația curentă este una accesibilă cu ajutorul internetului, deci soluția construită poate fi descrisă ca o aplicație de tip client-server cu următoarea componență:

- **Aplicația server:** reprezentată sub forma unui API REST² care expune extern informații, prin intermediul serviciilor, și care ajută la manipularea datelor;
- **Aplicația client:** reprezentată de o interfață grafică, ce utilizează serviciile oferite de către aplicația server, apelându-le prin intermediul protocolului HTTP³, și expune informațiile obținute către utilizatori oferindu-le, de asemenea, posibilitatea manipulării datelor;
- **Baza de date:** reprezentată de o bază de date de tip relațional, cu ajutorul căreia datele necesare aplicației sunt stocate persistent.

Abordare tehnică

Limbaje și cadre de programare

Entity Framework Core este versiunea independentă de platformă al cadrului Entity Framework, acesta fiind un ORM⁴ (Object Relational Mapping) care permite dezvoltatorilor .NET să lucreze cu o bază de date folosind obiecte. (Detalii disponibile în *Anexa 1 - Limbaje de programare*)

² Principii REST: <https://www.c-sharpcorner.com/uploadfile/kalisk/rest-fundamentals/>

³ Hypertext Transfer Protocol: <https://developer.mozilla.org/en-US/docs/Web/HTTP>

⁴ ORM: <https://searchwinddevelopment.techtarget.com/definition/object-relational-mapping>

SCSS (Sassy CSS) a avut ca și utilitate în construirea aplicației mele, în principal, utilizarea variabilelor pentru memoria diferitelor elemente (culoare, font) care ajută la scrierea unui cod curat, mentenabil și fără duplicate, având două exemple expuse în *Figura 1*, respectiv *Figura 2*. (Detalii disponibile în *Anexa 1 - Limbaje de programare*)

```
$text-font: 'Lucida Sans';  
$text-color: rgba(94, 64, 56, 1);
```

Figura 1: Declararea unor variabile SCSS

```
.sidebar_content{  
  cursor: pointer;  
  color: $text-color;  
  font-family: $text-font;  
  font-size: $text-size-header;  
  margin: 1% 5%;  
  outline: none;  
}
```

Figura 2: Folosirea unor variabile SCSS

Angular este un cadru de lucru structural constuit în întregime cu ajutorul limbajului TypeScript (formă îmbunătățită a limbajului JavaScript) ce poate fi folosit pentru a crea partea de client a unei aplicații. Avantajul pe care l-am avut folosind ambele tehnologii a fost ușurința transmiterii informațiilor între server și pagina web, fiind posibilă stocarea datelor în obiecte ușor manipulabile, validarea facilă a datelor și eliminarea codului duplicat prin utilizarea de metode. În figura de mai jos (*Figura 3*) se găsesc menționate alte patru avantaje ale utilizării Angular.

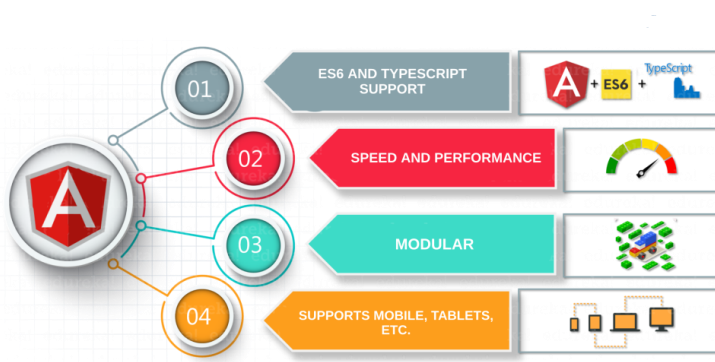


Figura 3: Avantajele utilizării Angular⁵

⁵ Sursa: <https://softwaredevelopment.ae/angular-best-solution-2018-web-app/>

Au mai fost folosite, de asemenea, următoarele limbaje și cadre de programare: **.NET Core**, **ASP.NET Core**, **C#**, **HMTL**, **CSS**. Despre toate acestea pot fi găsite detalii în cadrul *Anexei 1 – Limbaje și cadre de programare*.

Instrumente software

Visual Studio Code⁶ este un editor de cod sursă ușor de utilizat, dar puternic, care rulează pe desktop fiind disponibil pentru Windows, MacOS și Linux. Dispune de suport încorporat pentru TypeScript, acesta fiind principalul motiv pentru care am ales să îl folosesc pentru dezvoltarea interfeței cu utilizatorul.

Visual Studio este, de asemenea, un editor de cod sursă, cu ajutorul căruia pot fi construite diferite tipuri de aplicații (desktop, servicii, mobile). L-am utilizat datorită multitudinii de avantaje pe care acesta le oferă. Unele dintre ele sunt: sugestiile pe care acesta le oferă în timp real, modul în care sunt afișate și generate erorile de compilare și ușurința creării unei structuri optime de pentru aplicație.

Postman este o unealtă software care poate fi utilizată pentru a simula cererile HTTP care ar fi în mod normal trimise de către un client către server. Utilitatea lui a fost posibilitatea testării serviciilor oferite de către server, după cum se poate observa în *Figura 4*.

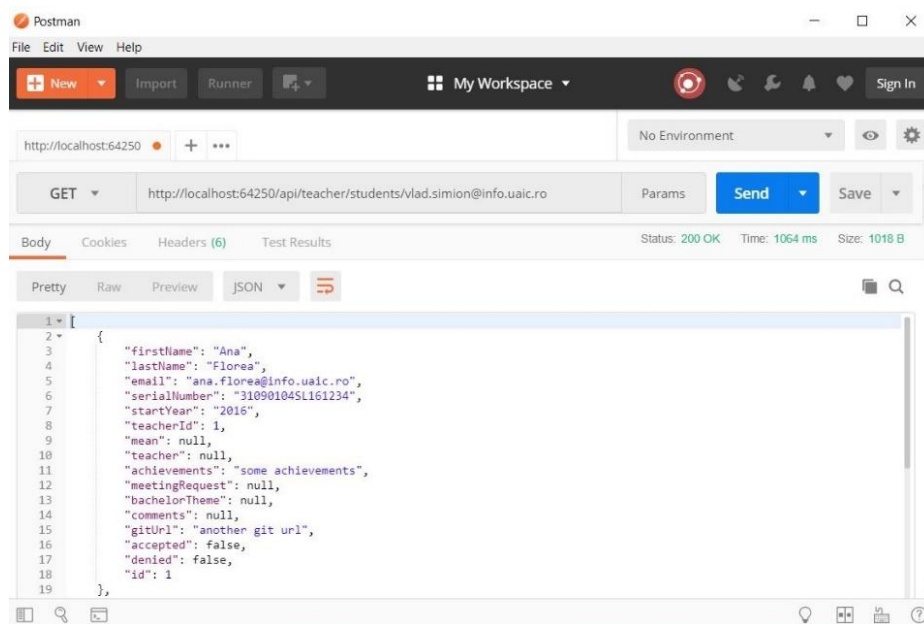


Figura 4: Cerere de tip GET și răspuns de la server, cu ajutorul Postman

⁶ Documentație Visual Studio Code: <https://code.visualstudio.com/docs>

SourceTree este o aplicație care joacă rolul de client pentru sistemul de versionare Git⁷. Lucrul cu sistemul de versionare este deci ușurat și încurajat cu ajutorul acestei interfețe grafice. Două din avantajele majore al folosirii acestora este siguranța pe care o oferă în stocarea codului sursă (și a orice altor documente) și posibilitatea folosirii oricărei versiuni anterioare de cod. Aceasta oferă cu ușurință informații despre toate modificările efectuate, inclusiv despre cele care se află în curs de modificare, lucru vizibil în *Figura 5*.

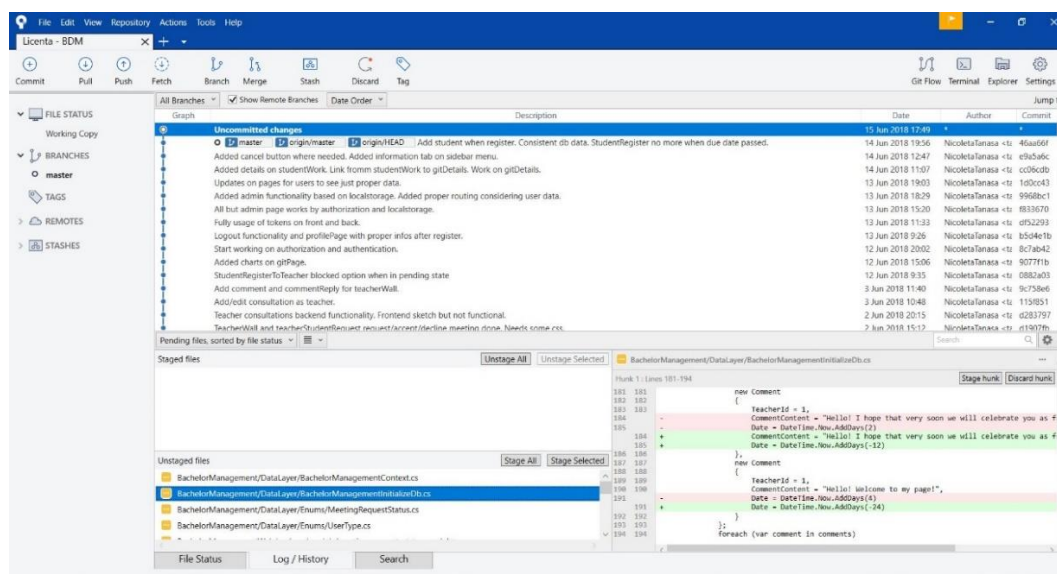


Figura 5: Interfața SourceTree

⁷ Detalii GitHub: <https://www.atlassian.com/git/tutorials/what-is-git>

1. Contribuții

Petrecând niște ani superbi în cadrul studenției mi-am propus ca lucrarea mea de licență să poată fi baza unei aplicații generice care să aibă scopul de a ajuta studenții. În acest sens m-am gândit inițial la o platformă pentru organizarea și gestionarea materialelor aferente cursurilor și seminariilor predate în cadrul facultății dar, având în vedere că domnii profesori deja folosesc anumite spații pentru a-și pune resursele la îndemâna studenților, m-am reorientat spre un tip de aplicație ce nu există încă în cadrul facultății și nici măcar în cadrul universității.

Pentru realizarea obiectivului, am creat o aplicație de tip client-server pe care am dezvoltat-o pe parcursul mai multor luni, folosind varii unelte și limbaje de programare. Inițial, am creat codul ce a generat structura bazei de date, apoi am creat câteva date ce populează baza menționată anterior, iar în cea mai mare parte a timpului am lucrat, în paralel, la construirea serverului și interfeței grafice.

Prin utilizarea unei multitudini de tehnologii moderne (spre exemplu Angular 4 și .NET core 2.0), respectarea bunelor practici atât în arhitectura cât și în implementarea aplicației și prin ușurința cu care aceasta poate fi extinsă, am reușit să creez o aplicație care să ajute studenții și chiar să îi provoace în a continua munca începută de mine.

Pe tot parcursul dezvoltării aplicației am folosit cunoștințe dobândite de-a lungul anilor de facultate, în cadrul mai multor materii, și anume:

- **Introducere în .NET:** limbajul C# și principii generale
- **Baze de date:** realizarea și managementul baze de date
- **Ingineria programării:** realizarea diagramelor și optimizarea modului de lucru
- **Rețele de calculatoare:** protocolul HTTP
- **Tehnologii web:** realizarea unei aplicații web și principii REST
- **Programare orientată obiect:** principiile programării orientate obiect

2. Descrierea problemei

În vremurile în care s-au pus bazele primelor universități din lume, în jurul anului 1100, profesorii erau forțați de către asociațiile studențești, formate în urma unor discuții cu cei aflați la conducerea orașelor, să respecte o serie de reguli, un exemplu fiind să nu lipsească de la cursuri fără aprobarea studenților. Aceștia trebuiau să se supună regulilor deoarece singurele lor venituri erau banii pe care studenții și familiile lor îi investeau în învățământ. Simțindu-se excluși din procesul de construire al acestor reguli, profesorii au început să impună de asemenea anumite condiții care să fie aplicate studenților, una dintre acestea fiind elaborarea unei lucrări de licență la finalul studiilor și prezentarea ei în fața unei comisii.

Astfel, în scurt timp, mediul academic universitar a prins o formă asemănătoare cu cea din zilele noastre, necesitatea dezvoltării unei lucrări de licență datând, deci de acum aproximativ 900 de ani iar importanța acesteia rămânând aceeași.

În zilele noastre, uneori studentul, poate din cauză că nu se consideră suficient de bine pregătit pentru a merge și a vorbi cu un anumit coordonator, pierde șansa de a realiza o lucrare excelentă și de a avea o colaborare minunată. Sau chiar, din cauza numărului mare de studenți, poate un profesor nu are șansa de a cunoaște fiecare student în parte și pierde și acesta, de asemenea, șansa unei colaborări eficiente din care să rezulte lucruri benefice.

Chiar și după ce primul pas a fost făcut, și studentul are asignat un profesor coordonator și o temă pentru viitoare lucrare de licență, din cauza modului curent de viață, de cele mai multe ori, nici viitorul absolvent, nici coordonatorul științific al acestuia nu dispun de o perioadă de timp suficient de mare încât să asigure o colaborare ce acoperă în totalitate nevoile studentului și așteptările coordonatorului.

3. Abordări anterioare

În acest moment nu există o aplicație folosită pe scară largă și care să poată să îmbunătățească acest proces iar parcursul urmat de către studenți depinde de fiecare coordonator științific, acesta alegându-și metoda prin care comunică aspecte cum ar fi: următoarea întâlnire pentru o nouă discuție, conținutul ce trebuie modificat, corectat sau eventual îmbunătățit, sfaturi pentru a-i ușura munca și pașii pe care trebuie să îi parcurgă până la următoarea lor discuție. Există totuși un site care găzduiește o platformă numită **EduSoft**⁸, aceasta fiind o asociație de tip ONG coordonată de către un specialist, sub îndrumarea căruia studenții pot alege să își elaboreze lucrarea de licență. Neajunsul acestei platforme este acela că numărul studenților care au folosit-o este extrem de redus iar din anul 2015 nu mai pare să existe activitate.

Datorită faptului că internetul și tehnologia, în general, au avut o evoluție foarte rapidă în ultimul timp, ele pot fi folosite cu succes pentru a implementa și, mai apoi, utiliza orice tip de aplicație menită să ne facă viața mai confortabilă și să ne ofere acces la tot felul de informații.

Am ales deci să construiesc un mod de a le ușura munca atât coordonatorilor cât și studenților prin a reduce cantitatea de timp pierdută prin aglomerația orașelor, prin așteptarea reciprocă pentru a avea o întâlnire dacă nu consideră că este neapărat necesară, prin discuțiile care pot divaga de la subiect și prin neînțelegerile ce pot apărea cu privire la anumite secțiuni din lucrare din cauza unor eventuale exprimări neclare. Aplicația este adresată studenților ciclului de licență și coordonatorilor acestora, venind în ajutorul lor cu tot felul de funcționalități gândite pentru a le ușura în special modul de manageriere a timpului și pentru a asigura o mai bună comunicare între cei doi, fiecare dintre aceștia putând actualiza datele și/sau fișierele oricând le permite timpul.

⁸ EduSoft: <https://www.edusoft.ro/>

4. Descrierea soluției

4.1 Principalele funcționalități

Prima pagină cu care utilizatorul are contact atunci când dorește să utilizeze aplicația curentă este cea care îi permite logarea, design vizibil în *Figura 6*. Aceasta se realizează cu ajutorul unui email, ce aparține Facultății de Informatică Iași, și o parolă. În cazul în care credențialele utilizate nu se regăsesc în sistem, înseamnă că acel cont nu există sau că, din greșeală, au fost introduse credențiale invalide. Utilizatorul este notificat de acest lucru, dacă este cazul, și i se permite să reintroducă un nou set de credențiale. Această pagină conține validări asupra adresei de email, mai exact aceasta să aparțină Facultății de Informatică și să fie asignată unui student existent. Email-ul poate fi introdus atât cu majuscule cât și cu minuscule, sistemul gestionând astfel de situații prin compararea șirurilor de caractere convertite în minuscule.

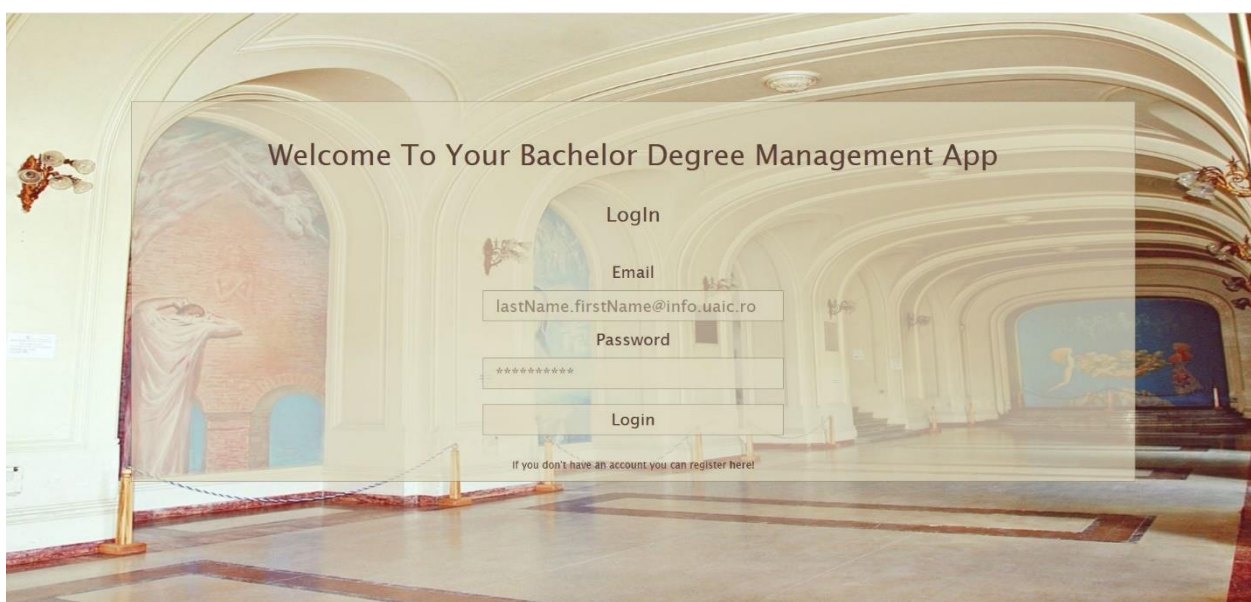


Figura 6: Pagină ce permite accesarea sistemului

În cazul în care contul pe care utilizatorul dorește să îl folosească nu există încă, situație explicată mai sus, acesta poate să se înregistreze în cadrul sistemului și mai apoi să se autentifice, existența noului cont fiind asigurată în caz de succes sau semnalată o eroare în caz contrar. Un utilizator nu poate să se înregistreze cu ajutorul unei adrese de email care a mai fost folosită în prealabil de către alt cont. *Figura 7* prezintă design-ul acestei pagini. Doar utilizatorii cu titlu de student își pot crea un cont nou, pentru utilizatorii de tip profesor existând un administrator care se

ocupă de crearea lor. Validările existente pe această pagină sunt atât referitoare la adresa de email cât și la parola introdusă. Adresa de email trebuie să fie una ce aparține Facultății de Informatică, să fie asignată unui student care își desfășoară în prezent studiile aici și să nu mai fi fost folosită la crearea niciunui alt cont în cadrul acestui sistem. Parola trebuie să aibă între șase și treizeci și două de caractere și să aibă în componență cel puțin o cifră, un caracter special, o literă mare și una mică. Orice fel de combinație care respectă cerințele anterioare este acceptată. Odată introdusă o parolă validă, aceasta trebuie confirmată prin reintroducerea ei într-un câmp apropiat. Dacă oricare din validările adresei de email sau parolei sunt încălcate, utilizatorul primește o notificare caracteristică fiecărei nereguli pentru a deveni conștient de acest lucru.

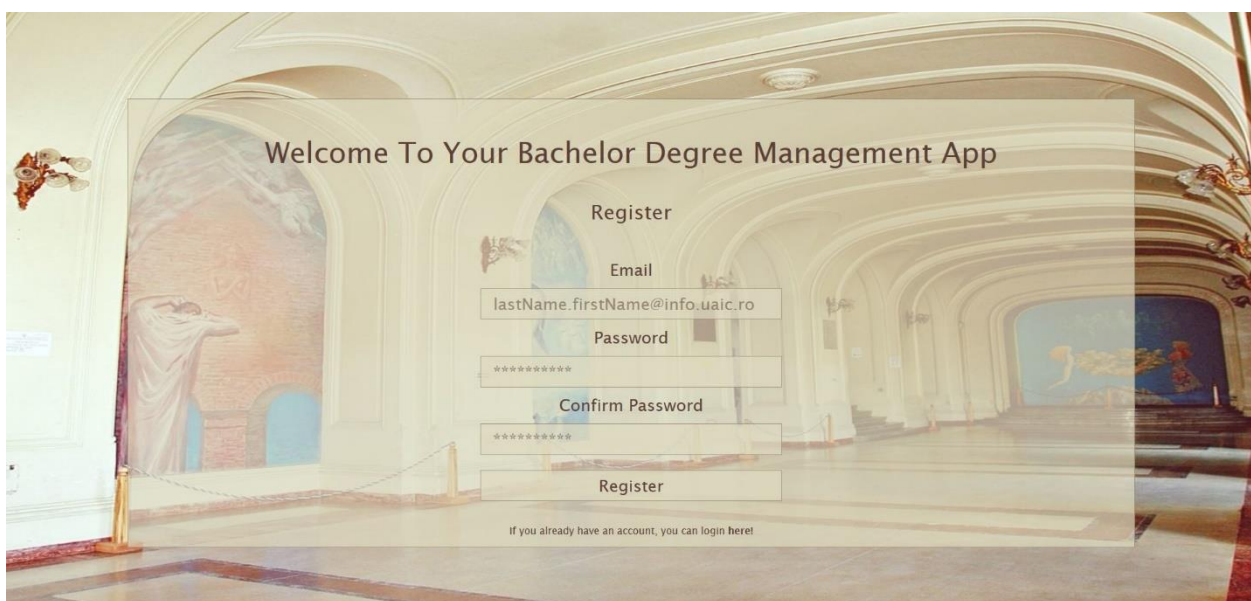


Figura 7: Pagină ce permite înregistrarea în sistem

Dacă autentificarea sau înregistrarea unui cont nou a reușit, utilizatorii au acces, în cazul în care doresc, la pagina de bun venit, din cadrul căreia pot alege care să fie următoarele lor acțiuni, disponibilă în *Figura 8*. Acțiunile care urmează pot fi: navigarea spre pagina în care profesorul postează diferite informații, navigarea spre pagina de profil a utilizatorului, navigarea către pagina care oferă informații despre munca fiecărui student, navigarea către pagina care oferă informații utile despre sesiune curentă de susținere a licenței (pagini accesibile atât pentru student cât și pentru profesor dar nu pentru administrator) și navigarea către pagina unde utilizatorul vede cererile de înscriere sau de întâlnire față în față (pagină accesibilă pentru profesor dar nu pentru student și

administrator). Pentru utilizatorii de tip student, pagina va deveni disponibilă după ce perioada specifică înscrierilor va trece și ei sunt asigurați către unul dintre coordonatorii disponibili.

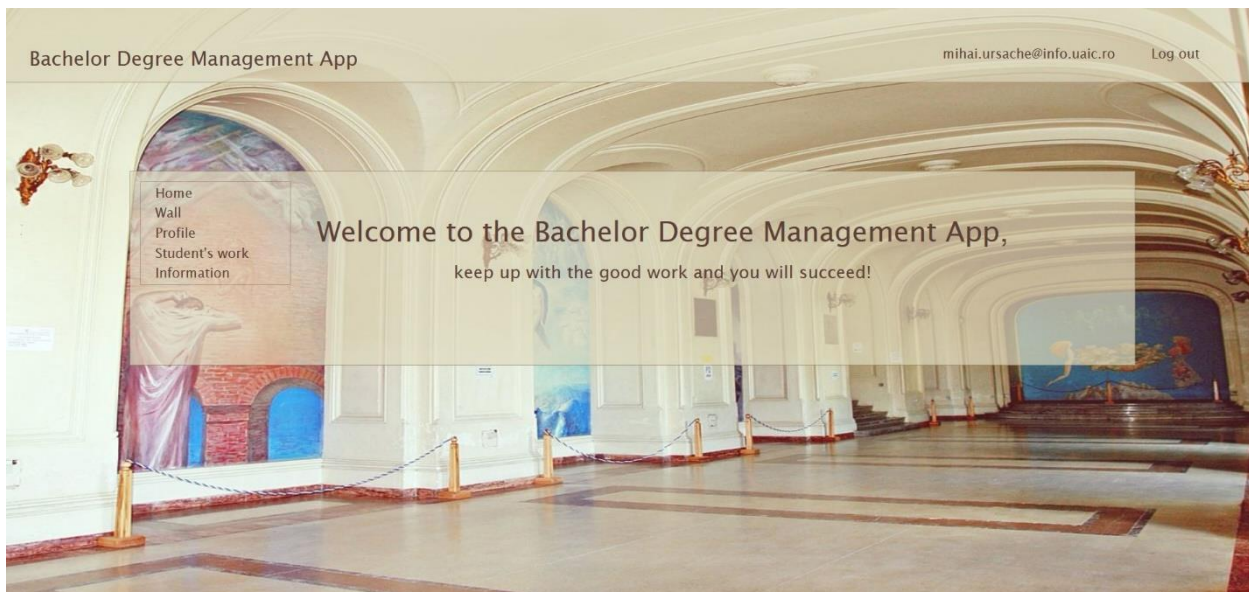


Figura 8: Pagină de “Bun venit”

O pagină similară cu cea de bun venit este cea de eroare, ce apare în cazul în care, din diverse motive, serverul nu este disponibil. Această pagină este disponibilă pentru toate cele trei tipuri de utilizatori existenți în sistem și se poate previzualiza în *Figura 9*.

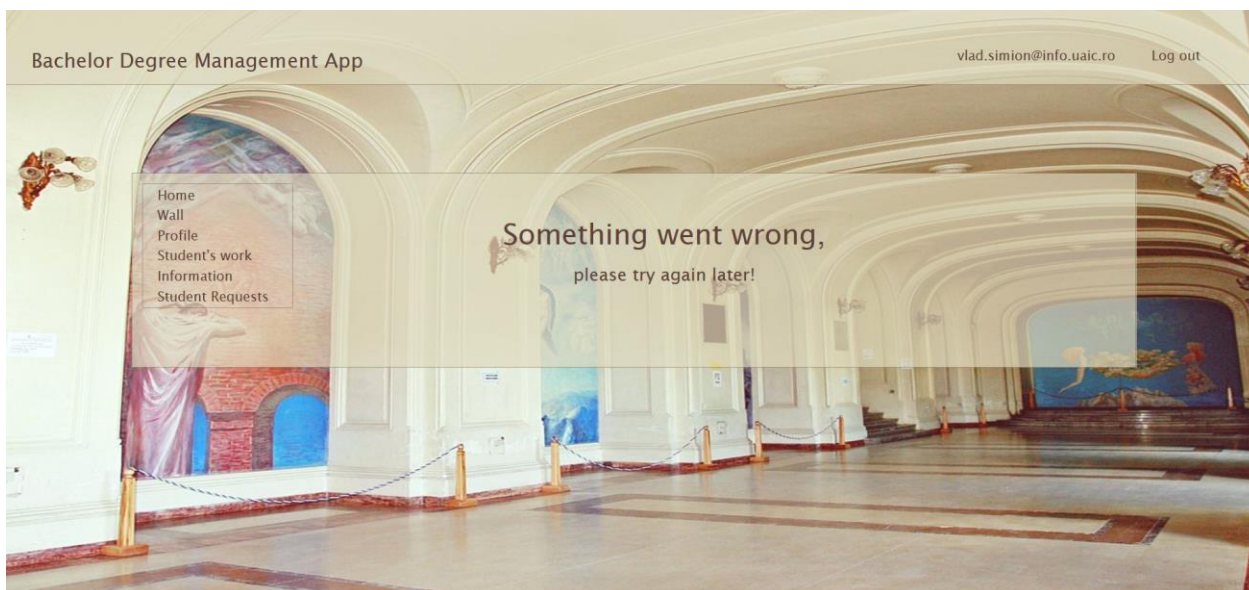


Figura 9: Pagină de eroare

Inițial, pentru toată durata perioadei de înscrieri, studentul ajunge pe o pagină care îi permite să își aleagă un profesor coordonator dintr-o listă completată de către cadrele didactice. El poate analiza opțiunile disponibile și, în funcție de aptitudinile și dorințele lui, poate continua procesul de selecție apăsând butonul vizibil (ca fiind inactiv) în *Figura 10*. Dacă studentul aplică la unul dintre coordonatorii listați, este redirecționat spre o pagină pe care adaugă anumite detalii și, mai apoi, reîntors pe aceeași pagină destinată înscrierilor dar cu opțiunea de aplicare blocată până când cadrul didactic îi aprobă sau respinge cererea de înscriere. De asemenea, înregistrarea corespondentă profesorului la care a aplicat este evidențiată.

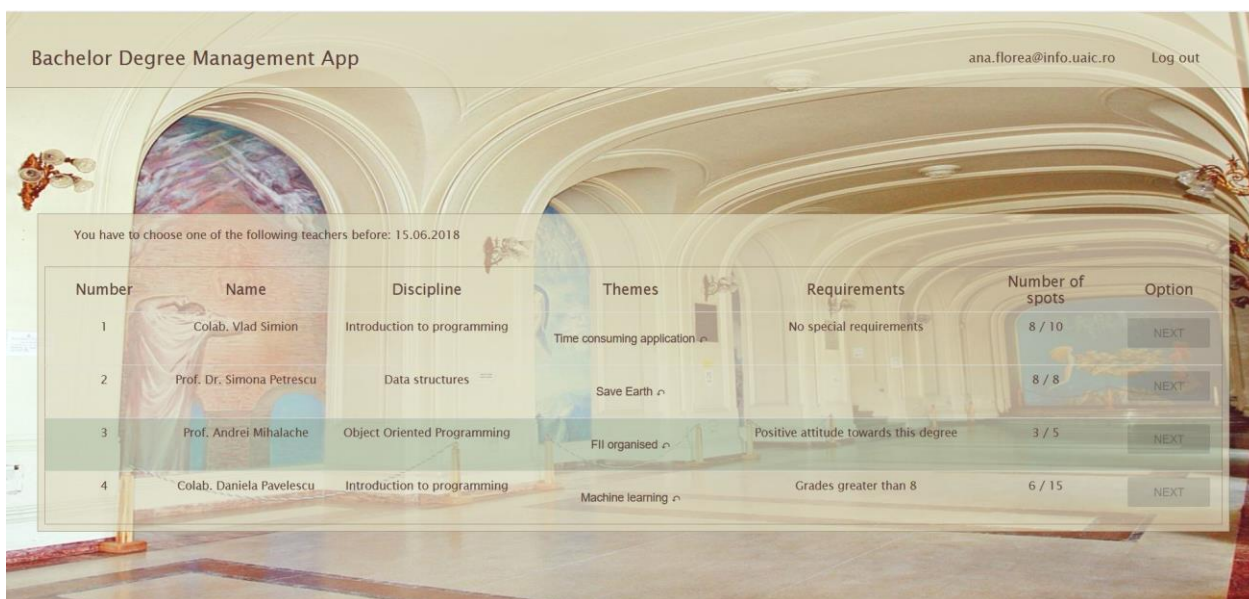


Figura 10: Pagină pentru înscrierea la un coordonator, cu opțiune blocată

Pentru utilizatorii de tip profesor, pentru aceeași perioadă a înscrierilor, este afișată lista cererilor studenților înscriși până în momentul respectiv, lucru vizibil în *Figura 11*. Aceștia pot vizualiza anumite detalii furnizate de către student, pe baza cărora vor decide dacă aprobă sau nu cererea studentului. Aceștia au în fiecare moment la dispoziție informații despre numărul de studenți pe care îi mai poate accepta și totalul numărului de studenți pe care vrea să îi accepte în sesiunea curentă, informație ce acesta o completează în prealabil. După ce perioada înscrierilor trece, lista cererilor de înscriere este înlocuită de lista cererilor de întâlniri în persoană pe care studenții le vor putea solicita, care conține adresa de email a studentului, data și ora la care a făcut cererea și statusul cererii (în așteptare, aprobat sau refuzat), lucru vizibil în *Figura 12*.

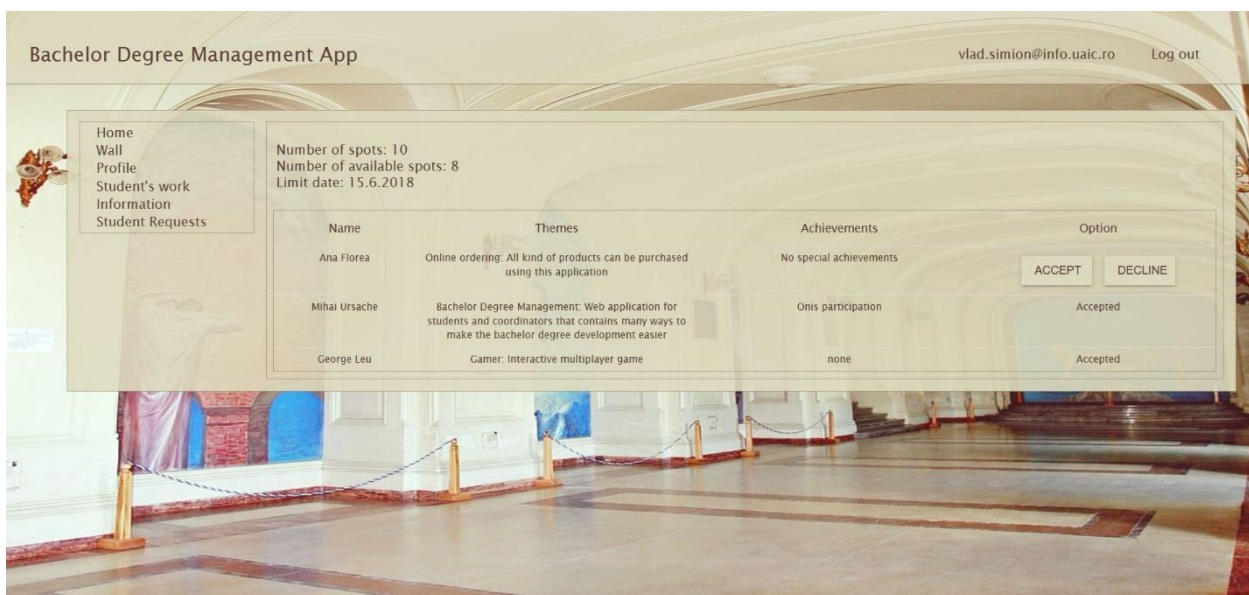


Figura 11: Pagină pentru vizualizarea cererilor studenților în timpul perioadei de înscriere

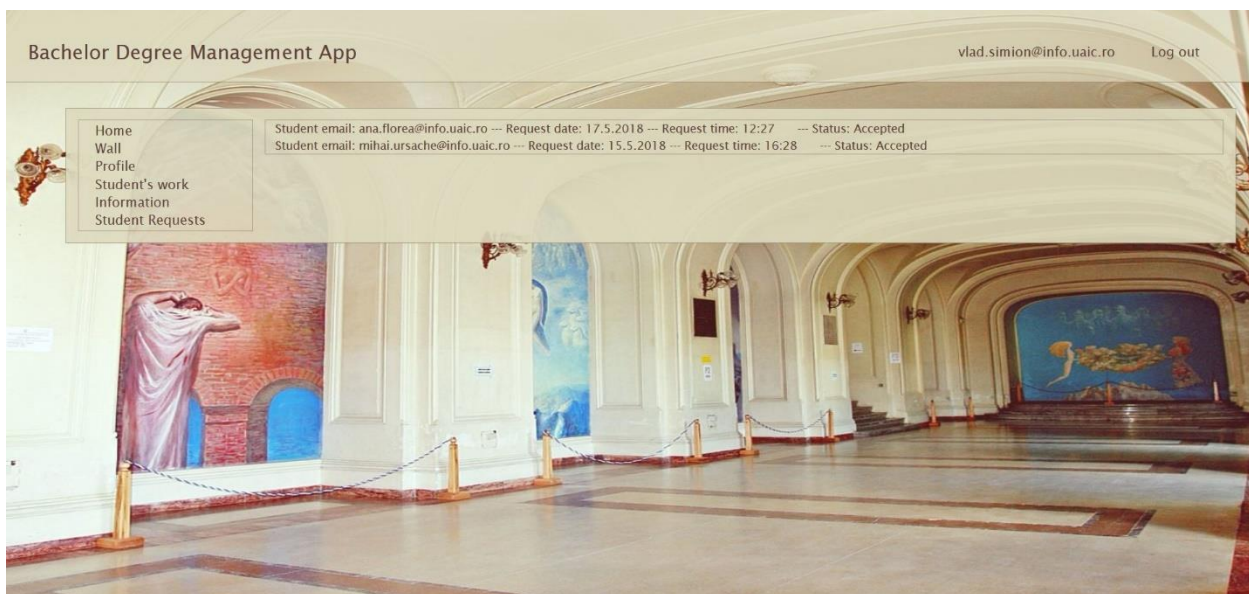
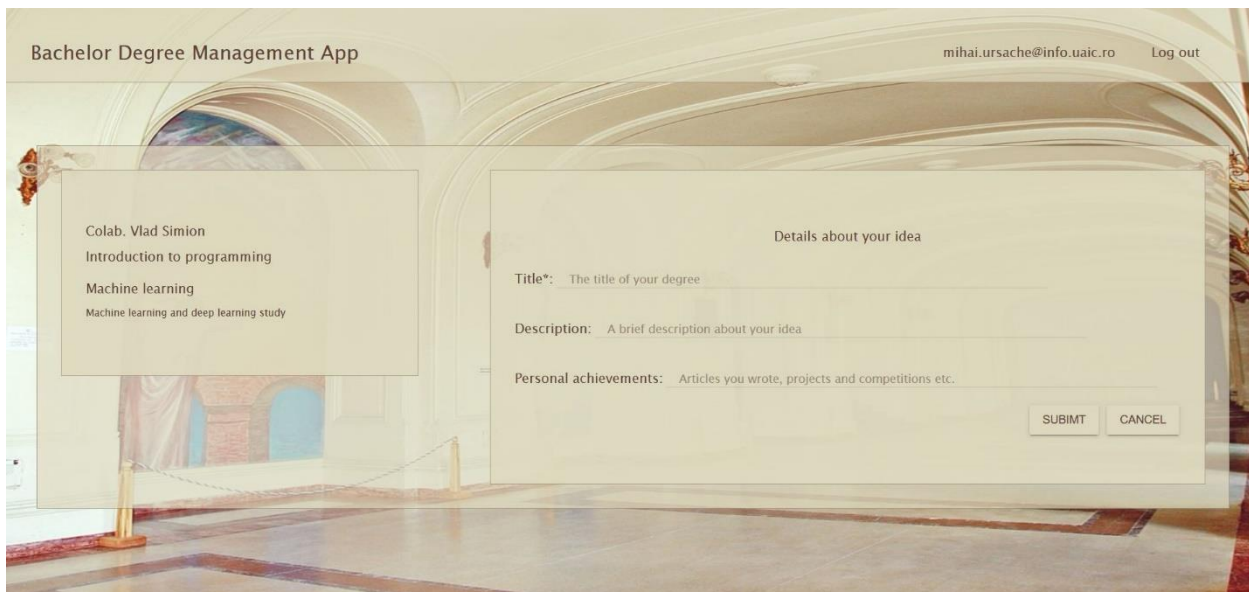


Figura 12: Pagină pentru vizualizarea cererilor studenților după perioada de înscriere

Așa cum menționam anterior, după ce un student selectează unul din profesorii coordonatori disponibili, este redirecționat către o pagină în cadrul căreia va putea oferi mai multe detalii despre ideea pe care o are, design vizibil în *Figura 13*. El completează date despre titlul lucrării, o mică descriere și, dacă dorește, menționează performanțe obținute în cadrul facultății.

Acestea din urmă ar putea însemna impresionarea profesorului coordonator ales. În partea stânga are disponibile informații despre profesorul coordonator la care studentul dorește să se înscrie.



The screenshot shows a web application titled "Bachelor Degree Management App". In the top right corner, there is a user email "mihai.ursache@info.uaic.ro" and a "Log out" link. The main content area is divided into two panels. The left panel, titled "Colab. Vlad Simion", lists three items: "Introduction to programming", "Machine learning", and "Machine learning and deep learning study". The right panel, titled "Details about your idea", contains a form with three fields: "Title*" with the placeholder "The title of your degree", "Description" with the placeholder "A brief description about your idea", and "Personal achievements" with the placeholder "Articles you wrote, projects and competitions etc.". At the bottom right of the form are two buttons: "SUBMIT" and "CANCEL". The background of the application is a light beige color with a faint image of a classical building interior.

Figura 13: Pagină pentru adăugarea detaliilor necesare lucrării de licență

Atunci când un cadru didactic este adăugat în sistem de către administrator, i se atribuie numai un set prestabilit de informații, profesorul trebuind, doar în cazul în care accesează sistemul (se loghează) pentru prima dată, să adauge mai multe detalii despre disponibilitatea și cerințele lui, așa cum este vizibil în *Figura 14*. Unele din aceste date pot fi editate ulterior dacă se dorește. Datele editabile sunt cele care corespund zilei și intervalului din zi în care domnii profesori coordonatori sunt disponibili pentru a avea întâlniri în persoană cu studenții. Tot profesorul coordonator trebuie să stabilească un număr maxim de studenți pe care îi va coordona (acesta fiind limitat superior de un număr hotărât de conducerea facultății și introdus în sistem de către administrator), niște cerințe minime pe care dorește ca studenții să de îndeplinească (dacă are astfel de cerințe și dorește să le facă publice), un titlu și o descriere pentru o lucrare de licență astfel încât studenții își pot da seama care este gradul de dificultate și care sunt așteptările pe care un profesor le are și o zi din săptămână, respectiv un interval orar din acea zi în care dânsul va fi disponibil pentru întâlniri în persoană cu studenții, în caz că aceștia solicită acest lucru.

Bachelor Degree Management App simona.petrescu@info.uaic.ro Log out

Details

Number of students per session:

Requirements: Achievements that you would like the students to have.

Theme title:

Theme description:

Consultation day:

Consultation interval:

[SAVE DETAILS](#)

Figura 14: Pagină pentru adăugarea detaliilor și cerințelor unui cadru didactic

După ce perioada destinată înscrierilor va trece, utilizatorii vor avea acces la o pagină ce conține majoritatea informațiilor pe care un student le dorește. Au legături către toate celelalte pagini, pot adăuga/anula o cerere pentru o întâlnire în persoană cu coordonatorul, pot vizualiza toate noutățile coordonatorului și iniția postări sau adăuga comentarii la postările deja existente. Profesorii coordonatori au la dispoziție aceleași informații, mai puțin posibilitatea de a adăuga/anula o cerere pentru o întâlnire în persoană. Aceasta este vizibilă în *Figura 15*.

Teacher Wall (vlad.simion@info.uaic.ro)

Available day and hour: Monday, 09:00 - 11:00

[CANCEL THE MEETING REQUEST](#)

Meeting request status: Pending

[Home](#)
[Wall](#)
[Profile](#)
[Student's work](#)
[Information](#)

Your post here.

Vlad Simion
Hello everybody! I've seen some of your papers. You did a very good job!
25-Jun-2018, 12:00
[Show comments](#)

Your comment here.

Vlad Simion
Hello! I hope that very soon we will celebrate you as fresh graduates. :)
3-Jun-2018, 12:27
[Hide comments](#)

Vlad Simion
I know we will be a great team!
4-Jun-2018, 14:40

Mihai Ursache
We hope this, too! See you soon.
5-Jun-2018, 0:27

Your comment here.

Figura 15: Pagină pentru adăugarea vizualizare detaliilor și cerințelor unui cadru didactic

Atât profesorul coordonator, cât și studenții au acces la o serie de informații despre toți candidații înscriși la același profesor, design vizibil în *Figura 16*. Aceste detalii sunt: numele, performanțele din cadrul facultății, adresa la care codul sursă este disponibil și ultimele 5 actualizări făcute de fiecare.

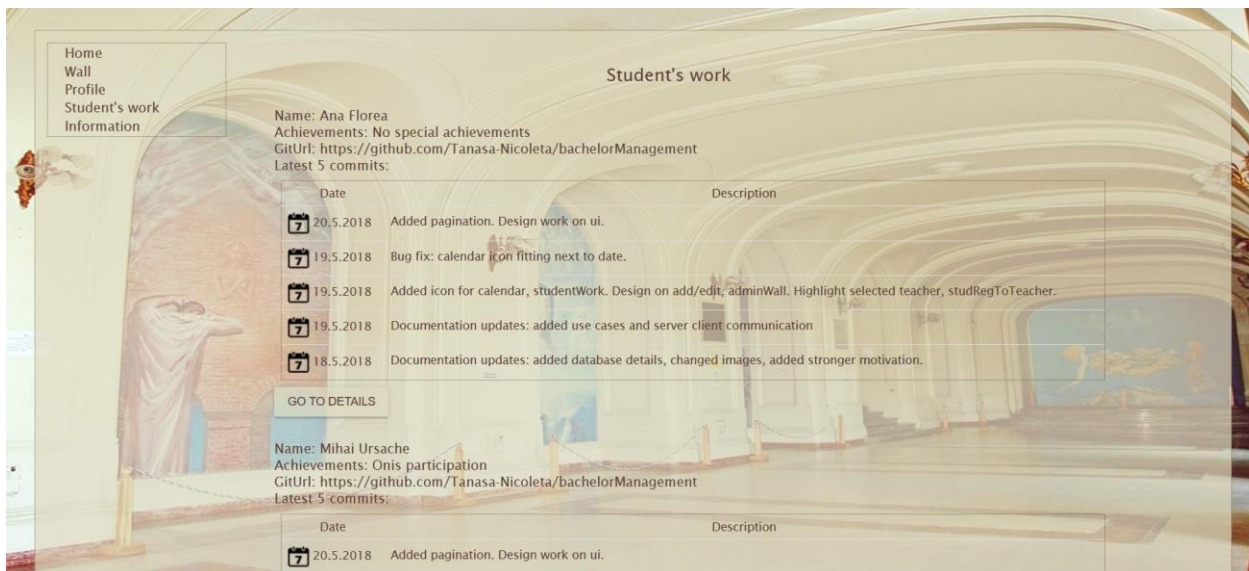


Figura 16: Pagină pentru vizualizare detaliilor fiecărui student

Pentru fiecare student care apare pe pagina menționată anterior, se poate naviga spre o pagină specifică acestuia în care vor fi prezentate informații mult mai detaliate cu privire la munca pe care acesta a demarat-o pe parcursul ultimelor luni. Se pot vedea informații despre codul sursă, cum ar fi: adresa la care acesta poate fi accesat și vizualizat, o diagramă care expune procentajul limbajelor de programare folosite (diagramă de tip disc cu ajutorul căreia informațiile se percep foarte ușor și care folosește culori calde pentru oferi utilizatorului o experiență plăcută), o diagramă ce expune progresul, măsurat ca și număr de linii de cod șterse și adăugate în ultimele 10 săptămâni (scara acestei diagrame este 1:100, ceea ce se traduce ca înmulțirea cu 100 a tuturor valorilor situate pe diagramă, culorile folosite sunt roșu pentru numărul de linii șterse respectiv verde pentru numărul de linii adăugate, acestea intercalându-se când este cazul și oferind utilizatorului o vedere de ansamblu exactă și rapidă) și ultimele 30 de actualizări pe care le-a făcut în dezvoltarea lucrării sale de licență (acestea fiind organizate în grupuri de câte cinci, disponibile în acest format cu ajutorul paginării ce are opțiuni pentru pagina anterioară, pagina următoare și trei dintre pagini disponibile pentru a fi selectate). Toate acestea se pot observa în *Figura 17*, respectiv *Figura 18*.

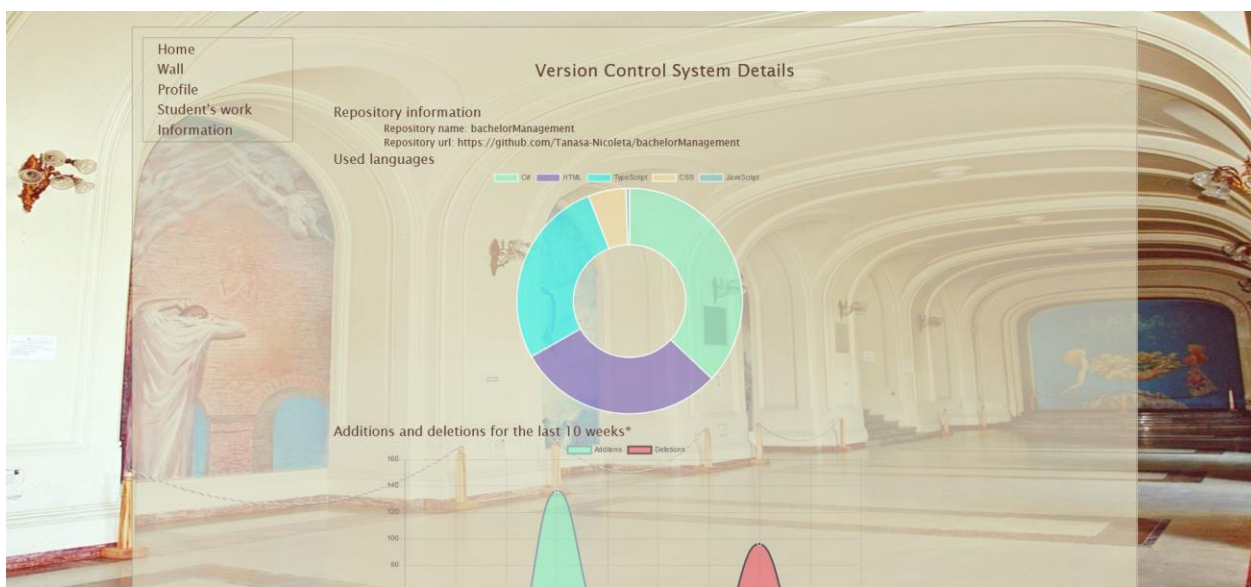


Figura 17: Pagină pentru vizualizare detaliilor unui student ales



Figura 18: Pagină pentru vizualizare detaliilor unui student ales – paginare

Fiecare utilizator, exceptând administratorul sistemului, are acces la o pagină cu informații specifice profilului său, design vizibil în *Figura 19*. Studenții au acces la informații de tipul: nume, numele profesorului coordonator asignat, numele temei lucrării de licență, descrierea temei respective, adresa la care aceasta poate fi găsită, numărul matricol, anul în care studentul a început studiile în cadrul facultății și mediile lui.

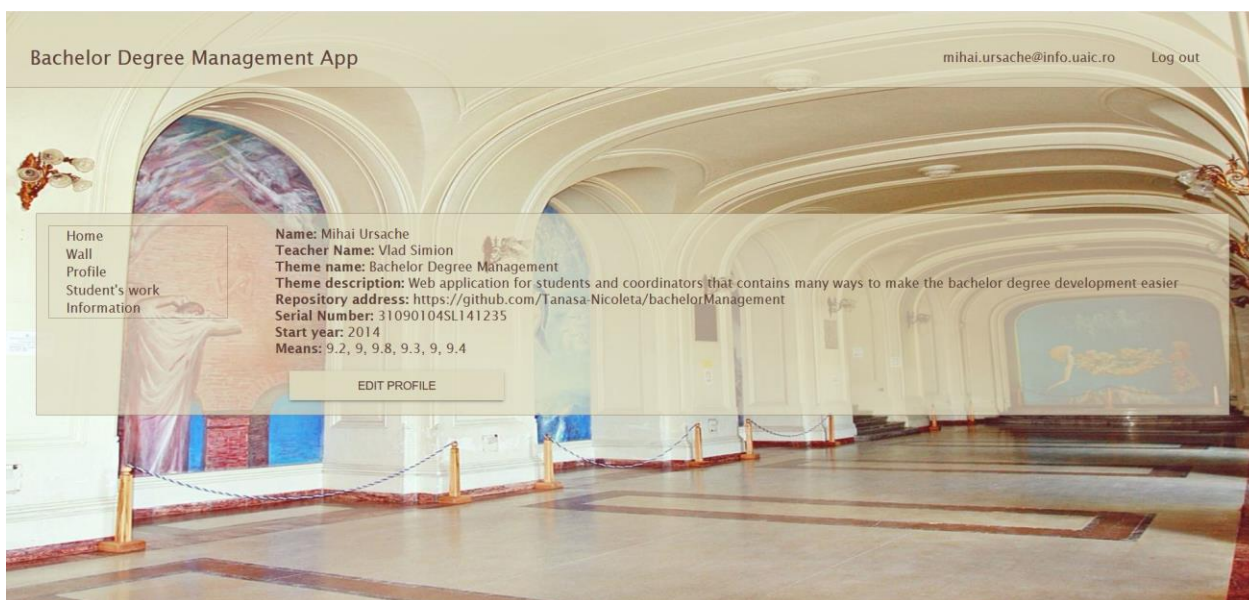


Figura 19: Pagină pentru vizualizarea profilului unui utilizator de tip student

De asemenea, utilizatorului de tip student i se permite editare unei anumite selecții de informații cum ar fi: adresa la care se găsește codul sursă, titlul lucrării pe care o dezvoltă și descrierea acestei lucrări, aspecte vizibile în *Figura 20*.

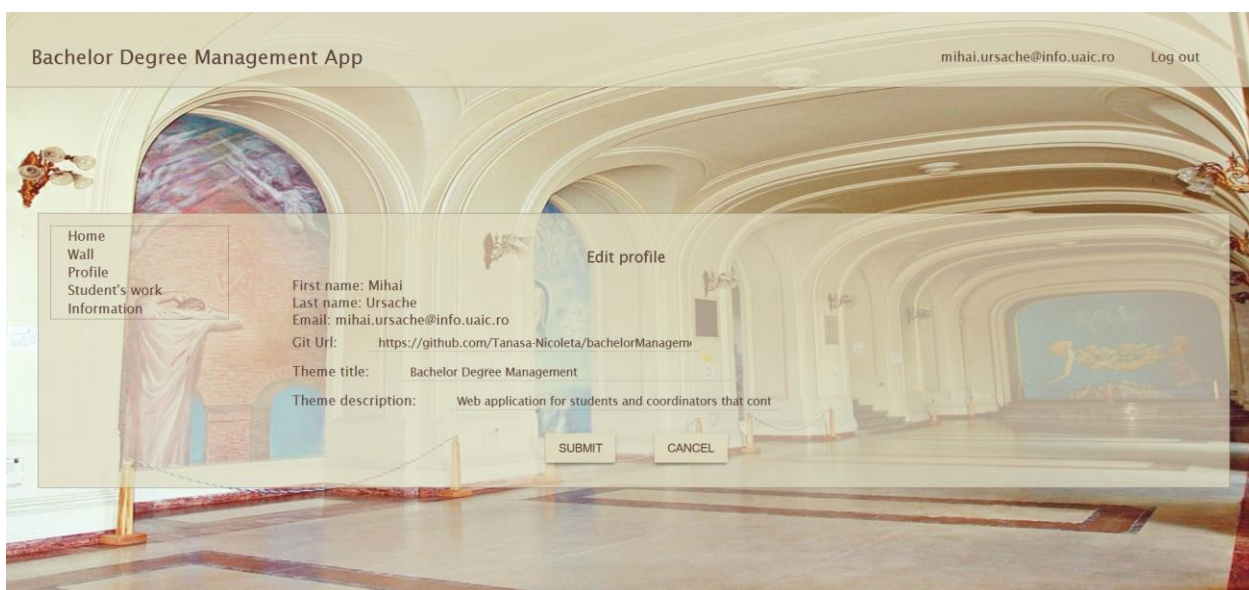


Figura 20: Pagină pentru editarea profilului unui utilizator de tip student

La fel ca în cazul studenților, și cadrele didactice au acces la o pagină de tip profil în care au disponibile informații despre numele și titlul lor, cerințele pe care le au de la studenți, ziua și ora în care au disponibilitate pentru întâlniri în persoană cu studenții, numele și descrierea temei

propușe de către ei, numele, titlul, descrierea și adresa lucrării de licență a fiecărui student înscris spre coordonarea de către acesta. Ei pot edita numai informațiile care privesc ziua și ora disponibilă pentru întâlniri cu studenții. Design-ul acestei pagini este disponibil în *Figura 21*. În cazul în care profesorul alege să își modifice ziua și intervalul stabilit pentru întâlniri în persoană cu studenții, pot apăsa butonul corespunzător acestei acțiuni și un mic formular își face apariția în cadrul aceleiași pagini. După ce informațiile sunt actualizate corespunzător, cadrul didactic apasă butonul caracteristic salvării detaliilor și, din nou, rămâne în aceeași pagină.

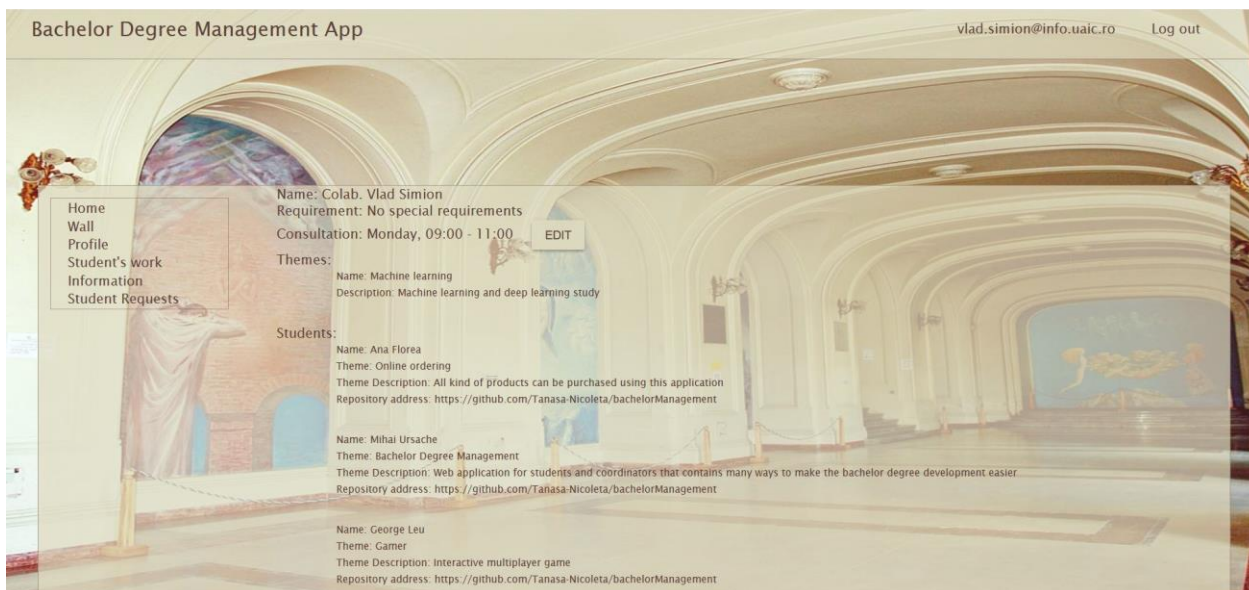


Figura 21: Pagină pentru vizualizarea profilului unui utilizator de tip profesor

Utilizatorii de tip profesor și cei de tip student au, de asemenea, acces la o pagină care oferă informații utile, *Figura 22*. Acestea se referă la perioadele de înscriere, de evaluare și de susținere a prezentării propriu-zise a lucrării de licență și documentele necesare pentru înscrierea în cadrul sesiunii curente. Se pot vizualiza și câteva sfaturi de redactare și prezentare, menite să le ușureze tot traseul de pregătire a lucrării de licență, ca parte practică dar și ca parte teoretică. Aceste informații sunt identice cu cele care sunt postate pe pagina oficială a Facultății de Informatică din Iași.

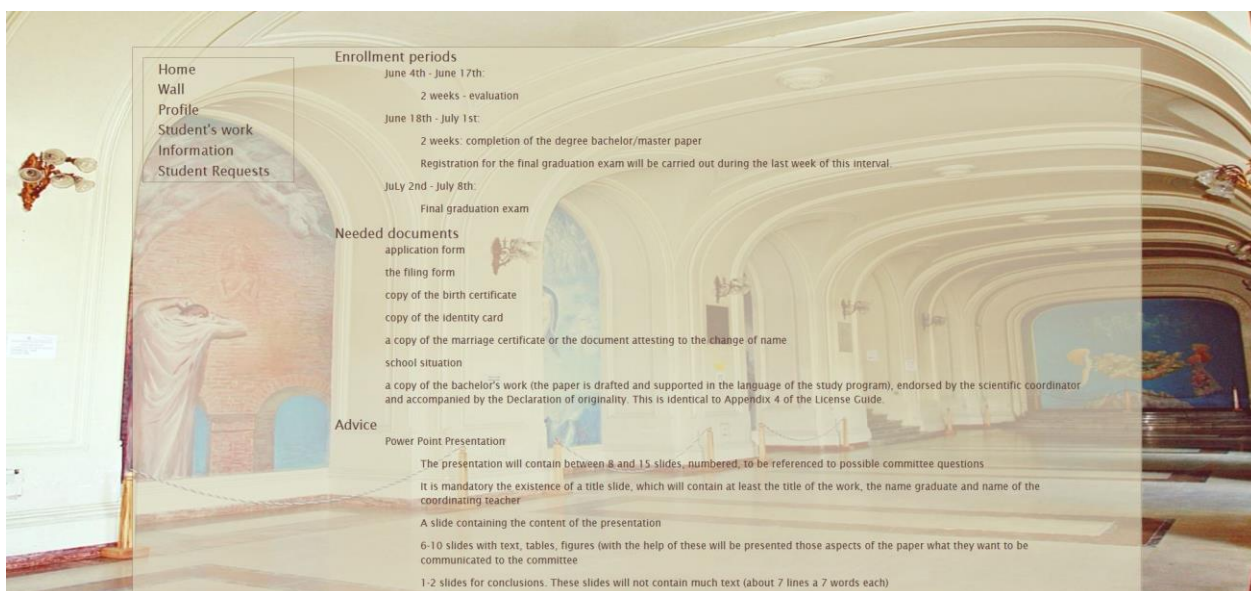


Figura 22: Pagină pentru vizualizarea informațiilor disponibile pentru sesiunea curentă

Un al treilea rol existent în cadrul aplicației este, după cum s-a menționat anterior, cel de administrator de sistem. Pagina pe care acesta o poate accesa este vizibilă în *Figura 23*. Responsabilitățile administratorului sunt de a menține lista de profesori mereu actualizată prin adăugarea, respectiv ștergerea conturilor cadrelor didactice, după caz. Există un singur cont de administrator, acesta putând fi împărțit de mai multe persoane, dacă se dorește.

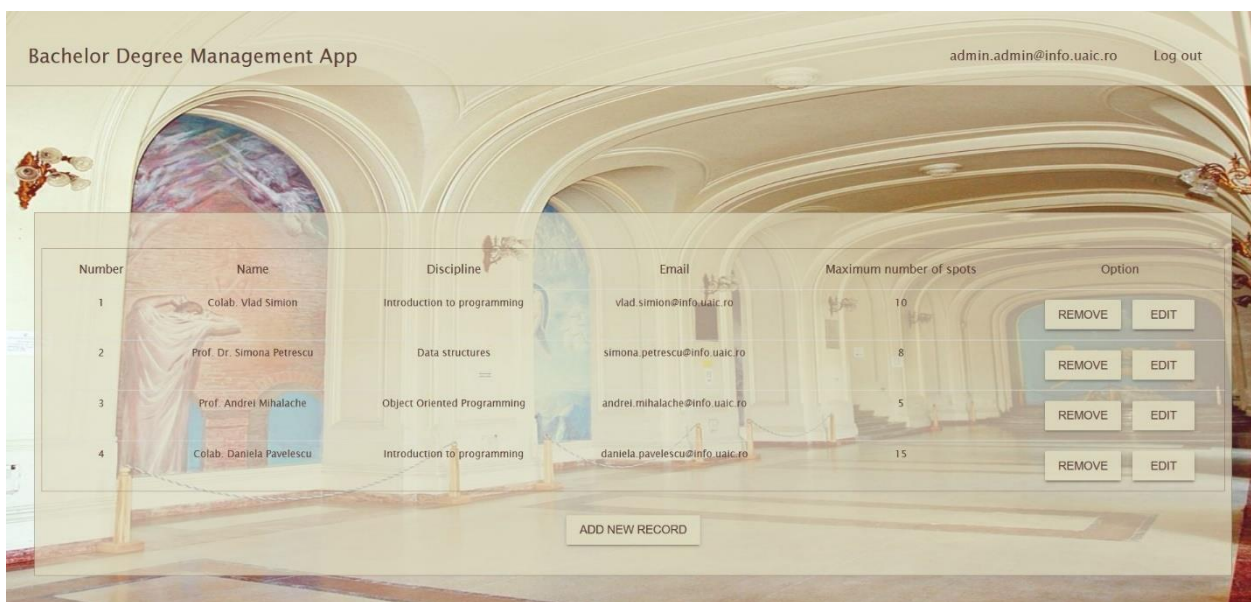


Figura 23: Pagină vizualizată de către administrator la accesul în sistem

Pentru a adăuga o nouă înregistrare de tip profesor coordonator, administratorul trebuie să dețină, în prealabil, informații despre numele, prenumele, adresa de email, disciplina pe care o predă și gradul profesorului, lucru vizibil în *Figura 24*. El adaugă, de asemenea, un număr maxim de studenți pe care un profesor îi poate accepta în sesiunea curentă. Acest număr reprezintă limita superioară la care poate ajunge un profesor cu numărul de studenți coordonați într-o anumită sesiune. În cazul în care, din diferite motive, administratorul nu vrea să finalizeze operația tocmai începută (și anume cea de a insera un cadru didactic nou în sistem) are la dispoziție un buton ce îi permite să o anuleze.

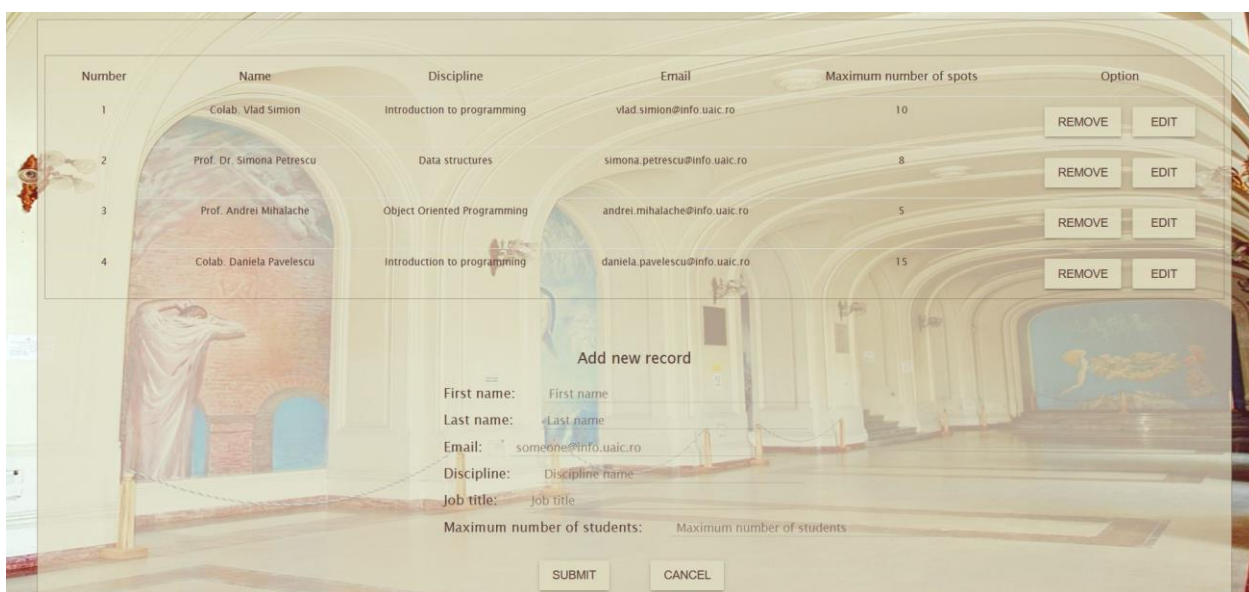


Figura 24: Pagină care permite adăugarea unei noi înregistrări de tip profesor, disponibilă administratorului de sistem

Dacă alege opțiunea de a edita o înregistrare de tip profesor, administratorului de sistem îi este permisă actualizarea doar a unei serii de date, adică modificarea numelui, prenumelui, disciplinei predate, gradului didactic și a numărului de studenți aferent sesiunii curente. La fel ca și în cazul adăugării unei înregistrări noi, există posibilitatea opririi operației de actualizare care este în progres dacă, din diferite motive, se dorește acest lucru. Atât adăugarea cât și actualizarea datelor au loc nominal (pentru un singur profesor la un moment dat) iar schimbările efectuate vor fi imediat vizibile pentru administrator. Pagina cu ajutorul căreia unele detalii ale unei înregistrări de tip profesor pot fi actualizate sunt disponibile în *Figura 25*.

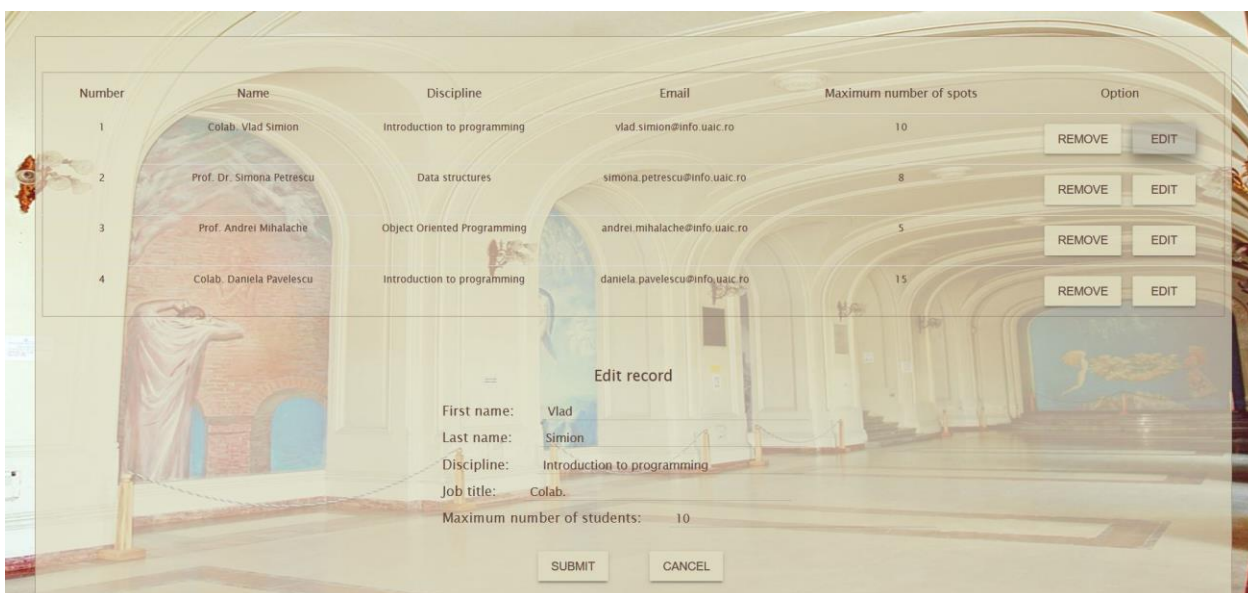


Figura 25: Pagină care permite editarea unei înregistrări de tip profesor, disponibilă administratorului de sistem

4.2 Diagrame

4.2.1 Diagrame pentru cazuri de utilizare

Fiecare utilizator trebuie să fie înregistrat în sistem înainte de a putea face orice fel de acțiune. După cum menționam anterior, în cadrul aplicației există definite trei tipuri de utilizatori: student, profesor coordonator și administrator de sistem. Fiecare dintre aceștia pot demara mai multe tipuri de acțiuni astfel interogând sau modificând baza de date asociată aplicației.

Toate cele trei tipuri de utilizatori pot vizualiza diferite tipuri de liste și pot acționa asupra elementelor acestora (adăugare, ștergere, editare, acceptare, refuzare) iar utilizatorii de tip student sau profesor coordonator pot să vizualizeze și editeze profilul personal și doar să îl vizualizeze pe cel al celorlalți participanți. Toate aceste acțiuni sunt explicate cu ajutorul unor diagrame în figurile de mai jos, respectiv intervalul *Figura 26 – Figura 30*.

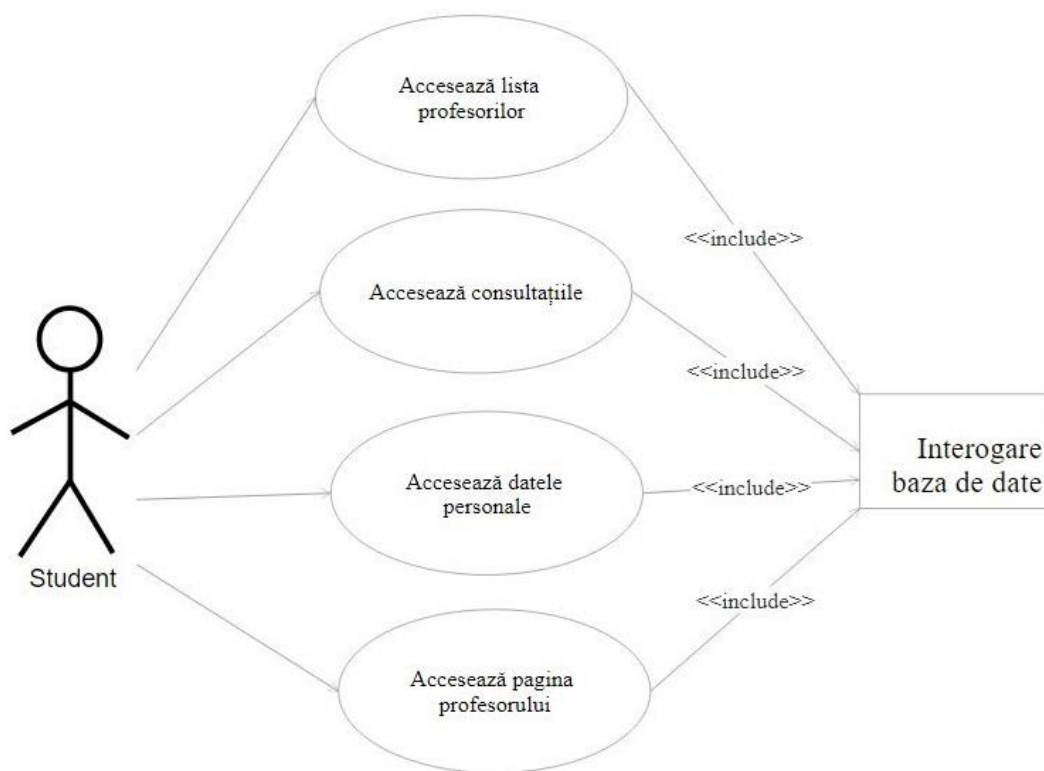


Figura 26: Acțiunile care generează interogarea bazei de date pentru studenți

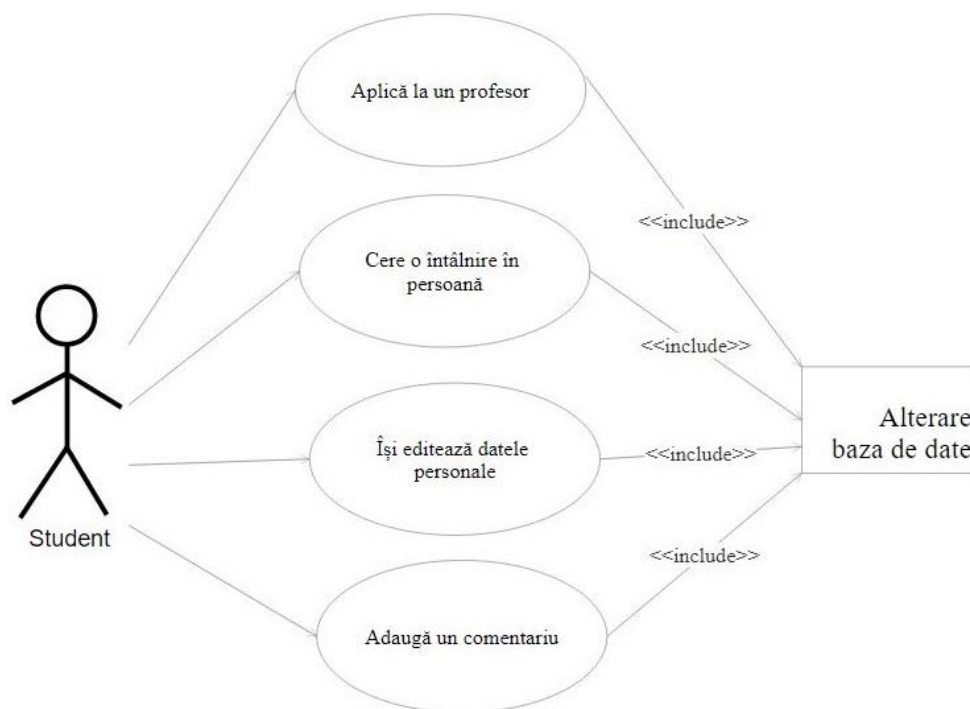


Figura 27: Acțiunile care generează alterarea bazei de date pentru studenți

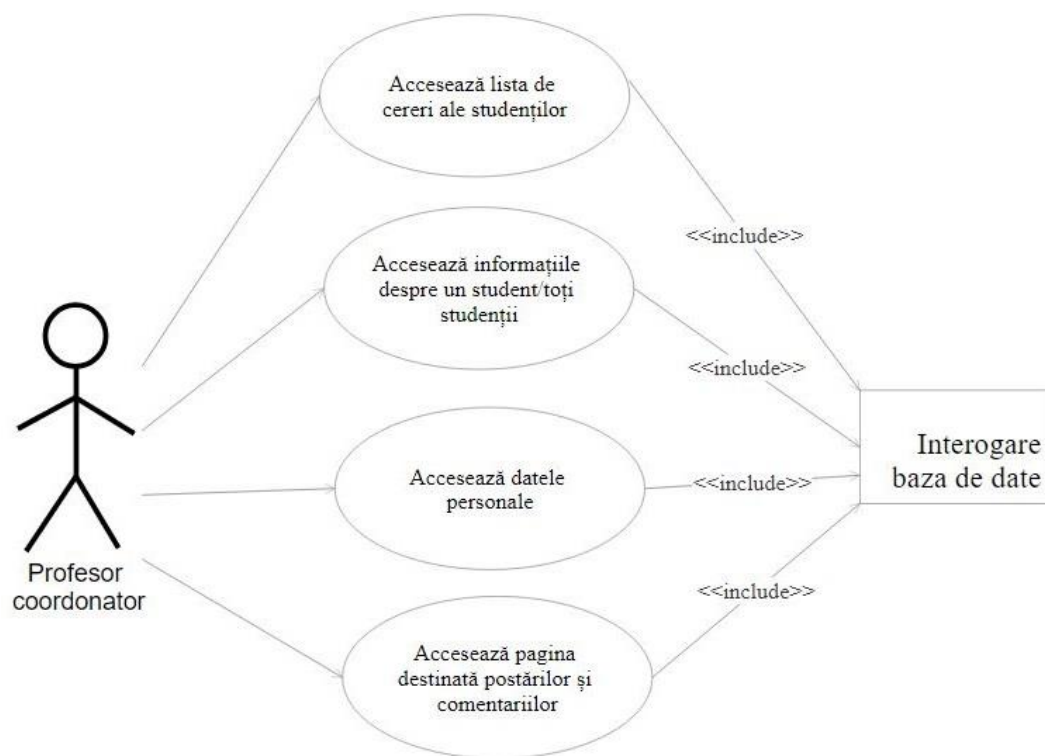


Figura 28: Acțiunile care generează interogarea bazei de date pentru profesori

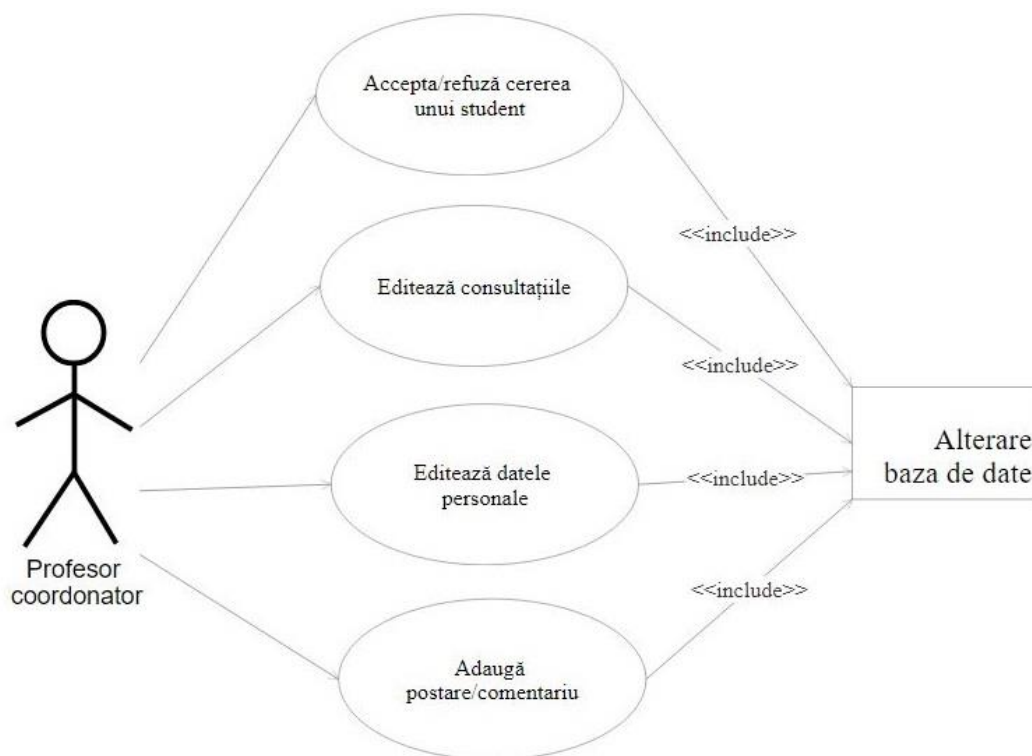


Figura 29: Acțiunile care generează alterarea bazei de date pentru profesori



Figura 30: Acțiunile care generează interogarea/alterarea bazei de date pentru administrator

4.2.2 Diagrama structurii soluției

Soluția este structurată pe mai multe nivele asigurându-se astfel, un mod optim de a se face managementul și mentenanța codului sursă, fapt vizibil în *Figura 31* și *Figura 32*. Fiecare nivel este reprezentat de un proiect care, la rândul lui este împărțit în dosare și subdosare pentru a permite localizarea ușoară a fișierelor.

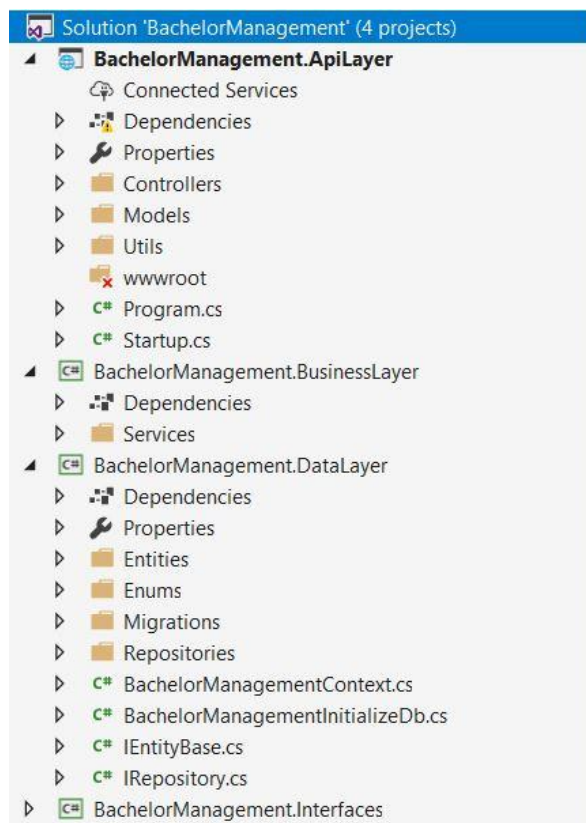


Figura 31: Structura soluției, Visual Studio

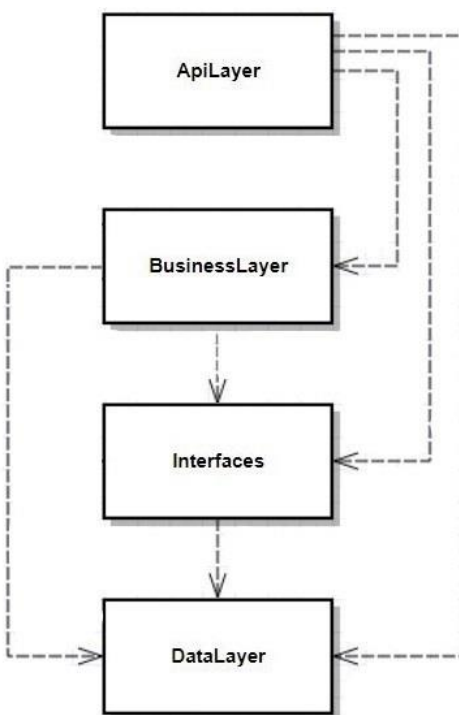


Figura 32: Structura soluției – proiecte și dependențe

ApiLayer – acesta este nivelul destinat API-ului REST construit. Acesta conține mai multe Controllere (clase ce expun servicii către partea de client a aplicației și care pot fi accesate cu ajutorul protocolului HTTP), mai multe modele (clase care descriu cum ar trebui să arate obiectele pe care serviciile le primesc de la partea client sau le trimit către partea client) și o clasă (în cadrul dosarului Utils) cu ajutorul căreia se verifică faptul că formatul atomului lexical (token) primit de la utilizatori este corect acest lucru eficientizând timpul de răspuns (această verificare se face imediat ce o cerere ajunge către server și în caz că nu sunt îndeplinite condițiile necesare se trimite imediat un răspuns corespunzător către client, fără a se mai executa cererea în sine).

BusinessLayer – acesta este nivelul destinat logicii ce are loc pentru a face răspunsul la cererea clientului posibil. Conține mai multe servicii, împărțite pe entități (cont, student, profesor, comentariu etc.) ce au responsabilitatea de a merge către locul de stocare a datelor și a efectua operații de interogare, adăugare, ștergere sau editare a acestora.

Interfaces – este nivelul care face, într-o anumită măsură, legătura dintre nivelul de date și nivelul de logică. Acesta conține doar interfețe ce descriu comportamentul principal pe care nivelul de logică trebuie să îl implementeze și care folosește obiecte definite în nivelul de date.

DataLayer – este nivelul responsabil cu manipularea bazei de date. Aici se regăsește codul aferent entităților (după cum am menționat anterior, mai întâi s-a creat codul corespunzător entităților și, mai apoi, acestea au fost generate), un depozit de date generic (pe care toate depozitele entităților îl moștenesc), repositoryele fiecărei entități, un context cu ajutorul căruia se construiește baza de date, o clasă ce inițializează baza de date prin introducerea unor valori în cazul în care aceasta este goală și un dosar cu migrări (Entity Framework funcționează cu ajutorul acestor migrări care sunt, practic, un istoric al modificărilor aduse structurii bazei de date).

4.3 Modelarea datelor

Datele necesare dezvoltării și manipulării aplicației sunt salvate într-o bază de date de tip relațional (adică formată dintr-un ansamblu de tabele și legăturile/relațiile dintre ele). Schema acestei baze de date este disponibilă în figura următoare, și anume *Figura 33*.

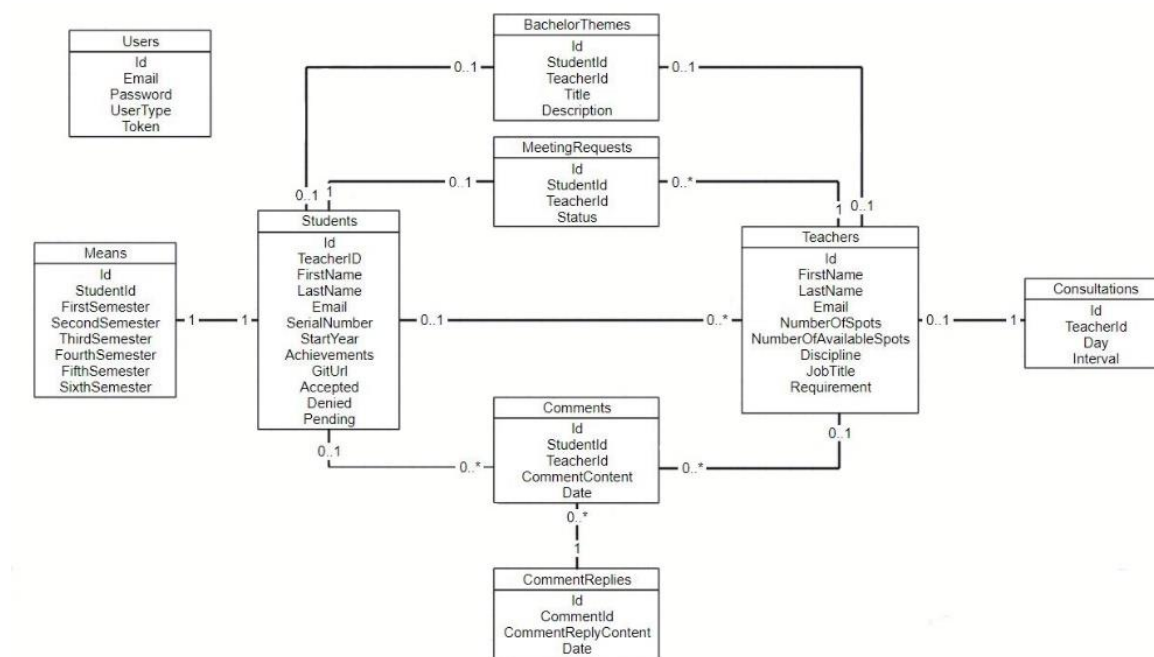


Figura 33: Diagrama bazei de date

Entități

Users – este tabela care definește orice utilizator ce intra în contact cu aplicația și aparține Facultății de Informatică din Iași. Fiecare utilizator are un email, o parolă, un tip de utilizator (din

cele trei posibile: student, profesor, administrator) și un semn (eng. token) cu ajutorul căruia se verifică autentificarea lui înainte de fiecare operație.

Students – este tabela care definește studenții care se înregistrează în aplicație. Aceștia au informații de tipul: nume complet, email, număr matricol, anul începerii facultății, reușitele lor din timpul facultății, o adresă la care va putea fi găsită lucrarea lor de licență și trei câmpuri indicatoare (dacă studentul este acceptat de către profesor, dacă este refuzat de către profesor sau dacă încă este în stare de așteptare pentru un răspuns din partea profesorului).

Teachers – este tabela care definește profesorii care pot juca rolul de coordonator. Aceștia au informații de tipul: nume complet, email, numărul maxim de studenți pe care îi poate coordona, numărul de studenți pe care dorește să îi coordoneze (limitat superior de numărul maxim de studenți), disciplina pe care o predau, gradul pe care îl au ca și cadre didactice și cerințele minime pe care studenții trebuie să le îndeplinească pentru a fi acceptați.

Means – este tabela asociată numai studenților care conține informații despre mediile pe care un student le are din primul semestru de facultate și până în semestrul curent, numărul maxim de semestre fiind șase.

BachelorThemes – este tabela care definește temele lucrărilor de licență ce vor fi realizate în sesiunea curentă. Acestea conțin informații despre deținătorul temei, despre titlul acesteia și o mică descriere asociată ei.

MeetingRequests – este tabela care definește statusul unei întâlniri solicitate de student către profesorul său coordonator, acesta putând fi: în așteptare, acceptat sau respins.

Comments – este tabela care definește comentariile ce pot fi adăugate, atât de studenți cât și de coordonatori pe pagina aferentă coordonatorilor. Aceasta conține informații despre autorul comentariului, conținutul comentariului și data la care acesta a fost adăugat.

CommentReplies – este tabela care definește răspunsurile la comentariile detaliate mai sus. Structura este asemănătoare cu cea a tabelii aferente comentariilor dar mai are în plus id-ul comentariului la care este asociat răspunsul.

Consultations – este tabela asociată doar profesorilor coordonatori și definește o zi și un interval din acea zi în care cadrul didactic este disponibil pentru a primi solicitări de întâlniri în persoană cu studenții pe care îi coordonează.

Legături între entități

Students – Teachers: este o relație de tip unul la mai mulți. Un student poate avea un singur profesor coordonator, însă un cadru didactic poate îndruma mai mulți studenți.

Students – MeetingRequests: este o relație de tip unul la unul. Un student poate avea, la un moment dat, o singură cerere de întâlnire personală cu profesorul său coordonator și o astfel de întâlnire poate avea asignat un singur student.

Students – Means: este o relație de tip unul la unul. Un student poate avea o singură înregistrare corespondentă care să îi indice notele pe parcursul semestrelor petrecute în cadrul facultății și o înregistrare de tip medii poate aparține unui singur student.

Students – BachelorThemes: este o relație de tip unul la unul. Un student poate avea o singură temă de licență asignată și o temă de licență poate aparține unui singur student.

Students – Comments: este o relație de tip unul la mai mulți. Un student poate avea mai multe înregistrări de tip comentariu asociate dar un comentariu poate avea un singur autor, mai exact un singur student.

Teachers – BachelorThemes: este o relație de tip unul la unul. Un profesor coordonator poate avea o singură temă de licență asignată și, în același timp, o temă de licență poate aparține unui singur cadru didactic.

Teachers – Comments: este o relație de tip unul la mai mulți. Un profesor coordonator poate avea mai multe înregistrări de tip comentariu asociate dar un comentariu poate avea un singur autor, mai exact un singur profesor.

Teachers – Consultations: este o relație de tip unul la unul. Un profesor coordonator poate avea un singur slot de timp asignat la un moment dat și, în același timp, o înregistrare de tip consultație poate aparține unui singur cadru didactic.

Comments – CommentReplies: este o relație de tip unul la mai mulți. Un comentariu poate avea mai multe înregistrări de tip răspuns, dar un răspuns poate aparține unei singure înregistrări de tip comentariu.

4.4 Comunicarea server-client

Comunicarea între cele două entități, server și client, se face cu ajutorul protocolului HTTP. Fiind un protocol utilizat pe scară largă, aplicația de tip client se poate schimba în orice moment, chiar fiind posibilă existența mai multor clienți de tipuri diferite în același timp (un client poate fi o aplicație web care folosește un motor de căutare, un client poate fi o aplicație de tip mobil sau o aplicație special concepută pentru a simula cererile unui client către server cum ar fi Postman, aplicație desktop folosită în timpul dezvoltării curetei lucrări de licență). Acesta pune la dispoziție o sumedenie de metode prin care se pot obține date iar cele folosite în aplicația curentă sunt:

- metoda GET: este o metodă cu ajutorul căreia se pot cere date de la o rută specificată. Așa cum se observă în *Figura 34* și în *Figura 35* metoda are nevoie de o rută pe care o va apela utiliza aplicația client pentru a obține datele necesare și, posibil, de niște parametri. Serverul poate răspunde cu ajutorul unui obiect sau doar cu o variabilă de orice tip.

```
[HttpGet("getAccessToken/{email}")]  
public IActionResult GetAccesToken(string email)
```

Figura 34: Signatura unei metode de tip get GET, AccountController

```
var commentResponse =  
this.http.get('http://localhost:64250/api/student/studentMeetingRequestSta  
tus/' + this.studentEmail + '/' + this.studentEmail + '/' + this.token,  
{ observe: 'response' });
```

Figura 35: Parte din apelul unei metode de tip get GET

- metoda POST: este o metodă cu ajutorul căreia se adaugă noi date. Așa cum se observă în *Figura 36* și în *Figura 37* serverul are nevoie să primească de la client un obiect populat cu anumite date pe care, mai apoi, le prelucrează în diferite moduri și, în cele din urmă, inserează o nouă înregistrare în baza de date.


```
[HttpPost("register")]
public IActionResult Register([FromBody] AccountDto accountDto)
```

Figura 36: Signatura unei metode de tip POST, AccountController

```
const commentReplyResponse =
this.http.post('http://localhost:64250/api/student/studentMeetingRequest',
this.meetingRequestBody, { observe: 'response' });
```

Figura 37: Parte din apelul unei metode de tip get POST

- metoda PUT: este o metodă cu ajutorul căreia, de cele mai multe ori, se actualizează datele deja existente în sistem. În cazul în care datele care se vor a fi actualizate nu există, metoda PUT preia din rolul metodei POST și inserează o nouă înregistrare. La fel ca și metoda anterioară, după cum se observă în *Figura 38* și *Figura 39*, și aceasta are nevoie ca aplicația client să trimită un obiect prepopulat cu datele necesare pe care serverul le administrează și decide care sunt cele ce trebuiesc actualizate sau dacă este nevoie de crearea unei înregistrări noi.

```
[HttpPut]
[Route("api/student/editStudent")]
public IActionResult EditStudent([FromBody] EditStudentDto editStudentDto)
```

Figura 38: Signatura unei metode de tip get PUT

```
const resp = this.http.put('http://localhost:64250/api/student/editStudent',
this.editProfileBody, { responseType: 'text' });
```

Figura 39: Parte din apelul unei metode de tip get PUT

Concluzii

Prezenta lucrare de licență demonstrează faptul că folosind o suită bine aleasă de tehnologii și instrumente software se poate realiza o aplicație web, cu scară de utilizare medie dar cu posibilitate de extindere, care să fie utilizată cu succes în mediul academic și care să ușureze modul de lucru al utilizatorilor săi.

Având în vedere că tehnologia se află într-o continuă dezvoltare, cel mai benefic pentru noi devine faptul că o putem utiliza oricând, merintându-se a fi create tot felul de aplicații care să ne ajute în atingerea acestui scop.

Așadar, aplicația “**Managementul lucrărilor de licență**” se pretează a veni ca sprijin atât pentru generațiile viitoare de studenți ce vor trece pragul Facultății de Informatică din Iași, cât și pentru cadrele didactice ce activează în cadrul acesteia. Aceasta reușește să ajute toți utilizatorii să economisească timp prețios, să se descopere între ei (studenții au acces la ideile domnilor profesori coordonatori și invers) și să asigure o colaborare eficientă și armonioasă.

Versiunea curentă a aplicației poate fi îmbunătățită în mai multe direcții:

- se poate adăuga funcționalitatea întâlnirilor în grupuri de persoane (nu doar întâlnirile student - coordonator), studenții putându-și acorda sfaturi între ei;
- având în vedere că aplicația este deja integrată cu GitHub API, se pot aplica diverși algoritmi asupra datelor extrase pentru a putea face predicții privind rata de succes a lucrării sau nevoie de accelerare a ritmului de muncă;
- tot din existența colaborare cu GitHub API, se poate implementa un mecanism care să declanșeze anumite acțiuni atunci când un student își actualizează codul sursă și acesta ajunge să fie încărcat în sistemul de versionare;
- funcționalitatea curentă se poate extinde pentru a avea utilizatori din toată Universitatea Alexandru Ioan Cuza din Iași sau poate din toată țara, dacă aplicația se dovedește a fi utilă, momentan ea acceptând doar utilizatori ce au o legătură directă cu Facultatea de Informatică din Iași.

Bibliografie

1. Principii REST: <https://www.c-sharpcorner.com/uploadfile/kalisk/rest-fundamentals/>
2. Hypertext Transfer Protocol: <https://developer.mozilla.org/en-US/docs/Web/HTTP>
3. GitHub: <https://github.com/>
4. ASP NET Core: <http://www.jomendez.com/2017/02/15/asp-net-core-1-0/>
5. ORM: <https://searchwindevelopment.techtarget.com/definition/object-relational-mapping>
6. Entity Framework Core: <http://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx>
7. HTML pentru începători: <https://www.w3schools.com/html/>
8. Angular : <https://softwaredevelopment.ae/angular-best-solution-2018-web-app/>
9. Cele mai populare tehnologii: <https://insights.stackoverflow.com/survey/2017#mostpopular-technologies>
10. Documentație Visual Studio Code: <https://code.visualstudio.com/docs>
11. Detalii GitHub: <https://www.atlassian.com/git/tutorials/what-is-git>
12. Edusoft: <https://www.edusoft.ro/s>
13. Curs Baze de Date, Lect. Dr. Cosmin Vârlan: https://profs.info.uaic.ro/~bd/wiki/index.php/Pagina_principal%C4%83
14. Curs Programare Orientată Obiect, Conf. Dr. Dragoș Gavriluț și Prof. Dr. Dorel Lucanu: <https://sites.google.com/site/fiicoursepoo/>
15. Curs Ingineria Programării, Conf. Dr. Adrian Iftene: <https://profs.info.uaic.ro/~adiftene/Scoala/2018/IP/index.htm>
16. Curs Tehnologii Web, Conf. Dr. Sabin Corneliu Buraga: <https://profs.info.uaic.ro/~busaco/teach/courses/web/>
17. Curs Rețele de Calculatoare, Conf. Dr. Lenuța Alboaie: <https://profs.info.uaic.ro/~adria/teaching-area.html>
18. Curs Dezvoltarea de aplicații pe platforma .NET, Lect. Ioan Asiminoaei: <https://profs.info.uaic.ro/~iasimin/>

Anexe

Anexa 1 - Limbaje și cadre de programare

.NET CORE are la bază limbajul C# și are ca principale avantaje faptul că poate rula pe mai multe platforme (nu doar Windows, cum eram obișnuiți, ci și pe Linux sau MacOS), compatibilitatea totală cu vechiul framework, menționat anterior, și faptul că există posibilitatea accesării codului sursă, acesta fiind public.

ASP.NET Core este complet accesibil și disponibil pe site-ul GitHub⁹, site de găzduire al proiectelor software. Utilitatea principală a acestui framework este că aplicațiile dezvoltate cu ajutorul lui pot rula atât pe tehnologia .NET Core, cât și pe .NET Framework (fapt vizibil în *Figura 40*).

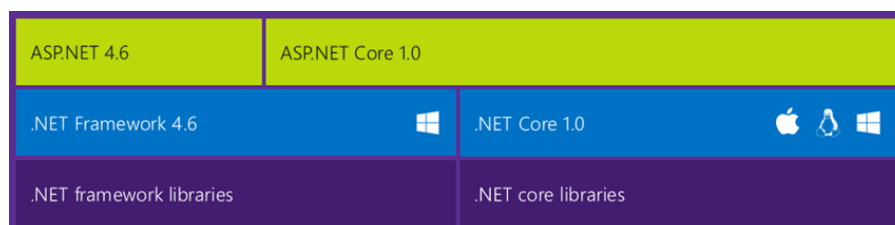


Figura 40: ASP.NET vs. ASP .NET Core¹⁰

În **Entity Framework Core** există două abordări posibile ale acestei versiuni, și anume Code-First (mai întâi se scriu modelele și mai apoi se generează tabelele bazei de date cu ajutorul lor) și Database-First (mai întâi se creează tabelele bazei de date și mai apoi se generează modelele aferente), după cum se observă în *Figura 41*.

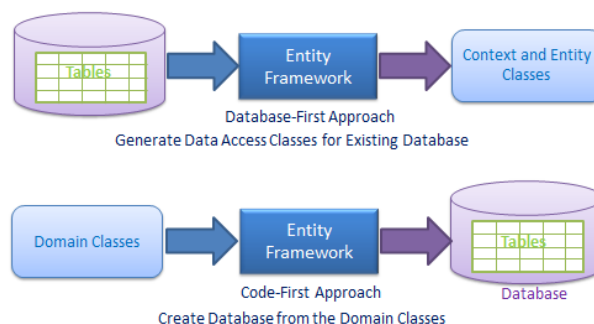


Figura 41: Abordări Entity Framework Core¹¹

⁹ GitHub: <https://github.com/>

¹⁰ Sursa: <http://www.jomendez.com/2017/02/15/asp-net-core-1-0/>

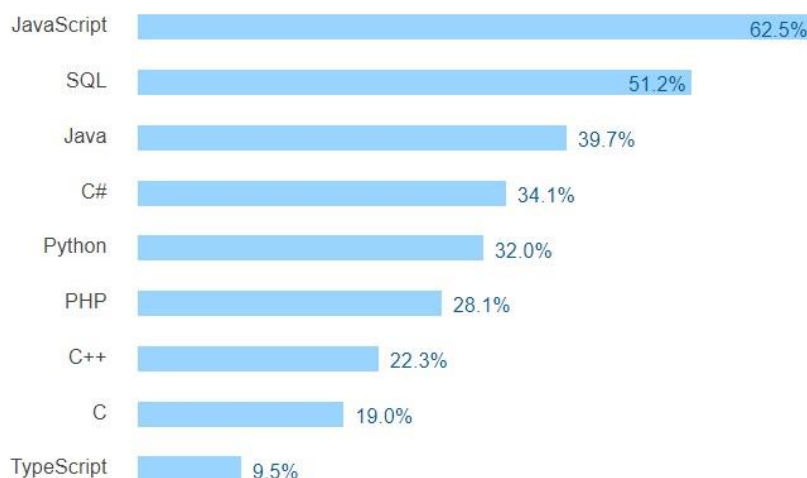
¹¹ Sursa: <http://www.entityframeworktutorial.net/efcore/entity-framework-core.aspx>

Dintre aceste două abordări, cea utilizată în cadrul aplicației curente este Entity Framework Core Code-First. Principalul motiv este acela că permite utilizarea structurilor de tip clasă pentru a reprezenta modelul pe care se bazează Entity Framework pentru a efectua orice fel de operații la baza de date. Codul este scris mai întâi iar apoi modelul este generat pe baza acestuia.

HTML (HyperText Markup Language) este un limbaj de marcare utilizat pentru crearea paginilor web ce pot fi afișate într-un browser scopul acestuia fiind de a prezenta informațiile (font, tabelă, paragraf) și putând fi construit folosind un simplu editor de texte. Elementele acestuia, reprezentate prin structuri numite tag-uri, alcătuiesc scheletul unei pagini, unele exemple fiind `<head>`, `<body>`, `<header>`, `<footer>`¹².

SCSS (Sassy CSS) este o formă îmbunătățită a limbajului CSS (Cascading Style Sheets) care se scrie în același mod dar permite folosirea caracteristicilor limbajului SASS (Syntactically Awesome Style Sheets), acesta fiind un preprocesor CSS.

C# este un limbaj de programare orientat-obiect conceput de Microsoft la sfârșitul anilor 90. .NET Core permite dezvoltarea aplicațiilor folosind oricare din limbajele C#, F# sau Visual Basic. Eu am ales folosirea primului dintre ele deoarece este unul dintre cele mai puternice, pe care îl utilizează de milioane de alți oameni după cum poate fi observat în *Figura 42*.



*Figura 42: Fragment dintr-o imagine care conține cele mai populare tehnologii în 2017*¹³

¹² HTML pentru începători: <https://www.w3schools.com/html/>

¹³ Studiu, imagine: <https://insights.stackoverflow.com/survey/2017#mostpopular-technologies>