

Universitatea “Politehnica” din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Includerea instrumentelor de predare-testare într-o aplicație e-learning

Proiect de diploma

Prezentată ca cerință parțială pentru obținerea
titlului de *Inginer*
în domeniul *Calculatoare și Tehnologia Informației*
programul de studii de licență *Ingineria Informației*

Conducător științific
Ș.L. Dr. Ing. Elena Cristina STOICA

Absolvent
Albert-Lucian LUȚĂ

Anul 2021

TEMA PROIECTULUI DE DIPLOMĂ
a studentului LUȚĂ I. Albert-Lucian , 444A

1. Titlul temei: Includerea instrumentelor de predare-testare într-o aplicație e-learning

2. Descrierea temei și a contribuției personale a studentului (în afara părții de documentare):

Se va proiecta și dezvolta o aplicație web de tip e-learning , care va conține următoarele componente:

- module dedicate pentru studenți, profesori, companie/universitate
- modul Studenți cu funcții pentru: acces cursuri, teste, calendar, note, chat/forum
- modul Profesori cu funcții pentru: management cursuri, adăugare materiale, assignments, teste, notare, acces chat/forum
- modul Companie/Universitate cu funcții pentru: management grupuri/subgrupuri(ani, serii, grupe) și participanți la acestea(studenți, profesori)

Aplicația va integra instrumente de predare-lucru-testare: creare, integrare instrumente de predare, editare materiale, utilizare diverse instrumente ca de ex. compilatoare ca suport pentru lucrul din platformă, activități colaborative în platforma e-learning (acces concurent la resurse), generarea de teste.

3. Discipline necesare pt. proiect:

Programarea calculatoarelor, Programare orientată obiect, Tehnologii de programare în Internet, Proiectarea bazelor de date

4. Data înregistrării temei: 2020-11-17 16:03:06

Conducător(i) lucrare,
Ș.L.Dr.Ing. Elena Cristina STOICA

Student,
LUȚĂ I. Albert-Lucian

Director departament,
Ș.L. dr. ing Bogdan FLOREA

Decan,
Prof. dr. ing. Mihnea UDREA

Cod Validare: afc75b0d0e

Declarație de onestitate academică

Prin prezenta declar că lucrarea cu titlul *Includerea instrumentelor de predare-testare într-o aplicație e-learning*, prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității “Politehnica” din București ca cerință parțială pentru obținerea titlului de *Inginer* în domeniul Inginerie Electronică și Telecomunicații/ Calculatoare și Tehnologia Informației, programul de studii *Ingineria Informației* este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor și măsurărilor pe care le prezint ca fiind făcute de mine, precum și metodele prin care au fost obținute, sunt reale și provin din respectivele simulări, experimente și măsurători. Înțeleg că falsificarea datelor și rezultatelor constituie fraudă și se sancționează conform regulamentelor în vigoare.

București, Septembrie 2021.

Absolvent: Albert-Lucian LUTA


.....

Cuprins

Lista figurilor	9
Lista acronimelor	11
Introducere	13
0.1. Scopul Proiectului	13
0.2. Descrierea proiectului	13
1. Proiectare	15
1.1. Funcționalități administrator	15
1.1.1. Structura universității	15
1.1.2. Managementul utilizatorilor	16
1.2. Funcționalități profesor	18
1.2.1. Crearea activităților	18
1.2.2. Notarea studenților	19
1.2.3. Crearea setului personal de întrebări	21
1.3. Funcționalități student	22
1.3.1. Accesarea detaliilor fiecărei activități	22
1.3.2. Încărcarea rezolvărilor pentru teme	23
1.3.3. Participarea la teste	25
1.4. Funcționalități comune	26
1.4.1. Participarea la discuții pe forum	26
2. Implementare	29
2.1. Modulul de Autentificare	31
2.2. Funcționalități administrator	33
2.2.1. Structura Universității	36
2.2.2. Managementul utilizatorilor	40
2.3. Funcționalități profesor	42
2.3.1. Crearea activităților	43
2.3.2. Accesarea activităților	47
2.3.3. Crearea setului personal de întrebări	51
2.4. Funcționalități student	53
2.4.1. Participarea la activități	53
Concluzii	57
Bibliografie	59
Anexa 1. Cod Sursa	61
1.1. Modul Autentificare - Funcționalități Server	61
1.2. Formular Autentificare	64
1.3. Formular Înregistrare	65

1.4. Validare autentificare server	68
1.5. Validare înregistrare server	68
1.6. Afişare lista universităţi	68
1.7. Bară de instrumente	71
1.8. Meniu Utilizator	72
1.9. Afişare listă facultăţi	74
1.10. Management listă utilizatori	77

Lista figurilor

1.1. Proiectare Software - Model Cascadă, Sursă: [1]	15
1.2. Diagramă Caz de utilizare - Creează Structura Universității	16
1.3. Diagramă Activitate - Creează Structura Universității	16
1.4. Diagramă Caz de utilizare - Adaugă Utilizator	17
1.5. Diagramă Activitate - Adaugă Utilizator	17
1.6. Diagramă Caz de utilizare - Creează Activitate	18
1.7. Diagramă Activitate - Creează Activitate	19
1.8. Diagramă Caz de utilizare - Notează Student	20
1.9. Diagramă Activitate - Notează Student	20
1.10. Diagramă Caz de utilizare - Creează Set Personal de Întrebări	21
1.11. Diagramă Activitate - Creează Set Personal de Întrebări	22
1.12. Diagramă Caz de utilizare - Accesează Activitate	23
1.13. Diagramă Activitate - Accesează Activitate	23
1.14. Diagramă Caz de utilizare - Încarcă Rezolvare Temă	24
1.15. Diagramă Activitate - Încarcă Rezolvare Temă	24
1.16. Diagramă Caz de utilizare - Participă la Test	25
1.17. Diagramă Activitate - Participă la Test	26
1.18. Diagramă Caz de utilizare - Participă la Forum	27
1.19. Diagramă Activitate - Participă la Forum	27
2.1. Arhitectură Client-Server	29
2.2. Exemplu design flexibil - Desktop	30
2.3. Exemplu design flexibil - Telefon Mobil	31
2.4. Formular Autentificare, Anexă: 1.2, 1.4	31
2.5. Formular Înregistrare, Anexă: 1.3, 1.5	32
2.6. Exemplu Eroare Validare - Client(stânga), Server(dreapta)	33
2.7. Tablou de bord Utilizator	33
2.8. Tablou de bord Utilizator - Bară de instrumente	34
2.9. Tablou de bord Utilizator - Listă Universități, Anexă: 1.6	34
2.10. Tablou de bord Utilizator - Formular Creare Universitate	35
2.11. Tablou de bord Utilizator - Meniu Universitate Administrator(stânga), Profe- sor/Student(dreapta)	35
2.12. Tablou de bord Utilizator - Formular Editare Universitate	36
2.13. Tablou de bord Utilizator - Întrebare siguranță ștergere(stânga), părăsire(dreapta)	36
2.14. Tablou de bord Universitate - Bară de Instrumente1.7	37
2.15. Tablou de bord Universitate - Meniu navigare rapidă	37
2.16. Tablou de bord Universitate - Meniu Utilizator Administrator(stânga), Profe- sor(mijloc), Student(dreapta)1.8	38
2.17. Tablou de bord Universitate - Listă facultăți Administrator1.9	38
2.18. Tablou de bord Universitate - Listă facultăți Student	39
2.19. Tablou de bord Universitate - Formular creare(stânga), editare(dreapta) Facultate	39

2.20. Tablou de bord Universitate - Formular creare(stânga), editare(dreapta) Materie	39
2.21. Tablou de bord Universitate - Meniu Facultate/Materie	39
2.22. Tablou de bord Management Utilizatori1.10	40
2.23. Tablou de bord Management Utilizatori - Formular Adăugare Utilizator	40
2.24. Tablou de bord Management Utilizatori - Formular Editare Utilizator	41
2.25. Tablou de bord Management Utilizatori - Alegere Rol Utilizator	41
2.26. Tablou de bord Management Utilizatori - Formular Înrolare la Facultăți	42
2.27. Tablou de bord Management Utilizatori - Formular Înrolare la Materii	42
2.28. Tablou de bord Materie - Listă secțiuni Profesor	43
2.29. Tablou de bord Materie - Formular creare(stânga), editare(dreapta) Secțiune . .	43
2.30. Tablou de bord Materie - Meniu alegere tip de Activitate	44
2.31. Tablou de bord Materie - Formular creare(stânga), editare(dreapta) Material/- Forum	44
2.32. Tablou de bord Materie - Câmpuri în plus pentru tipul Temă	44
2.33. Tablou de bord Materie - Câmp alegere Timp - dată(stânga), oră(mijloc), mi- nut(dreapta)	45
2.34. Tablou de bord Materie - Câmpuri în plus pentru tipul Test	46
2.35. Tablou de bord Materie - Formular adăugare întrebări	46
2.36. Tablou de bord Activitate - Detalii de bază	47
2.37. Tablou de bord Activitate - Detalii specifice temelor	47
2.38. Tablou de bord - Notare temă Student	48
2.39. Tablou de bord Activitate - Detalii specifice testelor	48
2.40. Tablou de bord Activitate - Întrebări test	49
2.41. Tablou de bord Activitate - Listă studenți Test	49
2.42. Tablou de bord - Notare test Student	50
2.43. Tablou de bord Activitate - Forum	50
2.44. Tablou de bord Activitate - Adăugare comentariu Forum	50
2.45. Tablou de bord Set Întrebări	51
2.46. Tablou de bord Set Întrebări - Formular creare(stânga), editare(dreapta) Categorie	51
2.47. Tablou de bord Set Întrebări - Formular Întrebare	52
2.48. Tablou de bord Set Întrebări - Indicații Notare Automată	53
2.49. Tablou de bord Materie - Listă secțiuni Student	53
2.50. Tablou de bord Activitate - Încărcare temă	54
2.51. Tablou de bord Activitate - Detalii test Student	54
2.52. Tablou de bord Test activ	55
2.53. Tablou de bord Test activ - Întrebare răspuns unic	55
2.54. Tablou de bord Test activ - Întrebare răspuns multiplu	55
2.55. Tablou de bord Verificare Test	56

Lista acronimelor

API = Application Programming Interface(Interfață de programare a aplicației)

ORM = Object-relational Mapping(Cartografiere obiect-relațională)

SGBD = Sistem de gestiune a bazei de date

TCP = Transmission Control Protocol(Protocol de control al transmisiei)

UI = User Interface(Interfață cu utilizatorul)

UX = User Experience(Experiența utilizatorului)

Introducere

0.1 Scopul Proiectului

Proiectul are ca scop dezvoltarea unei aplicații web e-learning, cu ajutorul căreia universitățile să-și poată desfășura activitatea de predare/testare și în mediu online. Această metodă nu are rolul de a înlocui desfășurarea activității în mediu fizic, ci de a o îmbunătăți. Spre exemplu, studenții pot avea acces de oriunde și oricând la materialele și notele fiecărei materii, iar profesorii pot primi temele și proiectele pentru curs, seminar sau laborator online, fără a mai fi nevoie să transporte foile de hârtie ale fiecărui student. Cu toate acestea, mediul online prezintă dezavantaje în momentul testării/examinării cunoștințelor proprii ale studenților, deoarece comunicarea online este foarte greu sau chiar imposibil de oprit. Din acest motiv, consider că examinările fizice sunt cea mai bună modalitate în acest caz, iar cele online pot fi păstrate pentru simulări, exerciții propuse sau variante din ani trecuți ai examenului.

0.2 Descrierea proiectului

Aplicația permite crearea și managementul unei universități. În cadrul universității pot exista 3 tipuri de utilizatori: utilizator cu rol de administrator, utilizator cu rol de profesor și utilizator cu rol de student, fiecare având acces la funcționalități diferite.

- Student
 - Accesarea materialelor(cursuri, seminarii, laboratoare)
 - Accesarea temelor și încărcarea rezolvărilor
 - Participarea la teste și simulări
 - Participarea la discuții pe forum
 - Vizualizarea notelor
 - Vizualizarea viitoarelor activități
- Profesor
 - Încărcarea materialelor
 - Crearea temelor
 - Crearea testelor/simulărilor
 - Notarea studenților
 - Participarea la discuții pe forum
 - Împărțirea activităților(materiale, teme, teste, forum) pe secțiuni
 - Crearea setului personal de întrebări și răspunsuri pentru teste
- Administrator
 - Managementul facultăților

- Managementul materiilor
- Managementul utilizatorilor

Capitolul 1

Proiectare

Înainte de a trece la implementarea propriu-zisă a proiectului, trebuie să înțelegem foarte bine cerințele și metodele prin care acesta va fi construit. Această treaptă preliminară implementării se numește proiectare, și a devenit din ce în ce mai importantă cu cât programele software au devenit mai complexe.

Proiectarea software reprezintă procesul de găsim soluțiilor conceptuale pentru realizarea unei aplicații, pe baza unui set de cerințe și așteptări. Acest pas poate reduce costurile și timpul de implementare, în cazul unor neînțelegeri inițiale sau a unor schimbări de cerințe, și ușurează procesul de estimare al timpului și costului de realizare al întregului proiect.[2]

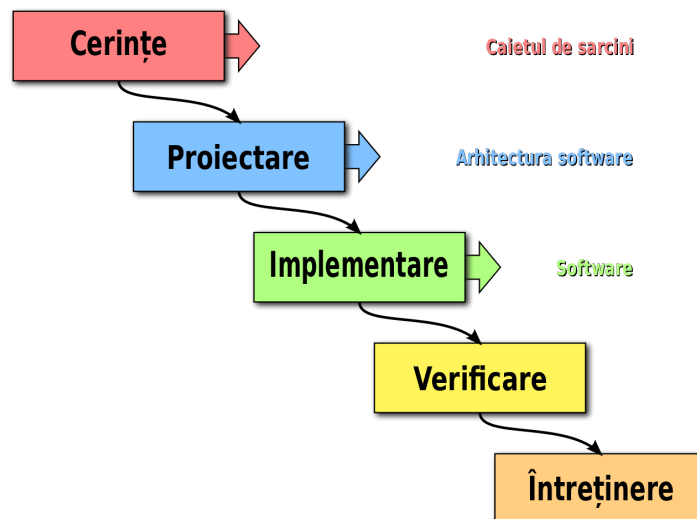


Figura 1.1: Proiectare Software - Model Cascadă, Sursă: [1]

1.1 Funcționalități administrator

Funcționalitățile utilizatorilor cu rol de administrator sunt următoarele:

- Managementul facultăților
- Managementul materiilor
- Managementul utilizatorilor

1.1.1 Structura universității

Înainte de a putea face managementul utilizatorilor, administratorul este nevoit să definească structura universității - facultățile și materiile componente. Acesta va putea adăuga facultăți, iar pentru fiecare dintre acestea va putea adăuga materiile asociate.

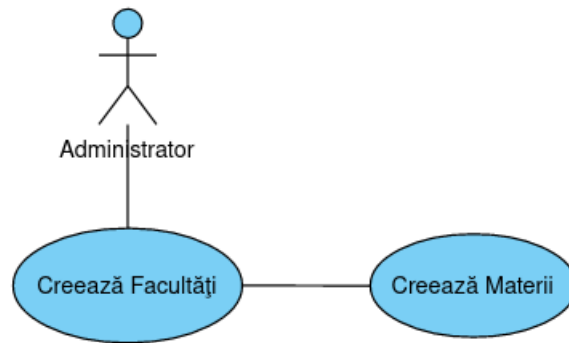


Figura 1.2: Diagramă Caz de utilizare - Creează Structura Universității

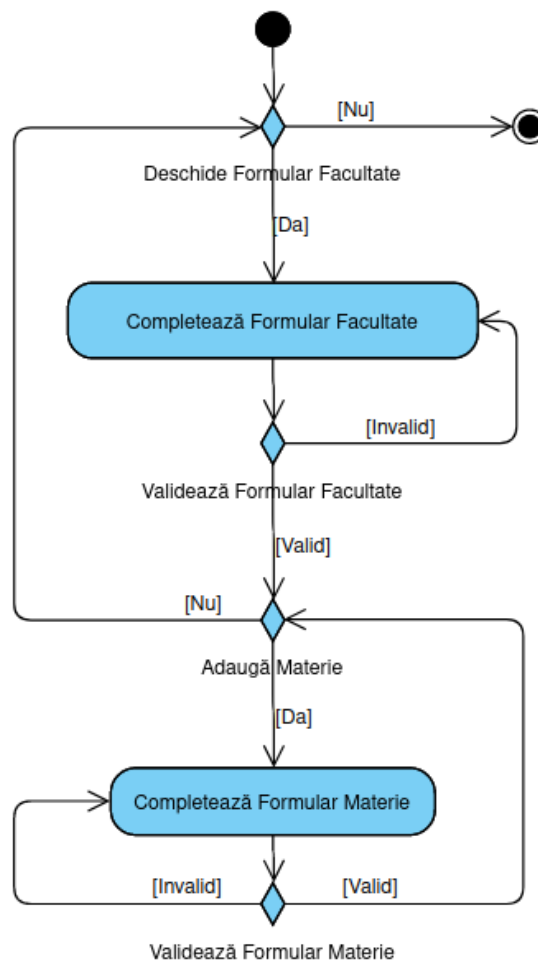


Figura 1.3: Diagramă Activitate - Creează Structura Universității

1.1.2 Managementul utilizatorilor

Odată stabilită structura universității, administratorul va putea adăuga utilizatori, le va putea atribui un anumit rol și îi va putea repartiza la facultățile și materiile la care vor avea acces.

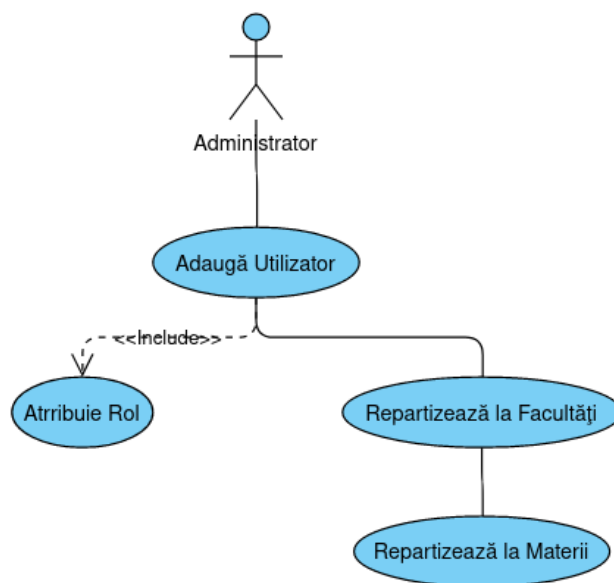


Figura 1.4: Diagramă Caz de utilizare - Adaugă Utilizator

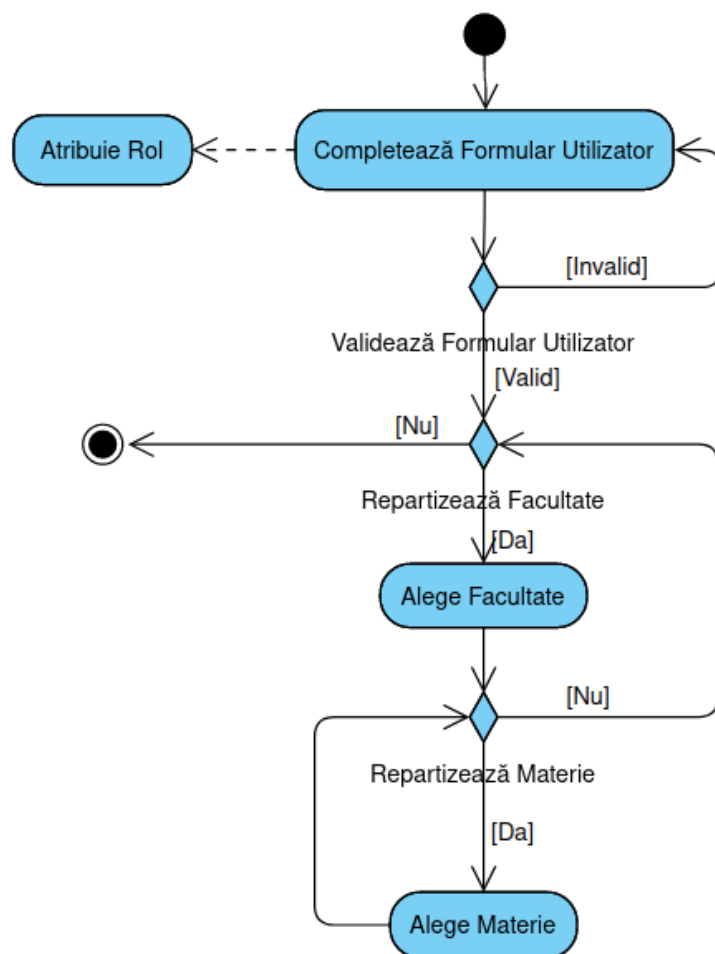


Figura 1.5: Diagramă Activitate - Adaugă Utilizator

1.2 Funcționalități profesor

Funcționalitățile utilizatorilor cu rol de profesor sunt următoarele:

- Încărcarea materialelor
- Crearea temelor
- Crearea testelor/simulărilor
- Notarea studenților
- Împărțirea activităților(materiale, teme, teste, forum) pe secțiuni
- Crearea setului personal de întrebări și răspunsuri pentru teste

1.2.1 Crearea activităților

Activitățile pot fi de 4 tipuri: materiale, temă, test și forum. Pentru crearea unei activități vom avea nevoie atât de detaliile de bază, pe care le vom găsi la orice tip de activitate, cât și de detaliile specifice fiecărui tip în parte. Pentru o mai bună repartizare logică a activităților, acestea vor fi împărțite pe secțiuni.

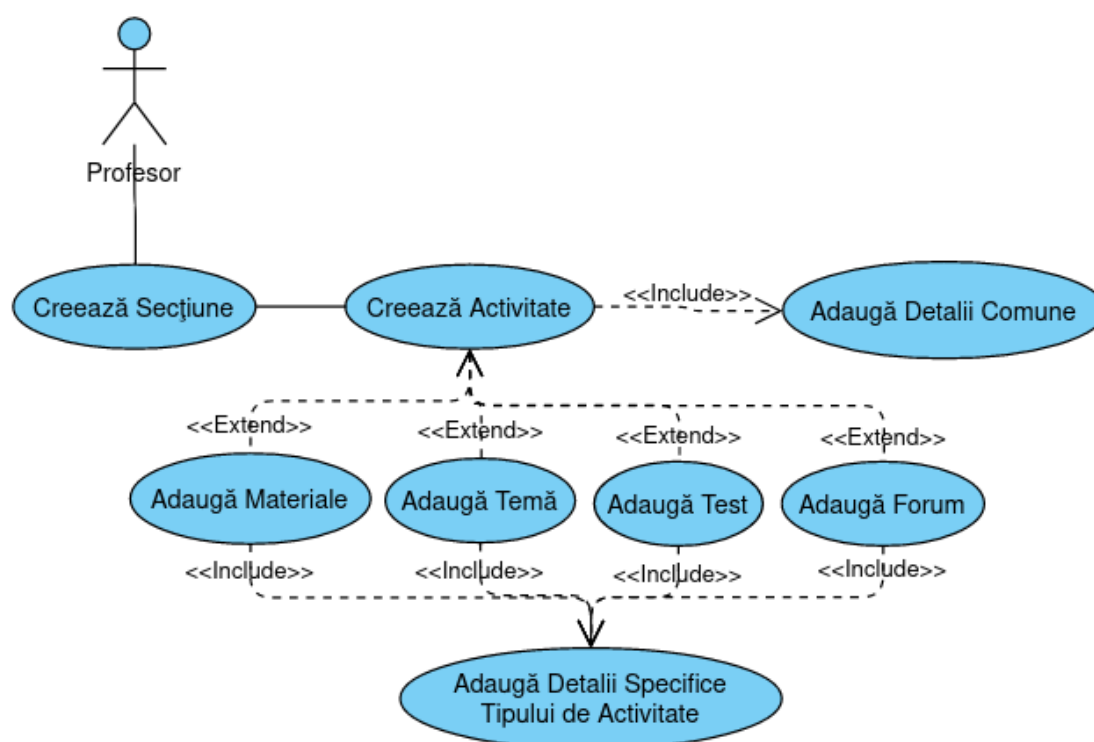


Figura 1.6: Diagramă Caz de utilizare - Creează Activitate

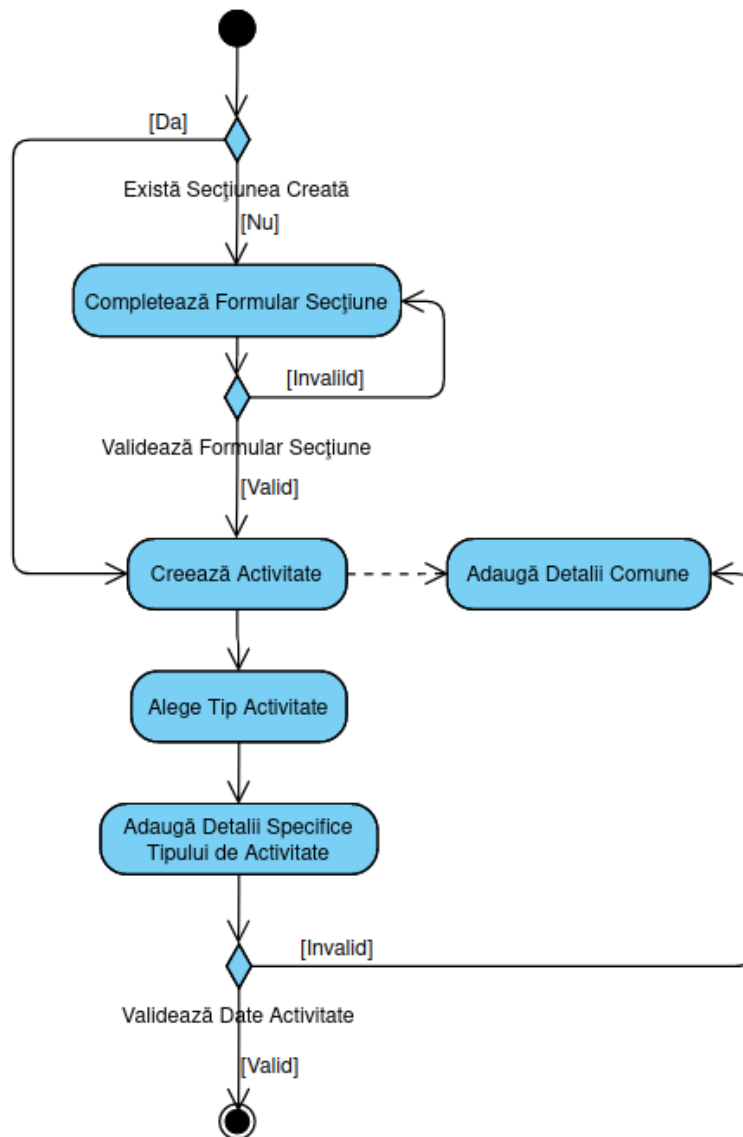


Figura 1.7: Diagramă Activitate - Creează Activitate

1.2.2 Notarea studenților

Pentru notare, profesorul este nevoit în primul rând să selecteze activitatea pentru care se va desfășura procesul. Odată aleasă activitatea, acesta va putea alege unul dintre studenții care au participat la aceasta și îi va putea nota.

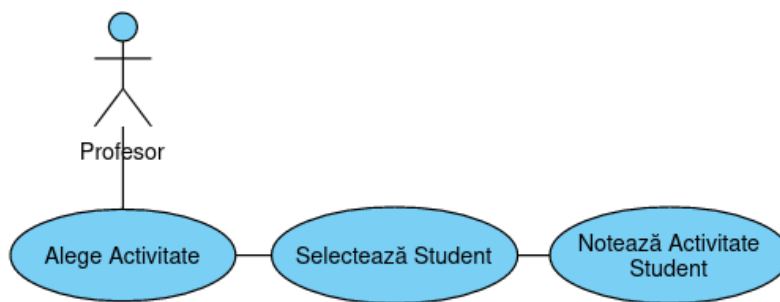


Figura 1.8: Diagramă Caz de utilizare - Notează Student

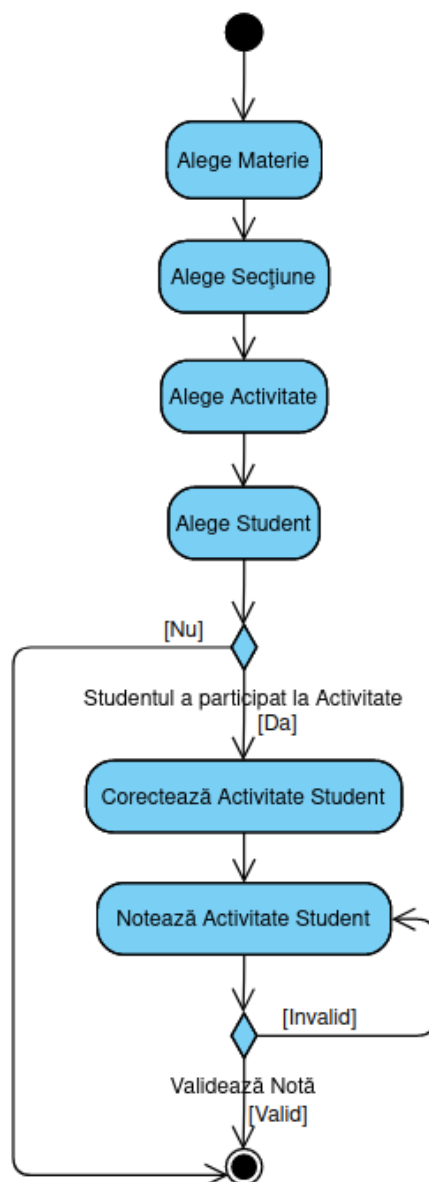


Figura 1.9: Diagramă Activitate - Notează Student

1.2.3 Crearea setului personal de întrebări

Pentru a face mai facilă refolosirea întrebărilor de la un an la altul, profesorii își vor putea crea un set personal de întrebări pe care îl va folosi la generarea testelor. Acesta va putea crea întrebări de diferite tipuri, repartizate în funcție de o categorie, adăuga răspunsuri și marca pe cele corecte și greșite.

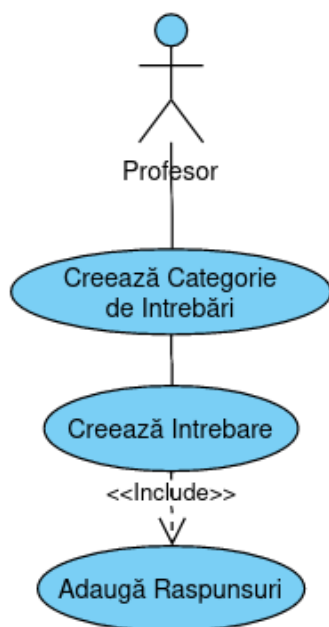


Figura 1.10: Diagramă Caz de utilizare - Creează Set Personal de Întrebări

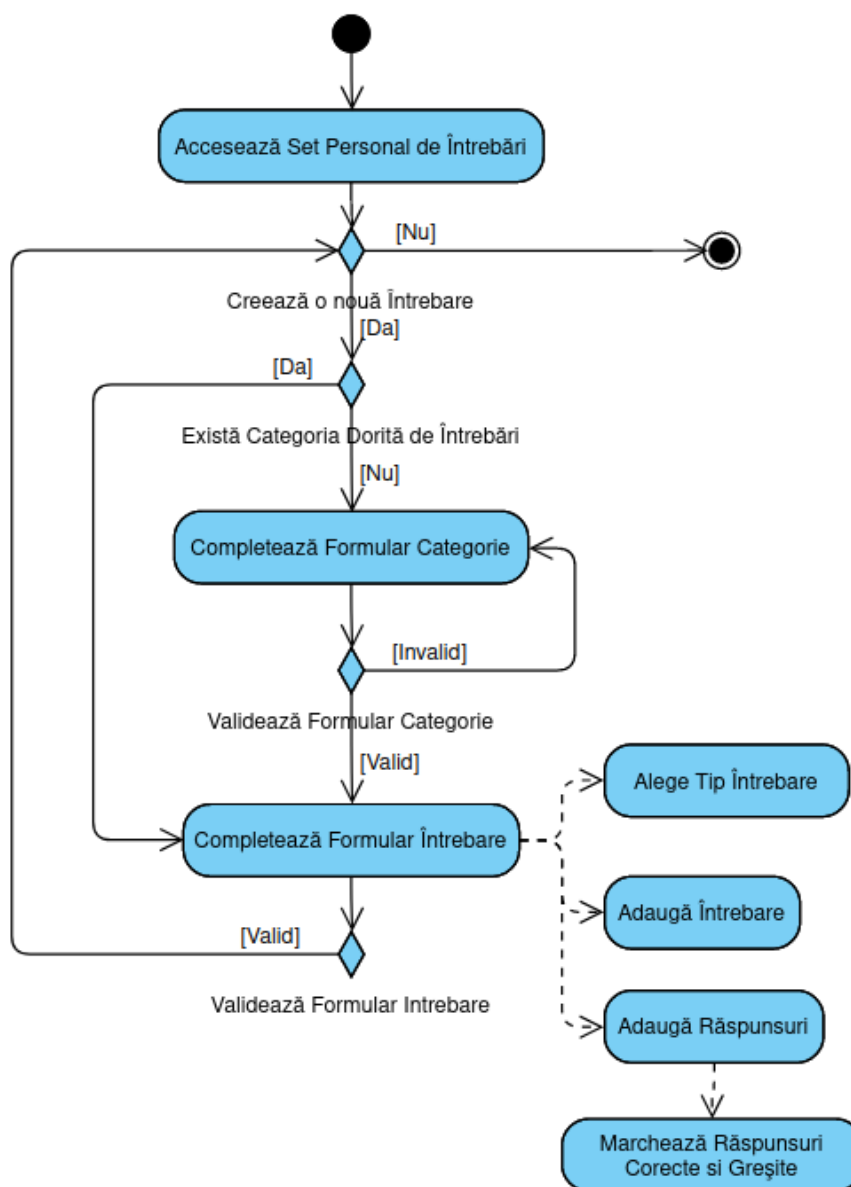


Figura 1.11: Diagramă Activitate - Creează Set Personal de Întrebări

1.3 Funcționalități student

Funcționalitățile utilizatorilor cu rol de student sunt următoarele:

- Accesarea materialelor(cursuri, seminarii, laboratoare)
- Accesarea temelor și încărcarea rezolvărilor
- Participarea la teste și simulări
- Vizualizarea notelor

1.3.1 Accesarea detaliilor fiecărei activități

După ce profesorul a creat structura materiei, secțiuni și activități, studenții vor avea acces la acestea. Ei își vor putea alege activitatea la care vor să ia parte, și vizualiza detaliile despre

aceasta.

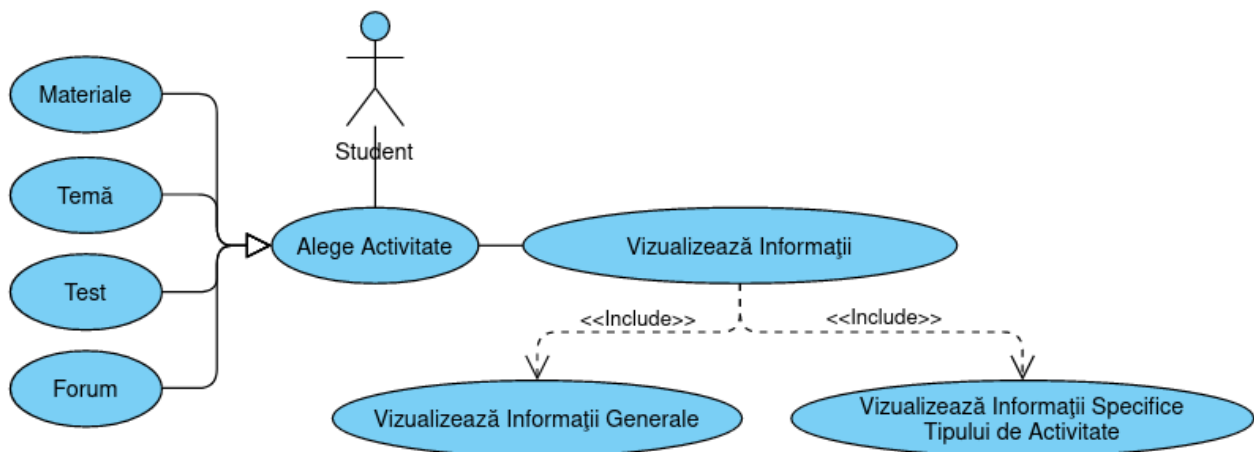


Figura 1.12: Diagramă Caz de utilizare - Accesează Activitate

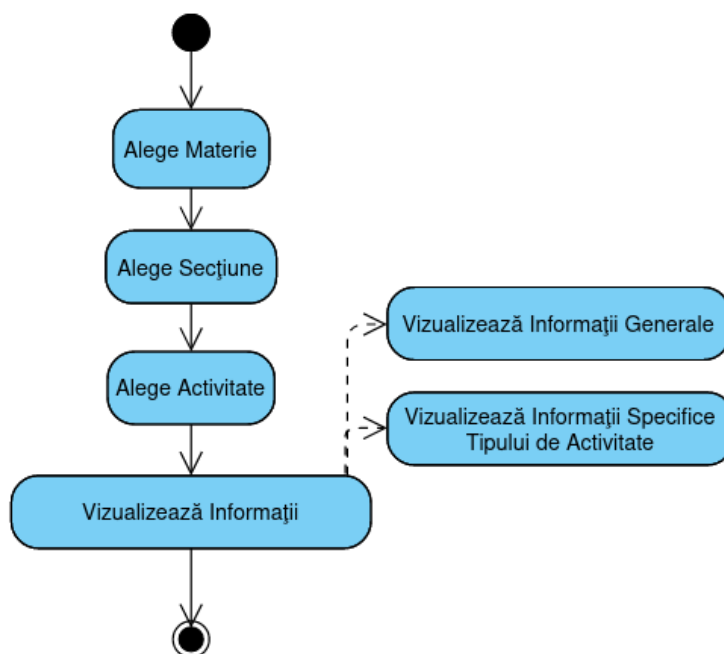


Figura 1.13: Diagramă Activitate - Accesează Activitate

1.3.2 Încărcarea rezolvărilor pentru teme

Pentru activitățile de tip temă, aceștia își vor putea încărca rezolvările și vizualiza nota, după ce tema a fost corectată și notată de către profesor.

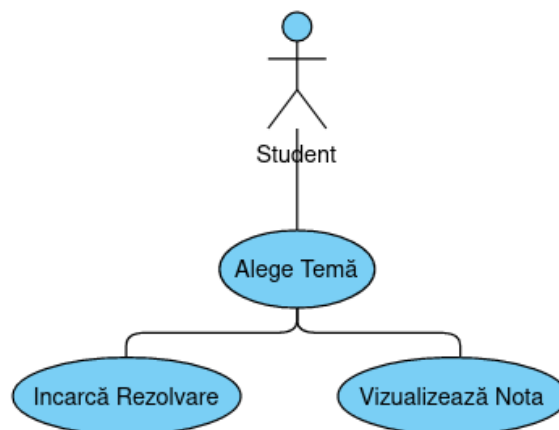


Figura 1.14: Diagramă Caz de utilizare - Încarcă Rezolvare Temă

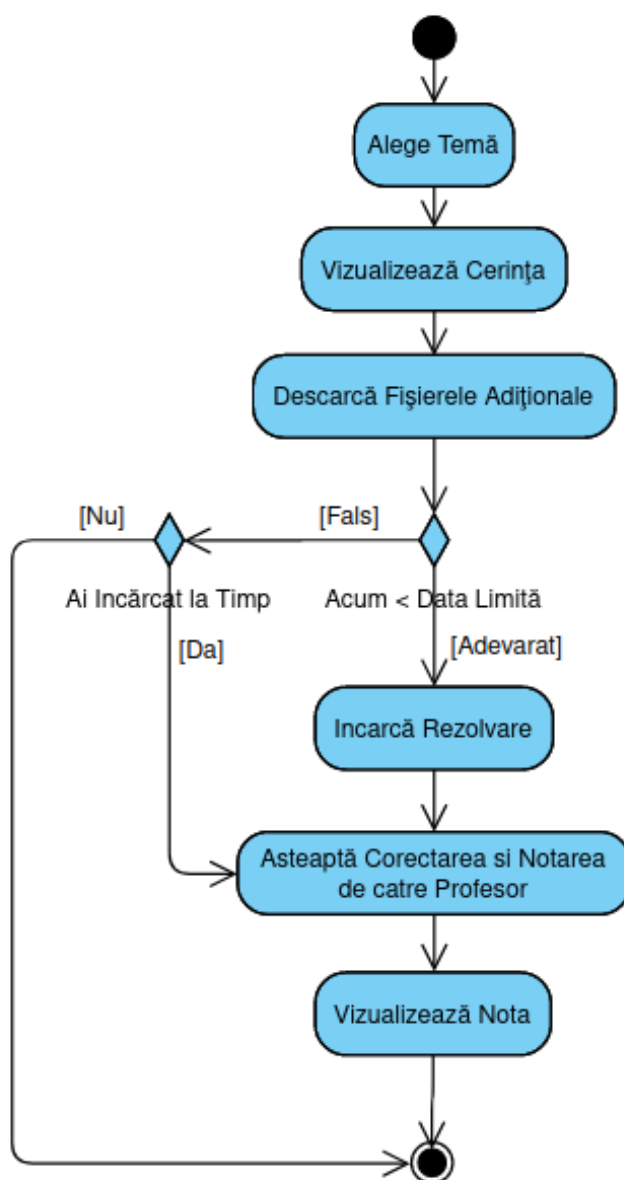


Figura 1.15: Diagramă Activitate - Încarcă Rezolvare Temă

1.3.3 Participarea la teste

Pentru activitățile de tip test, studenții vor putea interacționa în diferite moduri, în funcție de data curentă și detaliile testului. Vor avea posibilitatea de a începe o încercare sau a o continua, dacă testul este activ și verifica rezultatele întrebărilor, dacă testul a expirat.

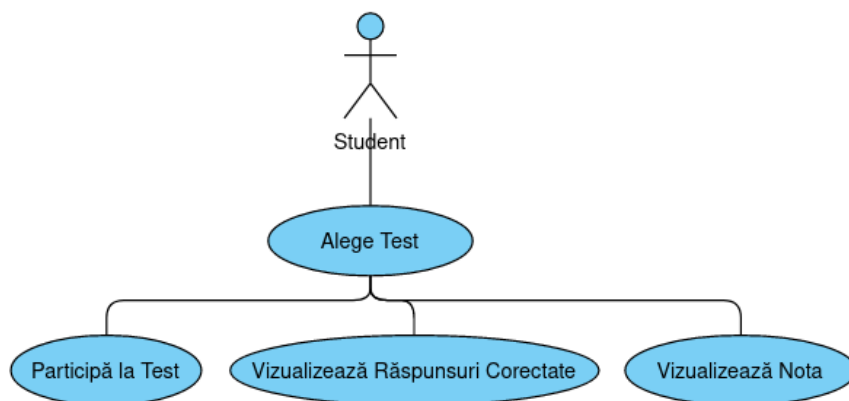


Figura 1.16: Diagramă Caz de utilizare - Participă la Test

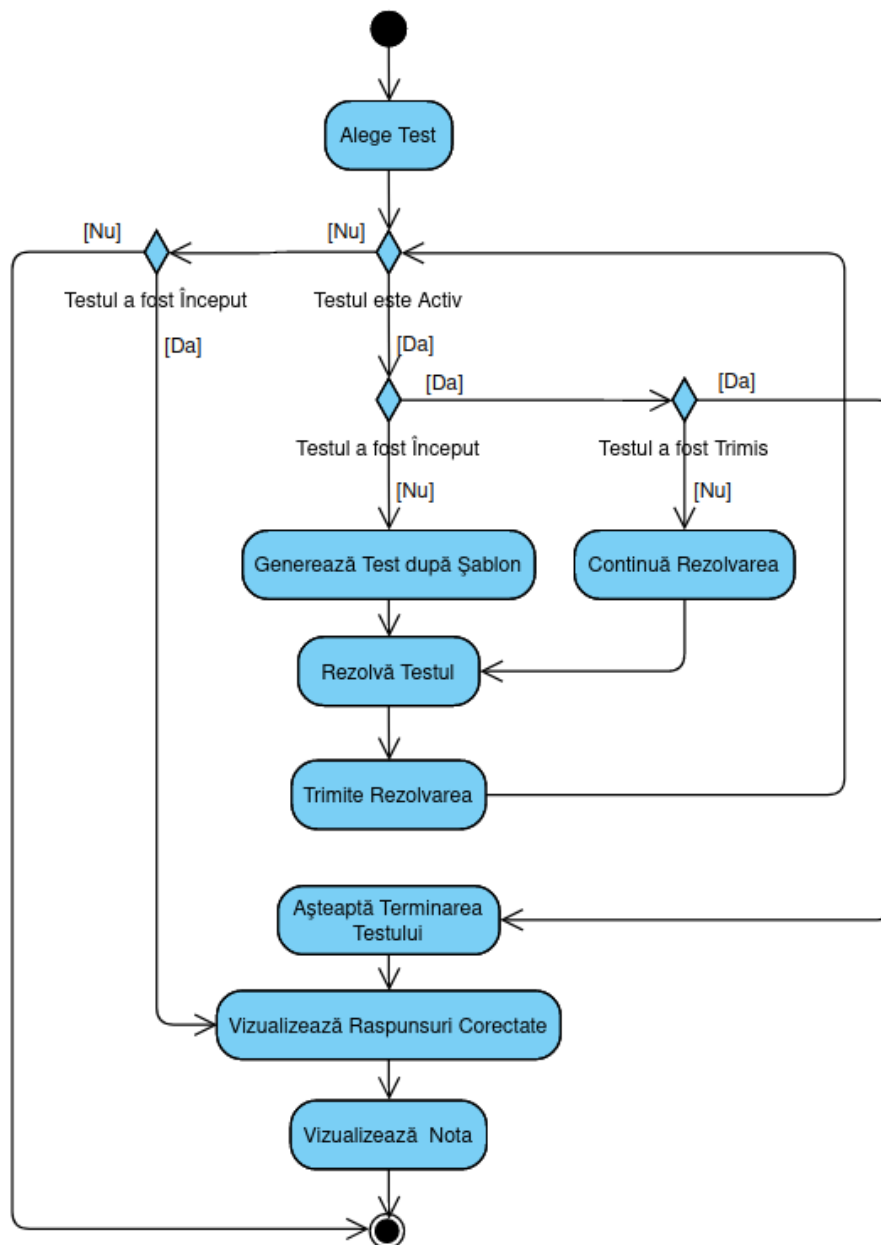


Figura 1.17: Diagramă Activitate - Participă la Test

1.4 Funcționalități comune

Funcționalitățile comune sunt următoarele:

- Participarea la discuții pe forum

1.4.1 Participarea la discuții pe forum

Indiferent de rolul pe care un utilizator îl are în universitate, acesta va putea participa la discuții pe forum. Cu toate acestea, doar administratorii sau profesorii pot crea discuții noi.

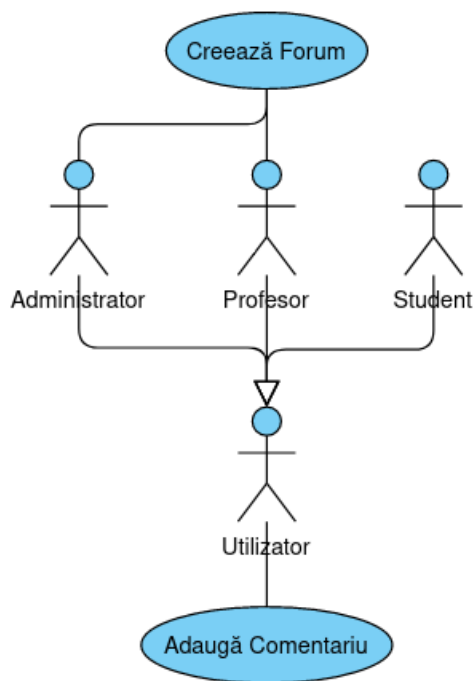


Figura 1.18: Diagramă Caz de utilizare - Participă la Forum

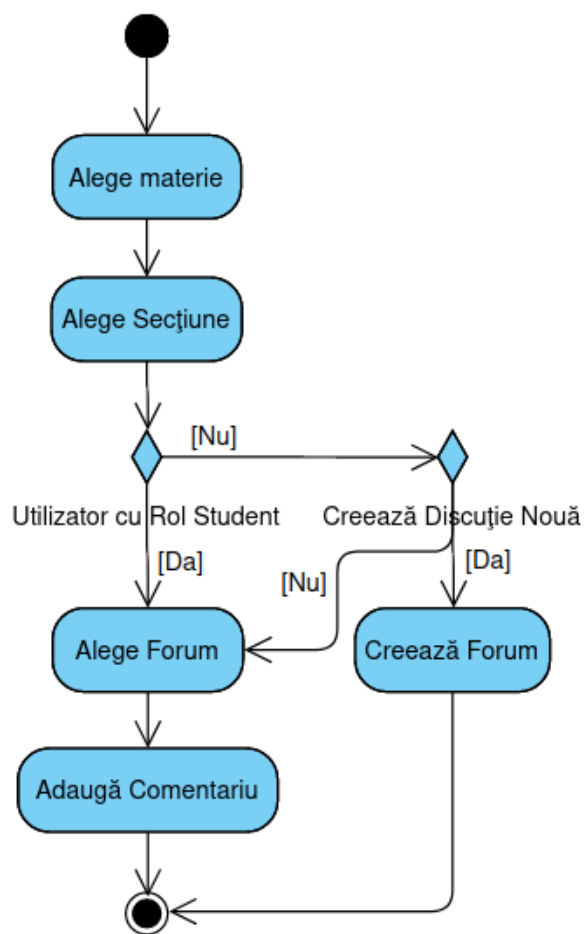


Figura 1.19: Diagramă Activitate - Participă la Forum

Capitolul 2

Implementare

Fiind o aplicație web, am folosit arhitectura standard de construire a acestora ”client-server”. Aceasta împarte aplicația în două părți, fiecare având un rol separat, comunicând între ele prin conexiuni TCP. Partea de client se referă la interfața vizuală cu care interacționează un utilizator în momentul folosirii, iar partea de server se referă la codul care rulează pe un calculator, de obicei în cloud¹, care răspunde cererilor primite de la client, cel din urma stocând datele necesare rulării aplicației, de cele mai multe ori într-o bază de date.[3]

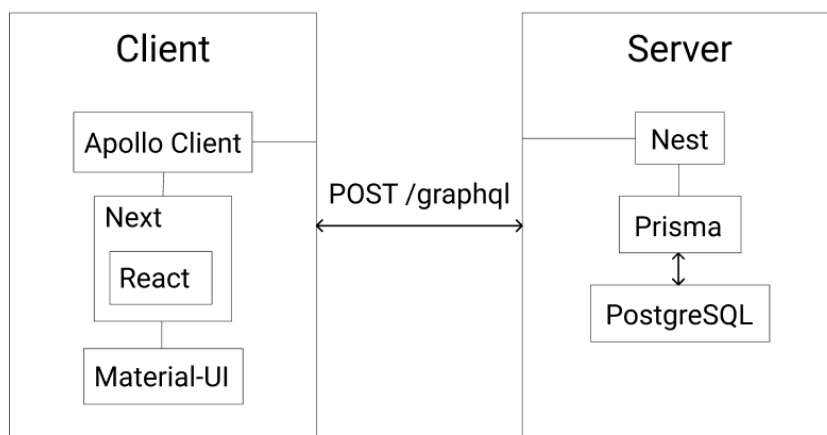


Figura 2.1: Arhitectură Client-Server

Întreaga aplicație este construită folosind limbajul TypeScript, atât pe partea de client, cât și pe partea de server, acesta fiind un superset² al limbajului JavaScript, care ne oferă un sistem sigur pentru tipuri de date.[4]

Tehnologiile pe care le-am folosit pe partea de client sunt:

- React[5] - Librărie de JavaScript pentru construirea interfețelor vizuale complexe
- Next[6] - Framework³ de React, care ne oferă o mulțime de unelte de optimizare
- Material-UI[7] - Librărie de componente UI de React, bazată pe design-ul celor de la Google(Material Design)
- Apollo Client[8] - Librărie de React, care ne ajută la folosirea unui API de tip GraphQL

¹Ansamblu distribuit de servicii de calcul

²Un limbaj de programare care conține toate caracteristicile unui limbaj dat și a fost extins sau îmbunătățit pentru a include și alte caracteristici

³Un produs software care oferă funcționalități generice pentru construirea mai facilă a aplicațiilor

Tehnologiile pe care le-am folosit pe partea de server sunt:

- Node[9] - Mediu de execuție pentru JavaScript, care ne va ajuta la construirea unui server
- Nest[10] - Framework de JavaScript, care ne va ajuta la construirea unui API de tip GraphQL
- PostgreSQL[11] - SGBD open-source⁴
- Prisma[12] - ORM de JavaScript, care ne va ajuta în gestionarea bazei de date

În ultimii ani, datorită creșterii utilizării internetului pe dispozitivele mobile, dezvoltatorii de aplicații web au fost nevoiți să adopte o abordare și un design flexibil⁵ când vine vorba de interfață. Din acest considerent, aplicația va fi gândită și dezvoltată să poată fi folosită pe orice dimensiune a ecranului, dar în continuare vom prezenta funcționalitățile folosind un ecran mare, deoarece acesta va fi folosit cel mai des de către utilizatori.[13]

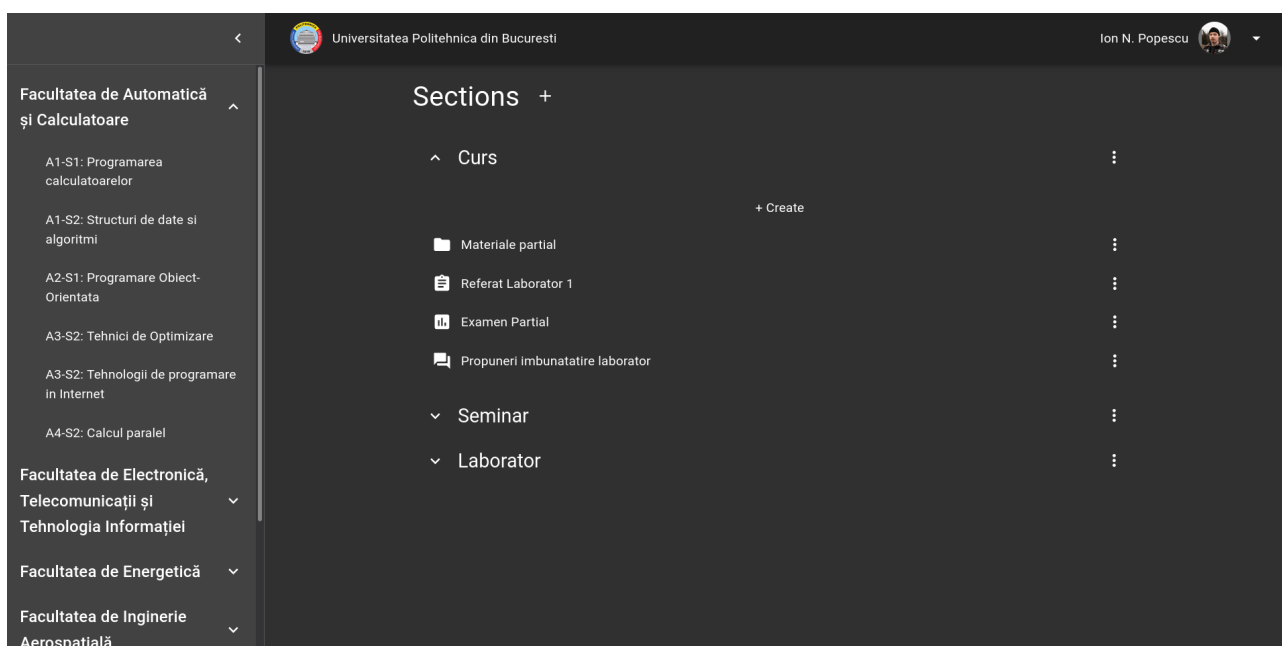


Figura 2.2: Exemplu design flexibil - Desktop

⁴Proiect software “public” – oricine are acces pentru a citi, adăuga sau modifica codul sursă, moderat de obicei de un grup restrâns

⁵Website-ul poate fi folosit atât de pe ecrane mari(ex. desktop), cât și de pe ecrane mici(ex. telefonul mobil)

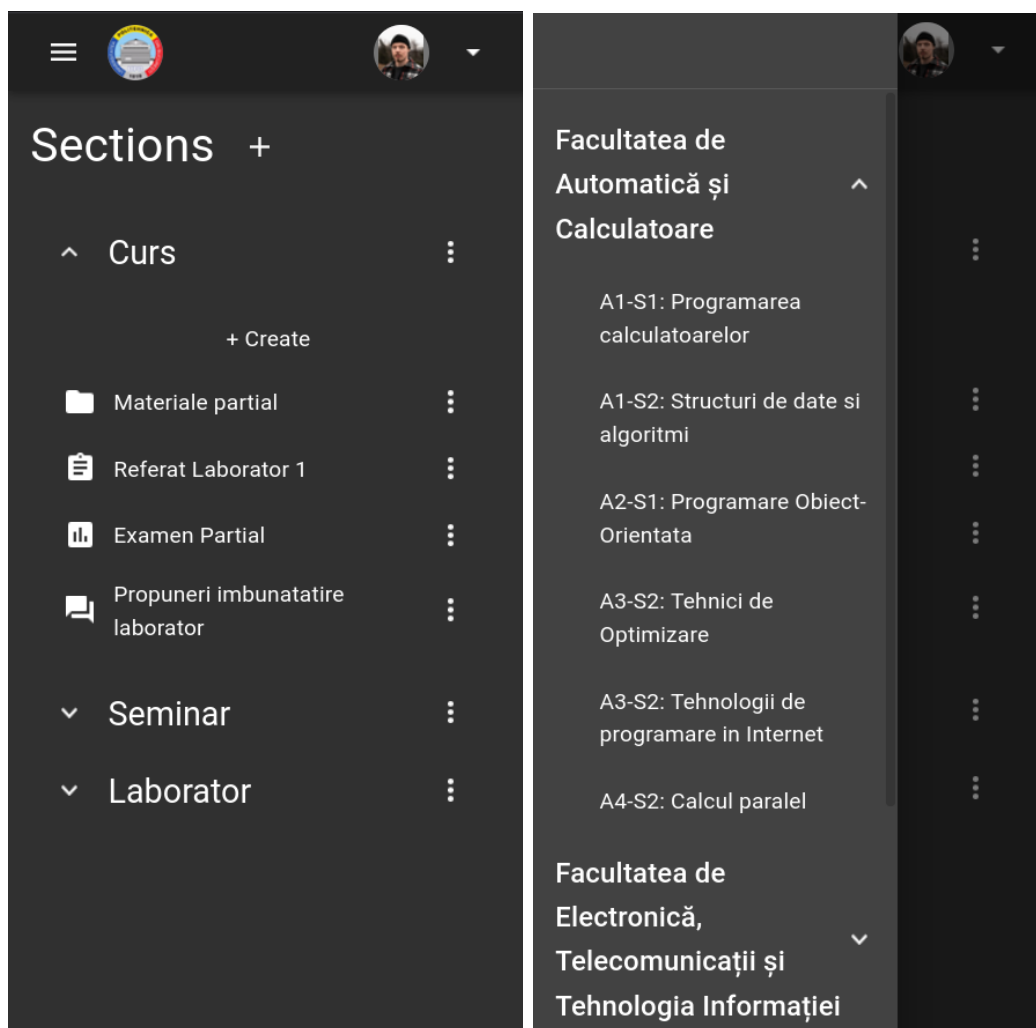


Figura 2.3: Exemplu design flexibil - Telefon Mobil

2.1 Modulul de Autentificare

Înainte ca utilizatorii să poată interacționa cu aplicația, aceștia vor trebui să-și creeze un cont, sau dacă au deja un cont să se autentifice, cu ajutorul modulului de autentificare. Acesta este compus dintr-o pagină cu 2 funcționalități Autentificare și Înregistrare.

Figura 2.4: Formular Autentificare, Anexă: 1.2, 1.4

The image shows a web interface with two tabs: 'LOGIN' and 'REGISTER'. The 'REGISTER' tab is selected. The form contains the following fields: 'First Name', 'Last Name', 'Father's Initial', 'Email', and 'Password'. Below these is an 'Avatar (optional)' section with a note 'Recommended image size: 40x40px' and a dashed box containing a paperclip icon and the text 'Click to add or drop files here'. At the bottom right is a blue button labeled 'REGISTER'.

Figura 2.5: Formular Înregistrare, Anexă: 1.3, 1.5

Pentru ca un utilizator să se poate înregistra, acesta va trebui să completeze următoarele câmpuri:

- nume de familie
- prenume
- inițiala tatălui
- email - unic printre toți utilizatorii
- parola - mai lungă de 6 caractere
- avatar(opțional)

Odată înregistrat, acesta se va putea autentifica folosindu-și email-ul și parola.

Aceste date, împreună cu orice alt formular din aplicație, vor fi validate de 2 ori, 1 dată pe partea de client și 1 dată pe partea de server. Validarea pe partea de client o vom face din motive de UX, utilizatorul va ști dacă datele au un format valid și respecta anumite reguli de validare imediat după ce termină un anumit câmp de completat. Validarea pe partea de server o vom face din 2 motive, în primul rând pe partea de client nu se pot face validările care țin cont de date din baza de date, spre exemplu nu putem verifica dacă email-ul introdus este unic, iar în al doilea rând, reprezintă o metodă de protecție împotriva celor cu intenții rele, care nu folosesc aplicația, ci interacționează manual cu API-ul acesteia.[14]

Dacă oricare dintre cele 2 validări găsesc erori, acestea vor fi afișate utilizatorului exact la câmpurile corespunzătoare.1.1

The image shows two side-by-side web forms for login and registration. Both forms have tabs for 'LOGIN' and 'REGISTER'. The left form (Client-side validation) shows an error for the email field: 'invalid email format' and 'You must enter a valid email address'. The right form (Server-side validation) shows an error for the email field: 'There is no user registered with this email'. Both forms have a password field with masked characters and a 'LOGIN' button.

Figura 2.6: Exemplu Eroare Validare - Client(stânga), Server(dreapta)

2.2 Funcționalități administrator

Odată autentificat, utilizatorul va fi întâmpinat cu un tablou de bord, de unde își poate alege una din universitățile la care este înrolat și dorește să participe, aceasta va fi funcționalitatea cea mai des folosită, sau chiar își poate crea o universitate la care va participa cu rol de administrator, funcționalitate destinată conturilor de administrator de universitate.

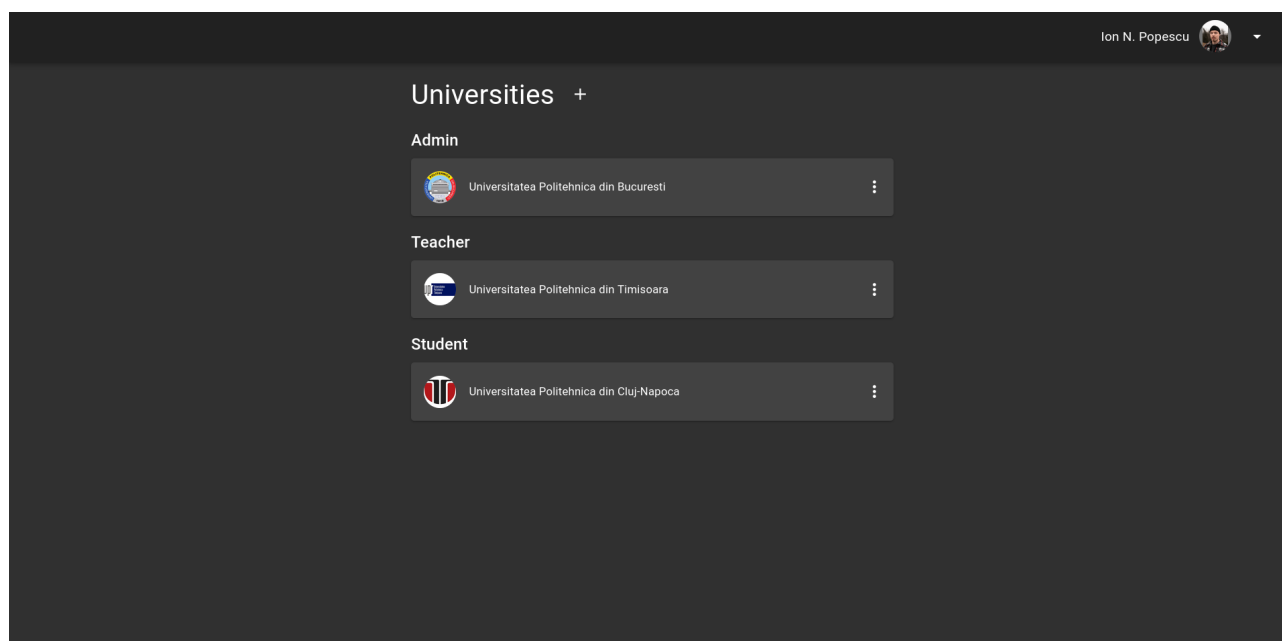


Figura 2.7: Tablou de bord Utilizator

Acest tablou este format din 2 componente, o bară de instrumente și lista cu universități.

Bara de instrumente este formată din 2 elemente: 1 buton cu datele utilizatorului, care va redirecționa spre acest tablou de bord, și o iconiță de meniu, care va conține un buton pentru deconectare.



Figura 2.8: Tablou de bord Utilizator - Bară de instrumente

Lista cu universități va afișa toate universitățile la care utilizatorul este înrolat, grupate după rolul pe care acesta îl deține în cadrul universității. Iconița de adăugat din dreapta titlului va deschide formularul de creare a unei noi universități, având câmpurile nume și siglă (opțional). Fiecare universitate este formată dintr-un buton, care va redirecționa spre tabloul de bord al universității, și dintr-un buton care va deschide un meniu special. În funcție de rolul pe care utilizatorul îl deține în cadrul universității, acest meniu va conține funcționalități diferite.

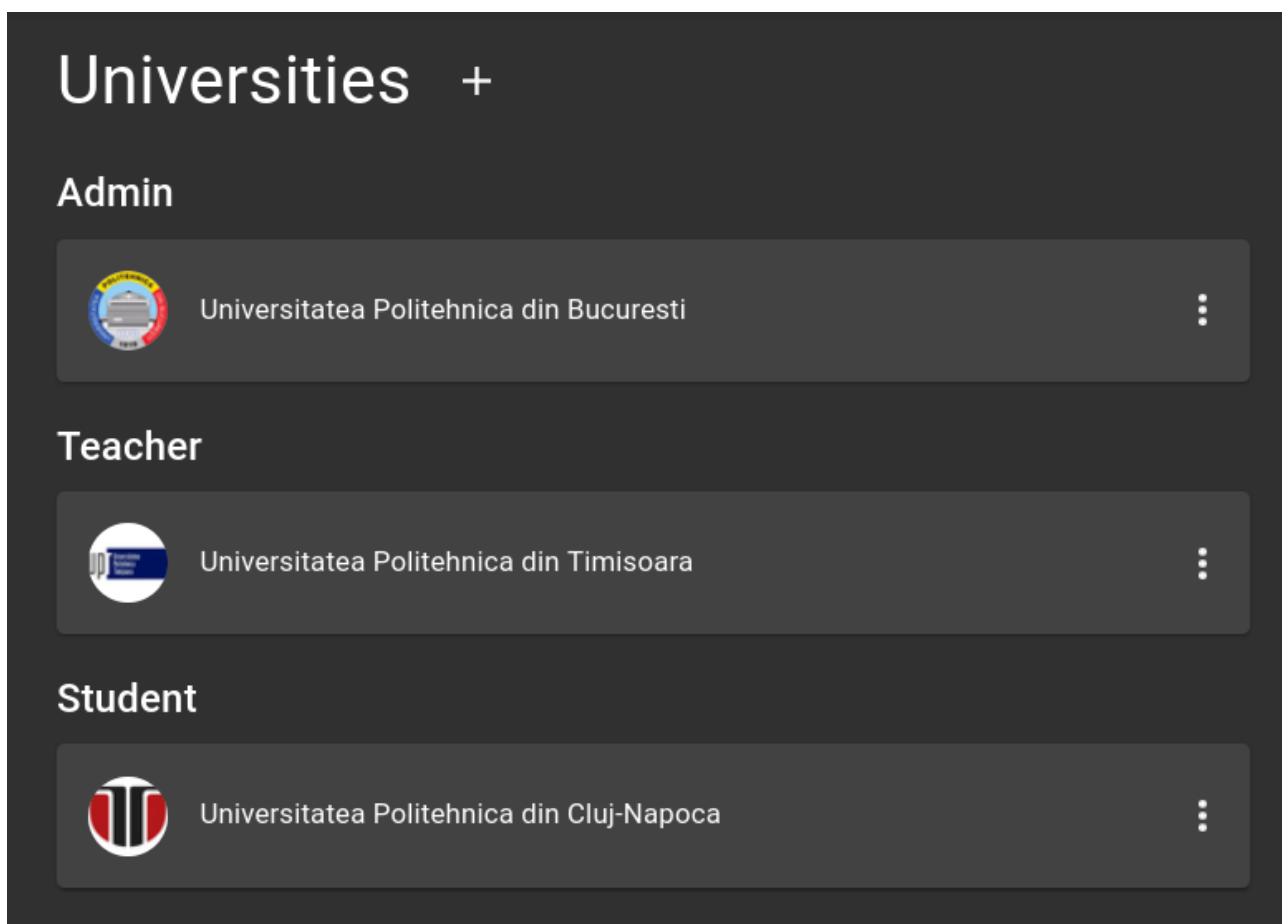


Figura 2.9: Tablou de bord Utilizator - Listă Universități, Anexă: 1.6

Create University

Name

Logo (optional)
Recommended image size: 40x40px

Click to add or drop files here

Files: 0/1
Accepted file extensions: .jpg, .jpeg, .gif, .png, .tiff, .bmp
Maximum file size: 10 MB

CREATE

Figura 2.10: Tablou de bord Utilizator - Formular Creare Universitate

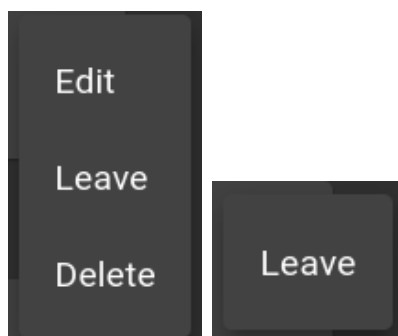


Figura 2.11: Tablou de bord Utilizator - Meniu Universitate Administrator(stânga), Profesor/Student(dreapta)

Butonul editare va deschide un formular de editare al universității, cel de părăsire va scoate utilizatorul din lista universității, iar cel de ștergere va șterge universitatea și orice date asociate. În cazul butoanelor de părăsire și ștergere, înainte să se execute comanda, utilizatorul va fi întrebat dacă este sigur ca operația este cea dorită.

Figura 2.12: Tablou de bord Utilizator - Formular Editare Universitate

Figura 2.13: Tablou de bord Utilizator - Întrebare siguranță ștergere(stânga), părăsire(dreapta)

2.2.1 Structura Universității

Odată selectată o universitate, utilizatorul va fi redirecționat spre tabloul de bord al universității. Aici vor fi introduse noi funcționalități în bara de instrumente, în partea de stânga vor apărea un buton cu detaliile universității, care va redirecționa spre tabloul de bord al universității, și o iconiță de meniu care poate fi comutată, pentru a ține meniul de navigare rapidă deschis sau pentru a-l închide. Tot aici, meniul utilizatorului va avea introduse butoane noi în funcție de rolul pe care îl deține în cadrul universității:

- Student
 - Note - vizualizarea rapidă a tuturor notelor grupate
- Profesor
 - Set Personal de Întrebări
- Administrator
 - Utilizatori - managementul utilizatorilor
- Comun

- Activități Viitoare - Vizualizarea rapidă a următoarelor activități în ordine cronologică



Figura 2.14: Tablou de bord Universitate - Bară de Instrumente1.7

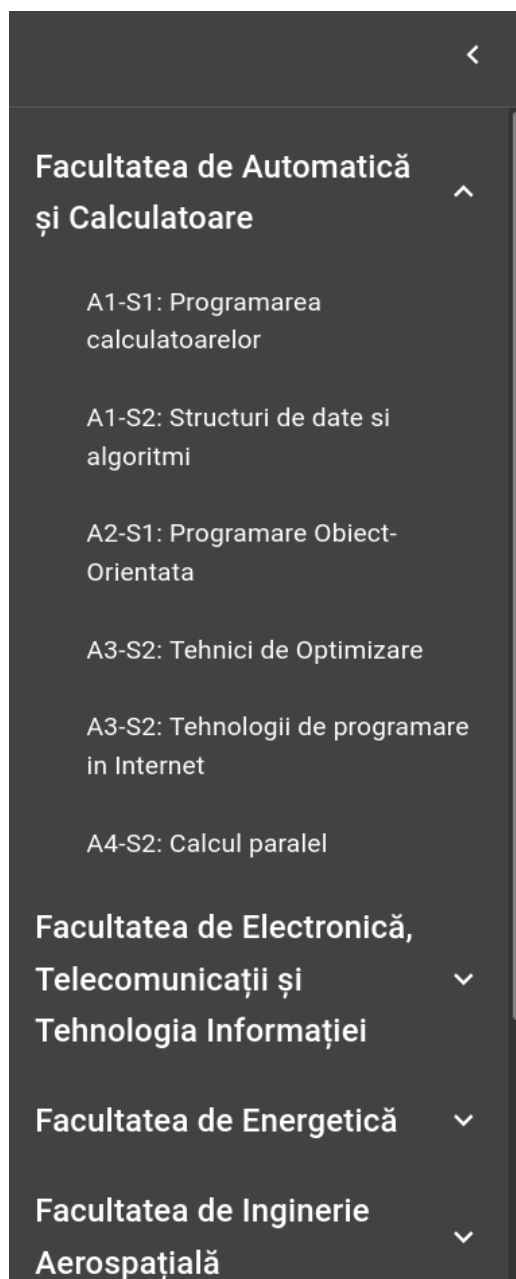


Figura 2.15: Tablou de bord Universitate - Meniu navigare rapidă

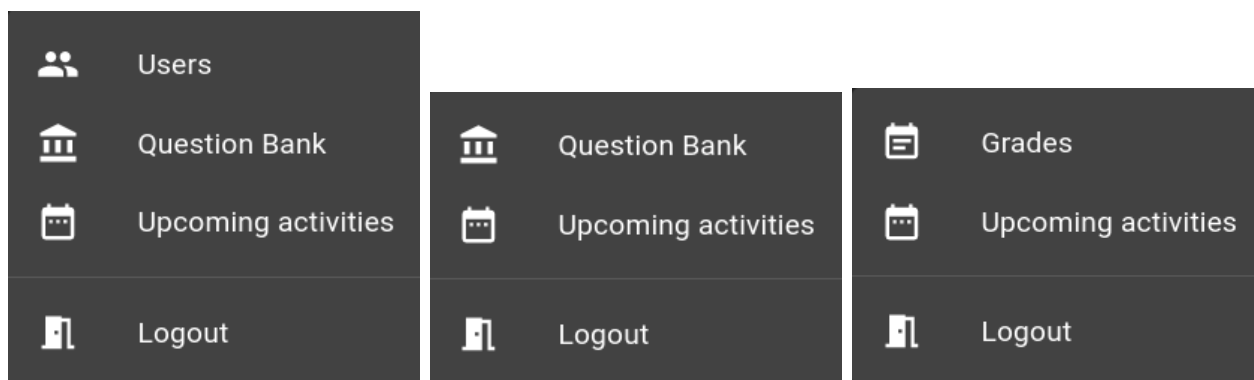


Figura 2.16: Tablou de bord Universitate - Meniu Utilizator Administrator(stânga), Profesor(mijloc), Student(dreapta)1.8

Ca și conținut, va fi afișată lista de facultăți din cadrul universității ca și secțiune colapsabilă, în care va fi afișată lista de materii din cadrul respectivei facultăți. Dacă utilizatorul deține rolul de administrator în cadrul universității, vor apărea butoane de adăugare, editare sau ștergere a facultăților și materiilor.

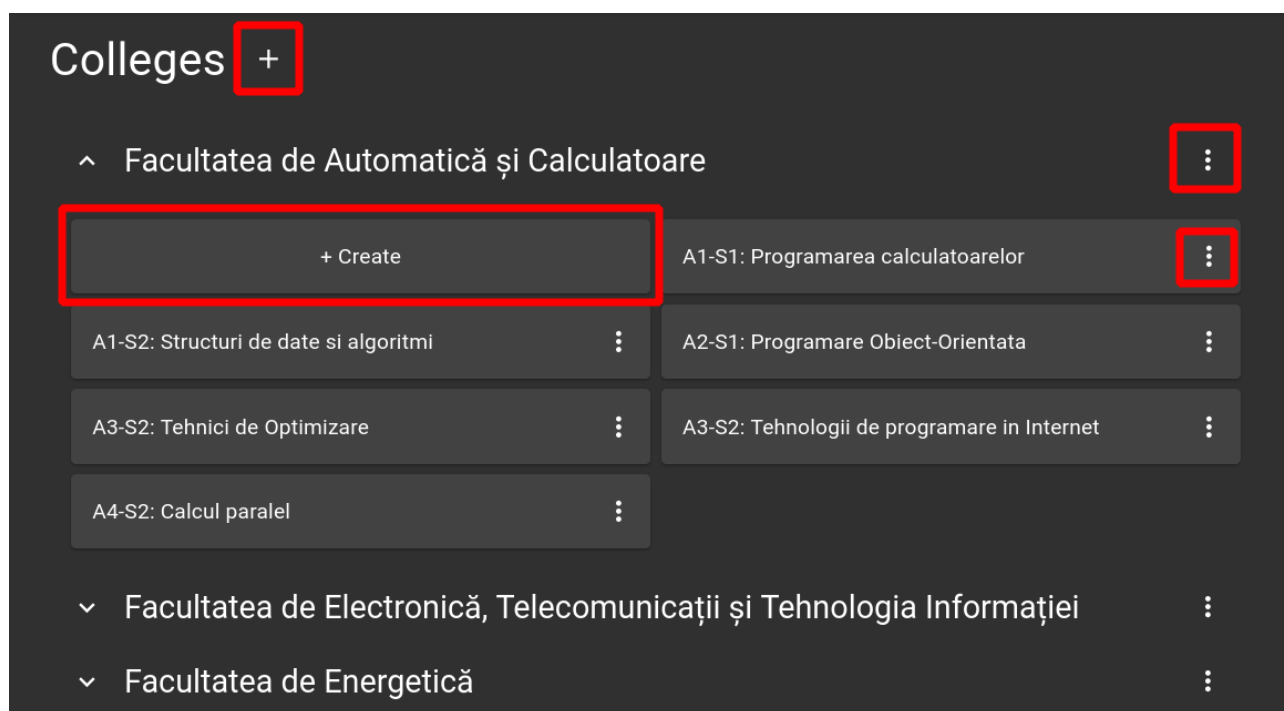


Figura 2.17: Tablou de bord Universitate - Listă facultăți Administrator1.9

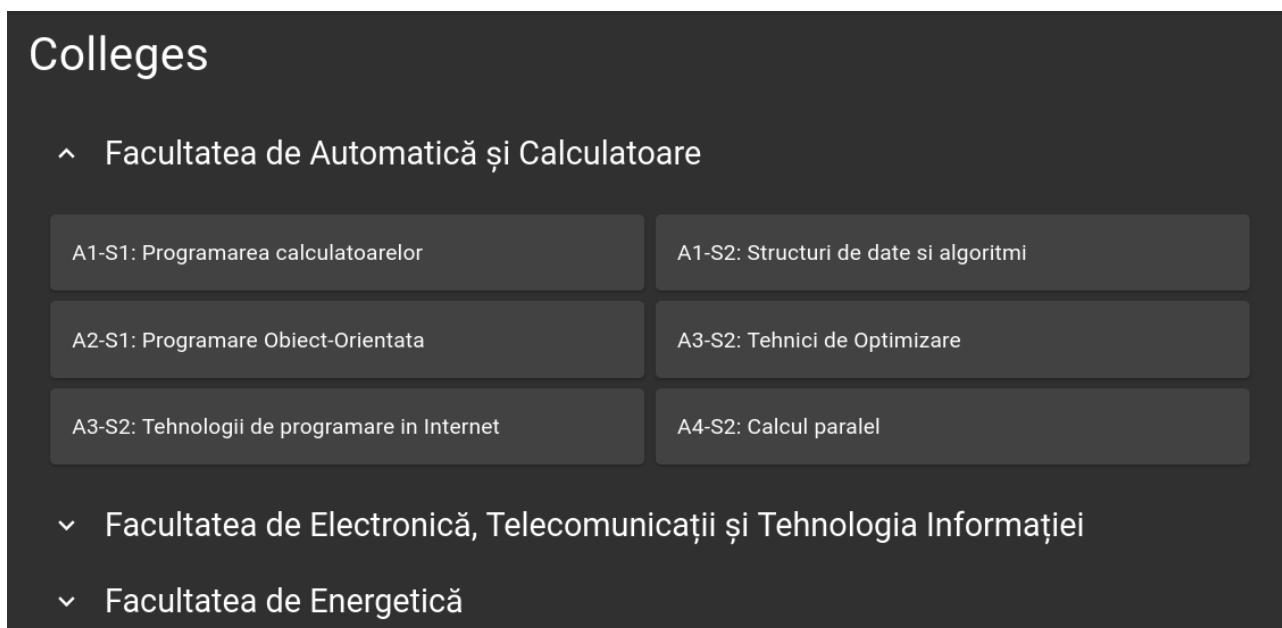


Figura 2.18: Tablou de bord Universitate - Listă facultăți Student

Create College

Name

CREATE

Update College

Name

Facultatea de Automatică și Calculatoare

UPDATE

Figura 2.19: Tablou de bord Universitate - Formular creare(stânga), editare(dreapta) Facultate

Create Course

Name

CREATE

Update Course

Name

A1-S1: Programarea calculatoarelor

UPDATE

Figura 2.20: Tablou de bord Universitate - Formular creare(stânga), editare(dreapta) Materie

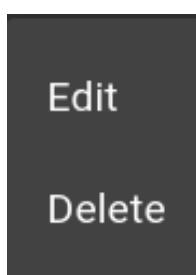
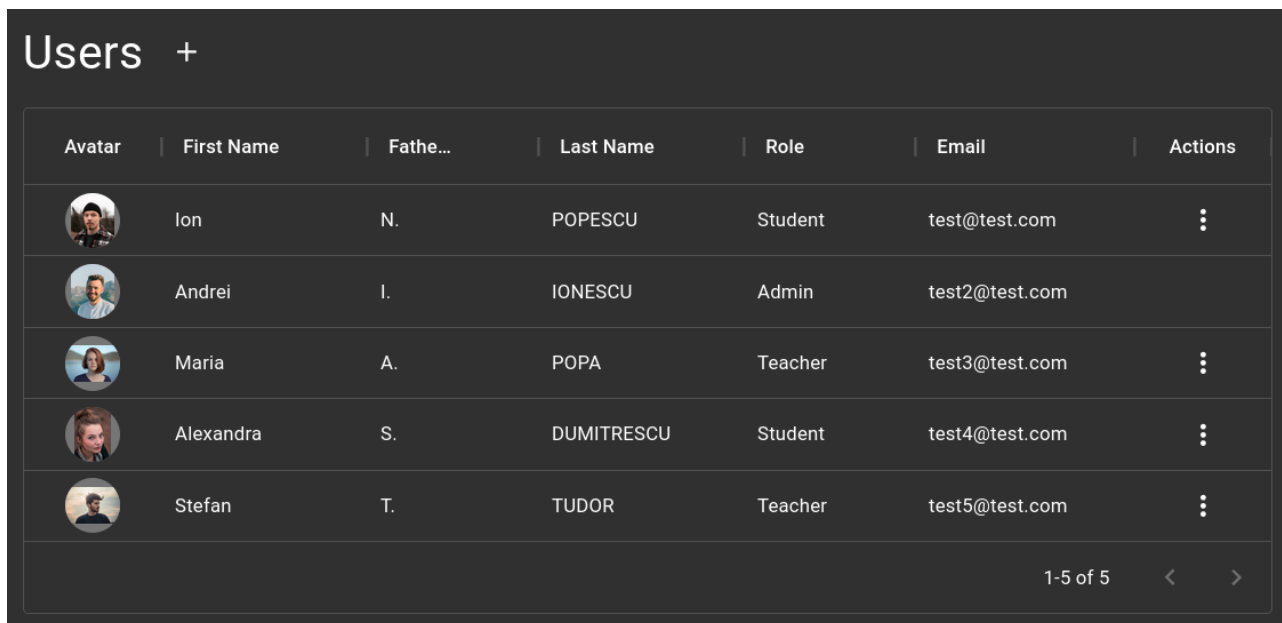











Figura 2.21: Tablou de bord Universitate - Meniu Facultate/Materie

2.2.2 Managementul utilizatorilor

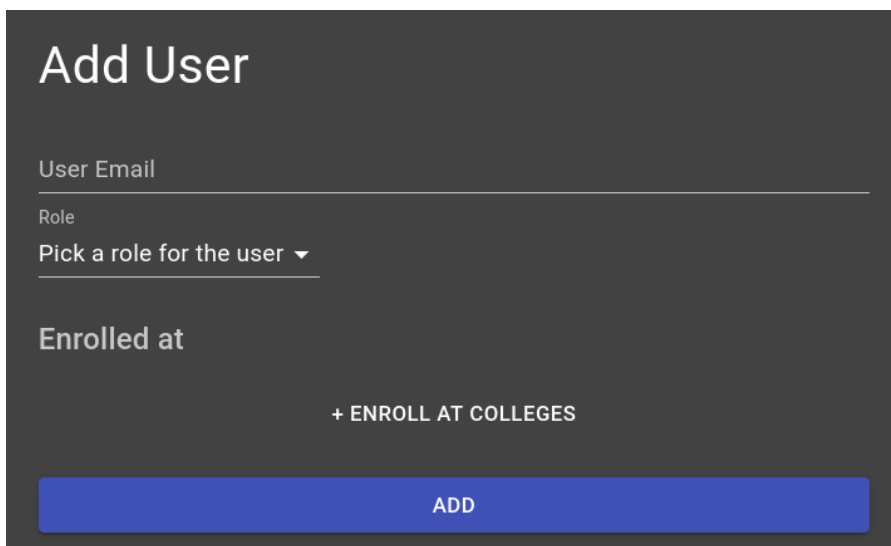
Odată apăsat butonul 'Utilizatori' din meniul dedicat administratorilor, utilizatorul va fi redirectionat către tabloul de bord al tuturor utilizatorilor din cadrul universității. Aici se vor putea adăuga noi utilizatori, sau modifica sau șterge utilizatorii deja existenți. Fiecare utilizator va avea un rol și va fi înrolat la anumite facultăți și materii. Un caz mai special îl reprezintă utilizatorii cu rol de administrator, care nu vor fi înrolați la anumite facultăți, aplicația îi va înrola automat la toate facultățile și materiile și îi va ține în această stare de fiecare dată când acestea se vor schimba. Totodată, administratorii nu vor avea posibilitatea ștergerii sau schimbării rolului altor administratori.



Avatar	First Name	Last Name	Role	Email	Actions
	Ion	POPESCU	Student	test@test.com	
	Andrei	IONESCU	Admin	test2@test.com	
	Maria	POPA	Teacher	test3@test.com	
	Alexandra	DUMITRESCU	Student	test4@test.com	
	Stefan	TUDOR	Teacher	test5@test.com	

1-5 of 5 < >

Figura 2.22: Tablou de bord Management Utilizatori1.10



Add User

User Email

Role

Pick a role for the user ▼

Enrolled at

+ ENROLL AT COLLEGES

ADD

Figura 2.23: Tablou de bord Management Utilizatori - Formular Adăugare Utilizator

Update User

Role
Student ▼

Enrolled at

Facultatea de Electronică, Telecomunicații și Tehnologia Informației ✕

Courses

- A1-S1: Programarea calculatoarelor
- A1-S2: Structuri de date si algoritmi
- A2-S1: Programare Obiect-Orientata ▼
- A3-S2: Tehnici de Optimizare
- A3-S2: Tehnologii de programare in Internet
- A4-S2: Calcul paralel

Facultatea de Automatică și Calculatoare ✕

Courses

- A1-S1: Programarea calculatoarelor
- A1-S2: Structuri de date si algoritmi
- A2-S1: Programare Obiect-Orientata ▼
- A3-S2: Tehnici de Optimizare

Figura 2.24: Tablou de bord Management Utilizatori - Formular Editare Utilizator

Pick a role for the user

- Admin
- Teacher
- Student

Figura 2.25: Tablou de bord Management Utilizatori - Alegere Rol Utilizator

Colleges

<input type="checkbox"/>	Name
<input type="checkbox"/>	Facultatea de Automatică și Calculatoare
<input checked="" type="checkbox"/>	Facultatea de Electronică, Telecomunicații și Tehnologia Informației
<input checked="" type="checkbox"/>	Facultatea de Energetică
<input type="checkbox"/>	Facultatea de Inginerie Aerospațială
<input type="checkbox"/>	Facultatea de Inginerie Electrică
<input type="checkbox"/>	Facultatea de Inginerie Mecanică și Mecatronică

2 rows selected 1-6 of 6 < >

ADD

Figura 2.26: Tablou de bord Management Utilizatori - Formular Înrolare la Facultăți

A1-S1: Programarea calculatoarelor
A1-S2: Structuri de date si algoritmi
A2-S1: Programare Obiect-Orientata
A3-S2: Tehnici de Optimizare
A3-S2: Tehnologii de programare in Internet
A4-S2: Calcul paralel

Figura 2.27: Tablou de bord Management Utilizatori - Formular Înrolare la Materii

2.3 Funcționalități profesor

Odată selectată o materie, utilizatorul va fi redirecționat spre tabloul de bord al materiei. Aici bara de instrumente își păstrează aceleași funcționalități, iar conținutul va fi reprezentat de o listă de secțiuni, fiecare secțiune fiind colapsabilă și conținând activități. Profesorul va

putea crea atât secțiuni, pentru o mai bună repartizare logică a activităților, cât și activități. Crearea, editarea și ștergerea secțiunilor au exact același format ca cel al facultăților.

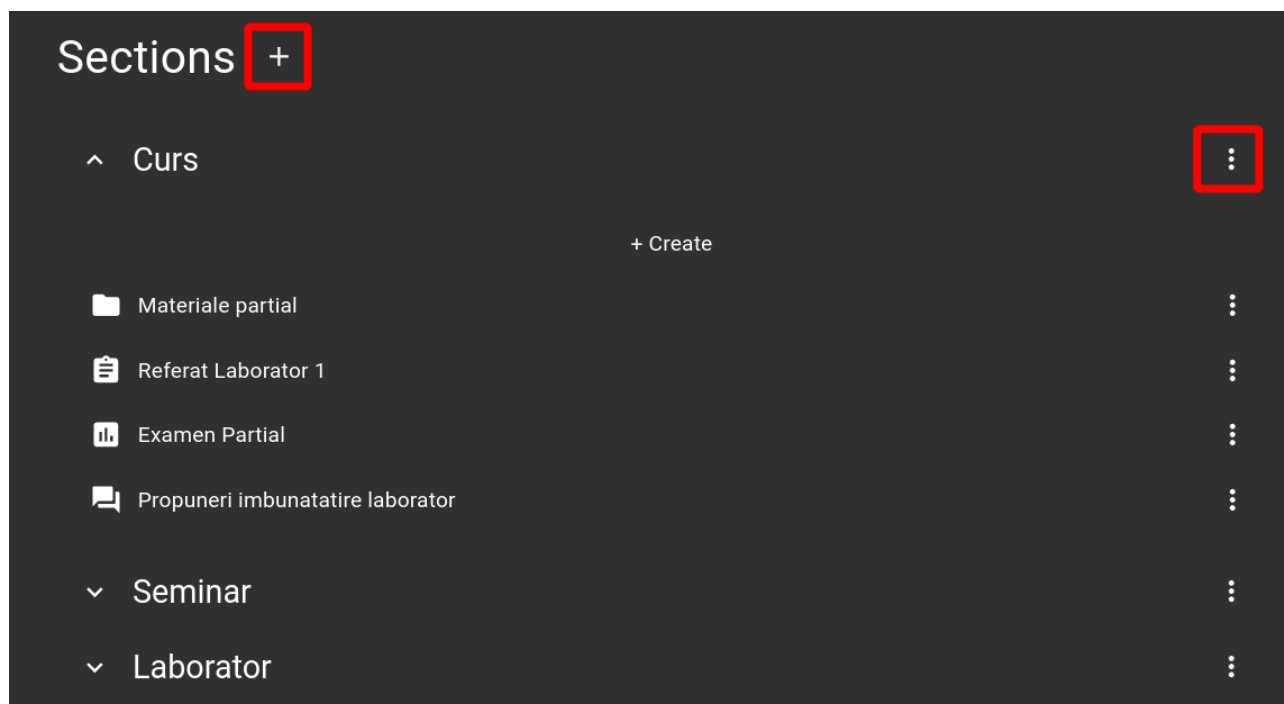


Figura 2.28: Tablou de bord Materie - Listă secțiuni Profesor

The image shows two side-by-side forms on a dark background. The left form is titled 'Create Section' in white. It has a white input field labeled 'Name' and a blue button labeled 'CREATE'. The right form is titled 'Update Section' in white. It has two white input fields, one labeled 'Name' and one labeled 'Curs', and a blue button labeled 'UPDATE'.

Figura 2.29: Tablou de bord Materie - Formular creare(stânga), editare(dreapta) Secțiune

2.3.1 Crearea activităților

Pentru crearea și editarea activităților, formularul se va schimba în funcție de tipul de activitate dorit.

Pentru tipurile materiale și forum, acestea nu vor conține nimic în plus față de formularul de bază al unei activități. Orice activitate are următoarele câmpuri de completat:

- Nume - numele activității
- Descriere(optional) - descriere sau textul activității
- Fișiere(optional) - fișiere ajutătoare pentru parcurgerea activității de către student

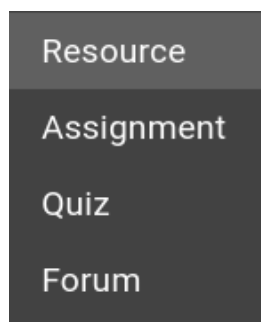


Figura 2.30: Tablou de bord Materie - Meniu alegere tip de Activitate

Create Activity

Type: **Resource** ▼

Name: _____

Description (optional): _____

Files (optional):

Click to add or drop files here

CREATE

Update Resource

Name: **Materiale partial**

Description (optional): M-am gandit ca v-ar ajuta sa grupez si sa ordonez toate materialele de care veti avea nevoie pentru partialul care va urma. Mai jos veti gasi toate cursurile si seminarile incarcate pana acum + cateva propuneri de exercitii.

Files (optional):

Click to add or drop files here

- Curs1.pdf ×
- Curs2.pdf ×
- Curs3.pdf ×
- Curs4.docx ×
- Curs5.docx ×

UPDATE

Figura 2.31: Tablou de bord Materie - Formular creare(stânga), editare(dreapta) Material/Forum

Pentru tipul temă, se vor mai adăuga 2 câmpuri:

- Nota maximă - nota maximă pe care o poate obține un student pentru realizarea temei
- Termen limită - data și ora până când tema poate fi încărcată

Nota maximă trebuie să fie mai mare sau egal cu 0, iar termenul limită trebuie să reprezinte o dată mai mare decât data curentă.

Max Grade

0

Deadline

August 23rd 17:37

Figura 2.32: Tablou de bord Materie - Câmpuri în plus pentru tipul Temă

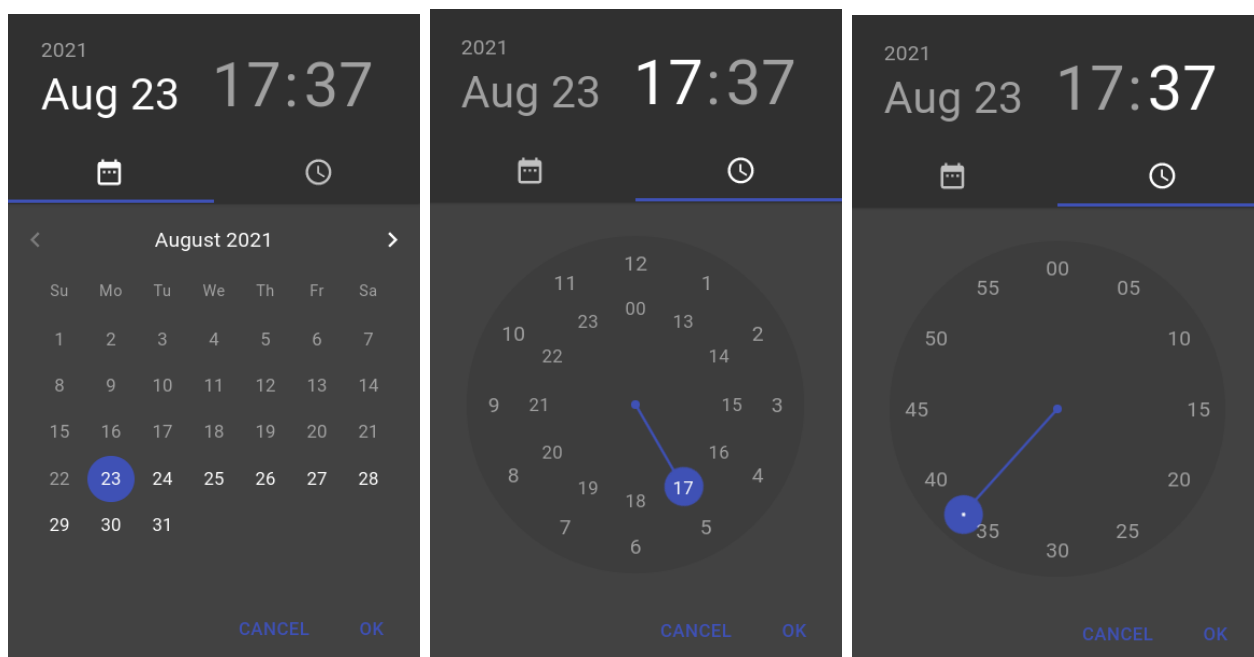


Figura 2.33: Tablou de bord Materie - Câmp alegere Timp - dată(stânga), oră(mijloc), minut(dreapta)

Pentru tipul test, se vor mai adăuga 6 câmpuri:

- Timp deschidere - timp de deschidere al testului
- Timp închidere - timp de închidere al testului
- Timp limită(minute) - timpul limită maxim admis
- Amestecare întrebări - amestecă aleator ordinea întrebărilor
- Amestecare răspunsuri - amestecă aleator ordinea răspunsurilor fiecărei întrebări
- Întrebări - setul de întrebări pentru generarea testelor studenților, iar pentru fiecare întrebare, vom avea 2 câmpuri
 - Ordinea - ordinea în lista de întrebări
 - Nota maximă - nota maximă pe care o poate obține un student pentru răspunsul corect la întrebare

Timpul de deschidere și cel de închidere trebuie să reprezinte o dată mai mare decât data curentă, timpul de închidere trebuie să fie după timpul de deschidere, timpul limită trebuie să fie mai mare decât 0, iar pentru fiecare întrebare în parte, nota maximă trebuie să fie mai mare sau egală cu 0.

Time Open
August 23rd 17:37

Time Close
August 24th 17:37

Time Limit (minutes)
90

Shuffle Questions ☒

Shuffle Answers ☒

Questions

= Question 1 ✕

☒ Mostenirea claselor Max Grade 5

= Question 2 ✕

☒ Structuri de date Max Grade 10

+ ADD QUESTIONS

Figura 2.34: Tablou de bord Materie - Câmpuri în plus pentru tipul Test

Questions

<input checked="" type="checkbox"/>	Category	Type	Name
<input checked="" type="checkbox"/>	Programare Obiect-Orientata	Single Choice	Mostenirea claselor
<input checked="" type="checkbox"/>	Programare Obiect-Orientata	Multiple Choice	Structuri de date
<input type="checkbox"/>	Structuri de date si algoritmi	Single Choice	Mostenirea claselor
<input type="checkbox"/>	Structuri de date si algoritmi	Multiple Choice	Structuri de date

2 rows selected 1-4 of 4 < >

ADD

Figura 2.35: Tablou de bord Materie - Formular adăugare întrebări

2.3.2 Accesarea activităților

Odată cu selectarea unei activități, utilizatorul va fi redirecționat către un tablou de bord dedicat fiecărui tip de activitate. Orice activitate va avea afișat la început detaliile de bază: nume, descriere, fișiere, după care va continua cu detalii diferite în funcție de tip.

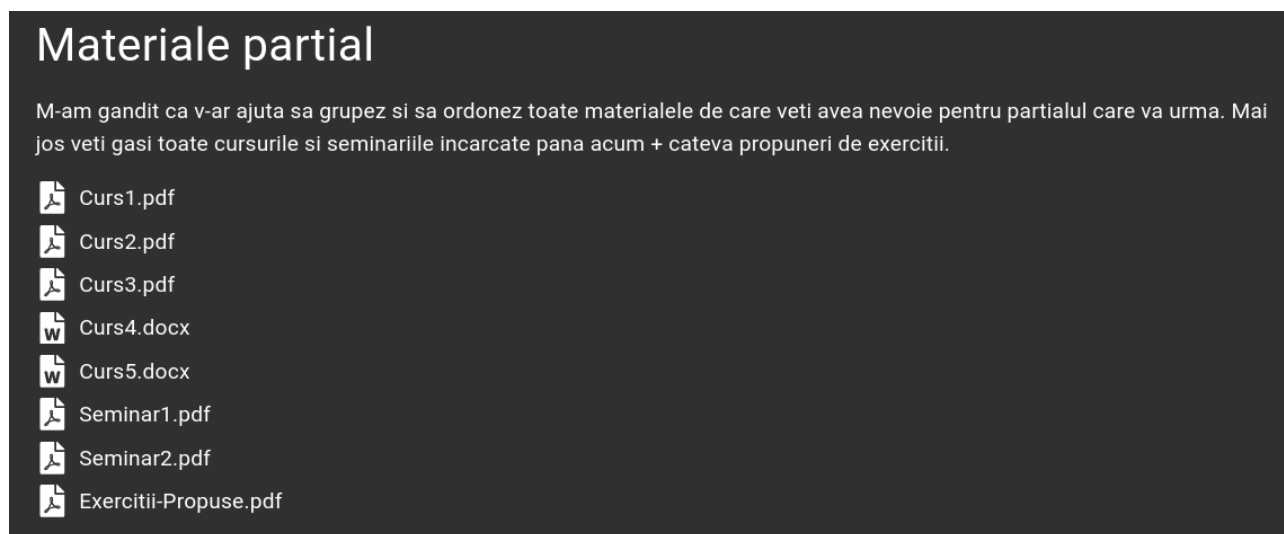


Figura 2.36: Tabloul de bord Activitate - Detalii de bază

În cazul temelor, se vor afișa cele 2 câmpuri în plus într-un tabel, Detalii Temă, iar sub tabel va fi lista cu studenții participanți, care au încărcat rezolvări la teme, împreună cu detaliile lor: avatar-ul, punctajul, numele și ultima dată de încărcare.

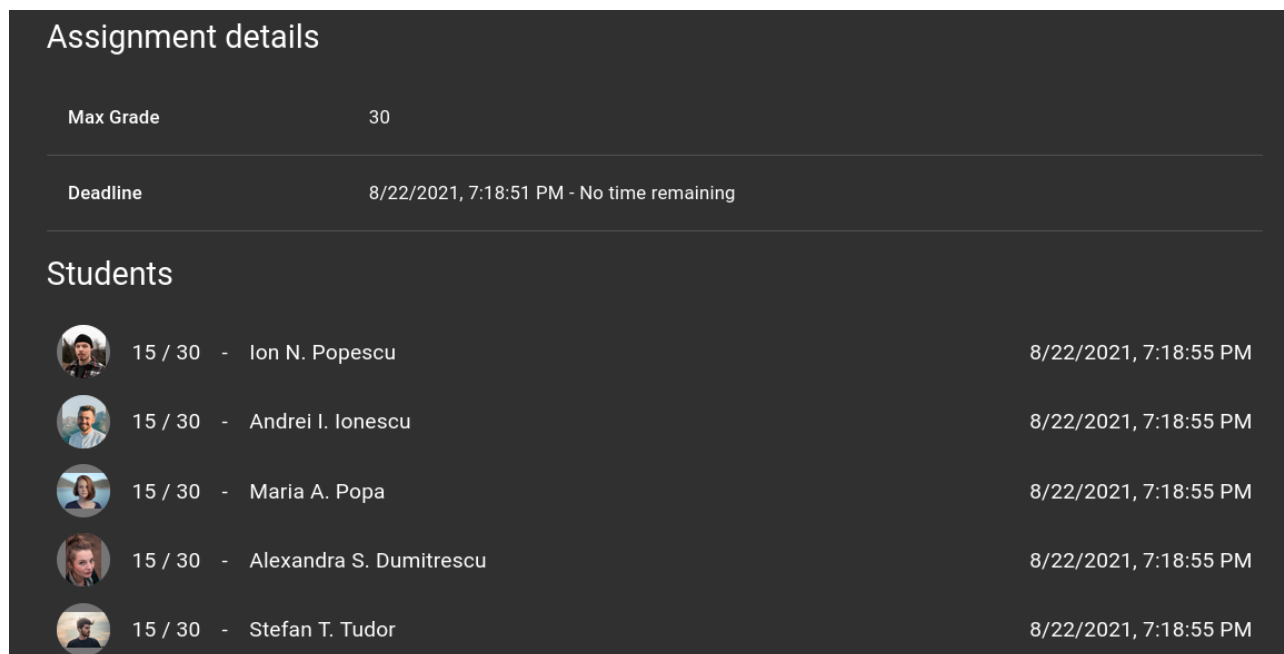


Figura 2.37: Tabloul de bord Activitate - Detalii specifice temelor

Odată apăsat unul dintre studenți, utilizatorul va fi redirecționat către un tablou de bord dedicat notării studentului, unde vor fi afișate datele despre temă, rezolvarea încărcată de acesta și un formular de notare în dreapta paginii.

Universitatea Politehnica din Timisoara

Ion N. Popescu

>

Referat Laborator 1

Referatul unei lucrari va fi alcatuit dintr-un singur fisier pdf numit astfel: "NumePrenume_grupa_TitluLucrariiPeScurt". Referatul va contine: pre-referatul (max 2 pagini, pe prima pagina apar urmatoarele: Numele, Prenumele, Grupa, Data, Titlul lucrarii), tabelul de date, graficele (daca este cazul), foi cu prelucrari si rezultatele finale. Mai jos veti gasi platforma de laborator 1.

Laborator1.pdf

Assignment status

Grade	15 / 30
Deadline	8/22/2021, 7:18:51 PM - No time remaining
Last time updated	8/22/2021, 7:18:55 PM - 1 day, 2 hours ago

Student uploads

IonPopescu_414A_Oscilatii-Mecanice.pdf

Alexandra S. Dumitrescu
(test4@test.com)

Grade
15

UPDATE

Figura 2.38: Tablou de bord - Notare temă Student

În cazul testelor, se vor afișa într-un tabel detaliile specifice testului, iar sub acesta vor exista 2 secțiuni, prima este cea care conține detalii despre întrebările care aparțin de test, aceasta fiind o secțiune colapsabilă, iar cea de-a doua este lista cu studenții care au participat la test.

Quiz details

Max Grade	145
Time Open	8/22/2021, 7:18:51 PM
Time Close	8/22/2021, 9:18:51 PM - No time remaining
Time Limit	1 hour, 30 minutes
Shuffle Questions	True
Shuffle Answers	True

Figura 2.39: Tablou de bord Activitate - Detalii specifice testelor

^ Questions			
Order	Max Grade	Type	Name
1	9	<input type="radio"/>	Mostenirea claselor
2	5	<input type="radio"/>	Mostenirea claselor
3	11	<input checked="" type="checkbox"/>	Structuri de date
4	4	<input checked="" type="checkbox"/>	Structuri de date
5	14	<input type="radio"/>	Mostenirea claselor
6	13	<input checked="" type="checkbox"/>	Structuri de date

Figura 2.40: Tablou de bord Activitate - Întrebări test





Students		
	Not graded yet / 116 - Andrei I. Ionescu	8/22/2021, 7:18:51 PM - Not finished yet
	Not graded yet / 116 - Maria A. Popa	8/22/2021, 7:18:51 PM - Not finished yet
	Not graded yet / 116 - Alexandra S. Dumitrescu	8/22/2021, 7:18:51 PM - Not finished yet
	24 / 116 - Stefan T. Tudor	8/22/2021, 7:18:51 PM - 8/23/2021, 9:42:42 PM

Figura 2.41: Tablou de bord Activitate - Listă studenți Test

Odată apăsând unul dintre studenți, utilizatorul va fi redirecționat către un tablou de bord dedicat notării studentului, dar deoarece notarea testelor se face automat, această pagină este strict pentru vizionarea testului. Ca și conținut se vor afișa întrebările și răspunsurile la acestea, respectiv nota obținută la fiecare întrebare, iar în dreapta se va regăsi o metodă de navigare rapidă prin întrebările testului.

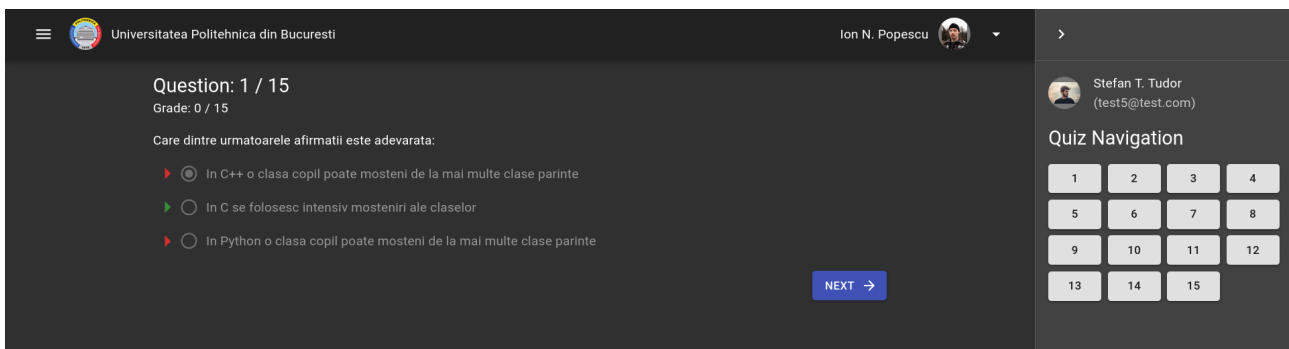


Figura 2.42: Tablou de bord - Notare test Student

În cazul forumurilor, se vor afișa comentariile adăugate de participanți, orice rol este permis.

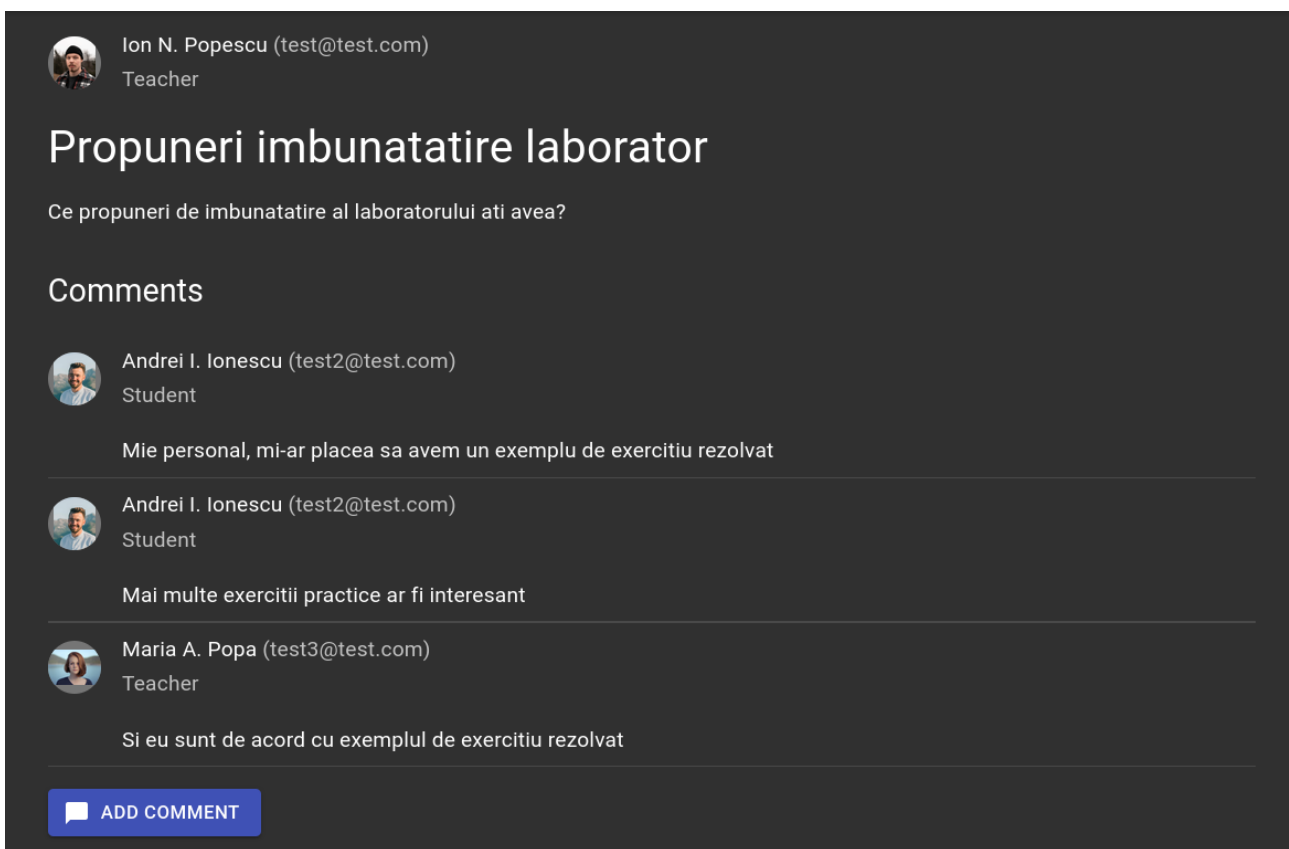


Figura 2.43: Tablou de bord Activitate - Forum

Add Comment

Text

ADD

Figura 2.44: Tablou de bord Activitate - Adăugare comentariu Forum

2.3.3 Crearea setului personal de întrebări

La selectarea Setului personal de întrebări din meniu, utilizatorul va fi redirecționat către tabloul de bord cu întrebări. Acesta va conține o listă de categorii de întrebări, pentru o mai bună repartizare logică a acestora, iar fiecare categorie va fi o secțiune colapsabilă care va conține întrebările specifice.

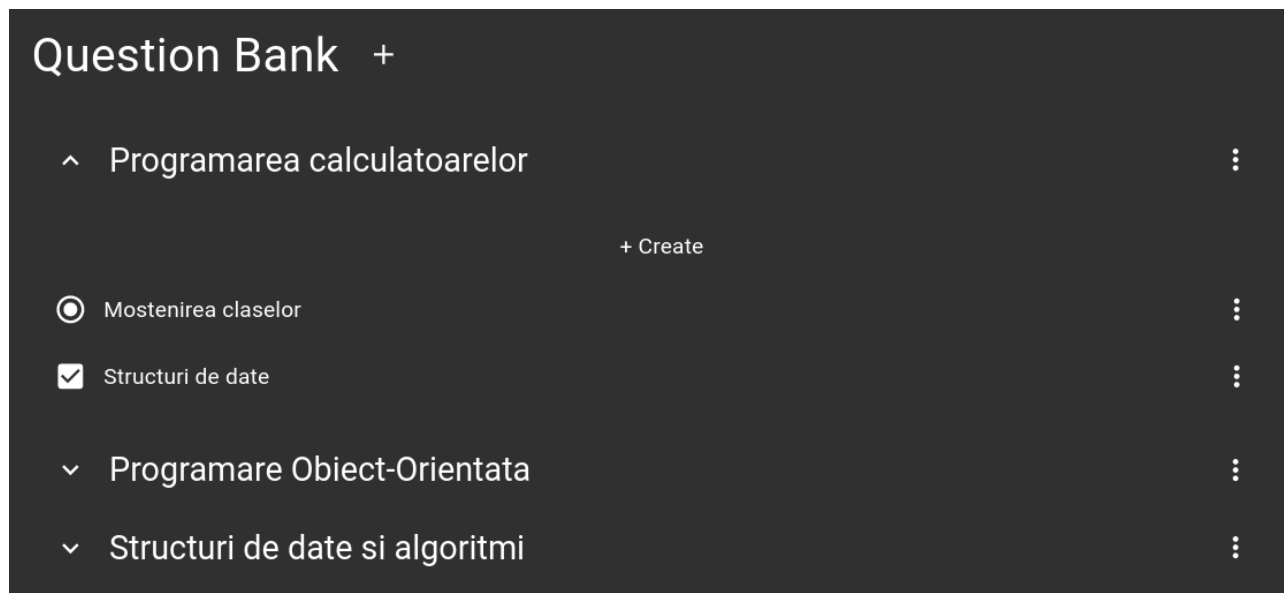


Figura 2.45: Tablou de bord Set Întrebări

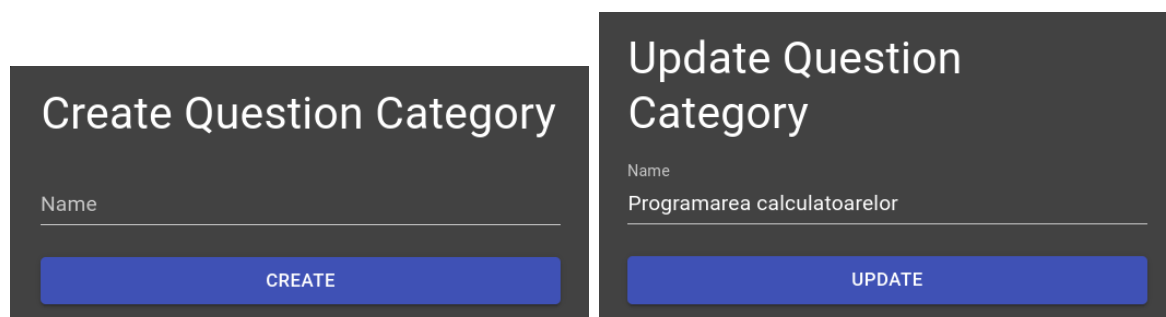


Figura 2.46: Tablou de bord Set Întrebări - Formular creare(stânga), editare(dreapta) Categorie

Formularul de creare a unei întrebări cuprinde următoarele câmpuri:

- Tip de activitate - care poate fi cu răspuns unic, sau cu răspuns multiplu
- Nume - numele întrebării
- Text - textul întrebării
- Răspunsuri - răspunsurile întrebării, iar pentru fiecare răspuns trebuie completate 2 câmpuri
 - Ordine - ordinea din lista de răspunsuri
 - Text - textul răspunsului
 - Procentaj/Fracțiune - ponderea pe care o are răspunsul respectiv pentru întrebare, poate lua valori între -100 și 100

Update Question

Type
Single choice ▼

Name
Mostenirea claselor

Text
Care dintre urmatoarele afirmatii este adevarata:

Answers ▼

= Answer 1 ✕

Text
In C++ o clasa copil poate mosteni de la mai multe clase parinte

Fraction
0

= Answer 2 ✕

Text
In C se folosesc intensiv mosteniri ale claselor

Fraction
100

Figura 2.47: Tablou de bord Set Întrebări - Formular Întrebare

Pentru o calculare automată corectă a notelor de la teste, exista câteva indicații care trebuie respectate.

- Procentajul poate lua valori între -100 și 100
- Pentru întrebările cu răspuns unic, doar 1 singur răspuns ar trebui să aibă procentajul = 100, restul ar trebui să fie = 0
- Pentru întrebările cu răspuns multiplu, suma tuturor procentajelor pozitive ar trebui să fie = 100
- Pentru întrebările cu răspuns multiplu, suma tuturor procentajelor negative ar trebui să fie = -100

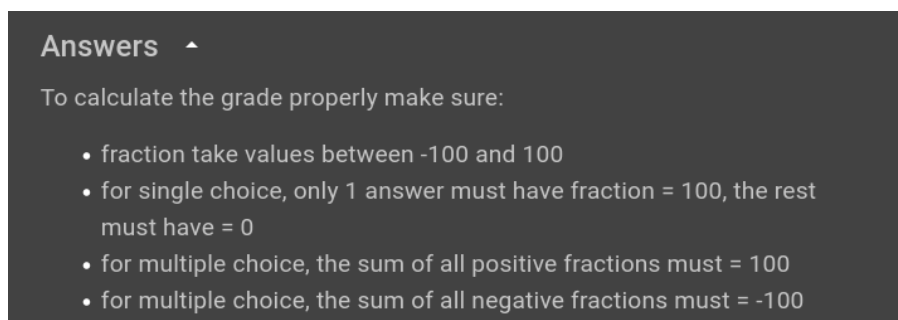


Figura 2.48: Tablou de bord Set Întrebări - Indicații Notare Automată

2.4 Funcționalități student

Ajungând în tabloul de bord al materiei, utilizatorii cu rol de student vor vedea aceleași informații ca și cei cu rol de profesor, în afară de butoanele de creare, editare sau ștergere ale secțiunilor și activităților.

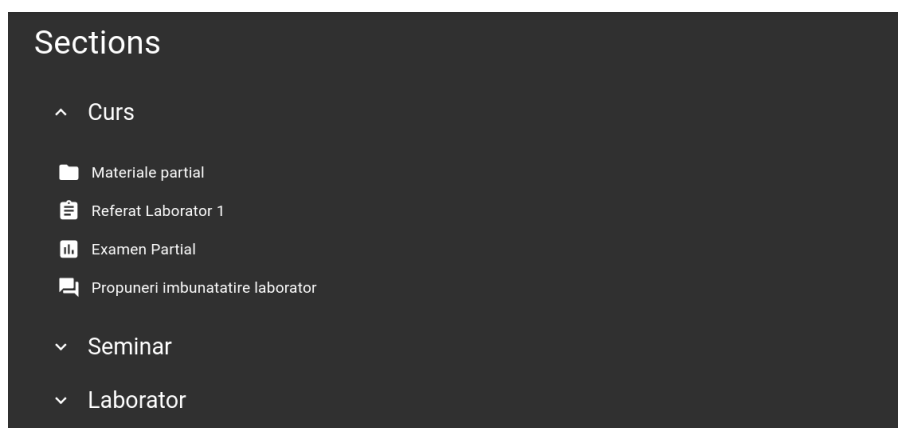


Figura 2.49: Tablou de bord Materie - Listă secțiuni Student

2.4.1 Participarea la activități


Odată selectată o activitate, utilizatorul va fi redirecționat către tabloul de bord al activității, care va afișa ca și în cazul utilizatorilor cu rol de profesor, detalii de bază despre activitate, dar și detalii specifice studentului. Diferențele notabile sunt pentru tipurile temă și test.

În cazul temelor, studentului i se vor afișa detalii personale legate de tema respectivă, notă, ultima dată de încărcare, dacă timpul limită nu s-a scurs și o modalitate de încărcare a temei.

Assignment status

Grade	15 / 30
Deadline	8/25/2021, 12:10:10 PM - 23 hours, 45 minutes remaining
Last time updated	8/24/2021, 12:10:14 PM - 15 minutes, 6 seconds ago

Uploads

 Click to add or drop files here


 IonPopescu_414A_Oscilatii-Mecanice.pdf ×

Figura 2.50: Tablou de bord Activitate - Încărcare temă

În cazul testelor, ca și în cazul temelor, studentului i se vor afișa detalii personale legate de testul respectiv, notă, timp deschidere test, timp trimitere test pentru corectare.

Quiz status

Grade	Not graded yet / 121
Time Open	8/24/2021, 12:10:10 PM
Time Close	8/24/2021, 2:10:10 PM - 1 hour, 39 minutes remaining
Time Limit	1 hour, 30 minutes
Time Start	8/24/2021, 12:10:10 PM
Time Finish	Not submitted yet

Figura 2.51: Tablou de bord Activitate - Detalii test Student

În funcție de timpul curent și detaliile testului, utilizatorului i se va afișa una dintre cele 5 stări ale butonului de interacționare:

- Nimic - testul nu este activ încă sau testul a expirat, utilizatorul nu și-a început încercarea
- Începe - testul este activ, utilizatorul nu și-a început încercarea
- Continuă - testul este activ, utilizatorul și-a început încercarea
- Verifică(blocat) - testul este activ, utilizatorul a trimis rezolvarea pentru notare
- Verifică - testul a expirat, utilizatorul și-a început încercarea

Butoanele Începe și Continuă vor redirecționa studentul spre încercarea curentă a testului. Starea testului, răspunsurile la întrebări, sunt salvate incremental, în timp ce studentul rezolva testul, în cazul unei urgențe sau deconectări, studentul nu va pierde tot progresul pe care l-a făcut.

Tabloul de bord al testului activ este format din meniul pentru navigare rapidă în dreapta paginii, iar ca și conținut se vor afișa întrebările, una câte una, detalii despre aceasta, și timpul rămas din încercare. Toate celelalte butoane și meniuri din bara de instrumente vor fi ascunse, pentru cât mai puține distrageri și/sau apăsări din greșeală.

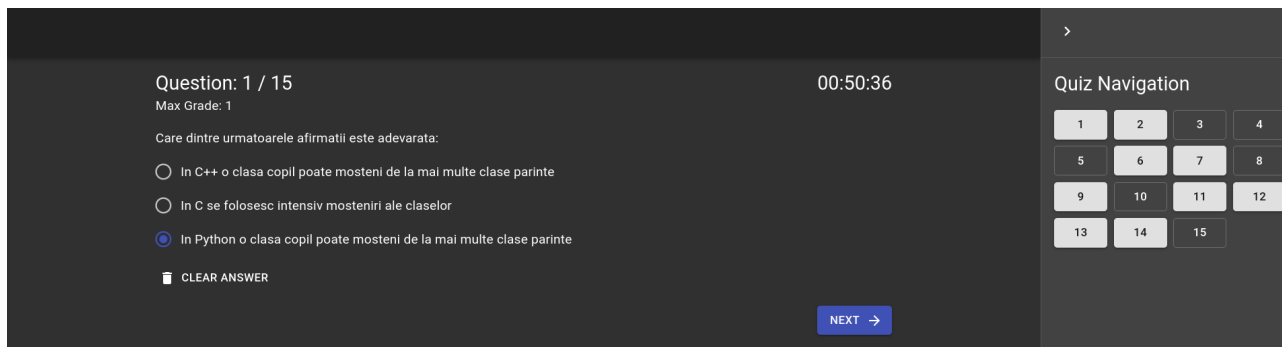


Figura 2.52: Tablou de bord Test activ

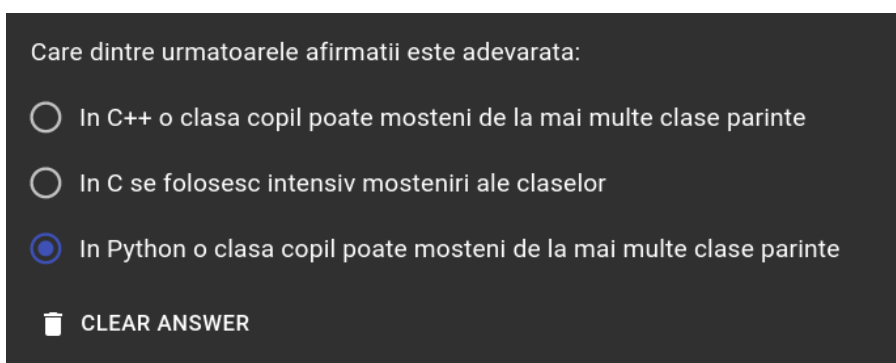


Figura 2.53: Tablou de bord Test activ - Întrebare răspuns unic

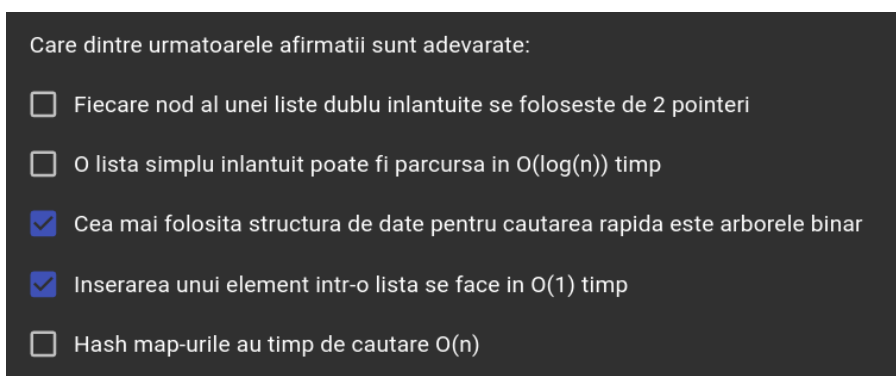


Figura 2.54: Tablou de bord Test activ - Întrebare răspuns multiplu

După ce testul a expirat, utilizatorul își poate verifica răspunsurile la întrebări și notele.

Universitatea Politehnica din Bucuresti

Ion N. Popescu

>

Question: 11 / 15

Grade: 6 / 12

Care dintre urmatoarele afirmatii sunt adevarate:

☒ Fiecare nod al unei liste dublu inlantuite se foloseste de 2 pointeri

☐ O lista simplu inlantuit poate fi parcursa in $O(\log(n))$ timp

☐ Cea mai folosita structura de date pentru cautarea rapida este arborele binar

☐ Inserarea unui element intr-o lista se face in $O(1)$ timp

☐ Hash map-urile au timp de cautare $O(n)$

PREVIOUS

NEXT

Quiz Navigation

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

Figura 2.55: Tablou de bord Verificare Test

Concluzii

Proiectul implementează funcționalitățile cele mai importante pentru desfășurarea activității de predare/testare în mediu online. Această metodă nu are rolul de a înlocui desfășurarea activității în mediu fizic, ci de a o îmbunătăți. Consider ca integrarea unei platforme de e-learning în cadrul oricărei universități vine cu beneficii atât pentru studenți, cât și pentru profesori. Trecerea în mediu online a unor tipuri de activități poate salva timp, efort, chiar și bani pentru laturile implicate.

Bibliografie

- [1] Proiectare software - model cascadă. https://ro.wikipedia.org/wiki/Proiectare_de_program, accesat la data: 14.08.2021.
- [2] Proiectare software. https://en.wikipedia.org/wiki/Software_design, accesat la data: 14.08.2021.
- [3] Arhitectură client-server. https://cio-wiki.org/wiki/Client_Server_Architecture, accesat la data: 23.08.2021.
- [4] Documentație typescript. <https://www.typescriptlang.org/>, accesat la data: 17.02.2021.
- [5] Documentație react. <https://reactjs.org/>, accesat la data: 05.02.2021.
- [6] Documentație next. <https://nextjs.org/>, accesat la data: 09.05.2021.
- [7] Documentație material-ui. <https://material-ui.com/>, accesat la data: 30.06.2021.
- [8] Documentație apollo client. <https://www.apollographql.com/docs/react/>, accesat la data: 17.06.2021.
- [9] Documentație node. <https://nodejs.org/en/>, accesat la data: 21.02.2021.
- [10] Documentație nest. <https://nestjs.com/>, accesat la data: 18.06.2021.
- [11] Documentație postgresql. <https://www.postgresql.org/>, accesat la data: 13.05.2021.
- [12] Documentație prisma. <https://www.prisma.io/>, accesat la data: 11.07.2021.
- [13] Design flexibil. <https://www.smashingmagazine.com/2011/01/guidelines-for-responsive-web-design/>, accesat la data: 23.03.2021.
- [14] Bune practici pentru realizarea formularelor in aplicatii web. <https://medium.com/@andrew.burton/form-validation-best-practices-8e3bec7d0549>, accesat la data: 20.04.2021.

Anexa 1

Cod Sursa

1.1 Modul Autentificare - Funcționalități Server

```
1 import { Resolver, Args, Mutation } from '@nestjs/graphql';
2 import { AuthService } from './auth.service';
3 import { RegisterUserInput } from './dto/register-user.input';
4 import { LoginUserInput } from './dto/login-user.input';
5 import { GraphQLRes } from 'src/my-graphql/decorators/graphql-res.decorator';
6 import { Authentication } from './dto/authentication.object';
7 import { GraphQLReq } from 'src/my-graphql/decorators/graphql-req.decorator';
8 import { UnauthorizedException } from '@nestjs/common';
9 import { Public } from './decorators/public.decorator';
10 import { ReqType, ResType } from 'src/my-graphql/my-graphql.types';
11 import { FileUpload, GraphQLUpload } from 'graphql-upload';
12
13 @Public()
14 @Resolver()
15 export class AuthResolver {
16   constructor(private readonly authService: AuthService) {}
17
18   @Mutation(() => Authentication)
19   register(
20     @GraphQLRes() res: ResType,
21     @Args('user') user: RegisterUserInput,
22     @Args('avatar', { type: () => GraphQLUpload, nullable: true })
23     avatar?: FileUpload
24   ): Promise<Authentication> {
25     return this.authService.register(res, user, avatar);
26   }
27
28   @Mutation(() => Authentication)
29   login(
30     @Args('user') user: LoginUserInput,
31     @GraphQLRes() res: ResType
32   ): Promise<Authentication> {
33     return this.authService.login(user, res);
34   }
35
36   @Mutation(() => Authentication, { nullable: true })
37   logout(): void {
38     throw new UnauthorizedException('logout');
39   }
40
41   @Mutation(() => Authentication)
42   refreshToken(
43     @GraphQLReq() req: ReqType,
44     @GraphQLRes() res: ResType
45   ): Promise<Authentication> {
46     return this.authService.refreshTokens(req, res);
47   }
48 }

```



```
1 import { Injectable, InternalServerErrorException } from '@nestjs/common';
2 import { PrismaService } from 'src/global/prisma/prisma.service';
3 import { RegisterUserInput } from './dto/register-user.input';
4 import argon2 from 'argon2';
5 import { PrismaError } from 'prisma-error-enum';
6 import { LoginUserInput } from './dto/login-user.input';
7 import { Authentication } from './dto/authentication.object';
8 import { TokensService } from './services/tokens.service';
9 import { MyBadRequestException } from 'src/general/error-handling/exceptions/my-bad-request-exception';
10 import { REFRESH_TOKEN_COOKIE_NAME } from './auth.constants';
```

```

11 import { TokensPayload } from './auth.types';
12 import { ReqType, ResType } from 'src/my-graphql/my-graphql.types';
13 import { FileUpload } from 'graphql-upload';
14 import { FileService } from 'src/global/file/file.service';
15
16 @Injectable()
17 export class AuthService {
18   constructor(
19     private readonly prisma: PrismaService,
20     private readonly tokensService: TokensService,
21     private readonly fileService: FileService
22   ) {}
23
24   async register(
25     res: ResType,
26     {
27       email,
28       fatherInitial,
29       firstName,
30       lastName,
31       password
32     }: RegisterUserInput,
33     avatar?: FileUpload
34   ): Promise<Authentication> {
35     try {
36       const hashedPassword = await argon2.hash(password.trim());
37       const createdUser = await this.prisma.user.create({
38         data: {
39           email,
40           firstName: firstName.trim(),
41           lastName: lastName.trim().toUpperCase(),
42           fatherInitial: fatherInitial.trim().toUpperCase(),
43           password: hashedPassword
44         }
45       });
46       if (avatar) {
47         const avatarPath = await this.fileService.createUserAvatar(
48           createdUser.id,
49           avatar
50         );
51         await this.prisma.user.update({
52           where: {
53             id: createdUser.id
54           },
55           data: {
56             avatar: avatarPath
57           }
58         });
59       }
60
61       const tokensPayload: TokensPayload = {
62         user: { id: createdUser.id, universities: {} }
63       };
64       const {
65         accessToken,
66         refreshToken
67       } = this.tokensService.generateTokens(tokensPayload);
68
69       this.tokensService.setRefreshTokenCookie(refreshToken, res);
70       return { accessToken };
71     } catch (e) {
72       if (
73         e.code === PrismaError.UniqueConstraintViolation &&
74         e.meta.target.includes('email')
75       ) {
76         throw new MyBadRequestException({
77           email: 'Email is already in use'
78         });
79       }
80
81       throw new InternalServerErrorException();
82     }
83   }
84
85   async login(
86     userProvided: LoginUserInput,

```

```

87     res: ResType
88 ): Promise<Authentication> {
89     const user = await this.prisma.user.findUnique({
90       where: {
91         email: userProvided.email
92       },
93       select: {
94         id: true,
95         password: true
96       }
97     });
98
99     if (user == null) {
100       throw new MyBadRequestException({
101         email: 'There is no user registered with this email'
102       });
103     }
104
105     try {
106       const hasCorrectPassword = await argon2.verify(
107         user.password,
108         userProvided.password
109       );
110       if (!hasCorrectPassword) {
111         throw new Error('incorrect password');
112       }
113     } catch (e) {
114       if (e.message === 'incorrect password') {
115         throw new MyBadRequestException({
116           password: 'Incorrect password'
117         });
118       }
119
120       throw new InternalServerErrorException();
121     }
122
123     const universities = await this.getUniversitiesScopes(user.id);
124     const tokensPayload: TokensPayload = {
125       user: { id: user.id, universities }
126     };
127     const { accessToken, refreshToken } = this.tokensService.generateTokens(
128       tokensPayload
129     );
130
131     this.tokensService.setRefreshTokenCookie(refreshToken, res);
132     return { accessToken };
133   }
134
135   async refreshTokens(req: ReqType, res: ResType): Promise<Authentication> {
136     const prevToken = req.cookies[REFRESH_TOKEN_COOKIE_NAME];
137     const prevTokensPayload = this.tokensService.getPayloadFromToken(
138       prevToken,
139       'refresh'
140     );
141     const universities = await this.getUniversitiesScopes(
142       prevTokensPayload.user.id
143     );
144     const newTokensPayload: TokensPayload = {
145       ...prevTokensPayload,
146       user: {
147         ...prevTokensPayload.user,
148         universities
149       }
150     };
151     const { accessToken, refreshToken } = this.tokensService.generateTokens(
152       newTokensPayload
153     );
154
155     this.tokensService.setRefreshTokenCookie(refreshToken, res);
156     return { accessToken };
157   }
158
159   private async getUniversitiesScopes(
160     userId: string
161 ): Promise<TokensPayload['user']['universities']> {
162     const universities = await this.prisma.universityUser.findMany({

```

```

163     where: {
164       userId
165     },
166     select: {
167       universityId: true,
168       role: {
169         select: {
170           scopes: {
171             select: {
172               name: true
173             }
174           }
175         }
176       }
177     }
178   });
179
180   return universities.reduce(
181     (universitiesAcc, universitiesCurr) => ({
182       ...universitiesAcc,
183       [universitiesCurr.universityId]: {
184         scopes: universitiesCurr.role.scopes.reduce(
185           (scopesAcc, scopesCurr) => ({
186             ...scopesAcc,
187             [scopesCurr.name]: true
188           }),
189           {}
190         )
191       }
192     }),
193     {}
194   );
195 }
196 }

```

1.2 Formular Autenticare

```

1 import { FormVerticalLayout } from 'domains/shared/components/form/FormVerticalLayout';
2 import { FormErrors } from 'domains/shared/constants/FormErrors';
3 import { Field, Form, Formik } from 'formik';
4 import { TextField } from 'formik-material-ui';
5 import { FC, memo } from 'react';
6 import * as yup from 'yup';
7 import { useLoginMutation } from 'generated/graphql';
8 import { accessTokenVar } from 'domains/auth/reactiveVars';
9 import { ButtonWithLoader } from 'domains/shared/components/buttons/ButtonWithLoader';
10 import { useFormikSubmit } from 'domains/shared/hooks/useFormikSubmit';
11 import { useRouter } from 'next/router';
12 import { Routes } from 'domains/shared/constants/Routes';
13
14 interface Values {
15   email: string;
16   password: string;
17 }
18
19 const initialValues: Values = {
20   email: '',
21   password: ''
22 };
23
24 const validationSchema = yup.object().shape({
25   email: yup
26     .string()
27     .trim()
28     .email(FormErrors.VALID_EMAIL)
29     .required(FormErrors.REQUIRED),
30   password: yup.string().trim().required(FormErrors.REQUIRED)
31 });
32
33 export const LoginForm: FC = memo(function LoginForm() {
34   const router = useRouter();
35
36   const [login] = useLoginMutation();
37   const handleLogin = useFormikSubmit<Values>(async (values) => {
38     const res = await login({

```



```

39     variables: {
40       user: values
41     },
42     fetchPolicy: 'no-cache'
43   });
44   accessTokenVar(res.data?.login.accessToken);
45   router.push(Routes.user.DASHBOARD.path);
46 });
47
48 return (
49   <Formik
50     initialValues={initialValues}
51     validationSchema={validationSchema}
52     onSubmit={handleLogin}
53   >
54     {( { isSubmitting } ) => (
55       <Form autoComplete="on">
56         <FormVerticalLayout
57           fields={
58             <>
59               <Field
60                 component={TextField}
61                 name="email"
62                 type="email"
63                 label="Email"
64                 fullWidth
65               />
66               <Field
67                 component={TextField}
68                 name="password"
69                 type="password"
70                 label="Password"
71                 fullWidth
72               />
73             </>
74           }
75           actions={
76             <ButtonWithLoader
77               variant="contained"
78               color="primary"
79               fullWidth
80               loading={isSubmitting}
81               type="submit"
82             >
83               Login
84             </ButtonWithLoader>
85           }
86         />
87       </Form>
88     )}
89   </Formik>
90 );
91 });

```

1.3 Formular Înregistrare

```

1 import { ChangeEvent, FC, memo } from 'react';
2 import * as yup from 'yup';
3 import { FormErrors } from 'domains/shared/constants/FormErrors';
4 import { Field, Form, Formik } from 'formik';
5 import { TextField } from 'formik-material-ui';
6 import { FormVerticalLayout } from 'domains/shared/components/form/FormVerticalLayout';
7 import { ValidationRegexp } from 'domains/shared/constants/ValidationRegexp';
8 import { useRegisterMutation } from 'generated/graphql';
9 import { accessTokenVar } from '../reactiveVars';
10 import { ButtonWithLoader } from 'domains/shared/components/buttons/ButtonWithLoader';
11 import {
12   FileUpload,
13   FileUploadProps
14 } from 'domains/shared/components/form/FileUpload';
15 import { FileType } from 'domains/shared/constants/file/FileType';
16 import { useFormikSubmit } from 'domains/shared/hooks/useFormikSubmit';
17 import { useRouter } from 'next/router';
18 import { Routes } from 'domains/shared/constants/Routes';
19 import { composeLabel } from 'domains/shared/utils/form/composeLabel';

```

```

20
21 interface Values {
22   firstName: string;
23   lastName: string;
24   fatherInitial: string;
25   email: string;
26   password: string;
27   avatar: Record<string, File>;
28 }
29
30 const initialValues: Values = {
31   firstName: '',
32   lastName: '',
33   fatherInitial: '',
34   email: '',
35   password: '',
36   avatar: {}
37 };
38
39 const validationSchema = yup.object().shape({
40   firstName: yup.string().trim().required(FormErrors.REQUIRED),
41   lastName: yup.string().trim().required(FormErrors.REQUIRED),
42   fatherInitial: yup
43     .string()
44     .trim()
45     .matches(ValidationRegexp.ALPHA, FormErrors.ONLY_ALPHA)
46     .required(FormErrors.REQUIRED),
47   email: yup
48     .string()
49     .trim()
50     .email(FormErrors.VALID_EMAIL)
51     .required(FormErrors.REQUIRED),
52   password: yup.string().trim().required(FormErrors.REQUIRED),
53   avatar: yup.object()
54 });
55
56 export const RegisterForm: FC = memo(function RegisterForm() {
57   const router = useRouter();
58   const [register] = useRegisterMutation();
59   const handleRegister = useFormikSubmit<Values>(async (values) => {
60     const { avatar: _, ...user } = values;
61     const avatar = Object.values(values.avatar)[0];
62     const res = await register({
63       variables: {
64         user,
65         avatar
66       },
67       fetchPolicy: 'no-cache'
68     });
69     accessTokenVar(res.data?.register.accessToken);
70     router.push(Routes.user.DASHBOARD.path);
71   });
72
73   return (
74     <Formik
75       initialValues={initialValues}
76       validationSchema={validationSchema}
77       onSubmit={handleRegister}
78     >
79     {({ isSubmitting, setFieldValue }) => (
80       <Form>
81         <FormVerticalLayout
82           fields={
83             <>
84               <Field
85                 component={TextField}
86                 name="firstName"
87                 label="First Name"
88                 fullWidth
89               />
90               <Field
91                 component={TextField}
92                 name="lastName"
93                 label="Last Name"
94                 fullWidth
95                 onChange={

```

```

96         e: ChangeEvent<HTMLInputElement>
97     ) =>
98         setFieldValue(
99             'lastName',
100             e.target.value.toUpperCase()
101         )
102     }
103 />
104 <Field
105     component={TextField}
106     name="fatherInitial"
107     label="Father's Initial"
108     inputProps={{ maxLength: 1 }}
109     fullWidth
110     onChange={
111         (
112             e: ChangeEvent<HTMLInputElement>
113         ) =>
114             setFieldValue(
115                 'fatherInitial',
116                 e.target.value.toUpperCase()
117             )
118     }
119 />
120 <Field
121     component={TextField}
122     name="email"
123     type="email"
124     label="Email"
125     fullWidth
126     autoComplete="on"
127 />
128 <Field
129     component={TextField}
130     name="password"
131     type="password"
132     label="Password"
133     fullWidth
134 />
135 <Field name="avatar">
136     {({
137         field: { value }
138     }): {
139         field: {
140             value: FileUploadProps['newFiles'];
141         };
142     }) => (
143         <FileUpload
144             label={composeLabel(
145                 'Avatar',
146                 'optional'
147             )}
148             helperText="Recommended image size: 40x40px"
149             newFiles={value}
150             onNewFilesUpdate={
151                 (
152                     getUpdatedFiles
153                 ) => {
154                     setFieldValue(
155                         'avatar',
156                         getUpdatedFiles(value)
157                     );
158                 }
159             }
160             maxFiles={1}
161             maxFileSize={10}
162             acceptedFileTypes={[FileType.IMAGE]}
163         />
164     )}
165 </Field>
166 </>
167 }
168 actions={
169     <ButtonWithLoader
170         variant="contained"
171         color="primary"
172         fullWidth
173         loading={isSubmitting}
174         type="submit"

```

```

172         >
173         Register
174     </ButtonWithLoader>
175     }
176     />
177 </Form>
178 })
179 </Formik>
180 );
181 });

```

1.4 Validare autentificare server

```

1 import { InputType, PickType } from '@nestjs/graphql';
2 import { RegisterUserInput } from '../register-user.input';
3
4 @InputType()
5 export class LoginUserInput extends PickType(RegisterUserInput, [
6   'email',
7   'password'
8 ] as const) {}

```

1.5 Validare înregistrare server

```

1 import { Field, InputType } from '@nestjs/graphql';
2 import { IsAlpha, IsEmail, IsNotEmpty, Length } from 'class-validator';
3
4 @InputType()
5 export class RegisterUserInput {
6   @Field()
7   @IsNotEmpty()
8   firstName: string;
9
10  @Field()
11  @IsNotEmpty()
12  lastName: string;
13
14  @Field()
15  @IsEmail()
16  email: string;
17
18  @Field()
19  @IsNotEmpty()
20  password: string;
21
22  @Field()
23  @IsAlpha()
24  @Length(1, 1)
25  fatherInitial: string;
26 }

```

1.6 Afișare lista universități

```

1 import { Box, Dialog, IconButton, Tooltip } from '@material-ui/core';
2 import { Add } from '@material-ui/icons';
3 import { Content } from 'domains/shared/components/layout/Content';
4 import { ContentHeader } from 'domains/shared/components/layout/ContentHeader';
5 import { MyHead } from 'domains/shared/components/MyHead';
6 import { MySkeleton } from 'domains/shared/components/MySkeleton';
7 import { useBooleanState } from 'domains/shared/hooks/useBooleanState';
8 import { CreateUniversityForm } from 'domains/university/components/UniversityForm/
  CreateUniversityForm';
9 import { UniversitiesCards } from 'domains/university/components/UniversitiesCards';
10 import { useMeQuery } from 'generated/graphql';
11 import { useRefreshTokens } from 'domains/auth/hooks/useRefreshTokens';
12 import { useCallback } from 'react';
13 import { composeDynamicRoute } from 'domains/shared/utils/route/composeDynamicRoute';
14 import { Routes } from 'domains/shared/constants/Routes';
15 import { useRouter } from 'next/router';
16
17 export default function App() {
18   const me = useMeQuery();

```

```

19  const [
20    isCreateUniversityDialogOpen,
21    openCreateUniversityDialog,
22    closeCreateUniversityDialog
23  ] = useBooleanState();
24
25  const router = useRouter();
26  const refreshTokens = useRefreshTokens();
27  const redirectToUniversity = useCallback(
28    async (universityId: string): Promise<void> => {
29      await refreshTokens();
30
31      router.push(
32        composeDynamicRoute(Routes.university.DASHBOARD.path, {
33          universityId
34        })
35      );
36    },
37    [refreshTokens, router]
38  );
39
40  return (
41    <>
42      <MyHead title="Dashboard" />
43      <ContentHeader
44        title="Universities"
45        action={
46          <Tooltip title="Create University">
47            <IconButton onClick={openCreateUniversityDialog}>
48              <Add />
49            </IconButton>
50          </Tooltip>
51        }
52      />
53
54      {me.loading ? (
55        Array(3)
56          .fill(0)
57          .map((_, i) => (
58            <Box key={i} pt={i && 1}>
59              <MySkeleton variant="round" height={72} />
60            </Box>
61          ))
62      ) : (
63        <UniversitiesCards
64          groupedByRoleUniversities={
65            me.data?.me.groupedByRoleUniversities
66          }
67        />
68      )}
69
70      <Dialog
71        open={isCreateUniversityDialogOpen}
72        onClose={closeCreateUniversityDialog}
73        fullWidth
74        maxWidth="xs"
75      >
76        <Content>
77          <ContentHeader title="Create University" />
78          <CreateUniversityForm onSuccess={redirectToUniversity} />
79        </Content>
80      </Dialog>
81    </>
82  );
83 }

```

```

1  import { Box, Typography } from '@material-ui/core';
2  import { UserRole } from 'domains/shared/constants/UserRole';
3  import { GroupedByRoleUniversitiesObject } from 'generated/graphql';
4  import { FC, Fragment, memo } from 'react';
5  import { UniversityCard } from './UniversityCard';
6
7  const getRoleName = (role: string) => {
8    switch (role) {
9      case UserRole.ADMIN:
10        return 'Admin';

```

```

11     case UserRole.TEACHER:
12         return 'Teacher';
13     case UserRole.STUDENT:
14         return 'Student';
15     default:
16         return 'Student';
17 }
18 };
19 const validRoles = Object.values(UserRole).reduce<Record<string, true>>(
20     (acc, curr) => (acc[curr] ? acc : { ...acc, [curr]: true }),
21     {}
22 );
23 const rolesPositions: Record<string, number> = {
24     [UserRole.ADMIN]: 0,
25     [UserRole.TEACHER]: 1,
26     [UserRole.STUDENT]: 2
27 };
28
29 interface UniversitiesCardsProps {
30     groupedByRoleUniversities?: GroupedByRoleUniversitiesObject[];
31 }
32
33 export const UniversitiesCards: FC<UniversitiesCardsProps> = memo(
34     function UniversitiesCards({ groupedByRoleUniversities }) {
35         if (groupedByRoleUniversities == null) {
36             return null;
37         }
38
39         const filteredAndSortedUniversitiesGroups = groupedByRoleUniversities
40             .filter(
41                 ({ role, universities }) =>
42                     !!validRoles[role] && !!universities.length
43             )
44             .sort((a, b) => rolesPositions[a.role] - rolesPositions[b.role]);
45
46         if (!filteredAndSortedUniversitiesGroups.length) {
47             return (
48                 <Typography color="textSecondary" align="center">
49                     You are not enrolled in any university
50                 </Typography>
51             );
52         }
53
54         return (
55             <>
56                 {filteredAndSortedUniversitiesGroups.map(
57                     ({ role, universities }, roleIndex) => (
58                         <Fragment key={role}>
59                             <Typography variant="h6" gutterBottom>
60                                 {getRoleName(role)}
61                             </Typography>
62                             {universities.map((university, universityIndex) => (
63                                 <Fragment key={university.id}>
64                                     <UniversityCard
65                                         role={role as UserRole}
66                                         university={university}
67                                     </>
68                                     {universityIndex !==
69                                         universities.length - 1 && (
70                                         <Box pb={1} />
71                                     )}
72                                 </Fragment>
73                             ))}
74                             {roleIndex !==
75                                 groupedByRoleUniversities.length - 1 && (
76                                 <Box pb={2} />
77                             )}
78                         </Fragment>
79                     )
80                 )}
81             </>
82         );
83     }
84 );

```

1.7 Bară de instrumente

```
1 import { IconButton, Toolbar, Tooltip } from '@material-ui/core';
2 import { ArrowDropDown, ChevronLeft, Menu } from '@material-ui/icons';
3 import { FC, memo, useEffect, useState } from 'react';
4 import { usePopupState, bindTrigger } from 'material-ui-popup-state/hooks';
5 import { UserDropDownMenu } from '../UserDropDownMenu';
6 import { AppBarStyled, EmptyMiddleSpace } from '../index.styles';
7 import { UserButton } from '../UserButton';
8 import { UniversityButton } from '../UniversityButton';
9 import { useRouter } from 'next/router';
10 import { isRouteMatching } from 'domains/shared/utils/route/isRouteMatching';
11 import { Routes } from 'domains/shared/constants/Routes';
12
13 interface AppBarProps {
14   // Menu
15   hideMenuDrawerButton: boolean;
16   isMenuDrawerDesktop: boolean;
17   isMenuDrawerOpen: boolean;
18   onMenuDrawerOpen: () => void;
19   // Extra content
20   showExtraContentDrawerButton: boolean;
21   isExtraContentDrawerDesktop: boolean;
22   isExtraContentDrawerOpen: boolean;
23   onExtraContentDrawerOpen: () => void;
24 }
25
26 export const AppBar: FC<AppBarProps> = memo(function AppBar({
27   hideMenuDrawerButton,
28   isMenuDrawerDesktop,
29   isMenuDrawerOpen,
30   onMenuDrawerOpen,
31   showExtraContentDrawerButton,
32   isExtraContentDrawerDesktop,
33   isExtraContentDrawerOpen,
34   onExtraContentDrawerOpen
35 }) {
36   const userMenuState = usePopupState({
37     variant: 'popover',
38     popupId: 'appBarUserMenu'
39   });
40
41   const router = useRouter();
42   const [isQuizActive, setIsQuizActive] = useState(
43     isRouteMatching(router.asPath, Routes.activity QUIZ_ACTIVE)
44   );
45   useEffect(() => {
46     if (isRouteMatching(router.asPath, Routes.activity QUIZ_ACTIVE)) {
47       setIsQuizActive(true);
48     } else {
49       setIsQuizActive(false);
50     }
51   }, [router.asPath]);
52
53   return (
54     <>
55     <AppBarStyled
56       color="default"
57       isMenuDrawerOpen={isMenuDrawerOpen}
58       isMenuDrawerDesktop={isMenuDrawerDesktop}
59       isExtraContentDrawerOpen={isExtraContentDrawerOpen}
60       isExtraContentDrawerDesktop={isExtraContentDrawerDesktop}
61     >
62     <Toolbar>
63       {!isQuizActive && (
64         <>
65           {!hideMenuDrawerButton && !isMenuDrawerOpen && (
66             <Tooltip title="Drawer">
67               <IconButton onClick={onMenuDrawerOpen}>
68                 <Menu />
69             </IconButton>
70             </Tooltip>
71           )}
72           <Tooltip title="University Dashboard">
73             <UniversityButton />
74           </Tooltip>
```

```

75         </>
76     )}
77
78     <EmptyMiddleSpace />
79
80     {!isQuizActive && (
81         <>
82             <Tooltip title="User Dashboard">
83                 <IconButton />
84             </Tooltip>
85             <Tooltip title="Menu">
86                 <IconButton {...bindTrigger(userMenuState)}>
87                     <ArrowDropDown />
88                 </IconButton>
89             </Tooltip>
90         </>
91     )}
92     {showExtraContentDrawerButton && !isExtraContentDrawerOpen && (
93         <Tooltip title="More Actions">
94             <IconButton onClick={onExtraContentDrawerOpen}>
95                 <ChevronLeft />
96             </IconButton>
97         </Tooltip>
98     )}
99     </Toolbar>
100 </AppBarStyled>
101
102     <UserDropDownMenu popupState={userMenuState} />
103 </>
104 );
105 });

```

1.8 Meniu Utilizator

```

1 import { useReactiveVar } from '@apollo/client';
2 import {
3   Divider,
4   ListItemIcon,
5   ListItemText,
6   MenuItem,
7   MenuList,
8   Popover
9 } from '@material-ui/core';
10 import {
11   MeetingRoom,
12   EventNote,
13   DateRange,
14   AccountBalance,
15   People
16 } from '@material-ui/icons';
17 import { Routes, RoutesGroups } from 'domains/shared/constants/Routes';
18 import { UserRole } from 'domains/shared/constants/UserRole';
19 import { useBooleanState } from 'domains/shared/hooks/useBooleanState';
20 import { composeDynamicRoute } from 'domains/shared/utils/route/composeDynamicRoute';
21 import { isRouteMatching } from 'domains/shared/utils/route/isRouteMatching';
22 import { selectedUniversityVar } from 'domains/university/reactiveVars';
23 import { useLogoutMutation } from 'generated/graphql';
24 import { PopupState, bindPopover } from 'material-ui-popup-state/core';
25 import { useRouter } from 'next/router';
26 import { FC, memo, useCallback, useEffect, useMemo } from 'react';
27 import { MenuLinkItem } from '../../list/MenuLinkItem';
28
29 interface UserDropDownMenuProps {
30   popupState: PopupState;
31 }
32
33 export const UserDropDownMenu: FC<UserDropDownMenuProps> = memo(
34   function UserDropDownMenu({ popupState }) {
35     const router = useRouter();
36
37     useEffect(() => {
38       router.prefetch(Routes.auth.LOGIN_REGISTER.path);
39     }, [router]);
40
41     const [logout] = useLogoutMutation({ fetchPolicy: 'no-cache' });

```



```

42     const handleLogout = useCallback(async () => {
43         try {
44             await logout();
45         } catch {
46             popupState.close();
47         }
48     }, [logout, popupState]);
49
50     const [
51         shouldShowUserUniversitySpecificButtons,
52         showUserUniversitySpecificButtons,
53         hideUserUniversitySpecificButtons
54     ] = useBooleanState();
55     useEffect(() => {
56         if (
57             isRouteMatching(router.asPath, RoutesGroups.OUTSIDE_UNIVERSITY)
58         ) {
59             hideUserUniversitySpecificButtons();
60         } else {
61             showUserUniversitySpecificButtons();
62         }
63     }, [
64         router.asPath,
65         showUserUniversitySpecificButtons,
66         hideUserUniversitySpecificButtons
67     ]);
68
69     const university = useReactiveVar(selectedUniversityVar);
70     const shouldShowTeacherOrAdminSpecificButtons = useMemo(
71         () =>
72             [UserRole.TEACHER, UserRole.ADMIN].includes(
73                 university?.role ?? UserRole.STUDENT
74             ),
75         [university?.role]
76     );
77
78     return (
79         <Popover
80             {...bindPopover(popupState)}
81             anchorOrigin={{
82                 vertical: 'bottom',
83                 horizontal: 'right'
84             }}
85             transformOrigin={{
86                 vertical: 'top',
87                 horizontal: 'right'
88             }}
89         >
90             {shouldShowUserUniversitySpecificButtons && (
91                 <>
92                     <MenuList>
93                         {university?.role === UserRole.STUDENT && (
94                             <MenuItem
95                                 href={composeDynamicRoute(
96                                     Routes.userUniversity.GRADES.path,
97                                     {
98                                         universityId: String(
99                                             router.query.universityId
100                                         )
101                                     }
102                                 )}
103                                 onClick={popupState.close}
104                                 button
105                             >
106                                 <ListItemIcon>
107                                     <EventNote />
108                                 </ListItemIcon>
109                                 <ListItemText primary="Grades" />
110                             </MenuItem>
111                         )}
112                         {university?.role === UserRole.ADMIN && (
113                             <MenuItem
114                                 href={composeDynamicRoute(
115                                     Routes.userUniversity.USERS.path,
116                                     {
117                                         universityId: String(

```

```

118         router.query.universityId
119     )
120 }
121 }}
122 onClick={popupState.close}
123 button
124 >
125     <ListItemIcon>
126         <People />
127     </ListItemIcon>
128     <ListItemText primary="Users" />
129 </MenuItem>
130 }}
131 {shouldShowTeacherOrAdminSpecificButtons && (
132     <MenuItem
133         href={composeDynamicRoute(
134             Routes.userUniversity.QUESTION_BANK
135             .path,
136             {
137                 universityId: String(
138                     router.query.universityId
139                 )
140             }
141         )}
142         onClick={popupState.close}
143         button
144     >
145         <ListItemIcon>
146             <AccountBalance />
147         </ListItemIcon>
148         <ListItemText primary="Question Bank" />
149     </MenuItem>
150 )}
151 <MenuItem
152     href={composeDynamicRoute(
153         Routes.userUniversity.UPCOMING_ACTIVITIES
154         .path,
155         {
156             universityId: String(
157                 router.query.universityId
158             )
159         }
160     )}
161     onClick={popupState.close}
162     button
163 >
164     <ListItemIcon>
165         <DateRange />
166     </ListItemIcon>
167     <ListItemText primary="Upcoming activities" />
168 </MenuItem>
169 </MenuList>
170
171 <Divider />
172 </>
173 )}
174
175 <MenuList>
176     <MenuItem button onClick={handleLogout}>
177         <ListItemIcon>
178             <MeetingRoom />
179         </ListItemIcon>
180         <ListItemText primary="Logout" />
181     </MenuItem>
182 </MenuList>
183 </Popover>
184 );
185 }
186 );

```

1.9 Afişare listă facultăți

```

1 import { useReactiveVar } from '@apollo/client';
2 import {
3     Box,

```

```

4   IconButton,
5   List,
6   Tooltip,
7   Typography,
8   Dialog
9 } from '@material-ui/core';
10 import { Add } from '@material-ui/icons';
11 import { CollegeDashboardCollapsible } from 'domains/college/components/
    CollegeDashboardCollapsible';
12 import { CreateCollegeForm } from 'domains/college/components/CollegeForm/CreateCollegeForm';
13 import { Content } from 'domains/shared/components/layout/Content';
14 import { ContentHeader } from 'domains/shared/components/layout/ContentHeader';
15 import { MyHead } from 'domains/shared/components/MyHead';
16 import { MySkeleton } from 'domains/shared/components/MySkeleton';
17 import { UserRole } from 'domains/shared/constants/UserRole';
18 import { useBooleanState } from 'domains/shared/hooks/useBooleanState';
19 import { selectedUniversityVar } from 'domains/university/reactiveVars';
20 import { useCollegesQuery } from 'generated/graphql';
21
22 export default function UniversityDashboard() {
23   const colleges = useCollegesQuery({
24     variables: {
25       universityId: selectedUniversityVar()?.id ?? 'placeholder'
26     }
27   });
28   const university = useReactiveVar(selectedUniversityVar);
29
30   const [
31     isCreateCollegeDialogOpen,
32     openCreateCollegeDialog,
33     closeCreateCollegeDialog
34   ] = useBooleanState();
35
36   return (
37     <>
38       <MyHead title="University" />
39       <ContentHeader
40         title="Colleges"
41         action={
42           university?.role === UserRole.ADMIN && (
43             <Tooltip title="Create College">
44               <IconButton onClick={openCreateCollegeDialog}>
45                 <Add />
46               </IconButton>
47             </Tooltip>
48           )
49         }
50       />
51
52     {(() => {
53       if (colleges.loading) {
54         return (
55           <Box py={1} px={2}>
56             <Array(5)
57               .fill(0)
58               .map((_, i) => (
59                 <Box key={i} pt={i && 1}>
60                   <MySkeleton
61                     variant="round"
62                     height={50}>
63                   </>
64                 </Box>
65               ))>
66             </Box>
67           );
68         }
69
70         if (!colleges.data?.colleges.length) {
71           return (
72             <Box py={1} px={2}>
73               <Typography color="textSecondary" align="center">
74                 There are no colleges created yet
75               </Typography>
76             </Box>
77           );
78         }

```

```

79
80     return (
81       <List>
82         {colleges.data.colleges.map((college) => (
83           <CollegeDashboardCollapsible
84             key={college.id}
85             college={college}
86           />
87         )}}
88       </List>
89     );
90   }())}
91
92   <Dialog
93     open={isCreateCollegeDialogOpen}
94     onClose={closeCreateCollegeDialog}
95     fullWidth
96     maxWidth="xs"
97   >
98     <Content>
99       <ContentHeader title="Create College" />
100       <CreateCollegeForm onSuccess={closeCreateCollegeDialog} />
101     </Content>
102   </Dialog>
103 </>
104 );
105 }

1 import { useReactiveVar } from '@apollo/client';
2 import { Box, Typography } from '@material-ui/core';
3 import { CreateCourseForm } from 'domains/course/components/CourseForm/CreateCourseForm';
4 import { AddCard } from 'domains/shared/components/card/AddCard';
5 import { ListItemCollapsible } from 'domains/shared/components/list/ListItemCollapsible';
6 import { ModifyResourceAction } from 'domains/shared/components/ModifyResourceAction';
7 import { UserRole } from 'domains/shared/constants/UserRole';
8 import { selectedUniversityVar } from 'domains/university/reactiveVars';
9 import {
10   CollegeFieldsFragment,
11   useDeleteCollegeMutation
12 } from 'generated/graphql';
13 import { FC, memo, useCallback } from 'react';
14 import { deleteCollegeUpdate } from '../graphql/updates/deleteCollegeUpdate';
15 import { UpdateCollegeForm } from '../CollegeForm/UpdateCollegeForm';
16 import { CourseDashboardCard } from '../CourseDashboardCard';
17
18 interface CollegeDashboardCollapsibleProps {
19   college: CollegeFieldsFragment;
20 }
21
22 export const CollegeDashboardCollapsible: FC<CollegeDashboardCollapsibleProps> = memo(
23   function CollegeDashboardCollapsible({ college }) {
24     const { id, name, courses } = college;
25
26     const university = useReactiveVar(selectedUniversityVar);
27
28     const [
29       deleteCollege,
30       { loading: deleteCollegeLoading }
31     ] = useDeleteCollegeMutation({ update: deleteCollegeUpdate });
32     const handleDeleteCollege = useCallback(() : void => {
33       deleteCollege({
34         variables: {
35           id
36         }
37       }).catch(() => null);
38     }, [deleteCollege, id]);
39
40     return (
41       <ListItemCollapsible
42         name={name}
43         action={
44           university?.role === UserRole.ADMIN && (
45             <ModifyResourceAction
46               // Shared
47               resourceName={name}
48               resourceType="College"

```

```

49         // Update
50         updateForm={ (onSuccess) => (
51             <UpdateCollegeForm
52                 college={college}
53                 onSuccess={onSuccess}
54             />
55         )}
56         // Delete
57         onDelete={handleDeleteCollege}
58         deleteLoading={deleteCollegeLoading}
59     />
60 )
61 }
62 >
63 {courses.length ? (
64     <Box
65         display="grid"
66         gridGap={8}
67         gridTemplateColumns="repeat(auto-fill, minmax(300px, 1fr))"
68     >
69         {university?.role === UserRole.ADMIN && (
70             <AddCard
71                 resourceType="Course"
72                 form={ (onSuccess) => (
73                     <CreateCourseForm
74                         onSuccess={onSuccess}
75                         collegeId={college.id}
76                     />
77                 )}
78             />
79         )}
80         {courses.map((course) => (
81             <CourseDashboardCard
82                 key={course.id}
83                 collegeId={id}
84                 course={course}
85             />
86         ))}
87     </Box>
88 ) : (
89     <>
90         {university?.role === UserRole.ADMIN && (
91             <Box mb={2}>
92                 <AddCard
93                     resourceType="Course"
94                     form={ (onSuccess) => (
95                         <CreateCourseForm
96                             onSuccess={onSuccess}
97                             collegeId={college.id}
98                         />
99                     )}
100                 />
101             </Box>
102         )}
103         <Typography color="textSecondary" align="center">
104             There are no courses created yet
105         </Typography>
106     </>
107 )}
108 </ListItemCollapsible>
109 );
110 }
111 );

```

1.10 Management listă utilizatori

```

1 import {
2     DataGrid,
3     GridCellParams,
4     GridCellValue,
5     GridColDef
6 } from '@material-ui/data-grid';
7 import { ModifyResourceAction } from 'domains/shared/components/ModifyResourceAction';
8 import { MyAvatar } from 'domains/shared/components/MyAvatar';
9 import { UserRole } from 'domains/shared/constants/UserRole';

```

```

10 import { formatUserRole } from 'domains/shared/utils/formatUserRole';
11 import {
12   UniversityUsersQuery,
13   useDeleteUniversityUserMutation
14 } from 'generated/graphql';
15 import { FC, memo, useCallback } from 'react';
16 import { deleteUniversityUserUpdate } from '../graphql/updates/deleteUniversityUserUpdate';
17 import { UpdateUniversityUserForm } from '../UniversityUserForm/UpdateUniversityUserForm';
18
19 type UniversityUser = UniversityUsersQuery['universityUsers'][number];
20
21 const sortComparator = (v1: GridCellValue, v2: GridCellValue) =>
22   (v1 as string).toString().localeCompare((v2 as string).toString());
23
24 const UserAvatar = ({ value, row }: GridCellParams) => {
25   return (
26     <MyAvatar
27       src={value as string}
28       alt={`${(row as UniversityUser).user.firstName} ${
29         (row as UniversityUser).user.lastName
30       } avatar`}
31     />
32   );
33 };
34
35 const UserActions = ({ row }: GridCellParams) => {
36   const [
37     deleteUniversityUser,
38     { loading: deleteUniversityUserLoading }
39   ] = useDeleteUniversityUserMutation({ update: deleteUniversityUserUpdate });
40   const handleDeleteUniversityUser = useCallback(() : void => {
41     deleteUniversityUser({
42       variables: {
43         id: (row as UniversityUser).id
44       }
45     }).catch(() => null);
46   }, [deleteUniversityUser, row]);
47
48   if ((row as UniversityUser).role.name === UserRole.ADMIN) {
49     return null;
50   }
51
52   return (
53     <ModifyResourceAction
54       // Shared
55       resourceName={
56         (row as UniversityUser).user.firstName +
57         ' ' +
58         (row as UniversityUser).user.lastName
59       }
60       resourceType="User"
61       // Update
62       updateForm={({ onSuccess }) => (
63         <UpdateUniversityUserForm
64           universityUserId={(row as UniversityUser).id}
65           onSuccess={onSuccess}
66         />
67       )}
68       // Delete
69       deleteSubtitle="Data about the user will be lost, you will not be able to recover it"
70       onDelete={handleDeleteUniversityUser}
71       deleteLoading={deleteUniversityUserLoading}
72     />
73   );
74 };
75
76 const columns: GridColDef[] = [
77   {
78     field: 'avatar',
79     headerName: 'Avatar',
80     align: 'center',
81     headerAlign: 'center',
82     disableColumnMenu: true,
83     sortable: false,
84     valueGetter: ({ row }) => (row as UniversityUser).user.avatar,
85     renderCell: UserAvatar

```

```

86   },
87   {
88     field: 'firstName',
89     headerName: 'First Name',
90     width: 150,
91     valueGetter: ({ row }) => (row as UniversityUser).user.firstName,
92     sortComparator
93   },
94   {
95     field: 'fatherInitial',
96     headerName: 'Father Initial',
97     width: 125,
98     valueGetter: ({ row }) => (row as UniversityUser).user.fatherInitial,
99     valueFormatter: ({ value }) => (value as string) + '.',
100    sortComparator
101  },
102  {
103    field: 'lastName',
104    headerName: 'Last Name',
105    width: 150,
106    valueGetter: ({ row }) => (row as UniversityUser).user.lastName,
107    sortComparator
108  },
109  {
110    field: 'role',
111    headerName: 'Role',
112    width: 125,
113    valueGetter: ({ value }) => (value as UniversityUser['role']).name,
114    valueFormatter: ({ value }) => formatUserRole(value as UserRole),
115    sortComparator
116  },
117  {
118    field: 'email',
119    headerName: 'Email',
120    flex: 1,
121    valueGetter: ({ row }) => (row as UniversityUser).user.email,
122    sortComparator
123  },
124  {
125    field: 'actions',
126    headerName: 'Actions',
127    align: 'center',
128    headerAlign: 'center',
129    disableColumnMenu: true,
130    sortable: false,
131    renderCell: UserActions
132  }
133 ];
134
135 interface UniversityUsersTableProps {
136   universityUsers: UniversityUsersQuery['universityUsers'];
137 }
138
139 export const UniversityUsersTable: FC<UniversityUsersTableProps> = memo(
140   function UniversityUsersTable({ universityUsers }) {
141     return (
142       <DataGrid
143         rows={universityUsers}
144         columns={columns}
145         autoHeight
146         autoPageSize
147         disableSelectionOnClick
148       />
149     );
150   }
151 );

```