

UNIVERSITATEA POLITEHNICA BUCUREȘTI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE



PROIECT DE DIPLOMĂ

LabGradR

Platformă de e-learning destinată laboratoarelor

George-Daniel Glod

Coordonator științific:

Prof. dr. ing. Mariana Mocanu

As. drd. ing. Alexandru Predescu

BUCUREȘTI

2021

UNIVERSITY POLITEHNICA OF BUCHAREST
FACULTY OF AUTOMATIC CONTROL AND COMPUTERS
COMPUTER SCIENCE DEPARTMENT



DIPLOMA PROJECT

LabGradR
E-learning platform for laboratories

Glod George-Daniel

Thesis advisor:

Prof. dr. ing. Mariana Mocanu
As. drd. ing. Alexandru Predescu

BUCHAREST

2021

CUPRINS

Lista de figuri.....	3
Sinopsis	5
Abstract	5
1 Introducere	6
1.1 Context	6
1.2 Problema	6
1.3 Obiective	7
1.4 Structura lucrării.....	7
2 Analiza și specificarea cerințelor	8
3 Abordări existente	12
4 Soluția propusă.....	15
4.1 Baza de date	15
4.2 Back-End	16
4.3 Front-End.....	17
4.4 Workflow	18
5 Detalii de implementare	19
5.1 Structura bazei de date	19
5.1.1 User	19
5.1.2 Series, Group & Course	20
5.1.3 Lesson, Grade, Presence & Document	21
5.1.4 Migrări.....	21
5.2 Structura aplicației	22
5.2.1 Entitățile de business	23
5.2.2 Data Access	24
5.2.3 Design Patterns	26
5.2.4 Logica de business.....	28
5.2.5 Interfața grafică	30
5.2.6 Integrare.....	34
6 Experiența utilizatorului	36
6.1 Pagina de Login	36

6.2	Utilizatorul autentificat	36
6.3	Centrul de administrare	37
6.4	Pagina de administrare a unei entități.....	38
6.5	Pagina de Dashboard	39
6.6	Pagina unui laborator (asistent/profesor)	40
6.7	Pagina de editare a unei lecții de laborator	42
6.8	Pagina unui laborator (student)	42
6.9	Paginile de adăugare a asistenților	43
6.10	Pagina de schimbare a parolei	44
7	Concluzii și dezvoltări ulterioare	45
7.1	Dezvoltări ulterioare	45
8	Bibliografie.....	47

LISTA DE FIGURI

Figura 2.1 Diagrama Use Case pentru administrare	8
Figura 2.2 Diagrama Use Case pentru utilizatori ai aplicației	10
Figura 4.1 Arhitectura LabGradR	15
Figura 4.2 Pattern-ul MVC	18
Figura 4.3 Schema bloc a aplicației	18
Figura 5.1 Diagrama bazei de date	19
Figura 5.2 Tabelele Series, Group, Course, legate de User.....	20
Figura 5.3 Tabelele Lesson, Grade, Presence, Document și legăturile lor.....	21
Figura 5.4 Migrarea Inițială – Metoda Up.....	22
Figura 5.5 Clasa Group	23
Figura 5.6 Tabela Group.....	23
Figura 5.7 Clasa context a aplicației LabGradR.....	24
Figura 5.8 Aplicarea configurărilor în clasa context	25
Figura 5.9 Set de reguli pentru configurarea unei clase C#.....	25
Figura 5.10 Clasa BaseRepository	26
Figura 5.11 Clasa ce implementează paradigma Unit Of Work.....	28
Figura 5.12 Arhitectura logicii de business	29
Figura 5.13 Serviciul pentru entitatea Group	29
Figura 5.14 Modelul unei lecții de laborator pentru un student.....	31
Figura 5.15 Crearea configurației unei mapări	31
Figura 5.16 Validări adăugate direct pe model.....	31
Figura 5.17 Header de acțiune din controller cu attributele Authorize și HttpGet	32
Figura 5.18 Structura de foldere a view-urilor	33
Figura 5.19 Pagină de view	33
Figura 5.20 Funcție AJAX pentru a șterge un document din pagină.....	34
Figura 5.21 Crearea unui fișier de tipul Excel cu ajutorul EPPlus	35
Figura 6.1 Pagina de login	36
Figura 6.2 Opțiunile prezente la hover pe numele utilizatorului	37
Figura 6.3 Pagina de administrare	37
Figura 6.4 Exemplu de pagină de administrare	38
Figura 6.5 Modelul unui fișier Excel.....	38
Figura 6.6 Erori apărute la adăugarea fișierului	38
Figura 6.7 Mesaj de confirmare pentru o importare de date cu succes	39
Figura 6.8 Dashboard profesor	39
Figura 6.9 Dashboard-ul unui student care este și asistent	40
Figura 6.10 Pagina goală unui laborator	40
Figura 6.11 Secțiunea de adăugare a unui lecții	41
Figura 6.12 Ștergerea unei lecții	41
Figura 6.13 Pagina de editare/vizualizare a unei lecții de laborator	41
Figura 6.14 Notarea studenților și adăugarea de documente pentru o lecție	42

Figura 6.15 Pagina unui laborator cu lecții din perspectiva unui student	43
Figura 6.16 Pagina cu materiile profesorului autentificat	43
Figura 6.17 Pagina de selectare asistenți pentru grupe	43
Figura 6.18 Pagina de schimbare parolă.....	44

SINOPSIS

Conceptul de e-learning reprezintă procesul de învățare prin intermediul Internetului și este unul dintre cele mai importante aspecte la nivel global ale momentului. Această lucrare de licență prezintă un proiect destinat notării studenților în cadrul laboratoarelor sau seminarelor cu ajutorul unei aplicații web care ușurează acest proces și oferă o interfață intuitivă și prietenoasă.

Prin proiectul creat am avut în vedere realizarea unei platforme web prin care profesorii și asistenții de laborator să poată adăuga și edita lecții pentru laboratoarele sau seminarele din cadrul facultății. După notare, studenții vor putea să își urmărească parcursul la aceste materii prin vizualizarea rezultatelor și prezențelor.

Lucrarea conține, împreună cu arhitectura pe care am construit platforma web pentru notarea laboratoarelor, pașii pe care i-am urmat în implementarea acesteia. În final, am expus prin imagini și exemple aplicația, rezultatele obținute și am propus câteva metode ulterioare de îmbunătățire.

ABSTRACT

The concept of e-learning is the Internet based learning process, and it is one of the most important aspects of today's world. This thesis presents a project for grading students in laboratories or seminars with the help of a web application that facilitates this process and offers them an intuitive and user-friendly interface.

The created program aims to deliver a web platform through which professors and laboratory assistants can add and edit lessons for laboratories and seminars as they desire. The students who are graded will be able to access the subject's page and check their results and attendances.

The paper contains, beside the architecture on which we built the concept of web platform for grading laboratories, the steps we followed in its implementation. Finally, we presented through images and examples the finished result of the application and we listed some methods to improve the program in the future.

1 INTRODUCERE

În acest capitol voi prezenta motivele pentru care am ales schițarea și implementarea acestui proiect și imaginea de ansamblu asupra acestuia pentru a introduce cititorul în temă cu subiectul și pentru a-l face conștient de problemele pe care aplicația LabGradR le rezolvă.

1.1 Context

În zilele noastre conceptul de e-learning reprezintă procesul de învățare prin intermediul Internetului și este unul dintre cele mai importante aspecte la nivel global ale momentului. Acest sistem folosește tehnologia conectată la o rețea de internet pentru a proiecta, implementa, gestiona, susține și extinde învățarea și va continua să îmbunătățească eficiența studiului în mediul virtual.

Datorită flexibilității de care dispune, cererea pentru acest tip de învățare este într-o continuă creștere. Acest concept capătă o popularitate foarte mare în rândul studenților și academicienilor pentru că nu depinde de timp și locație, ceea ce rezolvă multe dintre problemele de flexibilitate care apar atunci când se dorește învățarea de concepte noi.

De asemenea, pandemia globală de Covid-19 a cauzat o întrerupere a educației la nivelul întregii planete, rezultând în schimbarea modului de predare a multor școli și universități din întreaga lume din scenariul fizic în cel online, ceea ce a propulsat sistemele de e-learning într-o creștere la nivel mondial. Nu numai că studenții și elevii au trecut într-un alt stil de învățare, dar foarte mulți oameni, petrecând mai mult timp în case, s-au înscris la diverse cursuri online pe diferitele platforme de e-learning.

Cu toate că lumea dorește o întoarcere la normal, o mare parte dintre aceștia au descoperit o nouă oportunitate de învățare, mai rapidă și ușor accesibilă, care cu siguranță va mai fi utilizată. Facultățile și școlile implementează, chiar și după revenire, un program pentru un sistem hibrid, care va duce către o digitalizare masivă și o infrastructură solidă pentru viitorul învățării electronice.

1.2 Problema

În domeniul universitar românesc, în special cel al științelor exacte, unde seminarele se pot transforma în laboratoare practice, studenții primesc note pentru activitatea depusă în timpul laboratoarelor. Asistenții de laborator sau profesorii au nevoie de un bun sistem pentru notare și păstrare a evidenței prezențelor studenților. Pe lângă aceste necesități, unele laboratoare, sau chiar seminare, utilizează materiale drept suport pentru lecția curentă, iar acestea sunt, în majoritatea cazurilor, într-un format electronic și au nevoie de o platformă pe care să fie livrate către studenți.

O altă problemă importantă în cazul laboratoarelor este aceea a studenților care doresc să își verifice notele primite în cadrul lecțiilor de laborator. Pentru a se implementa acest lucru este nevoie, pe lângă o actualizare constantă a notelor și prezențelor după fiecare laborator de către o persoană autorizată, de un loc specific în care asistenții sau profesorii să pună notele și studenții să aibă posibilitatea să le vadă.

1.3 Obiective

Soluția propusă în această lucrare pentru atingerea cerințelor prezentate în subcapitolul anterior este o aplicație web numită LabGradR. Principalul scop al acesteia este să ușureze procesul de notare al studenților în cadrul laboratoarelor sau seminarelor. Acest program reprezintă o platformă pe care studenții se pot autentifica și apoi să-și verifice notele și prezențele la laboratoarele sau seminarele la care sunt înscriși.

Funcționalitățile aplicației sunt împărțite pe patru roluri: student, asistent, profesor și administrator. Studenții vor putea să își vizualizeze notele și prezențele la laboratoarele la care sunt înrolați, așa cum am prezentat în paragraful anterior. Profesorii și asistenții de laborator au ca principale îndatoriri adăugarea și întreținerea lecțiilor de laborator. În cadrul acestor lecții, profesorul sau asistentul vor putea nota și pune prezențe studenților. Profesorul are posibilitatea să asocieze câte un asistent pentru fiecare grupă. Administratorul este cel care va adăuga în baza de date informațiile despre toate persoanele care vor folosi platforma. Fără configurările acestui ultim rol portalul web nu poate fi accesat și folosit.

Pentru o experiență cât mai plăcută cu utilizatorii, aplicația oferă o interfață cât mai intuitivă și prietenoasă. De asemenea, ea are implementate sisteme pentru a facilita modurile prin care datele sunt adăugate și editate de către administrator.

1.4 Structura lucrării

Proiectul prezentat dorește să livreze o platformă complet funcțională de e-learning pentru urmărirea și notarea activităților studenților în cadrul laboratoarelor și seminarelor. Pentru a duce la bun sfârșit această idee, este nevoie de o arhitectură bine structurată, peste care să se construiască o implementare solidă.

Lucrarea aceasta va conține prezentarea detaliată ale arhitecturii, capabilităților platformei, implementării și rezultatelor finale obținute. Structura documentului este împărțită după cum urmează:

- Capitolul 2: Analiza și specificarea cerințelor – o mică descriere a funcționalităților propuse;
- Capitolul 3: Abordări existente – o scurtă prezentare a platformelor deja existente care rezolvă problema curentă;
- Capitolul 4: Soluția propusă – o descriere la obiect a arhitecturii programului, a tehnologiilor folosite și a fluxului de lucru din aplicație;
- Capitolul 5: Detalii de implementare – intrare amănunțită în modul de implementare al nivelelor aplicației și o prezentare detaliată a bazei de date;
- Capitolul 6: Experiența utilizatorului – o prezentare vizuală a funcționalităților aplicației prin imagini extrase din varianta finalizată a programului;
- Capitolul 7: Concluzii și dezvoltări ulterioare – o vedere de ansamblu a rezultatelor finale obținute și expunerea unor dezvoltări viitoare ale aplicației.

2 ANALIZA ȘI SPECIFICAREA CERINȚELOR

Proiectul are ca scop principal implementarea unei platforme web de e-learning pentru facultate prin intermediul căreia profesorii și asistenții de curs vor putea crea lecții pentru fiecare laborator, vor putea oferi note și seta prezențele studenților și vor adăuga documente auxiliare pentru fiecare lecție, iar studenții își vor putea vedea rezultatele de la aceste laboratoare. Pe lângă cele 3 tipuri de utilizator enumerate mai sus există și rolul de administrator, care are îndatorirea de a adăuga în sistem utilizatorii, materiile, grupele, laboratoarele, de a împărți studenții pe grupe și de a asocia profesorii cu materiile predate. Toate aceste date sunt păstrate într-o bază de date stocată în SQL Server [1].

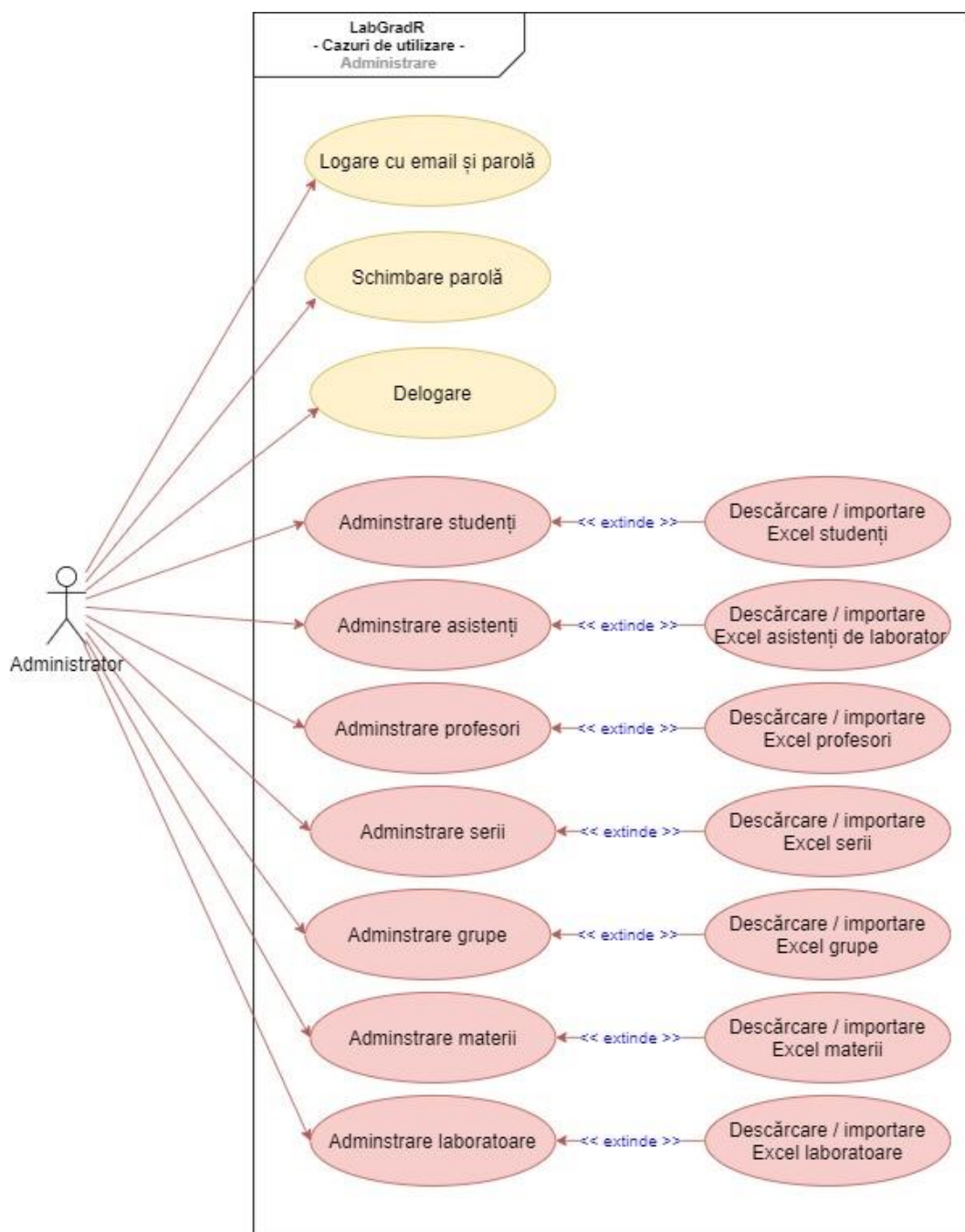


Figura 2.1 Diagrama Use Case pentru administrare

În Figura 2.1 putem observa diagrama Use Case pentru administrarea aplicației LabGradR, unde putem vizualiza toate funcționalitățile de care dispune o persoană cu rolul de administrator. În continuare voi prezenta cerințele funcționale ale aplicației prin prisma unui administrator.

Administrarea utilizatorilor din sistem

Această platformă de e-learning nu permite înregistrarea utilizatorilor, iar singura metodă de a-i adăuga în sistem este printr-o persoană autorizată. Aplicația oferă o metodă facilă și intuitivă de inserare a datelor, adică prin documente de tip Excel [2]. Administratorul trebuie să descarce documentul Excel pus la dispoziție de aplicație, să îl completeze cu numele, prenumele, adresa de email al utilizatorului și să îl încarce înapoi pe pagină. Dacă există date inserate greșit sau câmpuri necompletate vor apărea erori pe ecran care te vor îndruma către liniile care conțin greșeli din fișierul încărcat.

Pentru fiecare tip de utilizator există câte o pagină diferită de descărcare și încărcare a fișierului Excel. Dacă un utilizator face parte din două categorii de utilizator (ex: student și asistent de curs), acesta va fi trecut în ambele locuri, iar în sistem el va apărea cu două roluri.

Administrarea materiilor

La fel ca în cazul utilizatorilor, această inserare de date se va face cu fișier de tip Excel pentru a facilita execuția acestei sarcini de către administrator. Aici este nevoie doar de numele materiei și prescurtarea ei.

Administrarea seriilor

Seriile se vor adăuga într-un mod similar cu celelalte tipuri de date, doar că fișierul Excel va conține o singură coloană pentru nume.

Administrarea grupelor

Grupele se vor administra în același mod, de menționat ar fi că fișierul Excel va conține două coloane, una pentru serie și alta pentru numele grupei. Seriile inserate trebuie să fi fost deja inserate prin procedura anterioară pentru a nu primi eroare că seria adăugată nu există în sistem. De asemenea, numele grupei trebuie să conțină și numele seriei la care am asociat grupa respectivă (ex: 311CA este o grupă din cadrul seriei CA).

Administrarea laboratoarelor

Această funcție asociază un profesor cu o materie și o serie. Pentru adăugarea de laboratoare se vor respecta aceleași instrucțiuni ca la celelalte metode de inserare de date, dar în acest Excel vor fi trei coloane, reprezentând emailul profesorului, numele materiei și seria. Datele adăugate aici trebuie să fie deja inserate în sistem cu metodele precedente pentru a nu primi erori de inexistență.

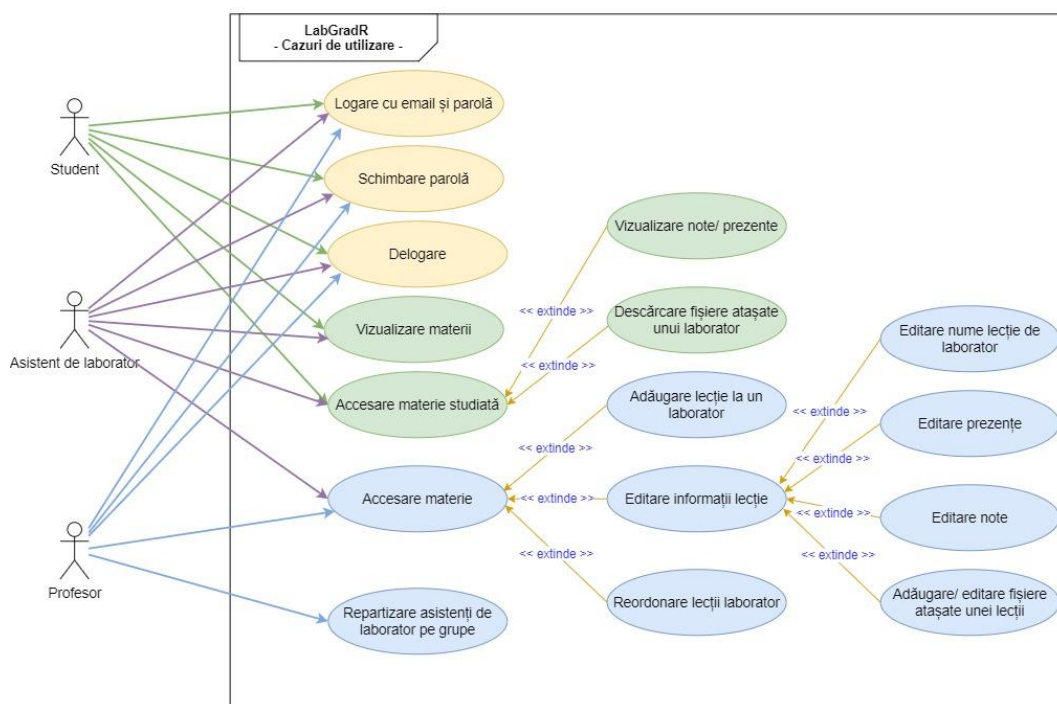


Figura 2.2 Diagrama Use Case pentru utilizatori ai aplicației

Diagrama din Figura 2.2 vizează toate capacitățile utilizatorilor principali, cei care vor folosi aplicația în majoritatea timpului, și anume profesori, asistenți de laborator și studenți. După cum ne prezintă imaginea, există acțiuni ce pot fi executate numai de anumite roluri și altele care sunt disponibile tuturor (logarea sau schimbarea de parolă), dar cel mai complex rol fiind cel de profesor pentru că deține mai mult control asupra celorlalți. Deși rolul de student are mai puține funcționalități, fără existența acestuia, celelalte tipuri de utilizator, dar și aplicația în sine își pierde tot scopul.

În următoarele paragrafe voi descrie succint cerințele funcționale din perspectiva utilizatorilor sus-menționați.

Autentificare

La deschiderea aplicației va apărea pagina de logare, unde se vor putea autentifica doar utilizatorii înregistrați în sistem de un administrator. La prima accesare a contului, utilizatorului i se va prezenta pagina de schimbare a parolei pentru că parolele inițiale sunt generate în funcție de nume și prenume.

Dashboard

Pagina de Dashboard este locul unde se vor putea vedea materiile cu care are legătură utilizatorul, în funcție de rol. În cazul unui student vor apărea materiile pe care acesta le studiază. Dacă este un profesor se vor vedea materiile, împreună cu grupele la care acesta predă. Unui asistent de curs îi vor fi prezentate materiile la grupele pe care acesta le notează. Există și cazuri când un utilizator poate avea două roluri și anume un student care poate fi

asistent la alte materii decât cele pe care le studiază, deci, pe lângă acestea, îi vor apărea și materiile la grupele la care are laboratoare.

Vizualizarea unei materii

Când un student accesează pagina unei materii el va vedea un tabel cu lecțiile de la laboratorul respectiv, notele și prezențele la acestea.

Pentru un profesor sau un asistent de curs această pagină este puțin diferită pentru că ei pot apăsa pe butonul de “creare lecție nouă”, pot reordona lista de lecții de la laboratorul curent și le pot edita.

Adăugarea unei lecții de laborator

Asistenții și profesorii au posibilitatea de a adăuga câte lecții doresc la un anumit laborator, acest lucru executându-se foarte ușor prin accesarea funcției de adăugarea care va avea nevoie doar de un nume pentru viitoarea lecție.

Editarea unei lecții din cadrul unui laborator

Această pagina de editare, care poate fi accesată numai de un profesor sau un asistent de laborator, constă din mai multe părți: o listă cu studenții de la grupa al cărui laborator este accesat, o rubrică unde se pot încărca documente pentru lecția curentă și o altă listă cu documentele deja încărcate, dacă există.

Fiecărui student din listă îi este asociat un buton pentru bifarea prezenței și un câmp pentru notă.

Documentele încărcate au un câmp pentru a fi redenumite, după caz. Dacă nu se introduce nimic în acesta, documentul își va păstra numele cu care a fost încărcat.

Asociere asistenți de curs cu grupele

Această pagină constă inițial în alegerea materiei pentru care se dorește stabilirea asistenților, iar apoi se alege câte o persoană pentru fiecare grupă.

Profesorul va alege asistenții de curs pentru grupele din seria la care el își predă materia. Această pagină este vizualizată numai de utilizatorul cu rol de profesor și el are responsabilitatea de a asocia persoanele autorizate cu grupele pe care le vor nota.

Funcția de logout

Utilizatorul are posibilitatea de a se deconecta de la aplicație. Această funcționalitate va trimite persoana la pagina de autentificare și este obligată să-și reintroducă datele pentru a intra înapoi în aplicație.

Schimbare parolă

Orice utilizator are acces la această pagină prin care își poate schimba oricând parola.

3 ABORDĂRI EXISTENTE

Universitățile și sistemele educaționale din România folosesc diverse platforme de e-learning pentru susținerea cursurilor, predarea temelor și notarea studenților. Un exemplu de sistem de management al cursurilor, pe care îl folosește și Universitatea Politehnica București, este platforma **Moodle** [3].

Moodle este un software internațional open-source, gratuit, bine-cunoscut pentru o platformă de învățare în mediul online, folosită de nenumărate școli, instituții de învățământ superior și chiar de locuri de muncă la nivel global. Această aplicație permite profesorilor din întreaga lume să creeze site-uri particulare, pline de funcții pentru proiectarea de cursuri și activități, optimizate pentru studiul în grup. Moodle.org este site-ul comunității unde Moodle este creat și unde se pot descoperi mai multe informații despre platformă.

Platforma Moodle este într-adevăr un program cu foarte multe funcționalități, care se pliază perfect pe strategia de învățământ din sistemul universitar Românesc din cadrul Universității Politehnica București. Astfel, la intrarea pe această aplicație, după ce trecem de pasul de autentificare, ni se prezintă pagina de Dashboard, unde se văd materiile în curs, la care studentul este înscris.

Pagina unei materii înfățișează, în prim plan, o zonă în care se găsesc documente, informații, link-uri, un forum pentru știri sau poate chiar o locație pentru încărcarea unei teme sau a unui document. Mai jos, putem observa o împărțire a paginii pe săptămâni, în funcție de perioada de întindere a cursului, iar fiecare secțiune poate conține documente, teme, teste, sau mici fragmente cu informații.

Aplicația LabGradR vine în ajutorul universităților pentru o notare mai ușoară și fluidă a laboratoarelor sau a seminarelor. Deși platforma Moodle are un sistem de notare foarte bine pus la punct, ea nu surprinde aspectele referitoare la verificarea prezențelor și notarea studenților din cadrul laboratoarelor sau seminarelor. Acest lucru se poate realiza pe Moodle în mai multe moduri. Prima metodă poate fi încărcarea pe pagina materiei a unui document de tip Excel, în care apar toți studenții și în care asistenții de laborator și profesorii trec notele. Unul dintre aspectele negative la acest lucru poate fi că un student va avea notele publice pentru ceilalți colegi. O altă modalitate de a oferi note pentru activitatea din cadrul laboratoarelor este prin inserarea săptămânală a unei secțiuni de tip assignment, în care fiecare student va avea trecută nota de către un asistent sau profesor. Această metodă păstrează confidențialitatea notelor între studenți, dar necesită mai mult timp. Platforma LabGradR este creată special pentru a păstra confidențialitatea notelor, dar și pentru a oferi persoanelor responsabile cu notarea o interfață care să faciliteze acest proces.

O altă platformă pentru e-learning, folosită în universități pentru prezentarea laboratoarelor online, este **OpenCourseWare (OCW)** [4]. OpenCourseWare este un site online pe care sunt publicate gratuit cursuri și laboratoare cu documente, linkuri, task-uri și foarte multe informații ajutătoare pentru studiul anumitor lecții predate în cadrul laboratoarelor sau

seminarelor. Fiecare universitate are propria structură pentru OCW, dar adesea, paginile sunt structurate pe materii, iar apoi pe capitole.

La fel ca în cazul Moodle, această platformă este perfectă pentru învățare și predare, dar nu și pentru notarea studenților. Fiind un site liber și gratuit pentru toată lumea nu se impune autentificarea. Acest lucru va genera câteva inconveniente, cum ar fi lipsa păstrării evidenței materiilor studiate, deci fiecare persoană care accesează site-ul va trebui să selecteze materia, capitolul și secțiunea pe care le caută. Un alt dezavantaj generat de lipsa autentificării este absența unei zonă de evaluare a studenților

Claroline [5] este un alt exemplu pentru un sistem de e-learning prin care universitățile sau școlile își pot crea propriul spațiu de predare, învățare și notare pentru studenții sau elevii din aceste instituții. Această platformă este folosită în mai mult de 80 de țări și este disponibilă în peste 30 de limbi. De asemenea, este compatibilă cu toate sistemele de operare și are la bază tehnologii gratuite precum PHP și MySQL.

Platforma Claroline este organizată astfel încât să acopere necesitățile unui program destinat cursurilor sau activităților pedagogice. Ea conține, la fel ca Moodle, o mulțime de funcționalități care le permit profesorilor să scrie informații despre cursuri, să încarce documente, să administreze forumurile publice sau private, să adauge secțiuni pentru încărcat teme sau lucrări și să vizualizeze statistici referitoare la activitățile studenților.

Comparând cu aplicația LabGradR, Claroline acoperă foarte multe funcționalități utile, dar punctând la cazul laboratoarelor sau seminarelor, trebuie create separat, pe pagina unei materii, secțiuni pentru laboratoare ca studenții să poată fi notați la fiecare lecție de laborator. Desigur că se poate adopta metoda specificată mai sus printr-un fișier de tip Excel în care toți studenții să apară, dar ne dorim ca notele să fie adresate fiecărui student pe contul propriu.

Pentru utilizarea acestei aplicații administratorii Claroline oferă două pachete abonament, unul Standard și unul Enterprise, cu prețuri variabile în funcție de numărul de persoane care o vor folosi.

Blackboard Learn [6] este o platformă de e-learning dezvoltată de compania Blackboard Inc. și care oferă, pe lângă capabilitățile de a crea și edita cursuri, opțiuni de a schimba arhitectura, designul și permite integrarea cu sisteme care stochează informații despre studenți și protocoale pentru autentificare.

La fel ca în cazurile anterior prezentate, această aplicație oferă profesorilor și studenților oportunitatea de a avea un sistem complet pentru activitățile destinate învățării online. Blackboard Learn prezintă foarte multe avantaje și beneficii, unul dintre cele mai importante fiind posibilitatea de a întreține apeluri video pe platformă. Alte aspecte foarte importante sunt monitorizarea prezențelor, integrarea statisticilor și posibilitatea de autentificare și folosire pe mai multe dispozitive cum ar fi telefoanele sau tabletele.

Voi continua acest capitol prin a prezenta și descrie anumite criterii care sunt importante în problema actuală și vom încerca să comparăm, pe baza acestora, programele sus-menționate:

- Gratuit – acest criteriu prezintă dacă platforma nu prezintă costuri de utilizare;
- Confidențialitate – va preciza dacă aplicația păstrează confidențialitatea datelor și notelor unui student;
- Prezență – acest lucru ne va indica dacă programul are o funcționalitate de monitorizare a prezențelor studenților;
- Notare – criteriu care va spune dacă aplicația are implementat un sistem de notare;
- Încărcare assignment – va specifica dacă aplicația prezintă funcționalitatea de a încărca o temă sau un document de către un student;
- Încărcare document – va indica dacă programul permite utilizatorilor privilegiați să încarce documente pentru studenți;
- Exportare note – criteriu care va spune dacă un program prezintă funcționalitatea de exportare a notelor studenților;
- Importare note – va spune dacă un program prezintă funcționalitatea de importare a notelor studenților;
- Suport alte dispozitive – va indica dacă platforma are versiuni pentru alte dispozitive;
- Specific laborator – va indica dacă aplicația prezintă măcar o funcționalitate destinată prezentării/notării lecțiilor de laborator;

Criteriu	LabGradR	Moodle	OCW	Claroline	Blackboard
Gratuit	✓	✓	✓		
Confidențialitate	✓	✓		✓	✓
Prezență	✓				✓
Notare	✓	✓		✓	✓
Încărcare assignment		✓		✓	✓
Încărcare document	✓	✓	✓	✓	✓
Exportare note		✓			
Importare note		✓			
Suport alte dispozitive		✓		✓	✓
Specific laborator	✓	✓		✓	✓

Tabel 3.1 Tabelul de comparații dintre LabGradR și celelalte tehnologii

Deși unele aplicații prezentate au mai multe funcționalități și sunt capabile să înlocuiască cursurile în format fizic cu cele online, platforma LabGradR care este destinată doar domeniului laboratoarelor și seminarelor are o aplicabilitate foarte mare pe acest sector și implementează un număr rezonabil de funcționalități pentru a facilita procesul de notare.

4 SOLUȚIA PROPUȘĂ

Acest capitol prezintă arhitectura aleasă pentru aplicația LabGradR, plusurile pe care le aduce implementarea acestei arhitecturi pe platforma noastră și tehnologiile folosite în construcția și dezvoltarea programului.

Făcând o analiză a specificațiilor pe care vrem să le avem pe platforma de e-learning și notare LabGradR, am schițat o arhitectură bine delimitată, care vizează principiile de bază în dezvoltarea de aplicații web, cât și “Separation of Concerns Principle” [7]. Acest principiu presupune secționarea programului în mai multe părți astfel încât să se poată urmări foarte clar traseul pe care îl urmează informația, modul cum este configurată, procesată, din punctul în care este preluată din baza de date și până când este afișată pe pagină, pentru a fi vizualizată de utilizator.

În diagrama din Figura 4.1 am prezentat arhitectura pe care am folosit-o în dezvoltarea aplicației LabGradR.

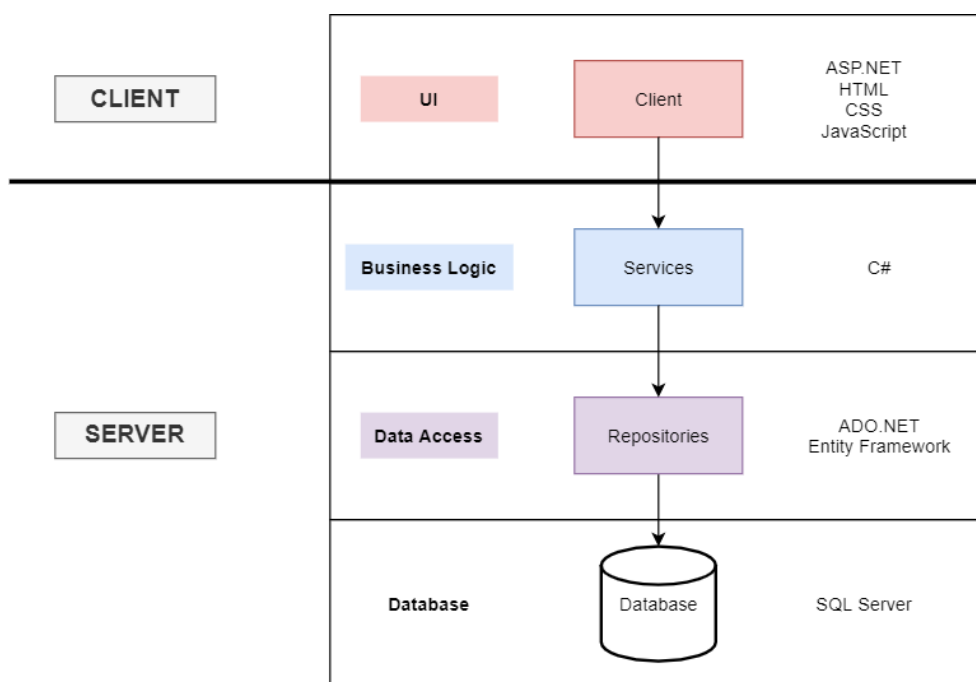


Figura 4.1 Arhitectura LabGradR

Voi prezenta, în continuare, fiecare layer al aplicației, în parte, specificând tehnologiile folosite și modurile prin care acestea comunică între ele.

4.1 Baza de date

Platforma web LabGradR are nevoie de o modalitate de stocare a unui număr foarte mare de date, cum ar fi informațiile despre studenți și profesori, despre laboratoare, materii, note etc. Dacă acestea nu sunt păstrate într-o bază de date, aplicația și funcționalitățile sale nu își ating scopul. În prezent există multe metode de păstrare a datelor, pornind de la o bază de date locală, până la una de tip Cloud. În acest moment baza de date a aplicației LabGradR este una locală, dar când aceasta va fi pregătită de a fi publicată și folosită de studenți și profesori de

la universitate, ea va putea fi importată într-un portal Azure [8] și va putea fi accesată de oricine.

Având în vedere arhitectura aplicației am ales să utilizez o bază de date de tip relațional precum Microsoft SQL Server. SQL Server folosește Transact-SQL (T-SQL) [9] și permite păstrarea de date de tip int (integer), bit, datetime, varbinary, varchar, care sunt perfecte pentru a acoperi funcționalitățile propuse pentru platforma LabGradR. Atunci când aplicația va face un request către SQL Server, el va fi integrat și tratat ca pe o tranzacție și, împreună cu design pattern-ul Unit Of Work [10] vor asigura că operațiile de adăugare, ștergere sau editare în bază sunt realizate corect.

4.2 Back-End

Entitățile de business

De obicei, în aplicațiile web, tabelele din baza de date ar trebui să se transpună în aplicația propriu zisă sub forma unor clase, care să respecte numele și tipul proprietăților. Acestea sunt prezente și în aplicația LabGradR și pot fi numite Entități de Business. Aceste entități vor fi direct legat de Layer-ul Data Access și ele vor transforma prin mapare informațiile din baza de date în obiecte din aplicație. După aceste mapări aplicația va avea transpuse toate tabelele din baza de date în clase cu aceleași nume și proprietăți ca ele. Aceste transformări se realizează cu ajutorul unui framework specific .NET [11], și anume Entity Framework [12], care este descris în cele ce urmează.

Layer-ul data access

Avansarea în ierarhia prezentată în Figura 4.1 presupune și o securizare a entităților, astfel că se dorește ca un utilizator să nu aibă acces la date pe care nu le cere și de care nu este nevoie. Trebuie să ne asigurăm că entitățile din baza de date nu vor ajunge niciodată direct într-o pagină de front-end și nu vor putea fi în totalitate la îndemâna utilizatorului, ci doar parțial, în funcție de situație.

Layerul care poate comunica direct cu baza de date este cel de Data Access. Scopul principal al acestui layer este să le ofere celorlalte layere superioare un mod abstractizat de accesare a informațiilor din baza de date. Scopul acestui strat a fost realizat printr-o tehnică de tipul ORN (Object Relational Mapping) cu Entity Framework. Acest framework ajută la maparea tabelelor în clase și la realizarea de query-uri direct în aplicație, acesta având grijă să le transpună în query-uri similare din SQL.

Entity Framework are integrate trei moduri cu care să se creeze legătura dintre aplicație și baza de date:

1. Database First [13] – prin această modalitate se înțelege ca prima dată să se creeze baza de date cu legăturile dintre tabele, iar apoi framework-ul le va proiecta automat în clase din aplicație cu proprietăți și cu legături. Aceste legături dintre tabele sunt numite proprietăți de navigare.

2. Code First [14] – în acest caz se va acționa în sens invers abordării anterioare. Se vor crea inițial clasele cu proprietățile de care avem nevoie și cu proprietățile de navigare care să înlocuiască legăturile de tipul primary key – foreign key. Entity Framework se va ocupa de crearea bazei de date cu tabelele aferente claselor, dacă acestea nu există deja. Dacă într-o clasă se găsește o proprietate numită Id sau care să conțină grupul Id (ex. RoleId), Entity Framework își dă singur seama ca această proprietate are rol de cheie primară în tabelă.
3. Model First [15] – această modalitate presupune generarea unui model de date de tip entitate (Entity Data Model) printr-un Wiziard pus la dispoziție de Visual Studio și apoi se va genera schema bazei de date pornind de la acest model.

În aplicația LabGradR am folosit prima metodă, adică Database First deoarece consider că se pot surprinde mult mai clar care sunt tabelele importante din bază și cum se leagă acestea în aplicație. De asemenea, SQL Server prezintă un utilitar care înfățișează într-un mod lizibil diagrama bazei de date.

Layer-ul logica de business

Layer-ul care conține logica de business este unul dintre cele mai importante din întreaga arhitectură. Acest strat abstractizează layer-ul de Data Access astfel încât, dacă acesta din urmă are parte de niște modificări majore, layer-ul cu logica de business nu ar suferi nicio schimbare și ar rula într-un mod obișnuit, fără erori, ca și cum nimic nu s-ar fi întâmplat.

Acest layer cuprinde metode și funcționalități care folosesc și prelucrează entitățile din bază ca mai apoi să trimită rezultatele obținute către Front-End-ul programului și vizualizate de utilizator.

4.3 Front-End

Pentru interfața cu utilizatorul am folosit design pattern-ul Model-View-Controller(MVC) [16] care separă într-un mod clar cele trei entități și le face să lucreze împreună pentru a oferi un avantaj important în dezvoltarea de programe web. Cu ajutorul modelului, un Controller primește date, le manipulează pentru a realiza în mod corect funcționalitățile aplicației și apoi le trimite mai departe. Componenta Controller se poate folosi iar de Model pentru a transmite datele într-un View care va conține ambalajul aplicației, partea pe care o va vedea utilizatorul, în final, pe ecran.

Se poate observa modul prin care componentele MVC-ului comunică și imaginea de ansamblu a funcționalităților acestui design pattern în Figura 4.2.

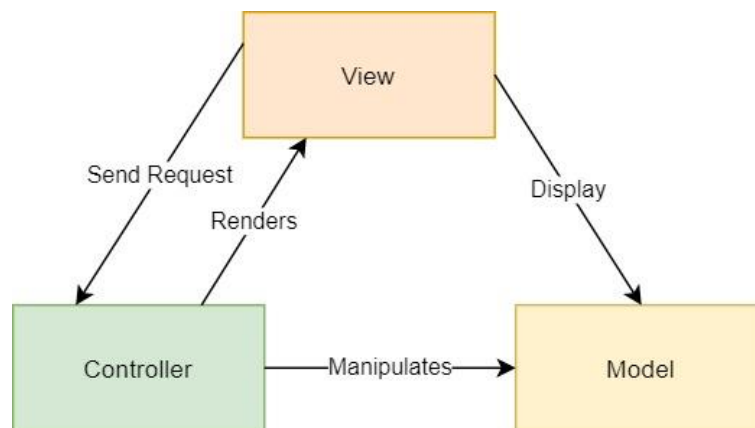


Figura 4.2 Pattern-ul MVC

4.4 Workflow

După scurta prezentare a secțiunilor în care este împărțită aplicația, voi încerca să descriu, pe scurt, traseul pe care îl urmează informația, din baza de date până când este vizualizată de utilizator pe ecran. Acesta se poate observa și în schema bloc surprinsă în Figura 4.3.

Datele existente în bază sunt extrase și păstrate cu grijă în program cu ajutorul mai multor containere care le vor stoca în funcție de tipul entității. Avem nevoie de un strat care să separe locul unde sunt ținute informațiile de partea din aplicație care se ocupă cu trimiterea lor către utilizator pentru a nu-i permite acestuia accesarea de date private. Cu acest scop ne-am creat o secțiune de servicii care vor folosi seturile de date la filtrări și modelări, iar apoi le vor trimite, la cerere, către Controller.

Exact cum am precizat în paragraful anterior, un Controller va apela serviciile doar atunci când are nevoie de datele din bază pe care metodele din servicii le pot procura. La nivelul Controller-ului se mapează entitățile primite în modele pentru a filtra sau crea noi obiecte care să conțină doar elemente pe care dorim să le afișăm pe ecranul utilizatorului. Modelul este apoi trimis către componenta de View, unde datele sunt extrase și aranjate în pagină într-un mod cât mai lizibil și prietenos pentru a oferi o experiență plăcută.

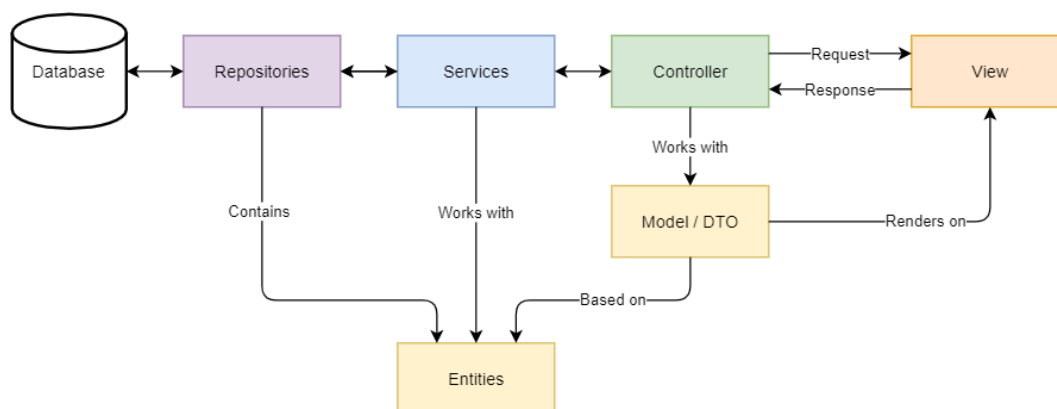


Figura 4.3 Schema bloc a aplicației

5 DETALII DE IMPLEMENTARE

În acest capitol voi prezenta în profunzime implementarea proiectului LabGradR. Vom începe de la cel mai jos nivel, adică baza de date, și vom urca până la cel mai înalt nivel, interfața cu utilizatorul.

5.1 Structura bazei de date

Baza de date este considerată epicentrul aplicației noastre, deoarece, fără o bază de date cu o structură bună și proiectată corect, programul poate avea probleme cu anumite funcționalități și vor apărea bug-uri. Baza de date a fost gândită în primul stadiu al implementării proiectului LabGradR și a fost proiectată în așa fel încât să suporte funcționalitățile propuse și să nu sufere foarte multe modificări de-a lungul procesului de programare al aplicației.

Figura 5.1 prezintă diagrama completă a bazei de date a platformei LabGradR.

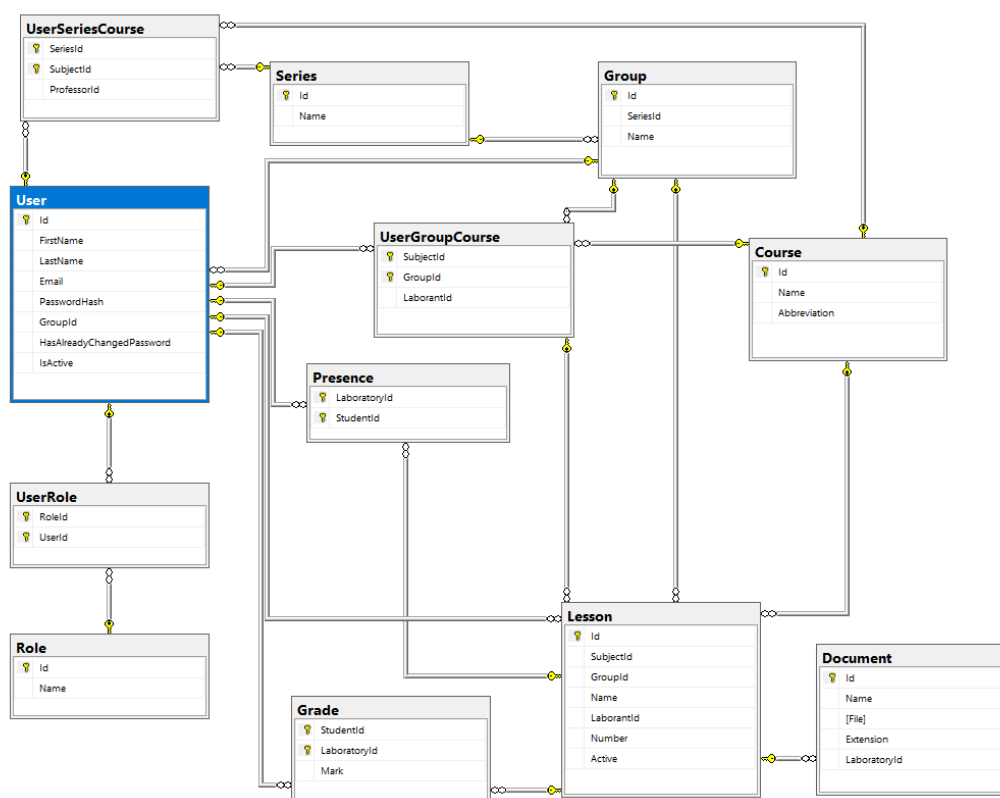


Figura 5.1 Diagrama bazei de date

Acest capitol cuprinde o prezentare a bazei de date, a tabelor și a legăturilor dintre acestea. Voi face o mică descriere a tabelor importante și a scopului lor în programul nostru.

5.1.1 User

Una dintre cele mai importante tabele din întreaga aplicație este tabela **User**, ea având foarte multe legături cu celelalte tabele, așa cum se poate observa și în Figura 5.1. În această tabelă

se stochează datele despre utilizatorii aplicației LabGradR. Coloana ID are proprietatea de Identitate, ceea ce înseamnă că fiecărei persoane i se generează automat un ID. Această coloană realizează o cheie primară împreună cu Email-ul utilizatorului. Datele de autentificare pe aplicație, Email-ul și Parola, se găsesc aici, împreună cu alte informații ale utilizatorului, cum ar fi Numele, Prenumele și date folosite la Logica de Business, IsActive, HasAlreadyChangedPassword.

Numeroasele legături cu această tabelă sunt realizate prin doua feluri: legătură directă, care se numește cheie străină (foreign key - FK) sau printr-o tabelă de legătură, atunci când relația dintre tabele este una de tipul Many-To-Many. Această tabelă de legătură are o cheie primară (primary key - PK) compusă din cheile primare ale tabelelor pe care le leagă.

5.1.2 Series, Group & Course

Scopul aplicației LabGradR este de a oferi universităților o platformă de e-learning pentru a urmări și nota activitatea studenților în cadrul laboratoarelor sau seminarelor. Astfel, în baza de date trebuie să avem o împărțire a studenților pe serii și grupe. Acestea sunt prezente în baza de date prin tabelele Series și Group, care sunt legate în mod indirect, dar și în mod direct la tabela utilizatorilor, User.

Tabela Course reprezintă o materie din cadrul facultății, care are asociate o serie și un profesor. Aceste legături sunt surprinse printr-o tabelă care leagă aceste trei entități, numită UserSeriesCourse. Cheile străine ale acestei tabele se leagă la toate cheile primare ale tabelelor pentru cursuri, serii și profesori.

Pe lângă logica pentru profesori există și o tabelă care realizează legăturile relației unui asistent de laborator cu celelalte entități. Astfel, am creat o altă tabelă care face legătura între un utilizator, care va fi asistentul asociat unui laborator, o grupă, pe care acesta o va nota în cadrul lecțiilor, și o materie. Tabela prezentată are denumirea de UserGroupCourse, după cum se poate observa și în Figura 5.2.

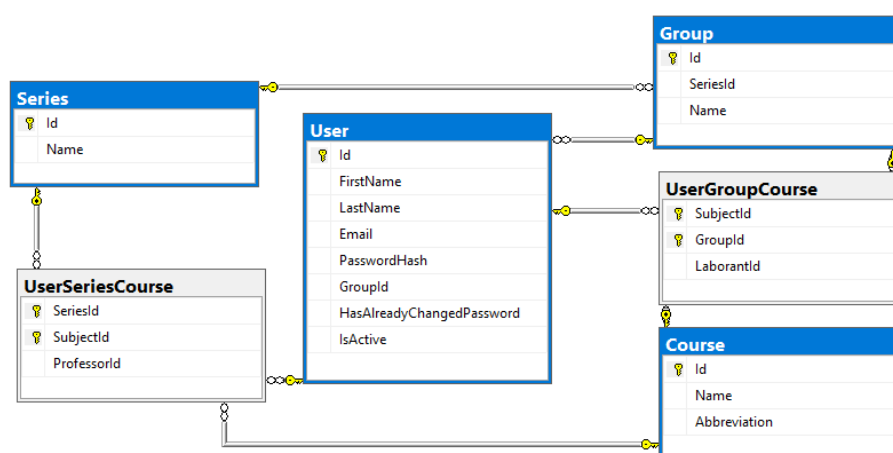


Figura 5.2 Tabelele Series, Group, Course, legate de User

5.1.3 Lesson, Grade, Presence & Document

Pentru a îndeplini scopul final al aplicației noastre avem nevoie de un sistem bine pus la punct pentru notare și monitorizare a activității studenților la seminare sau laboratoare, de aceea trebuie începută această funcționalitate de la cel mai jos nivel, adică baza de date. Tabela care reprezintă cel mai bine entitatea unei lecții de laborator este numită Lesson. Ea este legată de toate entitățile care stochează date despre laboratoare.

Grade este tabela în care se păstrează toate notele studenților de la laboratoare, ea având legături directe cu tabela User și Lesson prin chei străine.

Altă tabelă importantă și similară ca structură cu cea prezentată anterior este tabela Presence, care stochează prezențele studenților. Asemănarea tabelelor prezentate constă în legăturile pe care tabela curentă le are. La fel ca mai sus, există chei străine către tabelele pentru utilizatori și laboratoare.

O ultimă tabelă este Document, care, deși nu prezintă o foarte mare importanță pentru scopul principal al aplicației, ne ajută în implementarea unei funcționalități foarte practice pentru o platformă de e-learning și anume, adăugarea de documente auxiliare la orice lecție din cadrul unui laborator. Tabela Document are o singură legătură, cu entitatea Lesson, printr-o cheie străină și este folosită pentru păstrarea documentelor adăugate unei lecții.

Tabelele recent prezentate și legăturile dintre ele pot fi surprinse în diagrama din Figura 5.3.

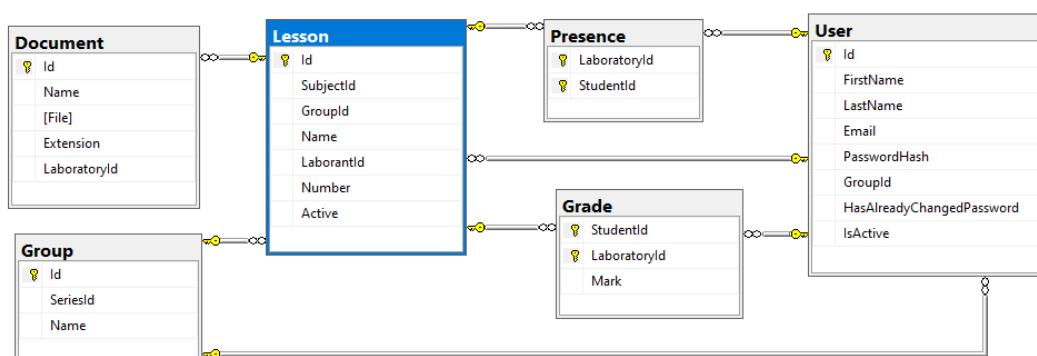


Figura 5.3 Tabelele Lesson, Grade, Presence, Document și legăturile lor

5.1.4 Migrări

Fiind vorba despre o aplicație web cu o complexitate medie care este la început, nu s-a putut realiza, din prima, designul perfect pentru baza de date, deci aceasta a suferit destule modificări de-a lungul perioadei de implementare. Ca fluxul aplicației să nu se întrerupă și să nu apară defecte de funcționare, operațiile și tabelele din baza de date trebuie să corespundă cu cele din aplicație.

Migrările sunt unelte puse la dispoziție de .Net Core și Entity Framework pentru a rezolva eventualele diferențe dintre baza de date și cod.

Migrările vor fi create și aplicate după ce am generat entitățile din bază cu ajutorul Entity Framework. Migrările sunt construite din două metode:

1. Up, metoda care conține toate modificările realizate de la ultima migrare. (Figura 5.4)
2. Down, care prezintă toate modificările de care avem nevoie pentru a ne întoarce la momentul precedent migrării. Această metodă este echivalentă cu demontarea unei migrări și revenirea la ce era înainte.

Vom face o nouă migrare când dorim să inserăm sau să modificăm o tabelă în baza de date, cum ar fi adăugarea, redenumirea, sau chiar ștergerea unei coloane. Migrările au adesea doar câteva modificări, excepție face migrarea inițială, care conține codul pentru crearea, în totalitate, a bazei de date.

Metoda migrărilor este folosită pentru a putea păstra un istoric al modificărilor și pentru eventuala revenire la versiuni anterioare ale bazei de date.

Se poate omite folosirea migrărilor prin modificări directe pe baza de date și apoi schimbări ale claselor din cod.

```
namespace LabGradR.DataAccess.Migrations
{
    1 reference | 0 changes | 0 authors, 0 changes
    public partial class InitialMigration : Migration
    {
        0 references | 0 changes | 0 authors, 0 changes
        protected override void Up(MigrationBuilder migrationBuilder)
        {
            migrationBuilder.CreateTable(
                name: "Roles",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false),
                    Name = table.Column<string>(maxLength: 30, nullable: false)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Roles", x => x.Id);
                });

            migrationBuilder.CreateTable(
                name: "Series",
                columns: table => new
                {
                    Id = table.Column<int>(nullable: false)
                        .Annotation("SqlServer:Identity", "1, 1"),
                    Name = table.Column<string>(maxLength: 10, nullable: true)
                },
                constraints: table =>
                {
                    table.PrimaryKey("PK_Series", x => x.Id);
                });
        }
    }
}
```

Figura 5.4 Migrarea Inițială – Metoda Up

5.2 Structura aplicației

Am prezentat cât mai detaliat baza de date a aplicației noastre, LabGradR, scoțând în prim plan modurile prin care baza de date ajută la îndeplinirea funcționalităților programului și care sunt legăturile dintre tabele, exemplificând și tipurile de legături folosite. Urmează o prezentare a felului în care tehnologiile se combină și ne ajută să implementăm de la cap la coadă arhitectura propusă aplicației.

5.2.1 Entitățile de business

O entitate de business este o clasă din aplicația propriu-zisă din C# care are drept corespondent o tabelă din baza de date. Proprietățile claselor și câmpurile din tabelă trebuie să corespundă unu la unu în ceea ce privește tipul lor și numele, dar acest aspect este vizat de Entity Framework.

Pentru a vedea cum se ocupă Entity Framework de aceste entități vom lua, ca exemplu, tabela Group și clasa ei corespondentă cu același nume din C# și le vom analiza proprietățile, pe rând, și le vom compara.

Comparând cele două imagini de mai jos (Figura 5.5 și Figura 5.6) se poate observa că Entity Framework a creat o clasă cu aceleași nume și tipuri de date ca ale tabelului din SQL. Coloanele Id și SeriesId au rămas amândouă int în C#, iar Name a trecut din nvarchar în string. Se pot vedea în clasa Group trei proprietăți pe care nu le vedem și în tabelă, iar asta pentru că acelea se numesc proprietăți de navigare și semnifică faptul că această clasă este legată de clasa Series, sau de clasa User prin proprietatea de tipul ICollection<User>.

Proprietatea de navigare „Series” ar putea fi tradusă ca „O grupă aparține unei serii”, iar proprietatea „Users” s-ar citi ca „O grupă are mai mulți utilizatori(users)”. Putem aplica același principiu pentru a traduce și proprietatea „Laboratories”.

```
public class Group : IEntity
{
    4 references | 1 author, 1 change
    public Group()
    {
        Users = new HashSet<User>();
        Laboratories = new HashSet<Laboratory>();
    }

    9 references | 1 author, 1 change
    public int Id { get; set; }
    10 references | 1 author, 1 change
    public int SeriesId { get; set; }
    19 references | 1 author, 1 change
    public string Name { get; set; }
    4 references | 1 author, 1 change
    public Series Series { get; set; }
    2 references | 1 author, 1 change
    public virtual ICollection<User> Users { get; set; }
    2 references | 1 author, 1 change
    public virtual ICollection<Laboratory> Laboratories { get; set; }
}
```

Figura 5.5 Clasa Group

Group		
	Column Name	Data Type
🔑	Id	int
	SeriesId	int
	Name	nvarchar(450)

Figura 5.6 Tabela Group

5.2.2 Data Access

În acest subcapitol voi prezenta modul de funcționare a layer-ului care se ocupă cu preluarea directă a datelor din bază, denumit mai sus „Layer-ul de Data Access”. Acesta va avea ca sarcină coordonarea request-urilor din cod care au ca obiectiv configurarea datelor, adică operațiile pe baza de date de tip CRUD (create, read, update, delete).

Acest strat al arhitecturii aplicației, având legături directe numai cu baza de date și cu layer-ul superior în care se găsește logica de business, este capabil numai de a manipula informații din bază, prin extragere și importare de date.

Pe lângă funcționalitățile de conversie în cod C# a tabelelor din SQL, Entity Framework ne oferă o modalitate de interogare a bazei de date cât mai abstractizată. Acest framework ne pune la dispoziție o clasă cu capabilități de care avem mare nevoie în aplicația noastră, și anume clasa context [17]. Ea va urmări în permanență modificările aduse bazei de date, astfel că va preveni accesarea unei resurse în același timp de către utilizatori diferiți sau de către același utilizator.

```
public class LabGradRContext : DbContext
{
    0 references | 1 author, 1 change
    public virtual DbSet<User> Users { get; set; }
    0 references | 1 author, 1 change
    public virtual DbSet<Role> Roles { get; set; }
    0 references | 1 author, 1 change
    public virtual DbSet<UserRole> UserRoles { get; set; }
    0 references | 1 author, 1 change
    public virtual DbSet<Document> Documents { get; set; }
    0 references | 1 author, 1 change
    public virtual DbSet<Grade> Grades { get; set; }
    0 references | 1 author, 1 change
    public virtual DbSet<Group> Groups { get; set; }
    0 references | 0 authors, 0 changes
    public virtual DbSet<Presence> Presences { get; set; }
    0 references | 1 author, 1 change
    public virtual DbSet<Series> Series { get; set; }
```

Figura 5.7 Clasa context a aplicației LabGradR

În Figura 5.7 se poate observa clasa LabGradRContext, care are generate, de către Entity Framework, proprietățile virtuale *DbSet<Nume_Entitate>* pentru fiecare entitate din bază, iar acestea sunt adăugate pentru a anunța contextul că programul va folosi obiecte de tipurile specificare (se pot observa în Figura 5.7 multiple entități, cum ar fi User, Role, UserRole, Document, Grade, Group etc.).

În dezvoltarea oricărui program trebuie să fim siguri că relațiile și comunicarea realizate cu baza de date sunt corecte și nu funcționează greșit. Pe lângă mapările inițiale din tabele în clase, Entity Framework mai creează, pentru fiecare entitate, și alte configurări, asemănătoare celor din SQL, în care se menționează exact cărui tabel îi corespunde o anumită clasă și care sunt limitările proprietăților.

```
protected override void OnModelCreating(ModelBuilder modelBuilder)
{
    base.OnModelCreating(modelBuilder);
    SeedData seedData = new SeedData();

    modelBuilder.ApplyConfiguration(new UserConfiguration());
    modelBuilder.ApplyConfiguration(new UserRoleConfiguration());
    modelBuilder.ApplyConfiguration(new RoleConfiguration());
    modelBuilder.ApplyConfiguration(new DocumentConfiguration());
    modelBuilder.ApplyConfiguration(new GradeConfiguration());
    modelBuilder.ApplyConfiguration(new GroupConfiguration());
    modelBuilder.ApplyConfiguration(new SeriesConfiguration());
    modelBuilder.ApplyConfiguration(new PresenceConfiguration());
}
```

Figura 5.8 Aplicarea configurărilor în clasa context

Entity Framework adaugă automat toate aceste reguli în metoda *OnModelCreating*, dar, pentru a facilita citirea și înțelegerea lor, am ales să împart aceste reguli în diferite clase, cum ar fi *GroupConfiguration*, în care vom găsi doar configurările asociate clasei *Group*. Am realizat această separare pentru toate entitățile folosite.

```
class GroupConfiguration : IEntityTypeConfiguration<Group>
{
    0 references | 1 author, 1 change
    public void Configure(EntityTypeBuilder<Group> entity)
    {
        entity.ToTable("Group");

        entity.HasKey("Id");

        entity.Property(g => g.Name)
            .IsRequired();

        entity.HasIndex(g => g.Name)
            .HasName("UQ_grp_name")
            .IsUnique();

        entity.HasOne(g => g.Series)
            .WithMany(s => s.Groups)
            .HasForeignKey(g => g.SeriesId)
            .OnDelete(DeleteBehavior.Restrict)
            .HasConstraintName("fk_group_series");
    }
}
```

Figura 5.9 Set de reguli pentru configurarea unei clase C#

În Figura 5.9 se poate observa setul de reguli generate pentru clasa *Group*. Se pot vedea mai multe tipuri de metode:

- *ToTable(„NumeTabel”)* – această metodă informează contextual programului că în clasa „Group” se va mapa tabela „Group”;
- *HasKey()* – metoda aceasta îi specifică contextului care este cheia primară din această tabelă. Această funcție poate primi ca parametru și o cheie primară compusă;
- *Property()* – avertizează contextul care sunt coloanele din baza de date ale tabelului specificate;

- HasIndex() – această metodă ne ajută să adăugăm constrângerile din bază pentru o anumită coloană. În figură se poate observa că este vorba despre constrângerea de unicitate;
- HasOne() – contextul este anunțat că această tabelă face parte dintr-o relație de tipul One-To-Many (o serie poate avea mai multe grupe) cu tabela „Series”, cu ajutorul proprietății „SeriesId”, care reprezintă și cheie străină. Cele două funcții „HasOne()” și „WithMany()” atribuie proprietățile de navigare din cele două clase între care va exista legătura.

5.2.3 Design Patterns

Pentru a păstra în aplicația LabGradR un cod bine structurat, care poate fi ușor de citit, de modificat, dar și care să se plieze perfect pe arhitectura prezentată, descrisă și folosită, am ales să utilizez trei paradigme de proiectare a codului, cunoscute în domeniul informatic ca „Design Patterns”. Principalul lor scop, pe lângă oferirea unei bune lizibilități codului, este de a evita apariția anumitor probleme.

Voi face o prezentare detaliată pentru fiecare dintre cele trei paradigme de proiectare utilizate în aplicația LabGradR.

Repository

Pentru a evita folosirea de mult cod duplicat pentru cele doisprezece entități cu care lucrează programul nostru, am ales să abstractizez operațiile prin care se fac adăugarea, citirea, ștergerea și editarea (operațiile CRUD) cu ajutorul conceptului de Repository [10].

```
public class BaseRepository<TEntiy>
    where TEntiy : class, IEntity
{
    private readonly LabGradRContext Context;

    12 references | 1 author, 1 change
    public BaseRepository(LabGradRContext context)
    {
        this.Context = context;
    }

    72 references | 1 author, 1 change
    public IQueryable<TEntiy> Get()
    {
        return Context.Set<TEntiy>().AsQueryable();
    }

    1 reference | 1 author, 1 change
    public void RemoveRange(IEnumerable<TEntiy> entities)
    {
        Context.Set<TEntiy>().RemoveRange(entities);
    }
}
```

Figura 5.10 Clasa BaseRepository

Conceptul presupune crearea unei clase care să manipuleze obiecte de tipuri generice și care să realizeze operațiile pe baza de date, punctate mai sus. Această clasă Repository va deveni podul dintre layer-ul de Data Access și cel în care se găsește logica de business și îi va livra

celui din urmă datele, astfel încât el să le utilizeze direct, fără să știe în ce mod se realizează relația cu baza de date. Metodele din clasa Repository vor lucra cu obiecte generice sau colecții de obiecte de tipul IEnumerable sau IQueryable (Figura 5.10).

Se poate observa în Figura 5.10 că această clasă, BaseRepository, primește contextul pe care l-a configurat layer-ul de Data Access printr-o modalitate numită Dependency Injection [18] și implementează metode pentru operațiile cu entități: adăugare, adăugare multiplă, modificare, ștergere, ștergere multiplă etc.

Unit Of Work

Pentru implementarea funcționalităților propuse, aplicația noastră trebuie să lucreze cu mai multe entități în același timp, iar schimbările acestora să fie făcute simultan, altfel pot apărea probleme în logica programului. Ca să nu întâmpinăm aceste probleme am optat pentru o execuție a modificărilor din bază în mod tranzacțional, adică vom putea opera oricând toate entitățile de care avem nevoie și vom salva la sfârșit toate aceste modificări.

Vom exemplifica conceptul pentru o înțelegere amănunțită. Presupunem că administratorul adaugă în sistem un utilizator. În momentul când acesta trimite fișierul Excel în care se găsesc datele utilizatorului, programul pregătește, în spate, pentru a fi adăugate în bază, un obiect de tipul User și încă unul de tipul UserRole, care este asociat tabelii de legătură dintre User și Role. În mod ideal ar trebui ca în acest caz să se introducă în baza de date, concomitent, o înregistrare în tabela User și una în tabela UserRole. Dacă nu am folosi Unit Of Work s-ar putea să ajungem în situația în care una dintre cele două încercări de adăugare în bază să genereze o eroare, dar să nu întrerupă procesul celeilalte înregistrări. În acest caz pot rezulta două situații:

1. Se introduce cu succes înregistrarea în tabela User, dar cea din tabela UserRole nu. Atunci când utilizatorul se va loga în aplicație, programul nu va ști ce rol are și nu îi va asocia nicio permisiune.
2. Se reușește introducerea înregistrării în tabela UserRole, dar nu și introducerea în tabela User. Tabela UserRole nu va permite un ID de User care nu există în bază, va apărea o eroare și aplicația se va opri.

La fel ca Repository, Unit Of Work primește, prin Dependency Injection, contextul deja configurat de către layer-ul inferior.

Vedem în Figura 5.11 implementarea design pattern-ului Unit Of Work, cuprinzând și toate entitățile care se folosesc de această clasă (Users, Roles etc.).

```

public class UnitOfWork
{
    private readonly LabGradRContext Context;

    0 references | 1 author, 1 change
    public UnitOfWork(LabGradRContext context)
    {
        this.Context = context;
    }

    17 references | 1 author, 1 change
    public void SaveChanges()
    {
        Context.SaveChanges();
    }

    private BaseRepository<User> users;
    14 references | 1 author, 1 change
    public BaseRepository<User> Users => users ?? (users = new BaseRepository<User>(Context));

    private BaseRepository<Role> roles;
    2 references | 1 author, 1 change
    public BaseRepository<Role> Roles => roles ?? (roles = new BaseRepository<Role>(Context));
}

```

Figura 5.11 Clasa ce implementează paradigma Unit Of Work

Dependency Injection

Este crucial într-o aplicație să nu existe niciodată în memorie mai multe instanțe ale unor clase de care nu avem nevoie deoarece ar putea apărea erori de logică. Dacă în programul nostru am avea două obiecte de tipul Context s-ar putea genera erori de concurență, ajungându-se să nu se știe cu care dintre ele se lucrează și cine execută modificări asupra lor. Datorită acestor detalii, am decis să folosesc în implementarea platformei LabGradR injectarea dependențelor, lucru care ne oferă niște avantaje considerabile:

1. Atunci când începe aplicația să ruleze, creăm obiectele de care avem nevoie o singură dată și le trimitem gata definite acolo unde sunt necesare, prin injectarea acestora în constructor.
2. Aceste obiecte au o durată de viață aleasă de noi, dintre cele trei tipuri:
 - Transient – adică obiectele au nevoie de o nouă instanță pentru fiecare request;
 - Scoped – obiectele sunt aceleași în cadrul acelui request;
 - Singleton – obiectele sunt aceleași în orice request.

5.2.4 Logica de business

Partea de Back-end se încheie cu stratul care include logica de business. În această parte a aplicației ne vom folosi de datele entităților primite de la layer-ul de Data Access pentru a le manipula și configura după bunul plac pentru a scoate, într-un final, elemente folosite în funcționalitățile dorite. Am ales să împart logica de business în mai multe clase, numite servicii, care să conțină metodele ce lucrează cu o singură entitate, astfel se păstrează o structură curată în proiect.

În Figura 5.12 se pot observa toate serviciile existente în logica de business, denumite după entitatea cu care se lucrează în funcții.

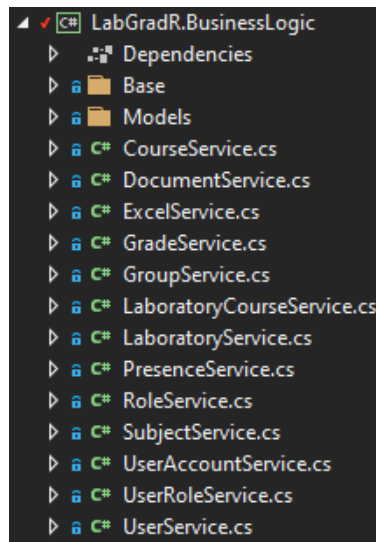


Figura 5.12 Arhitectura logicii de business

Voi prezenta, cu ajutorul unui exemplu din cod, cum funcționează serviciile și cum trimite mai departe datele de care are nevoie partea de Front-End. Serviciul prezentat va fi cel pentru entitatea Group și se poate surprinde parțial în Figura 5.13Figura 5.12.

```
public class GroupService : BaseService
{
    0 references | 1 author, 1 change
    public GroupService(UnitOfWork unitOfWork) : base(unitOfWork)
    {
    }

    1 reference | 2 authors, 3 changes
    public List<Group> GetGroupsByProfAndSubj(int profId, int subjId)
    {
        var series = UnitOfWork
            .Courses
            .Get()
            .Include(c => c.Series)
            .Where(c => c.ProfessorId == profId && c.SubjectId == subjId)
            .Select(c => c.SeriesId)
            .ToList();

        var groups = UnitOfWork
            .Groups
            .Get()
            .Where(g => series.Contains(g.SeriesId))
            .ToList();

        return groups;
    }
}
```

Figura 5.13 Serviciul pentru entitatea Group

După cum se vede din imagine, în acest serviciu interogăm în mai multe moduri baza de date cu ajutorul Design Pattern-ului descris anterior, Unit Of Work, și extragem informații ale entităților sub formele dorite. În situația dată ne dorim să obținem o listă cu toate grupele care au o anumită materie și un anumit profesor. În servicii se mai pot găsi și funcții de adăugare, ștergere, modificare sau citire pentru entități.

Interogările făcute în bază din C# trebuie să fie lizibile, asemănătoare query-urilor din SQL, de aceea am ales folosirea unei componente specifice aplicațiilor .NET, numită Language

Integrated Query (LINQ) [19]. Câteva dintre metodele acesteia, utilizate și în aplicația noastră sunt:

- Where – asemănător clauzei WHERE din SQL, acesta returnează înregistrările care îndeplinesc anumite condiții și este folosit pentru filtrare;
- Select – este utilizat pentru a extrage doar anumite proprietăți dintr-o entitate sau pentru a crea proiecții, adică noi obiecte create;
- Include – este folosit pe post de JOIN din SQL;
- Any – va returna valoarea true dacă există vreo entitate cu proprietățile specificate și false în caz contrar;
- FirstOrDefault – această metodă returnează primul element găsit, iar dacă nu există se va întoarce valoarea default a tipului de date cerut;
- Sum, Average, Min, Max, Count – metode ce vor întoarce suma, media, minimul, maximul și numărul entităților dintr-o listă.

5.2.5 Interfața grafică

În această parte a aplicației, și anume interfața utilizatorului (UI), vom prezenta cum va comunica programul cu utilizatorul în mod direct, cum se trimit request-urile către layer-urile inferioare și cum se afișează răspunsurile acestora pentru utilizator. Pentru acest layer al aplicației m-am folosit de design pattern-ul Model-View-Controller, specificat în secțiunea 4.3.

În alineatele următoare voi descrie componentele conceptului de MVC, prezentând rolul fiecăreia și modul în care acestea comunică între ele.

Model

Platforma LabGradR folosește această componentă de MVC pentru a selecta doar o parte dintre informațiile unei entități care vin din baza de date, astfel, utilizatorul nu are acces la date private, pe care nu ar trebui să fie capabil să le vizualizeze. De exemplu, putem considera situația când un asistent de curs dorește să noteze studenții la un laborator. Fiecare element din această listă de studenți trebuie să conțină doar numele și nota la acest laborator (dacă deja i-a fost asociată o notă), deși în baza de date se găsesc mai multe informații despre un user, cum ar fi email-ul sau parola.

Rezolvarea acestei probleme se va face prin maparea entităților care vin din baza de date la niște clase noi care să cuprindă doar proprietățile pe care vrem să le afișăm pe ecran. Am adăugat în numele acestor modele sufixul “Vm”(View Model) și pe care l-am folosit ca să specific că aceste clase sunt folosite pentru a le trimite unei componente View.

În Figura 5.14 se poate vedea modelul folosit pentru a stoca informațiile pe care le va vedea un student despre o lecție a unui laborator. În bază se află toate informațiile despre toate lecțiile, ale tuturor studenților. Le vom filtra și extrage numai cele ale studentului logat, iar apoi le vom mapa la această entitate *StudentLessonVm*.


```

public class StudentLessonVm
{
    3 references | 0 authors, 0 changes
    public int LabId { get; set; }
    2 references | 0 authors, 0 changes
    public string LabName { get; set; }
    2 references | 0 authors, 0 changes
    public float? Grade { get; set; }
    2 references | 0 authors, 0 changes
    public bool IsPresent { get; set; }
    1 reference | 0 authors, 0 changes
    public List<Document> Documents { get; set; }
}

```

Figura 5.14 Modelul unei lecții de laborator pentru un student

Mapările din aplicația LabGradR se realizează cu ajutorul unei librării, adesea folosită în aplicațiile .NET, care se numește AutoMapper [20]. Se pot face mapări directe dintr-o proprietate în alta dacă numele acestora coincide sau se pot specifica sursa și destinația. Aceste mapări trebuie specificate într-o clasă care trebuie să moștenească casa de bază Profiler, pusă la dispoziție de AutoMapper. În Figura 5.15 putem vedea configurația mapării entității de lecție în modelul prezentat mai sus.

```

CreateMap<Lesson, StudentLessonVm>()
    .ForMember(l => l.LabId, opt => opt.MapFrom(lab => lab.Id))
    .ForMember(l => l.LabName, opt => opt.MapFrom(lab => lab.Name));

```

Figura 5.15 Crearea configurației unei mapări

Un alt avantaj al folosirii modelelor este acela că putem adăuga validări pe anumite proprietăți și acestea nu vor permite adăugarea de inputuri care să nu corespundă cerințelor specificate. De exemplu, în cazul schimbării parolei, pe lângă faptul că toate câmpurile sunt obligatorii, am adăugat o validare ca numărul de caractere să fie cel puțin șapte. Nerespectarea acestora vor duce la afișarea în dreptul câmpului a erorii stocate în *ErrorMessage* (Figura 5.16).

```

[MinLength(7, ErrorMessage = "Noua parola trebuie sa contina cel putin 7 caractere")]
[Required(ErrorMessage = "Noua parola este obligatorie")]
[PasswordPropertyText]
5 references | 1 author, 1 change
public string NewPassword { get; set; }

```

Figura 5.16 Validări adăugate direct pe model

Controller

Această componentă reprezintă podul dintre utilizator și program pentru că pe aici trec toate request-urile de la utilizator către aplicație și layerele inferioare. În controller aducem, folosind dependency injection, serviciile pe care le vom apela și tool-ul cu care vom mapa entitățile în modele.

Când un utilizator accesează o funcționalitate, se trimite un request către controller, care va folosi serviciile și va prelua din layer-ul logicii de business elementele pe care utilizatorul le așteaptă, le va mapa, cu ajutorul Mapper-ului, într-un model și apoi va trimite rezultatul obținut componentei de View.

Toate acestea sunt împărțite în mai multe metode, numite acțiuni, depinzând de datele dorite și funcționalitățile așteptate. Numele acțiunilor este foarte important pentru că acesta va apărea, împreună cu numele controllerului în URL-ul browser-ului. Dacă administratorul dorește adăugarea de studenți va ajunge pe controller-ul destinat administrării și se va intra pe acțiunea de *AddStudent*, URL-ul pe care acesta îl vede este de forma */Administration/AddStudent*.

Scopul unui controller este de a manipula request-urile de tip HTTP. Acțiunile pot primi niște attribute pentru a ști cu ce tip de request va lucra. În cazul nostru, tipurile de attribute folosite sunt *[HttpGet]* și *[HttpPost]*. Dacă se intră pe o acțiune cu un alt tip de request față de cel dorit, utilizatorului i se va preciza că cererea nu a fost corectă.

Atunci când, spre exemplu, un asistent de laborator dorește să schimbe numele unei lecții de laborator, el va apăsa butonul de “editare”, se va trimite un request de *HttpGet* pentru a aduce din baza de date informațiile deja existente despre acea lecție, cum ar fi numele, documentele încărcate, notele studenților. Când termină modificările, el va apăsa pe butonul de “save” care va trimite un request de tip *HttpPost* pentru a salva noile date în bază.

Pentru că în cod avem două acțiuni cu numele “EditLab”, una cu atributul de *GET* și alta de *POST*, controller-ul trebuie să știe către ce acțiune să te direcționeze în fiecare caz.

Pe lângă attributele prezentate mai sus, un controller sau o acțiune poate primi și attribute de *[Authorize]*, care indică faptul că acea entitate poate fi accesată doar de rolurile autorizate (Figura 5.17).

```
[Authorize(Roles = "Laborant, Professor")]  
[HttpGet]  
2 references | 2 authors, 3 changes  
public IActionResult LabPageAdmin(int subjectId, int groupId)  
{  
    }
```

Figura 5.17 Header de acțiune din controller cu attributele *Authorize* și *HttpGet*

View

Această componentă a design pattern-ului MVC cuprinde paginile pe care le vede utilizatorul și pe care sunt afișate informațiile obținute și trimise de pe Controller. Fișierele de view au extensia “.cshtml”, ceea ce înseamnă că în ele se poate scrie, atât cod HTML, cât și cod C#.

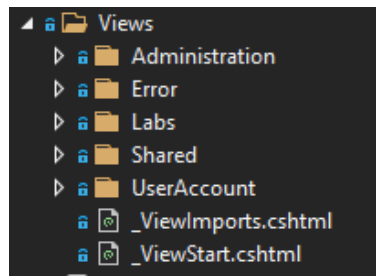


Figura 5.18 Structura de foldere a view-urilor

După cum se poate vedea și în Figura 5.18, pentru a păstra o structură uniformă a fișierelor, paginile de view vor fi împărțite, pe foldere, în funcție de controller-ul care folosește respectivele view-uri. Acestea adesea iau numele acțiunilor din controller pentru a anunța sistemul că acea pagină îi este asociată și pentru a ști controller-ul cărui view să trimită modelul.

Paginile de view, primesc prin componenta de Model informațiile pe care, mai apoi, le adaugă în elemente de HTML [21] pentru a le afișa pe ecran. Aceste pagini sunt stilizate cu ajutorul CSS [22] și JavaScript [23].

Pe lângă view-urile destinate doar anumitor controllere, există și cele de tipul Shared, care, de obicei, sunt partial-views, ceea ce înseamnă că reprezintă doar un fragment dintr-o pagină. Acestea pot fi utilizate în orice alt view. Un bun exemplu este bara de navigare care cuprinde meniul aplicației și care este vizualizată pe toate paginile programului.

```
@model LabGradR.WebApplication.Models.UserAccount.LoginVm
{
    ViewData["Title"] = "Login";
}
<div class="loginbox">
    <div class="login">Login</div>
    <hr class="loginline" />
    <div class="row">
        <div class="emailpassbox">
            <form asp-action="Login">
                <div asp-validation-summary="ModelOnly" class="text-danger"></div>
                <div class="form-group">
                    <label asp-for="Email" class="control-label emailtext"></label>
                    <input asp-for="Email" class="form-control" />
                    <span asp-validation-for="Email" class="text-danger"></span>
                </div>
                <div class="form-group">
                    <label asp-for="Password" class="control-label passtext">Parola</label>
                    <input asp-for="Password" class="form-control" type="password" />
                    <span asp-validation-for="Password" class="text-danger"></span>
                </div>
                <div class="form-group form-check">
                </div>
                <div class="form-group">

```

Figura 5.19 Pagină de view

În Figura 5.19 se poate observa view-ul destinat paginii de login care primește modelul *LoginVm*. Aici se poate vedea cum framework-ul de ASP.NET ne ajută să legăm câmpurile din HTML cu proprietățile din model prin attributele de *asp-for* date tag-urilor *label*, *input* sau *span*. Erorile validărilor adăugate pe model trebuie afișate pe ecran, de aceea am ales să le introduc sub câmpurile de unde sunt extrase informațiile pentru respectivele proprietăți.

5.2.6 Integrare

În acest subcapitol voi prezenta cele mai importante tehnologii cu care am implementat unele dintre funcționalitățile de bază ale aplicației, care nu sunt cuprinse în framework-ul folosit. Ele au fost instalate cu ajutorul tool-ului NuGet [24], prin care programele .NET pot descărca și folosi cod din surse publice.

Asynchronous JavaScript and XML (AJAX)

Pentru că dorim ca utilizatorul aplicației noastre să aibă parte de o experiență cât mai plăcută în timpul folosirii acesteia, am ales folosirea AJAX [25].

Acest tool, adesea folosit în dezvoltarea programelor web, ne ajută să comunicăm cu serverul printr-un obiect de tipul *XMLHttpRequest*. Cel mai mare avantaj în utilizarea acestuia este natura lui asincronă, prin care putem trimite request-uri către server și primi date fără a face refresh la pagină. HTML și CSS pot fi combinate cu AJAX pentru a oferi programului elemente dinamice și pentru a stiliza paginile. În aplicația LabGradR am utilizat AJAX pentru a șterge elemente de pe pagină în mod dinamic, fără a mai fi nevoie de refresh (Figura 5.20).

```
window.addEventListener("load", function () {
    let url = "/Labs/DeleteDocument";
    let deldoc = (e) => {
        let documentId = e.target.dataset.documentid;
        $.ajax({
            type: "DELETE",
            url: `${url}/${documentId}`,
            contentType: "application/json",
            success: function () {
                let parent = $(e.target.parentElement);
                let parentOfParent = parent.parent();
                parent.detach();
                if (parentOfParent.children().length <= 2) {
                    parentOfParent.detach();
                }
            },
            error: function (error) {
            }
        })
    }
    $(".del_doc_btn").on("click", deldoc);
})
```

Figura 5.20 Funcție AJAX pentru a șterge un document din pagină

Se poate observa în imagine că această funcție de AJAX primește mai mulți parametri în care se specifică tipul de request, url-ul, tipul de conținut al obiectului și două funcții *success* și *error*, care se vor executa dacă funcționalitatea este dusă la bun sfârșit cu bine, sau dacă intervin probleme pe parcurs. Funcția *success*, în cazul de față, va elimina de pe pagină, după reușirea ștergerii din bază a documentului, și elementele de HTML în care acesta se afla. La fel ca mai sus, în *error* ne putem folosi de elementele de HTML pentru a afișa erorile venite înapoi de pe server.

EPPlus

Funcționalitățile propuse aplicației noastre implică, în mare parte, manevrarea și configurarea fișierelor de tipul Excel, de la adăugarea și ștergerea de utilizatori, indiferent de rolul lor, până la managementul materiilor sau laboratoarelor.

Aceste operații am decis să le implementez cu ajutorul unei librării specifice aplicațiilor .NET, numită EPPlus [2] și care este specializată în configurarea foilor de calcul (spreadsheets) de tipul Office Open XML. Lucrul cu acest tool pentru configurarea, parcurgerea și extragerea informațiilor din fișiere de tip Excel este asemănător manipularea unei matrice, cele doua entități având în comun împărțirea pe linii și coloane.

După cum se poate observa în Figura 5.21, crearea coloanelor și câmpurilor fișierului Excel pot include numeroase proprietăți, cât și restricții de tipuri sau validări. În imagine ni se prezintă formarea header-ului fișei de calcul destinată utilizatorilor, prin adăugarea coloanelor de Nume, Prenume, Email și editarea acestora.

```
using (var package = new ExcelPackage())
{
    var worksheet = package.Workbook.Worksheets.Add("Utilizatori");

    worksheet.Column(1).Width = 20;
    worksheet.Column(2).Width = 20;
    worksheet.Column(3).Width = 40;

    worksheet.Cells[1, 1].Value = "Nume";

    worksheet.Cells[1, 1].Style.Font.Size = 12;
    worksheet.Cells[1, 1].Style.Font.Bold = true;
    worksheet.Cells[1, 1].Style.Border.Top.Style = ExcelBorderStyle.Hair;

    worksheet.Cells[1, 2].Value = "Prenume";

    worksheet.Cells[1, 2].Style.Font.Size = 12;
    worksheet.Cells[1, 2].Style.Font.Bold = true;
    worksheet.Cells[1, 2].Style.Border.Top.Style = ExcelBorderStyle.Hair;

    worksheet.Cells[1, 3].Value = "Email";

    worksheet.Cells[1, 3].Style.Font.Size = 12;
    worksheet.Cells[1, 3].Style.Font.Bold = true;
    worksheet.Cells[1, 3].Style.Border.Top.Style = ExcelBorderStyle.Hair;
}
```

Figura 5.21 Crearea unui fișier de tipul Excel cu ajutorul EPPlus

În momentul importării unui document de tipul Excel, tot prin utilitarul EPPlus, fișierul este parcurs, trecut printr-o serie de validări și apoi trimise datele către bază. Dacă apar erori și validările nu sunt trecute, se vor afișa pe ecran, într-o listă, toate liniile și coloanele care nu sunt conform standardelor impuse.

6 EXPERIENȚA UTILIZATORULUI

În capitolele anterioare am cuprins structura programului cu descrierea arhitecturii, funcționalităților și tehnologiilor utilizate, iar acum urmează să prezentăm, folosind exemple vizuale din aplicația propriu-zisă, experiența unui utilizator care ia contact cu platforma LabGradR.

6.1 Pagina de Login

Pagina cu care se deschide aplicația este cea de login, unde utilizatorul își introduce adresa de email și parola pentru a se autentifica în contul său. Aplicația LabGradR nu dispune de pagină destinată înregistrării pentru că, fiind vorba de o platformă de e-learning din cadrul facultății, studenților li se dau conturile deja configurate.

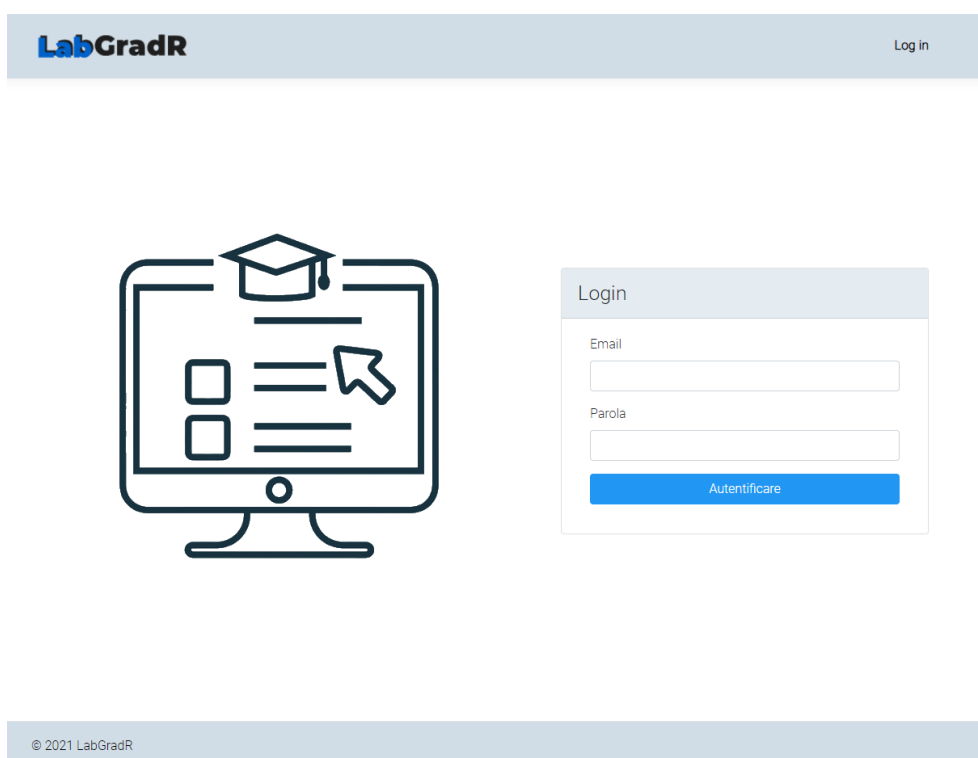


Figura 6.1 Pagina de login

Datorită acestui aspect, programul va include în bază, înainte de adăugarea altor tipuri de utilizatori, toate datele persoanelor care vor administra aplicația prin adăugarea de utilizatori, materii, serii, grupe etc.

6.2 Utilizatorul autentificat

Odată intrat în cont, fiecare tip de utilizator va vedea doar ceea ce îi va permite rolul. Astfel că o persoană autentificată drept administrator va putea vizualiza centrul de administrare al aplicației, iar profesorii, asistenții și studenții vor avea posibilitatea de a accesa materiile prezente pe pagina lor de Dashboard.

Aplicația LabGradR prezintă o secțiune în header care cuprinde, pe lângă numele întreg al persoanei logate, și o listă cu rolurile sale, iar atunci când ducem cursorul mouse-ului

deasupra numelui, vor coborî opțiunile de a schimba parola și de logout. Acestea se pot vedea în Figura 6.2.

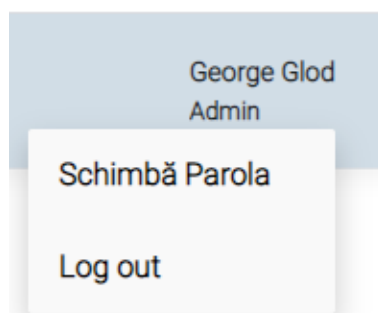


Figura 6.2 Opțiunile prezente la hover pe numele utilizatorului

6.3 Centrul de administrare

Această secțiune a aplicației este utilizată și poate fi accesată doar de persoanele vizate din cadrul facultăților, precum secretari sau alți angajați ai sectorului universitar, autorizați în acest sens, care vor respecta cu strictețe acordul GDPR (General Data Protection Regulation) și vor îndeplini atribuțiile acestui tip de utilizator.

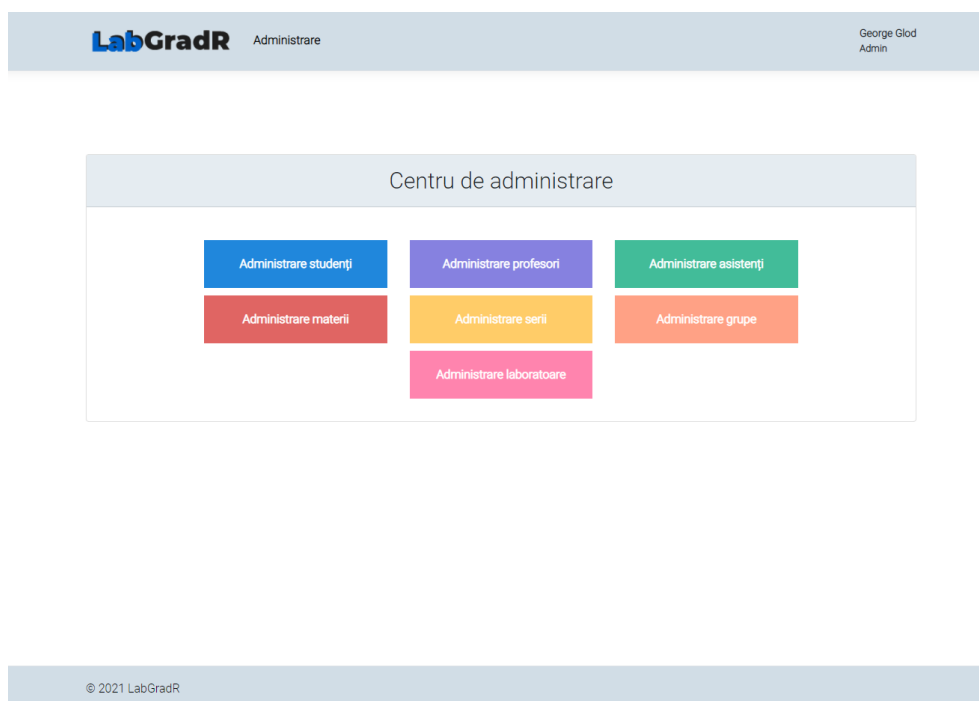


Figura 6.3 Pagina de administrare

În imaginea de mai sus se poate vedea pagina de Home a unui administrator și paginile pe care acesta le poate accesa.

6.4 Pagina de administrare a unei entități

Din acest loc se va descărca documentul de tip Excel care va conține informațiile deja existente despre entitatea aleasă din baza de date și utilizatorul va putea edita, adăuga sau șterge entități. Structura acestei pagini este simplă și intuitivă: se descarcă fișierul Excel din secțiunea specificată, se completează cu datele dorite, iar apoi se încarcă înapoi pe site.

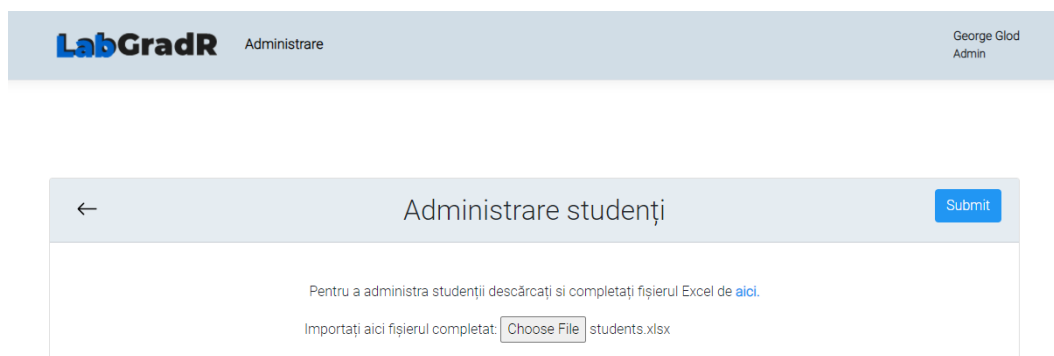


Figura 6.4 Exemplu de pagină de administrare

Figura 6.5 prezintă un exemplu de fișier Excel descărcat. Câmpurile ce trebuie completate pentru fiecare entitate vor fi specificate de numele coloanelor, iar unele vor avea deja configurată o modalitate de selecție dintr-un dropdown pentru a nu introduce sau scrie o resursă care nu se găsește în baza de date.

	A	B	C	D
1	Nume	Prenume	Email	Grupa
2	Popescu	Alexandru	alexandrupopescu@acs.pub.ro	311CA
3	Rusu	Andrei	andreirusu@acs.pub.ro	<div>311CA</div> <div>312CA</div>
4				
5				

Figura 6.5 Modelul unui fișier Excel

Programul nu va permite inserarea de fișiere care nu respectă regulile impuse, astfel, dacă documentul inserat are câmpuri necompletate, cu tipuri de date care nu se potrivesc cerințelor, nume de entități care nu există în bază sau dacă au fost modificate numele coloanelor deja existente vor apărea erori în care se va specifica locul producerii acestora, prin numărul linei și al coloanei, și o mică explicație pentru edificare (Figura 6.6).

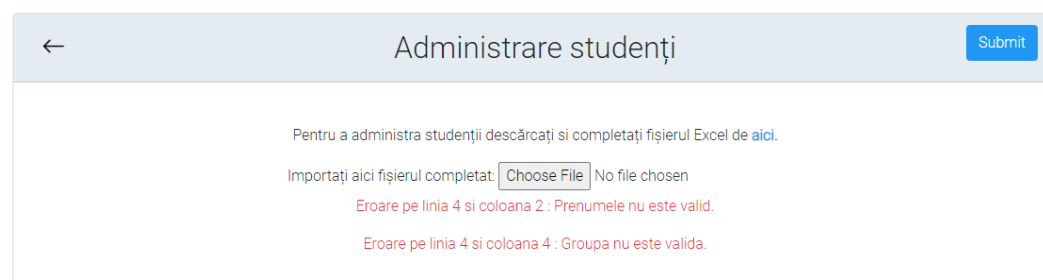
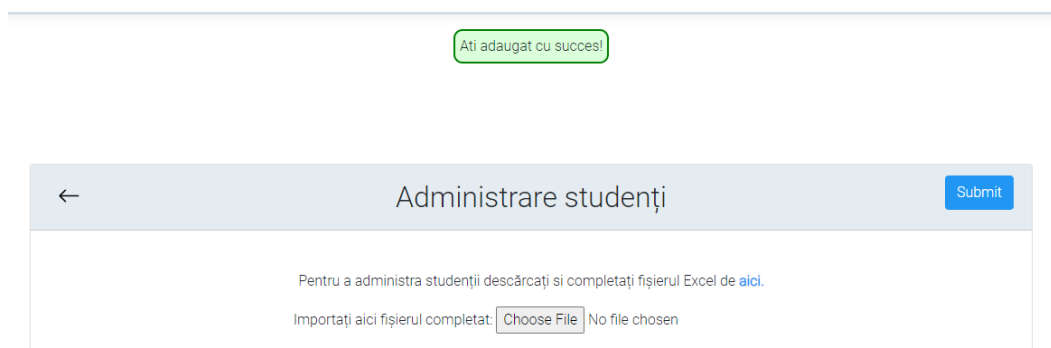


Figura 6.6 Erori apărute la adăugarea fișierului

În cazul în care acțiunea de importare trece de toate validările, programul va returna un mesaj specific pentru a anunța utilizatorul de finalizarea cu succes a operației.



The screenshot shows a web interface with a light blue header bar. In the center of the header is the text "Administrare studenți". On the left of the header is a back arrow icon, and on the right is a blue "Submit" button. Below the header, there is a white box containing the text: "Pentru a administra studenții descărcați și completați fișierul Excel de [aici](#)." followed by "Importați aici fișierul completat: Choose File | No file chosen". Above this white box, a green rounded rectangle contains the text "Ați adăugat cu succes!".

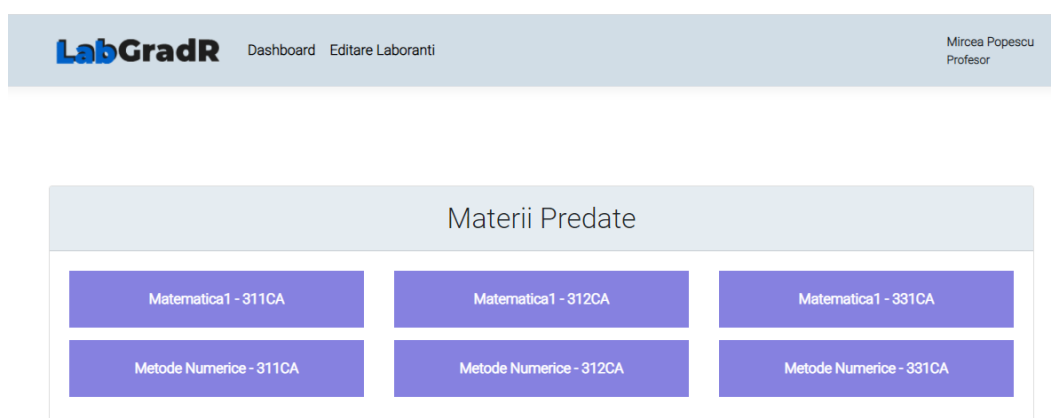
Figura 6.7 Mesaj de confirmare pentru o importare de date cu succes

Aceste funcționalități se aplică pentru toate paginile de administrare, acestea având un mod similar de operare, dar cu fișiere Excel care au coloane și validări diferite.

6.5 Pagina de Dashboard

Această locație este considerată pagina de Home a aplicației noastre, iar aici profesorii, asistenții sau studenții vor putea vedea și accesa materiile cu care au legătură. Fiecare tip de utilizator va vedea și accesa diferit aceste laboratoare. De exemplu, unui profesor i se vor afișa pe Dashboard-ul său, împreună cu grupa, materiile pe care acesta le predă (Figura 6.8).

După cum se poate vedea în Figura 6.8, profesorul autentificat predă Matematică1 și Metode Numerice la seria CA deoarece are lista tuturor grupelor din această serie. Deși încă nu a asociat asistenți pentru aceste laboratoare, el va putea crea lecții și nota studenții fără a face acest lucru.



The screenshot shows a dashboard for a professor. The top header bar is light blue and contains the "LabGradR" logo, the text "Dashboard Editare Laboranti", and the user's name "Mircea Popescu Profesor". Below the header, there is a white box titled "Materii Predate". Inside this box, there are six purple buttons arranged in a 2x3 grid. The top row contains buttons for "Matematica1 - 311CA", "Matematica1 - 312CA", and "Matematica1 - 331CA". The bottom row contains buttons for "Metode Numerice - 311CA", "Metode Numerice - 312CA", and "Metode Numerice - 331CA".

Figura 6.8 Dashboard profesor



Figura 6.9 Dashboard-ul unui student care este și asistent

În Figura 6.9 avem cazul când un utilizator are două roluri, și anume student și asistent, după cum se poate vedea și sub numele lui din header-ul aplicației. Dashboard-ul îi va împărți laboratoarele în așa fel încât să nu existe confuzie între ele. Materiile la care este asistent vor apărea în chenarul cu “Materii la care asişti”, iar cele pe care le studiază se vor găsi în chenarul de deasupra.

Încă un mod de a diferenția laboratoarele este după culori, acest aspect ieșind în evidență, atât în Figura 6.9, cât și în Figura 6.8 (materiile predate vor apărea cu mov, cele studiate cu albastru, iar cele notate de un asistent cu verde).

6.6 Pagina unui laborator (asistent/profesor)

La fel ca Dashboard-ul, această pagină este influențată de rolul care o accesează, prin urmare studenții vor vedea altceva în comparație cu profesorii sau asistenții. Inițial, când este accesat un laborator pentru prima dată, vom primi un mesaj pentru a ne înștiința că nu au fost adăugate lecții la acel laborator (Figura 6.10).

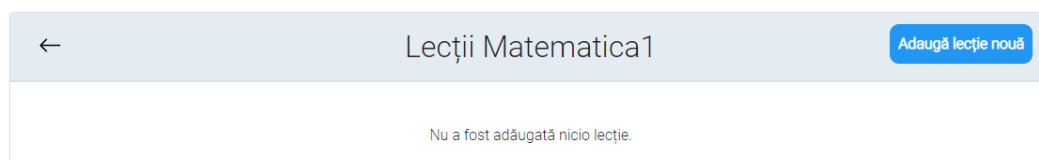


Figura 6.10 Pagina goală unui laborator

Apăsând pe butonul de adăugare a unei lecții ni se va deschide o nouă pagină cu un câmp în care trebuie scris numele pentru lecția ce urmează să fie adăugată (Figura 6.11)

Figura 6.11 Secțiunea de adăugare a unui lecții

După adăugarea mai multor lecții, lista va arăta ca în Figura 6.12. Tabelul în care se văd liniile recent adăugate prezintă acțiunile de editare, mutare a unei linii pe o altă poziție în listă și ștergere. Imaginea de mai jos înfățișează cazul unei ștergeri de lecție. Ștergerea entităților este o operație foarte importantă, care se poate uneori accesa din greșeală, de aceea, va apărea un modal pentru a confirma ștergerea unei lecții.

Nume	Editează	Mutare	Șterge
Matrice	Editează	↓	⊗
Operații complexe cu matrice	Editează	↑ ↓	⊗
Ecuatii de gradul 2	Editează	↑ ↓	⊗
Ecuatii diferențiale	Editează	↑	⊗

Figura 6.12 Ștergerea unei lecții

Butonul de “Editează” ne va trimite pe pagina unde vom putea vedea toate informațiile despre acea lecție de laborator și un mic catalog al studenților din grupa în al căru laborator ne aflăm.

Nume	Prezență	Notă
Balan Costel	<input type="checkbox"/>	0
Dinu Anda	<input type="checkbox"/>	0
Popa Mihai	<input type="checkbox"/>	0
Rosu Andreea	<input type="checkbox"/>	0
Rusu Andrei	<input type="checkbox"/>	0

Figura 6.13 Pagina de editare/vizualizare a unei lecții de laborator

6.7 Pagina de editare a unei lecții de laborator

În această secțiune a aplicației vom putea nota și pune prezențe studenților. După cum se poate vedea în figurile Figura 6.13 și Figura 6.14, această pagină este împărțită pe mai multe bucăți, fiecare având propria funcționalitate.

Nume	Prezență	Notă
Balan Costel	<input checked="" type="checkbox"/>	11
Dinu Anda	<input checked="" type="checkbox"/>	10
Popa Mihai	<input checked="" type="checkbox"/>	10
Rosu Andreea	<input type="checkbox"/>	0
Rusu Andrei	<input type="checkbox"/>	0

Figura 6.14 Notarea studenților și adăugarea de documente pentru o lecție

În partea din stânga sus există un câmp dedicat schimbării numelui lecției. Sub acesta se găsește locul unde putem adăuga documente lecției curente. Aici avem o casetă pentru a adăuga un nume diferit fișierului recent importat și butoane pentru ștergerea documentelor în caz că dorim acest lucru. La fel ca la lista de lecții, vom fi întrebați printr-un modal dacă suntem siguri că vrem să ducem la capăt operațiunea.

Cea mai importantă funcționalitate a acestei pagini, adică notarea studenților, se află în partea dreaptă, grupa de studenți fiind vizualizată sub forma unui tabel ce cuprinde numele complet, prezența și nota. Odată schimbate informațiile din listă va trebui apăsat butonul de “save” pentru a trimite modificările efectuate și în baza de date, iar dacă operația a fost efectuată cu succes vom primi un mesaj colorat în verde care ne va informa acest lucru.

6.8 Pagina unui laborator (student)

După ce un profesor sau asistent adaugă lecții, apoi note, prezențe sau documente, studenții din grupele respective le pot vizualiza pe pagina laboratorului. Ei vor vedea într-un tabel, de data această fără alte butoane, lecțiile cu notele pe care le-au primit și documentele fiecăreia. Dacă la un laborator, un asistent creează o lecție și nu o modifică deloc, studentului îi va apărea lecția, dar va avea absență.

În Figura 6.15 se poate vedea lista de lecții, dintre care prima și ultima au documente disponibile pentru descărcare, de la laboratorul de Matematică1.

← Matematica1		
	Prezenta	Nota
Matrice	Prezent	11
<div>Document1.xlsx</div> <div>Document2.xlsx</div>		
Ecuatii diferențiale	Prezent	9
Ecuatii de gradul 2	Absent	
<div>Document1.xlsx</div>		

Figura 6.15 Pagina unui laborator cu lecții din perspectiva unui student

6.9 Paginile de adăugare a asistenților

Pentru a adăuga asistenți la grupe, un profesor trebuie să apese pe butonul specific numai rolului său, aflat în header-ul aplicației. Pagina pe care este trimis conține toate materiile la care utilizatorul predă și va trebui selectată cea la care se dorește asocierea asistenților cu grupele (Figura 6.16)

LabGradR
Dashboard
Adăugare asistenți

Mircea Popescu
Profesor

Materii

Matematica1

Adăugați asistenți

Metode Numerice

Adăugați asistenți

Figura 6.16 Pagina cu materiile profesorului autentificat

După selecția materiei vor apărea grupele seriei sau seriilor la care profesorul curent își predă materia. În dreptul fiecărei grupe este plasat un dropdown pentru a selecta unul dintre asistenții înregistrați în sistem (Figura 6.17).

Adăugare asistenți la

Matematica1

Salvează

Grupa	Asistent de Curs
311CA	Bianca Stan
312CA	Alexandru Popescu
331CA	Alexandra Stanciu

Figura 6.17 Pagina de selectare asistenți pentru grupe

După ce aleg asistenți pentru toate grupele, utilizatorul trebuie să apese pe butonul de “Save”, care va trimite asocierile făcute către baza de date și va redeschide pagina cu materiile disponibile pentru a ușura o viitoare selecție de materie.

6.10 Pagina de schimbare a parolei

Această pagină este disponibilă pentru toate tipurile de utilizator și prezintă o funcționalitate simplă, dar totuși foarte utilă. Deoarece aplicația nu cere administratorului să adauge nicio parolă la introducerea de utilizatori în sistem, programul trebuie să genereze una automat. Astfel, la inserarea în bază a unui student, profesor sau asistent, în secțiune de parolă din bază se adaugă o parola creată prin alipirea numelui și prenumelui persoanei.

La prima autentificare în aplicație, utilizatorul va intra automat pe pagina de schimbare a parolei pentru a mări securitatea contului său. Acest pas este opțional pentru că poate fi evitat prin apăsarea butoanelor din header, dar este foarte indicat pentru o mai bună protecție a contului. Platforma nu se va opri să te redirecționeze la autentificare pe pagina de schimbare a parolei până când nu treci prin această etapă.

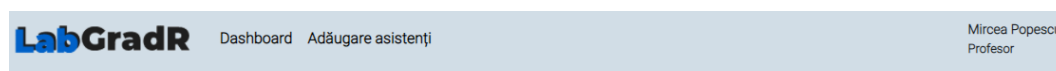
The image shows a form titled 'Schimbare parolă' (Change Password). It contains three input fields: 'Parola actuala:' (Current Password), 'Parola noua:' (New Password), and 'Confirmare parola:' (Confirm Password). Below the fields is a blue button labeled 'Salvare' (Save).

Figura 6.18 Pagina de schimbare parolă

7 CONCLUZII ȘI DEZVOLTĂRI ULTERIOARE

În această lucrare am realizat o prezentare detaliată a unei platforme web cu scopul de a ușura modul de notare și păstrare a rezultatelor studenților din cadrul laboratoarelor sau seminarelor. Prin această aplicație destinată universităților sau sistemelor de învățământ superior, am împărțit funcționalitățile pe patru roluri, care împreună definesc comportamentul final al programului. Inițiatorul sistemului descris este reprezentat de persoanele destinate cu adăugarea utilizatorilor în sistem, adică cei cu rolul de administrator. Ei vor putea monitoriza și păstra evidența entităților adăugate în baza de date. Celelalte roluri vor fi reprezentate de persoanele care vor utiliza platforma.

LabGradR ajută profesorii și asistenții de laborator oferindu-le posibilitatea de a adăuga și configura lecțiile laboratoarelor și de a nota studenții. După autentificare, aceștia din urmă își vor putea urmări parcursul și descărca documentele auxiliare de pe paginile laboratoarelor. Funcționalitățile menționate, împreună cu toate cele implementate, sunt explicate pe larg în Capitolul 2, unde am realizat și o împărțire a lor pe roluri.

Deși câteva dintre aplicațiile deja existente pe piață, menționate și descrise în Capitolul 3, conțin funcționalități cu care se pot adăuga și nota laboratoarele, ele nu sunt punctual create pentru a satisface această nevoie, iar uneori sunt incomode și durează mult timp. Comparățiile dintre aplicații arată că LabGradR implementează multe dintre necesitățile unei platforme de e-learning, ceea ce îl face foarte util în facilitarea procesului de punctare și contabilizare al prezențelor la seminare sau laboratoare.

Arhitectura pe baza căreia s-a construit aplicația este detaliată în Capitolul 4 al lucrării și este concepută în așa fel încât să respecte principiile de bază ale programării web pentru o împărțire ideală a conceptelor. Din descrierea layer-elor observăm o contopire armonioasă a tehnologiilor care rezultă într-un workflow bine echilibrat.

Implementarea complexă care este descrisă amănunțit în Capitolul 5 surprinde platforma ca fiind un program bine încheiat și modular. Acest rezultat se datorează și folosirii design pattern-urilor care, pe lângă facilitarea multor operații complicate, oferă aplicației capabilitatea de a se extinde foarte ușor în viitor.

Rezultatul final al platformei este prezentat printr-o serie de scenarii în Capitolul 6, unde se poate observa că programul descris în secțiunile inițiale ale proiectului este complet funcțional și aduce o interfață intuitivă și prietenoasă.

7.1 Dezvoltări ulterioare

Cu toate că aplicația LabGradR are acum o versiune stabilă, viabilă și pregătită pentru lansare, se pot aplica diverse îmbunătățiri care vor aduce un surplus de valoare acesteia. Astfel, voi enumera și descrie pe scurt câteva dintre cele mai importante schimbări și dezvoltări care se pot aplica peste structura curentă.

Din comparația cu celelalte software-uri de e-learning am reținut că multe dintre acestea au funcționalități comune care ar îmbunătăți platforma curentă. În primul rând, se poate observa că pentru a ridica standardul este nevoie de o disponibilitate a aplicației pe dispozitive iOS sau Android, cum ar fi telefoanele sau tabletele. Rapiditatea și ușurința accesării aplicației de pe un alt gadget decât calculatorul pot oferi unui asemenea program o mare popularitate în rândul academicienilor.

În al doilea rând, secțiunile pentru încărcare de fișiere, în care studenții vor adăuga teme sau chiar activitatea din timpul laboratorului, ar putea veni în ajutorul asistenților. Aceștia vor avea posibilitatea să le corecteze și în afara programului de lucru cu studenții.

În ultimul rând, din comparațiile făcute s-a mai evidențiat că descărcarea și încărcarea notelor din exterior este o funcționalitate care este deseori folosită de asistenți. Fișierul descărcat care conține rezultatele va avea formatul unui catalog a întregii grupe, în care fiecărui student îi vor apărea notele în funcție de laborator.

Se poate ajunge în situația în care anumite laboratoare sau seminare sunt ținute de către doi asistenți, sau se întâmplă să existe un ajutor. Această problemă nu este acoperită momentan în implementarea curentă, dar cu ușurință se poate introduce o modalitate de selectare multiplă a asistenților la o grupă.

De asemenea, dacă din nefericire un student trebuie să repete o materie el are nevoie să fie introdus la laboratorul unei grupe cu un an mai mic, iar acest lucru este momentan imposibil. Cu ajutorul unei migrări, prin care vom configura structura bazei de date, se va schimba modalitatea de păstrare a legăturii dintre laborator și student. Ea nu se va mai baza pe grupă, ci se va realiza o relație directă între aceste entități printr-o tabelă de legătură nou creată.

Așa cum toate aplicațiile vin cu actualizări periodice, LabGradR nu este o excepție și, cu siguranță, se vor mai găsi în viitor moduri de îmbunătățire și de sporire a performanței acesteia.

8 BIBLIOGRAFIE

- [1] Microsoft, „SQL Server 2019 | Microsoft,” Microsoft, 2019. [Interactiv]. Available: <https://www.microsoft.com/en-us/sql-server/sql-server-2019>. [Accesat 20 June 2021].
- [2] EPPlus Software AB, „Excel Spreadsheetlibrary for .NET Framework/ Core,” EPPlus Software AB, [Interactiv]. Available: <https://epplussoftware.com/en>. [Accesat 20 June 2021].
- [3] Moodle, „MoodleNet whitepaper- MoodleDocs,” September 2019. [Interactiv]. Available: https://docs.moodle.org/dev/MoodleNet_whitepaper. [Accesat 19 June 2021].
- [4] Open CourseWare, „Open CourseWare [CS Open CourseWare],” 24 May 2021. [Interactiv]. Available: <https://ocw.cs.pub.ro/>. [Accesat 20 June 2021].
- [5] Claroline, „Claroline - Dare to use innovative pedagogy with our LMS,” Claroline, 2021. [Interactiv]. Available: <https://claroline.net/>. [Accesat 20 June 2021].
- [6] Blackboard Inc., „Blackboard Learn - An Advanced LMS,” Blackboard Inc., 2021. [Interactiv]. Available: <https://www.blackboard.com/teaching-learning/learning-management/blackboard-learn>. [Accesat 20 June 2021].
- [7] Microsoft Patterns & Practices Team, Microsoft Application Architecture Guide, 2nd Edition, Microsoft Press, 2009.
- [8] Microsoft, „What is Azure-Microsoft Cloud Services,” Microsoft, 2021. [Interactiv]. Available: <https://azure.microsoft.com/en-us/overview/what-is-azure/>. [Accesat 20 June 2021].
- [9] Microsoft, „Writing Transact-SQL Statements - SQL Server,” Microsoft, [Interactiv]. Available: <https://docs.microsoft.com/en-us/sql/t-sql/tutorial-writing-transact-sql-statements?view=sql-server-ver15&viewFallbackFrom=sql-server-2019>. [Accesat 20 June 2021].
- [10] Microsoft, „Implementing the Repository and Unit of Work Patterns in an ASP.NET MVC Application,” Microsoft, 30 July 2013. [Interactiv]. Available: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/older-versions/getting-started-with-ef-5-using-mvc-4/implementing-the-repository-and-unit-of-work-patterns-in-an-asp-net-mvc-application>. [Accesat 20 June 2021].

- [11] Microsoft, „What is .NET? An open-source developer platform,” Microsoft, 2021. [Interactiv]. Available: <https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet>. [Accesat 20 June 2021].
- [12] Microsoft, „Entity framework Documentation,” Microsoft, 2021. [Interactiv]. Available: <https://docs.microsoft.com/en-us/ef/>. [Accesat 20 June 2021].
- [13] Microsoft, „Getting Started with Entity Framework 6 Database First Using MVC 5,” Microsoft, 2019. [Interactiv]. Available: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/database-first-development/>. [Accesat 20 June 2021].
- [14] Microsoft, „Tutorial: Get Started with Entity Framework 6 Code First using MVC 5,” Microsoft, 2019. [Interactiv]. Available: <https://docs.microsoft.com/en-us/aspnet/mvc/overview/getting-started/getting-started-with-ef-using-mvc/creating-an-entity-framework-data-model-for-an-asp-net-mvc-application>. [Accesat 20 June 2021].
- [15] Microsoft, „Model First - EF6,” Microsoft, 23 10 2016. [Interactiv]. Available: <https://docs.microsoft.com/en-us/ef/ef6/modeling/designer/workflows/model-first>. [Accesat 20 June 2021].
- [16] Microsoft, „ASP.NET MVC Pattern | .NET,” Microsoft, [Interactiv]. Available: <https://dotnet.microsoft.com/apps/aspnet/mvc>. [Accesat 20 June 2021].
- [17] Microsoft, „Working with DbContext,” Microsoft, 23 October 2016. [Interactiv]. Available: <https://docs.microsoft.com/en-us/ef/ef6/fundamentals/working-with-dbcontext>. [Accesat 20 June 2021].
- [18] Microsoft, „Dependency injection in ASP.NET Core,” Microsoft, 21 July 2020. [Interactiv]. Available: <https://docs.microsoft.com/en-us/aspnet/core/fundamentals/dependency-injection?view=aspnetcore-5.0>. [Accesat 20 June 2021].
- [19] Microsoft, „Language Integrated Query (LINQ) in C#,” Microsoft, [Interactiv]. Available: <https://docs.microsoft.com/en-us/dotnet/csharp/linq/>. [Accesat 20 June 2021].
- [20] J. Bogard, „AutoMapper,” [Interactiv]. Available: <https://automapper.org/>. [Accesat 20 June 2021].
- [21] WHATWG, „HTML Standard,” WHATWG, 17 June 2021. [Interactiv]. Available: <https://html.spec.whatwg.org/>. [Accesat 20 June 2021].

- [22] World Wide Web Consortium, „Cascading Style Sheets,” World Wide Web Consortium, 17 June 2021. [Interactiv]. Available: <https://www.w3.org/Style/CSS/Overview.en.html>. [Accesat 20 June 2021].
- [23] Mozilla, „JavaScript,” 2021. [Interactiv]. Available: <https://developer.mozilla.org/en-US/docs/Web/JavaScript>. [Accesat 20 June 2021].
- [24] Microsoft, „NuGet documentation,” Microsoft, 2021. [Interactiv]. Available: <https://docs.microsoft.com/en-us/nuget/>. [Accesat 20 June 2021].
- [25] OpenJS Foundation, „Ajax | JQuery API Documentation,” OpenJS Foundation, [Interactiv]. Available: <https://api.jquery.com/category/ajax/>. [Accesat 20 June 2021].
- [26] Massachusetts Institute of Technology, „Get Started with OCW | MIT Open CourseWare,” 2001 - 2021. [Interactiv]. Available: <https://ocw.mit.edu/help/get-started-with-ocw/>. [Accesat 20 June 2021].
- [27] DigitalTasha, „COVID-19 Impact on the E-learning Industry,” Adobe, 11 August 2020. [Interactiv]. Available: <https://elearning.adobe.com/2020/08/covid-19-impact-on-the-e-learning-industry/>. [Accesat 20 June 2021].
- [28] M. Vora, H. Barvaliya, P. Balar și N. Jagtap, „E-Learning Systems and MOOCs - A Review,” *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*, pp. 636-641, September 2020.
- [29] J. Albahari și B. Albahari, *C# 7.0 in a Nutshell*, O'REILLY, 2018.
- [30] Microsoft, „Microsoft Excel Spreadsheet Software,” Microsoft, 2021. [Interactiv]. Available: <https://www.microsoft.com/en-ww/microsoft-365/excel>. [Accesat 20 June 2021].