Universitatea “Dunărea de Jos” din Galați
Facultatea de Automatică, Calculatoare, Inginerie Electrică și Electrotehnică
Specializarea: Calculatoare și Tehnologia Informației

LUCRARE DE LICENȚĂ

Coordonator,
Prof. univ. ing Diana Ștefănescu

Absolvent,
Andrei Luca

2015

Universitatea “Dunărea de Jos” din Galați
Facultatea de Automatică, Calculatoare, Inginerie Electrică și Electrotehnică
Specializarea: Calculatoare și Tehnologia Informației

Teach Me
Aplicație web folosind suita de framework-uri M.E.A.N

Coordonator,
Prof. univ. ing Diana Ștefănescu

Absolvent,
Andrei Luca

2015

REZUMATUL LUCRARII

Lucrarea “Teach Me - Aplicație web folosind framework-urile M.E.A.N” reprezintă în primul rând un studiu asupra tehnologiei web de ultimă oră. Framework-urile folosite în această lucrare reprezintă viitorul în ceea ce priveşte dezvoltarea web. Având acces aproape în permanență la o conexiune la Internet, este necesar să venim în întâmpinarea utilizatorilor cu aplicații cât mai folositoare, accesibile de oriunde, cu o interfață atrăgătoare şi simplă de utilizat.

Pentru prezentarea şi studiul asupra acestor tehnologii web, a fost dezvoltată o aplicație ce simplifică comunicarea dîntre profesor şi student în cadrul unei facultăți. Partea scrisă a acestui proiect cuprinde definirea cerințelor şi a specificațiilor, obiectivele aplicației, dar şi prezentarea tehnologiilor folosite. Pe lângă acestea, a fost documentat şi procesul de concepție şi proiectare dar şi procesul de implementare. La finalul lucrării a fost prezentat manualul utilizatorului.

Exemplificarea acestor concepte s-a făcut prin partea aplicativă a acestei lucrări, aplicația funcționând în felul următor : atât profesorul cât şi studentul îşi vor crea un cont în cadrul platformei, unde după autentificare vor avea acces la diferite facilități ale aplicației. Meniul studentului va fi diferit față de cel al profesorului, atât prin elementele componente cât şi prin funcționalitate. Comunicarea presupune mai mult decât o conversație, de aceea aplicația permite şi transmiterea diferitelor informații de la profesor la student cum ar fi notele obținute în urma unor teste efectuate în cadrul platformei, sau a informațiilor legate de cursurile disponibile.

Aplicația “Teach Me” poate fi accesată de pe orice dispozitiv cu o conexiune la Internet, baza de date în care sunt stocate toate informațiile fiind găzduită de asemenea pe Internet, datele aflându-se în permanență în sincronizare.

Această lucrare a fost realizată sub îndrumarea doamnei Prof. univ. ing Diana Ştefănescu, din cadrul Facultății de Automatică, Calculatoare, Inginerie Electrică şi Electronică, Galați, căreia doresc să-i mulțumesc pentru interesul manifestat şi materialele de specialitate oferite.

CUPRINS

http://www.stumbleupon.com/su/9pM63c/!I9nN9f9:LaSXqWGb/www.holub.com/goodies/uml/index.html

INTRODUCERE

În societatea modernă a zilelor noastre în care educația trebuie să ocupe un loc important, se ridică problema comunicării dîntre profesor şi student. Profesorul clasic, acel personaj sobru şi autoritar, începe să fie înlocuit de profesorul democratic ce se implică în formarea studentului, nu doar în informarea lui. Este responsabilitatea profesorului să creeze un climat ce permite o comunicare liberă, deschisă, asigurând confortul necesar în timpul conversației.

În aceeaşi societate modernă, tehnologia informației are parte de o dezvoltare rapidă la care trebuie să ne adaptăm. Dacă în trecut existau manualele în format fizic, pe hârtie, astăzi se introduc încă din clasele primare manualele electronice ce pot fi descărcate cu uşurință pe tablete sau smartphone-uri.În contextul în care progresul tehnologic ne permite să fim conectați aproape în permanență la Internet, cursurile clasice încep să le ia locul celor online, prin video-urile înregistrate chiar de profesori, iar seminariile şi examenele scrise pe foaia de hârtie sunt înlocuite de examenele online. Astfel intervine comunicarea online, prin e-mail-uri, rețele de socializare, software-uri ce permit chat-ul în rețelele conectate la Internet sau rețele locale (LAN).

În această lucrare este propusă o platformă web ce permite comunicarea între profesor şi student prin intermediul unui chat, examinarea studenților prin diferite interactivități dar şi notarea acestora, toate acestea fiind oferite în cadrul aceleiaşi platforme. Pe lângă aceste facilități, studentul va putea vizualiza notele obținute în catalogul online şi se va putea înregistra la cursurile introduse de către profesor.

M.E.A.N reprezintă o suită de tehnologii ce au fost folosite în dezvoltarea acestei aplicații : MongoDB, Express.js, AngularJS, Node.js.

Comunitatea dezvoltatorilor de aplicații web a fost foarte receptivă în momentul în care aceste tehnologii au fost introduse, dovadă fiind numărul foarte mare de pachete şi librării ce au fost dezvoltate într-un timp atât de scurt, dar şi adoptarea lor de către marile companii consacrate deja în lumea web.

Pe lângă aceste motive, dorința de a învăța aceste tehnologii şi dorința de a-mi extinde cunoştințele în ceea ce priveşte aplicațiile web au condus la alegerea acestei teme.

Această lucrare, “Teach Me - Aplicație web folosind framework-urile M.E.A.N”, este un studiu asupra acestor noi tehnologii bazate în principal pe JavaScript, exemplificarea fiind făcută în cadrul unei aplicații care cred că este de un interes ridicat în lumea universitară.

CAPITOLUL I. DEFINIREA OBIECTIVULUI LUCRĂRII

DEFINIREA CERINTELOR SI A SPECIFICATIILOR

Primii paşi pentru a dezvolta aplicația „Teach Me - Aplicație web folosind framework-urile M.E.A.N” au presupus definirea cerințelor şi a specificațiilor pentru a facilita dezvoltarea lucrării.

A urmat apoi un studiu asupra tehnologiilor şi ulterior înțelegerii şi aprofundării acestora a fost concepută o versiune finală a acestor cerințe, prezentată în cele ce urmează.

CERINTE:

- Crearea unei aplicații web accesibilă pe Internet;
- Crearea conturilor utilizatorilor;
- Crearea meniurilor diferențiate în funcție de tipul utilizatorului;
- Crearea unui panou de control pentru profilul profesorului;
- Crearea unei interfețe grafice sugestivă, atrăgătoare, simplă de utilizat.

SPECIFICATII:

Acest produs software îşi propune să țină evidența studenților dar şi a profesorilor din cadrul unei Facultăți. Datele stocate referitoare la utilizator vor cuprinde numele, prenumele, adresa de e-mail dar şi tipul contului (student / profesor).

Produsul trebuie să permită crearea unui cont (student sau profesor), prin introducerea adresei de e-mail, a numelui şi a prenumelui dar şi a unei parole. Conectarea în cont se va face prin adresa de e-mail şi parolă.

Din punctul de vedere al unui profesor, aplicația permite :

- Vizualizarea sumară a informațiilor legate de cursurile ce au loc în ziua curentă, a mesajelor primite pe perioada “offline” a contului, dar şi a utilizatorilor înregistrați la cursuri;
- Introducerea unui curs prin specificarea materiei, a sălii, a zilei şi a orei de începere a cursului, a numărului de credite ce poate fi obținut în urma absolvirii cursului dar şi o descriere sumară a cursului;
- Introducerea unei materii ce va fi folosită în momentul în care se introduce un curs, prin introducerea numelui materiei;

- Posibilitatea de a permite/bloca accesul unui student la diferite teste predefinite (testele pot fi de tipul unui examen, în care utilizatorul are un număr predefinit de încercări sau de tipul unei teme în care utilizatorul nu are un număr maxim de încercări)
- Introducerea evenimentelor în cadrul unui calendar, evenimente ce vor putea fi vizualizate doar de către posesorul contului;
- Afişarea notelor pentru un utilizator înregistrat la unul din cursurile sale;
- Introducerea notelor pentru un anumit student în mod clasic, prin selectarea numelui materiei şi a notei
- Afişarea studenților ce sunt înscrişi la cursurile introduse de către profesor
- Modificarea datelor personale : nume, prenume
- Introducerea unei poze de profil

Din punctul de vedere al unui student, aplicația permite :

- Vizualizarea sumară a informațiilor legate de mesaje primite pe perioada "offline" a contului, cursurile ce au loc în ziua curentă, dar şi a examenelor la care utilizatorul are acces
- Introducerea evenimentelor în calendar, evenimente ce vor putea fi vizualizate doar de către posesorul contului;
- Vizualizarea notelor
- Filtrarea notelor după numele materiei
- Afişarea cursurilor disponibile, înregistrate în sistem de către toți profesorii
- Înregistrarea la un curs
- Afişarea informațiilor legate de un curs
- Accesarea task-urilor de tip "examen" sau "tema"
- Modificarea datelor personale : nume, prenume
- Introducerea unei poze de profil

Sub-meniul "Chat" permite :

- Conectarea la chat prin accesarea sub-meniului "Chat", implicit utilizarea chat-ului
- Recepționarea mesajelor de la un utilizator
- Trimiterea mesajelor către un utilizator chiar dacă utilizatorul nu este conectat în momentul respectiv (mesaje de tip "offline")

- Recepționarea mesajelor de la un utilizator într-o casetă de tip “pop-up”, chiar dacă pagina curentă nu este cea a chat-ului
- Afişarea tuturor utilizatorilor
- Afişarea status-ului utilizatorilor (online / offline)

OBIECTIVELE LUCRARII

Lucrarea “Teach Me - Aplicație web folosind framework-urile M.E.A.N” are ca obiectiv demonstrarea uşurinței cu care se pot folosi aceste tehnologii prin prezentarea informațiilor într-o aplicație web cu o interfață “user-friendly”, folosind Bootstrap pentru a permite ca informațiile să fie vizibile într-un mod cât mai plăcut indiferent de dimensiunea ecranului : smartphone, tabletă, monitor desktop.

Un alt obiectiv este să permită accesul gratuit la aplicație dar şi disponibilitatea online şi de asemenea, actualizarea şi trimiterea mesajelor între utilizatori în cadrul “Chat-ului” în timp real.

CAPITOLUL II. ANALIZA PROBLEMEI

2.1 INTRODUCERE

Aplicația prezentată în această lucrare, pentru a putea fi clasificată ca o aplicație web, necesită în primul rând un server web. În urma analizei problemei, s-a decis că folosirea platformei Node.js va facilita crearea unui astfel de server. Server-ul (computer-ul) ce va găzdui întreaga aplicația va trebuie să aibă instalat Node.js pentru ca aplicația să poată rula.

Aplicația dispune de mai multe rute pe care un utilizator le poate accesa, fie ele restricționate pentru utilizatorii autentificați, sau publice, accesibile tuturor tipurilor de utilizatori (autentificați / anonimi). Tehnologiile folosite în acest sens au fost AngularJS împreună cu Express.js, ambele fiind framework-uri ce permit ca întreg codul să poată fi scris în JavaScript, fiind concepute special pentru aplicații web.

Interfața grafică a fost dezvoltată atât cu ajutorul Bootstrap-ului, dar şi folosind AngularJS pentru a manipula informațiile recepționate de la server sau datele introduse de către utilizator. HTML şi CSS au stat la baza paginilor web ce sunt afişate la accesarea oricărei rute din cadrul aplicației.

Informațiile vor putea fi stocate în interiorul unei baze de date. În prealabil, au fost concepute scheme ale obiectelor ce vor putea fi introduse şi alegerea naturală a fost MongoDB, în detrimentul MySQL.

Următorul pas în analiza problemei a constat în înțelegerea noțiunilor de autentificare, persistența datelor, dar şi persistența conexiunii în momentul în care utilizatorul s-a autentificat cu datele folosite la crearea contului, în cadrul aceluiaşi browser. Datele de autentificare au fost salvate într-un “cookie” al browser-ului, nefiind necesară repetarea procesului de conectare în cazul în care browser-ul a fost închis.

Numărul mare de conexiuni simultane ce se aşteaptă în cadrul acestei platforme a condus către alegerea platformei Node.js, care este de tip non-blocking. Această tehnologie permite conexiuni multiple fără ca performanțele aplicației să scadă.

Tehnologia prezentată anterior permite folosirea limbajului de programare JavaScript pe partea de back-end a aplicației. Am luat în considerare şi faptul că pe partea de front-end datele vor fi manipulate cu AngularJS, framework ce impune folosirea acestui limbaj, de aceea alegerea personală a fost de a folosi această suită. Ca urmare a faptului că toate

componentele ei funcționează foarte bine împreună, ele au fost denumite “MEAN stack”.

În concluzie, analiza problemei a dus la următorul mod de funcționare al aplicației : utilizatorul îşi va crea cont în cadrul căruia îi vor fi puse la dispoziție mai multe funcționalități, informațiile introduse de către profesor sau de către student vor fi trimise către server sub formă unor obiecte de tip JSON, iar acolo ele vor fi procesate şi stocate în baza de date.

2.2 STUDIUL EXISTENTULUI

Analiza problemei presupune şi studiul existentului. În acest pas au fost descoperite aplicații similare şi serviciile folosite de aceste aplicații.

Majoritatea website-urilor, aplicațiilor web şi a platformelor web ce există astăzi pe Internet au la bază un cunoscut limbaj de programare web, PHP şi folosesc bazele de date MySQL. Acest pachet de software-uri a fost folosit pentru mult timp dar, alături de avantajele oferite au fost şi nişte dezavantaje, cum ar fi apelurile sincrone către server.

Aplicațiile următoare nu au la bază Node.js, ceea ce aduce un avantaj major aplicației “Teach Me”, prin utilizarea callback-urilor şi a apelurilor non-blocking.

În urma sondajelor efectuate pe diverşi studenți şi absolvenți ai Facultăților din România sau chiar din Marea Britanie, am observat că deşi există o platformă web unde studenții pot intra să obțină informații despre cursuri, ei nu pot comunica efectiv cu profesorii în cazul unor nelămuriri sau eventuale întrebări. De asemenea lipseşte un catalog online unde un student să poată urmări progresul propriu prin notele obținute la diferite materii.

2.2.1 EDU.CSED.UGAL.RO

Deşi „Departamentul de Calculatoare şi Tehnologia Informației al Facultății de Automatică, Calculatoare, Inginerie Electrică şi Electronică de la Universitatea Dunărea de Jos Galați” dispune de un site educațional care are o parte din facilitățile aplicației „Teach Me”, multe din Facultățile acestui oraş rămân fără o astfel de platformă online.

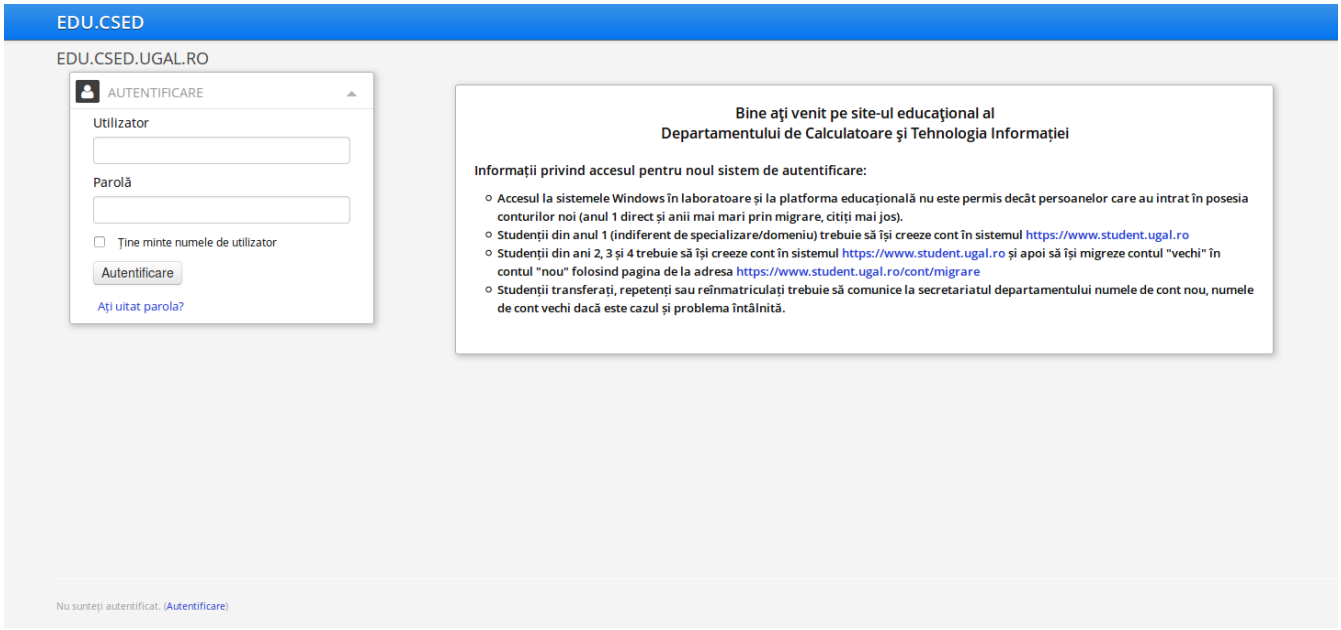

Figura 2.2.1.1 Ilustrarea platformei EDU.CSED.UGAL.RO

Chiar şi față de acest site educațional, aplicația prezentată în această lucrare prezintă câteva îmbunătățiri majore : facilitatea de comunicare între studenți sau între un student şi un profesor se realizează instant, “real-time”. Vizualizarea situației şcolare se face pe acelaşi site, nefiind nevoie de o conectare pe un site diferit pentru a afla notele.

Pe lângă aceste lucruri se mai adaugă şi partea de organizare a diferitelor evenimente ce se poate face accesând meniul Calendar atât dintr-un cont de student cât şi dintr-un cont de profesor. O altă funcționalitate şi totodată un plus față de alte platforme de acest gen este reprezentată de posibilitatea oferită studentului de a se înscrie la cursurile dorite în limita a unui număr de credite / semestru.

2.2.2 TEACHERS.NET

Teachers.net (http://teachers.net/) este o aplicație web ce permite comunicarea între profesori sau între profesori şi studenți. Prezența reclamelor pe acest site, a interfeței complicate şi foarte încărcate duc la obosirea utilizatorului şi în scurt timp la renunțarea de a mai folosi în continuare această platformă. Teachers.net nu prezintă un rezumat a ceea ce se poate realiza în cadrul aplicației.

Figura 2.2.2.1 Ilustrarea aplicației web Teachers.net

2.2.3 MY STUDY LIFE

My Study Life (https://www.mystudylife.com/) este o aplicație web adresată atât studenților cât şi profesorilor, dar îi lipsesc multe din funcționalitățile importante pentru un student. Un alt punct forte al aplicației „Teach Me" în comparație cu „My Study Life" este reprezentat de testele online ce pot fi accesate de către studenți în măsura în care profesorul le permite accesul la aceste teste.

Multe din aplicațiile deja existente permit utilizatorului doar să îşi programeze diferite evenimente într-un calendar şi posibilitatea de a fi atenționat cu o anumită perioadă înainte de începerea unui curs.

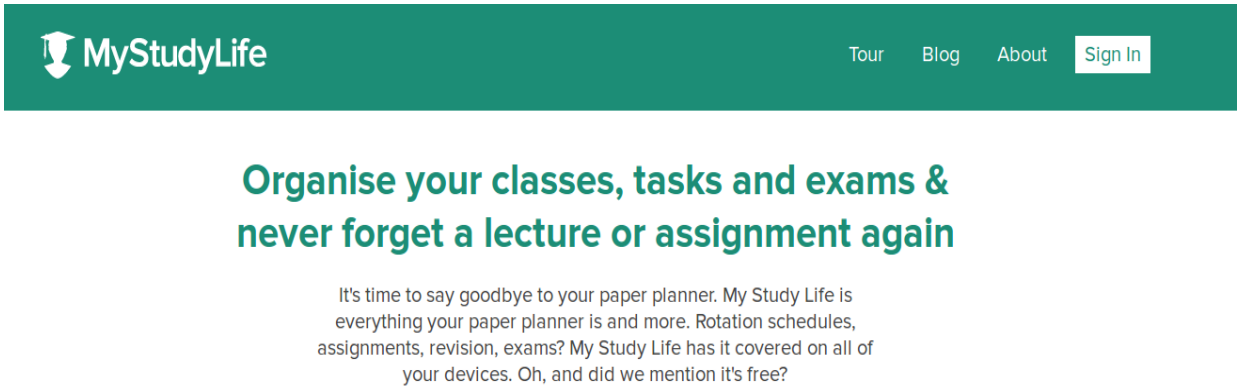

Figura 2.2.3.1 Ilustrarea aplicației My Study Life

Construită cu ajutorul unui framework pentru HTML şi CSS denumit Bootstrap, care este de fapt un fişier cu extensia .css, folosirea aplicației "Teach Me" pe telefonul mobil face ca interfața cu utilizatorul să rămână în continuare "user-friendly". În funcție de dimensiunea ecranului elementele din pagina HTML se vor repoziționa şi redimensiona, iar astfel aplicația nu va fi constrânsă doar de o anumită dimensiune şi rezoluție a ecranului ci va putea fi folosită pe orice dispozitiv cu o conexiune la Internet.

CAPITOLUL III. CONCEPȚIE ŞI PROIECTARE

3.1 MODELARE UML

Fişierele vor fi servite din folderul “public/modules” al aplicației. Fiecare modul al aplicației contine la rândul lui două foldere : “Controllers” – cel care va contine fişierele JavaScript ce conțin codul controller-ului, împreună cu toate metodele ce pot fi apelate – şi folderul “Templates” – cel care va contine fişierele HTML. Aplicația are mai multe module, iar acestea sunt :

- “Authentication”
- “Calendar”
- “Elev”
    - “Chat”
    - “Classes”
    - “Dashboard”
    - “Grades”
    - “Tasks”
    - “Home”
- “Prof”
    - “Chat”
    - “Classes”
    - “Dashboard”
    - “Grades”
    - “Tasks”
- “Settings”
- “Tests”

3.2 DESCRIEREA TABELELOR

Aplicația are nevoie de o bază de date în care să se stocheze informațiile introduse de utilizatori. În faza de proiectare, în momentul în care au fost concepute schemele pentru informațiile ce vor fi introduse a fost aleasă şi baza de date care corespundea cerințelor proiectului. Alegerea potrivită a fost MongoDB.

În completarea acestei baze de date s-a ales Mongoose. Mongoose este de fapt o librărie Node.js care oferă maparea obiectelor MongoDB în obiecte JavaScript pentru a putea fi folosite în aplicație.

Alegerea a fost făcută pe baza faptului că fiecare utilizator va avea mai multe informații : evenimente, note, cursuri la care un student s-a înscris sau care au fost adăugate de către un profesor. Toate aceste date au putut fi introduse în array-urile proprii ale obiectului "User" ("utilizator").

Deoarece aplicația se axează pe utilizator şi este modelată pentru a întâmpina cerințele acestuia, se poate spune că această colecție, Users, este principala colecție a aplicației "Teach Me". Pe lângă informațiile de contact ale utilizatorului şi alte obiecte stocate în array-uri în cadrul unui obiect de tip User, au mai fost folosite şi alte variabile de control ale obiectului.

Aplicația mai foloseşte şi alte colecții pentru a stoca mesajele din cadrul chat-ului ("Chat"), materiile şi cursurile introduse de profesori ("Classes" şi "Subjects"), sau temele şi examenele la care utilizatorii au acces ("Tasks"). În momentul în care se realizează autentificarea cu succes, se creează o sesiune a utilizatorului, ce va fi salvată în cookie-ul browser-ului dar şi în colecția "User_Session".

În operațiile de inserare sau de căutare au fost folosite API-urile de la MongoDB cât şi de la Mongoose.

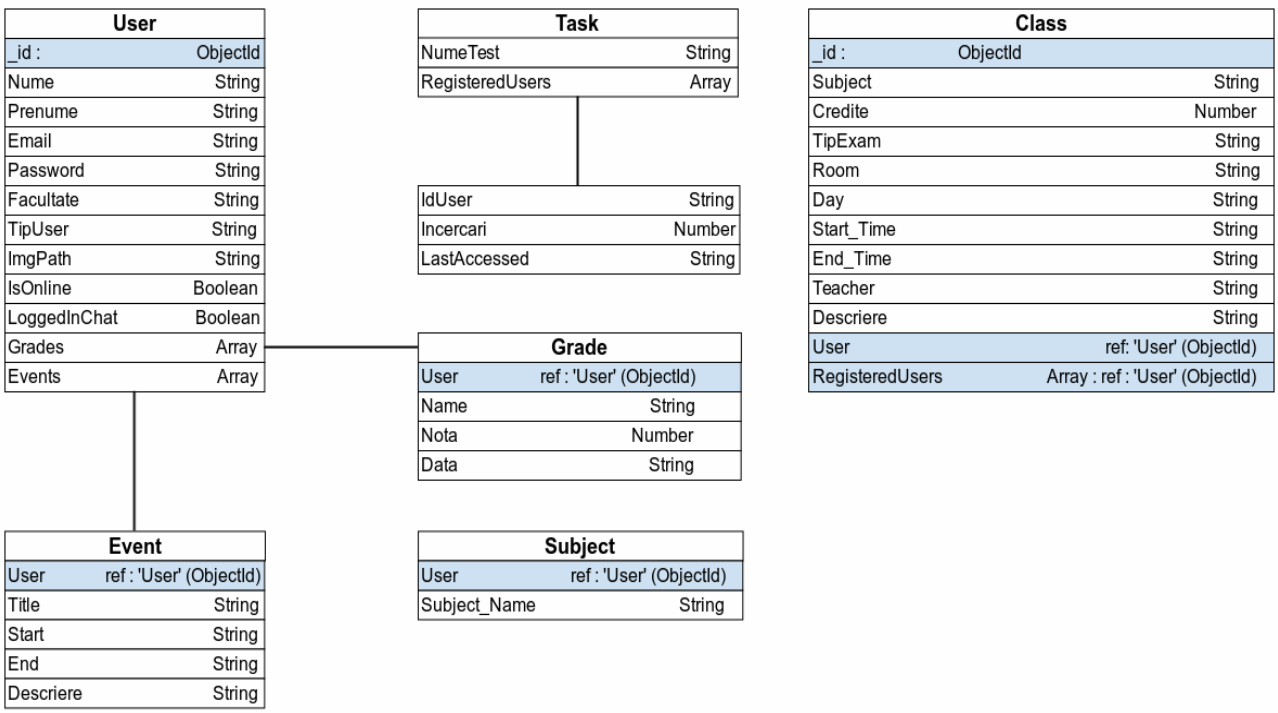

Figura 3.3.1 Schema bazei de date

CAPITOLUL IV. IMPLEMENTARE

Partea aplicativă a lucrării, aşa cum a fost menționat, foloseşte o combinație de tehnologii ce reuşesc să se completeze pentru a prezenta în final un produs software complet, sigur, scalabil, uşor de folosit.

Tehnologiile prezentate s-au dovedit a fi foarte uşor de utilizat. Unele din ele au fost dezvoltate de către Google sau sunt întreținute la momentul actual de către Google. API-ul foarte bine explicat atât prin definiții cât şi prin exemplele oferite direct pe site-ul deținătoarelor tehnologiilor au facilitat dezvoltarea aplicației. Pe lângă acestea, comunitatea de dezvoltatori este una în continuă creştere, fiind foarte primitori chiar şi cu începătorii, răspunzând la întrebările acestora.

Dovadă a acestei comunități stă numărul foarte mare de librării, plugin-uri şi pachete open-source oferite spre utilizare către toți dezvoltatorii. GitHub, despre care voi aminti şi mai târziu, găzduieşte multe proiecte open-source. Unele dîntre cele mai populare, cele mai căutate şi descărcate pachete şi proiecte sunt cele dezvoltate pentru aplicații care au ca platformă Node.js.

4.1 TEHNOLOGII FOLOSITE

În continuare se vor prezenta tehnologii, concepte şi limbaje de programare utilizate la dezvoltarea lucrării: JavaScript, MongoDB, Express.js, AngularJS, Node.js, Bootstrap, Git.

4.1.1 JAVA SCRIPT

JavaScript, cunoscut şi sub numele de ECMAScript, este un limbaj de programare dinamic. Asta înseamnă că în momentul rulării, acest limbaj de programare de nivel înalt, execută multe comportamente comune pe care limbajele de programare statice le realizează în momentul compilării. Aceste comportamente pot include extensia unui program prin adăugarea a noi linii de cod, sau extinzând obiectele din cadrul codului sursă.

Cel mai des este întâlnit în cadrul browserelor web a căror implementare permite script-urilor clienților să interacționeze cu utilizatorul, controlul browser-ului, comunicarea asincronă sau modificarea conținutului documentului afişat.

JavaScript este clasificat ca fiind un limbaj de scripting bazat pe prototipuri ce utilizează funcțiile de clasă primară. Asta înseamnă că limbajul permite trimiterea unei funcții ca parametru pentru altă funcție. Această combinație de funcționalități face ca JavaScript să fie un limbaj de programare multi-paradigmă. Scopul limbajelor de acest tip este să permită programatorilor să folosească cea mai bună componentă pentru un anumit task, recunoscând faptul că o singură paradigmă nu poate rezolva toate probleme în cel mai simplu şi eficient mod. Limbajul suportă programarea orientată-obiect, imperativă şi funcțională.

O paradigmă de programare este un stil fundamental de programare.
Severin Bumbaru a□□ Structuri de date, Algoritmi şi tehnici de programare- Editura Fundației Universitare,, Dunărea de josa□□a□□ Galați- 2002;

Programarea orientată obiect (POO, engleză OOP) este o paradigmă de programare, axată pe ideea grupării datelor (încapsulare) şi a codului care operează asupra lor într-o singură structură. Un alt concept important este polimorfismul, care permite abstractizări ce permit o descriere conceptuală mai simplă a soluției.

Principiile de bază ale programării orientate pe obiect sunt :
DE CĂUTAT ÎN CURSUL LUI SEVERIN BUMBARU PRINCIPIILE ŞI SCRIS CU CUVINTELE LUI ŞI CITAT LA SFÂRŞIT

Ideea programării orientată pe obiecte este de a crea programele ca o colecție de obiecte, unități individuale de cod care interacționează cu altele, în loc de simple liste de instrucțiuni sau de apeluri de proceduri.

Obiectele POO sunt de obicei reprezentări ale obiectelor din viața reală, astfel încât programele realizate prin tehnica POO sunt mai uşor de înțeles, de depanat şi de extins decât programele procedurale.

Programarea imperativă este o paradigmă de programare care descrie calculul ca instrucțiuni ce modifică starea unui program. Programele imperative sunt o secvență de comenzi pentru acționarea calculatorului. Programarea procedurală este o metodă obişnuită de executare a programării imperative. Procedurile, numite şi rutine, subrutine, metode sau funcții conțin o serie de paşi care trebuie executați. Orice procedură poate fi apelată la orice moment din execuția unui program, inclusiv de alte proceduri sau chiar de ea însăşi (recursivitate). Aşa cum a fost menționat mai sus, o procedură poate fi trimisă ca parametru unei alte proceduri în limbajul JavaScript.

JavaScript este folosit şi în alte aplicații care nu sunt bazate pe o conexiune la Internet, cum ar fi anumite widget-uri pentru desktop. Maşinile virtuale JavaScript mai noi şi mai

rapide şi platformele construite pe baza acestor maşini virtuale au crescut popularitatea acestui limbaj de programare în ceea ce priveşte aplicațiile web.

Pe partea de client, JavaScript a fost implementat în mod tradițional ca un limbaj interpretat, dar browserele mai recente au realizat compilare de tipul "just-in-time". Aceasta înseamnă că procesul de compilare se efectuează în momentul rulării, în timpul execuției unui program, şi nu înainte de execuția propriu-zisă a programului. De cele mai multe ori, aceasta constă în translatarea codului în cod-maşină care este apoi executat imediat. Acest tip de compilare este o combinație între cele două abordări tradiționale : compilare înainte de execuție şi interpretare. Apărut pentru prima dată acum 20 de ani, în 1995, a fost influențat de limbajele de programare C, Python, Java sau Scheme şi a influențat la rândul lui ActionScript, JScript .NET sau CoffeeScript.

Limbajul JavaScript este imperativ şi structurat. Multe din specificațiile acestui limbaj provin din C (structurile repetitive "while", "do-while", "for", structurile condiționale "if", "switch", etc), dar există şi o anumită diferență pentru cei obişnuiți cu limbajul de programare C : instrucțiunea ";" poate fi omisă în JavaScript.
Tipul unei variabile este asociat cu valoarea atribuită variabilei. De exemplu, unei variabile A îi poate fi atribuită inițial valoarea numerică "1", iar mai apoi i se poate atribui un şir de caractere "Acesta este un text".

JavaScript este aproape în întregime orientat pe obiecte. Proprietățile unui obiect pot fi accesate în două moduri : A.text = "Acesta este un text" sau A["text"] = "Acesta este un text". Datorită faptului că se bazează pe C multe din elementele de bază ale limbajului provin de la C.

4.1.2 NODE JS

Node.js este o platformă software folosită pentru a construi aplicații de rețea scalabile, scrisă în C, C++ şi JavaScript, lansată inițial la data de 27 Mai 2009.
Node.js este de tipul cross-platform, adică rulează independent de tipul maşinii şi este un mediu de dezvoltare folosit pe partea de server în cadrul unei aplicații. Aplicațiile Node.js pot fi rulate pe următoarele sisteme de operare : OS X, Microsoft Windows, Linux, FreeBSD, Nonstop şi IBM i. Ideea de cross-platform este atribuită unui software ce este implementat şi poate opera pe diverse platforme computer. Software-ul sau metodele cross-platform se mai numesc şi independente de platforma pe care rulează.

Termenul platformă se poate referi la tipul procesorului, “motorului” sau a altor componente hardware în cadrul căruia un sistem de operare sau o aplicație rulează. Un exemplu al unei platforme este Microsoft Windows ce rulează pe o arhitectură de tipul x86.

Node.js este un model de dezvoltare de tipul open-source, adică accesul este universal prin intermediul unei licențe gratis şi poate fi redistribuit de către oricine, chiar dacă au fost aduse modificări sau îmbunătățiri la modelul inițial.

Codul “open-source” este menit pentru a combina eforturile mai multor dezvoltatori de aplicații la care fiecare aduce îmbunătățiri şi pot distribui modificările în cadrul comunității. Această comunitate poate fi formată din programatori individuali dar şi mari companii.

Un foarte bun exemplu din categoria Software face parte sistemul de operare foarte bine cunocut : Android. Mentenanța acestui sistem de operare este realizată de către Google, dar natura foarte deschisă a companiei încurajează o imensă comunitate de dezvoltatori şi entuziaşti în a folosi codul lor open-source ca un fundament pentru diverse proiecte prin adăugarea unor funcționalități noi pentru utilizatorii mai avansați sau pentru a utiliza acest sistem de operare pe dispozitive care la lansare au avut alt sistem de operare (sisteme embedded).

Platforma Node.js foloseşte V8 JavaScript Engine dezvoltat de Google pentru browser-ul web Google Chrome. Prima versiune a acestui motor a fost lansată odată cu prima versiune a browser-ului creat de Google, Chrome, la data de 2 Septembrie 2008. Un motor JavaScript (JavaScript Engine) este o maşină virtuală ce interpretează şi execută cod JavaScript. Deşi se poate folosi în diverse medii, un motor JavaScript este folosit cel mai des în cadrul browser-elor web.

V8 înainte de a executa codul, îl compilează în cod maşină nativ (IA-32, x86-64, ARM sau MIPS ISA) , în loc să utilizeze tehnici mai tradiționale cum ar fi interpretarea codului sau compilarea întregului program în cod maşină şi să îl execute dintr-un fişier al sistemului. Codul compilat de către engine-ul V8 este optimizat şi re-optimizat în mod dinamic în momentul rulării.

Tehnicile de optimizare includ :

- inlining-ul - compilatorul generează codul corespunzător funcției în poziția apelului în loc să genereze secvența de apel în momentul rulării
- copy elision - compilatorul elimină copierea unor obiecte ce nu este necesară
- inline caching - memorarea rezultatelor obținute în urma apelurilor unor

metode

V8 poate fi folosit atât în cadrul unui browser (performanțele ridicate fiind în cadrul browser-ului Google Chrome) dar şi ca un motor de performanță ridicată, de sine stătător care poate fi integrat în proiecte independente, cum ar fi cod JavaScript pentru partea de server în Node.js.

V8 este creat special pentru web şi de aceea este eficient cu conceptele de bază ale internetului precum HTTP, DNS sau TCP.

Aşa cum s-a menționat anterior, Node.js foloseşte motorul Google V8 pentru a executa codul şi un număr mare de module de bază sunt scrise în JavaScript. Node.js are inclus în pachetul de bază o librărie ce permite aplicațiilor să se comporte ca un server Web fără a mai fi nevoie de software-uri adiționale cum ar fi Apache HTTP Server sau IIS.
Node.js capătă tot mai multă atenție şi este folosit de companii mari precum Microsoft, Yahoo!, Walmart, Groupon, LinkedIn sau Paypal.

O aplicație Node.js este scrisă în JavaScript.

Node.js oferă o arhitectură bazată pe evenimente şi un API (application programming interface) de tipul asincron (non-blocking I/O) ce permite altor calcule, procese să continue înainte ca transmisiunea inițială să se termine. - AICI MAI TREBUIE REFORMULAT, IAR LA PARTEA CU ARHITECTURĂ BAZATĂ PE EVENIMENTE AR TREBUI SĂ IAU TOT DIN CURSUL LUI BUMBARU DACĂ GĂSESC CEVA.

4.1.4.2.2 Interfețele bazate pe evenimente

Evenimentele descriu atunci când anumite acțiuni se petrec. Un eveniment poate fi o acțiune a utilizatorului, precum un click pe un element, sau poate fi un răspuns la o cerere HTTP. Când un utilizator interactionează cu un buton, acolo este o reacție, nu doar cu un eveniment care se petrece, ci cu mai multe. Există un eveniment atunci când se trece cu mouse-ul pe deasupra butonului, unul când se face click şi un eveniment atunci când cursorul părăseste butonul. Programatorul poate adăuga un ascultător de evenimente care să execute un anumit bloc de cod atunci când oricare din aceste evenimente se petrec.

Ascultarea de evenimente nu este strict legată de interfața grafică. Sunt şi evenimente de sistem care se petrec tot timpul. Când se face un apel HTTP, sunt evenimente ataşate statusului apelului care să asculte începutul, sfârsitul sau eşecul.

În 2011, un manager de pachete a fost introdus în librăria Node.js numit “npm”. Managerul de pachete permite comunității publicarea şi distribuirea librăriilor open-source Node.js şi simplifică foarte mult instalarea, updatarea şi dezinstalarea librăriilor.

Diferența majoră şi unul din motivele pentru care am ales Node.js în locul unui alt limbaj des întâlnit în programarea web, PHP, este că PHP este un limbaj de programare de tip blocking (comenzile se execută doar după ce comanda anterioară a fost terminată), în timp ce Node.js este un limbaj de programare non-blocking (comenzile se execută în paralel şi folosesc callback-urile pentru a semnala terminarea execuției unei comenzi). Aceasta înseamnă că Node.js operează pe un singur fir de execuție, folosind apeluri non-blocking, permițându-i să suporte zeci de mii de conexiuni concurente fără a afecta performanțele.

```
	//Librăriile din node.js se numesc module şi se încarcă acum modulul pentru HTTP
	var http = require(“http”);
	//Se creează un server HTTP nou care răspunde la toate requesturile cu un header 200 şi textul Hello World
	var server = http.createServer(function (request, response) {
		response.writeHead(200, {"Content-Type": "text/plain"});
		response.end("Hello World!n");
	});
	//Se ascultă pe portul 8000, adresa de IP default este 127.0.0.0
	server.listen(8000);
	console.log("Serverul rulează la http://127.0.0.1:8000/");
```

Se va salva acest cod într-un fişier numit hello.js şi se execută în linia de comandă : „node hello.js”, iar apoi se va deschide browser-ul la 127.0.0.0:8000. Ar trebui să apară Hello World!

4.1.3 MONGODB

MongoDB (litera M din acronimul M.E.A.N) este o bază de date NoSQL open-source orientată pe documente. Versiunea inițială a apărut în anul 2009 deşi dezvoltarea, începută de către compania 10gen, a avut loc în 2007.

MongoDB este o bază de date nerelațională, scrisă în C++, un avantaj major fiind faptul că nu există câmpuri predefinite spre deosebire de bazele de date relaționale, cum ar fi MySQL, unde există coloanele definite în momentul creării tabelelor. În MongoDB nu există o schemă pentru câmpurile unui document sau pentru tipurile lor de date, ele având

posibilitatea să varieze în funcție de ceea ce dorește utilizatorul să introducă; fiecare document trebuie să aibă totuşi un câmp "_id" care reprezintă cheia unică a acelui document - acest câmp poate fi creat fie de creatorul bazei de date, fie este generat automat după un algoritm astfel încât oricâte documente ar fi introduse, el nu se va repeta. Vom analiza mai târziu Mongoose, un ODM (Object Data Manager - sistem de organizare/manageriere a bazelor de date) pentru MongoDB ce permite crearea schemelor. În lucrarea aceasta, ambele concepte au fost ilustrate : atât folosirea MongoDB în forma clasică, cât şi executarea unor query-uri cu ajutorul Mongoose.

MongoDB memorează datele ca documente în format BSON (Binary JSON - format de interschimb şi transfer al datelor în rețea), avantajul fiind oferit de faptul că se reduce nevoia de join.

Crearea indecşilor secundari şi compuşi reprezintă un alt avantaj al MongoDB, orice atribut având posibilitatea de a fi indexat.

Mulți dezvoltatori ce lucrează cu baze de date sunt mult mai obişnuiți cu MySQL, de aceea în următoarele rânduri se va prezenta tabelul care realizează comparația între MySQL şi MongoDB :

Termeni MySQL	**Concepte MongoDB**
Bază de date	Bază de date
Tabelă	Colecție
Index	Index
Rând	Document BSON
Coloană	Câmp BSON
Join	Încapsulare și legatură
Cheie primară	Câmpul cheie unică _id
Group by	Agregare

Tabelul 4.1.3.1 Tabel comparativ între MySQL şi MongoDB

Dezvoltatorilor li se permite alegerea unei chei pentru “sharding”, care va determina ca interogările distribuite să fie rapide. Citirile şi scrierile vor fi distribuite pe partiții. Pe baza cheii datele vor fi distribuite în mai multe shard-uri (un shard master cu unul sau mai multe shard-uri slave).

Limbajul de interogare este mult îmbunătățit dar păstrează multe din principiile SQL şi C++.

Mongoose oferă o soluție bazată pe scheme ale documentelor ce permite o modelare facilă a datelor aplicației şi are inclusă built-in conversia de tip, validarea şi construirea interogărilor.

Totul în Mongoose începe cu Schema. Orice schemă se mapează la o colecție MongoDB şi defineşte forma documentelor din acea colecție.

AICI VA TREBUIE SĂ DAU UN EXEMPLU DE SCHEMA DE LA MINE DIN PROIECT, UN PRINTSCREEN CEVA

Tipurile de date permise în Mongoose sunt :

- String
- Number
- Date
- Buffer
- Boolean
- Mixed
- ObjectId
- Array

Pentru a putea folosi schema, ea trebuie convertită într-un Model. Modelul este un constructor compilat din definirea schemei. Instanțele modelelor reprezintă documentele care pot fi salvate în baza de date sau returnate din baza de date.

Returnarea documentelor este extrem de uşoară cu Mongoose, care foloseşte sintaxa foarte bogată din MongoDB. Documentele pot fi găsite cu ajutorul următoarelor funcții : find(), findById, findOne.

Ştergerea documentelor se poate face folosind metoda remove(), iar update-ul se poate face cu update() sau findOneAndUpdate().

În MongoDB se pot face modificări foarte rapide, apelând la următoarele operații :

- $inc - { $inc : { camp : valoare } } – incementează câmpul “camp” cu numărul dat ca valoare
- $set -{ $set : { camp : valoare } } – câmpul „camp” ia valoarea dată
- $push -{ $push : { camp : valoare } } – se adaugă valoarea dată câmpului specificat
- $addToSet -{ $addToSet : { camp : valoare } } – adaugarea valorii într-un vector

- $pop -{ $pop : { camp : 1 } } –șterge ultimul element dintr-un vector
- $pop -{ $pop : { camp : -1 } } –șterge primul element dintr-un vector
- $pull -{ $pull : { camp : _valoare } } – șterge toate valorile egale cu “valoare” din câmpul “camp”

Desigur, la toate metodele enumerate mai sus, trebuie specificat câmpul după care se face căutarea în momentul în care se execută un find, remove sau update.

MongoDB oferă un driver pentru Node.js şi astfel conexiunea dîntre platforma noastră şi baza de date se realizează mult mai uşor.

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://127.0.0.1/licentaDB');
var db = mongoose.connection;
```

Figura 4.1.3.1 Conectarea la baza de date “licentaDB”

Înainte de a continua cu introducerea în celelalte tehnologii, este necesar să vorbim despre REST, implicit API-ul REST care este creat în aplicația prezentată.
REST reprezintă un acronim pentru Representational State Transfer. Este un stil arhitectural a cărui implementare este inspirată din serviciile web. Protocolul pentru transferul informațiilor într-un serviciu REST este HTTP. REST foloseşte un model client-server, unde serverul este de tip HTTP iar clientul face apeluri HTTP folosind următoarele “verbe” : GET, POST, PUT, DELETE. Pe lângă aceste acțiuni, se trimite şi un URL (o rută) şi alte variabile ca şi parametri. URL-ul descrie ruta asupra căreia se vor face modificări, iar serverul va returna un cod de rezultat şi un obiect JSON (JavaScript Object Notation AICI AR TREBUI SĂ ARĂT UN PIC CE E ĂLA JSON)

Combinația dîntre Node.js şi MongoDB are un cuvânt de spus în acest moment. Datorită faptului că serverul răspunde cu un obiect JSON, manipularea este extrem de uşoară deoarece MongoDB interacționează foarte bine cu obiectele de acest tip.

Verbele HTTP se mapează foarte bine la cele patru funcții de bază ale stocării persistente - CRUD. CRUD este un acronim pentru CREATE, READ, UPDATE, DELETE. În tabelul următor se prezintă efectiv această mapare între operație, verbul HTTP şi corespondentul în SQL

Operatie	**SQL**	**HTTP**
Create	INSERT	PUT / POST
Read (Retrieve)	SELECT	GET
Update (Modify)	UPDATE	PUT / PATCH

Delete (Destroy)	DELETE	DELETE

Tabel 4.1.3.2 Tabel comparativ între numele operatiei, corespondentul in SQL si corespondentul in HTTP

- POST este folosit atunci când clientul doreşte să însereze sau să creeze un obiect în baza de date.
- GET este folosit când clientul doreşte să "citească" un obiect
- PUT este folosit când clientul doreşte să facă update la un obiect, iar DELETE este folosit în momentul ştergerii unui obiect.

Serverul răspunde şi cu un cod de rezultat. Cele mai comune sunt :

- 200 - "OK"
- 400 - "BAD REQUEST" - când lipsesc anumiți parametri din apel
- 401 - "UNAUTHORIZED" - când lipsesc parametrii de autentificare a celui care face apelul
- 403 - "FORBIDDEN" - când parametrii de autentificare există dar utilizatorul nu are destule drepturi pentru a face acest apel
- 404 - "NOT FOUND"

Rezumând, dacă cititorul este familiar cu tehnologia bazelor de date relaționale, se poate face o corespondență între acestea şi bazele de date nerelaționale. O bază de date MongoDB conține mai multe colecții, care sunt asemănătoare cu tabelele din bazele de date relaționale. Colecțiile conțin documente, iar fiecare document al colecției are ca şi corespondent un rând al unui tabel.

----------->>>>>>>>>>>>>>>>>> LA MONGODB AR TREBUI SA DAU UN EXEMPLU ELEMENTAR DE FIND CA SA ARAT SINTAXA. APOI AR TREBUI SA MAI DAU NISTE INFORMATII LEGATE DE COLECTII - CUM SE CREAZA, CUM SE STERG, CUM SE INTRODUC DATE INTR-O COLECTIE, CUM SE FACE UPDATE LA "COLOANE" (NU SE FACE UPDATE, TOCMAI ASTA E IDEEA, CA NU EXISTA O SCHEMA A DATELOR) ->> AR TREBUI ŞI CU CEVA EXEMPLE/PRINTSCREENURI DIN TERMINAL. GEN DROP PT STERGERE, ŞI USE MYDB - SE CREEAZA AUTOMAT

CAND DAU EXEMPLE LEGATE DE BAZA DE DATE AR TREBUI SA IAU CA ŞI EXEMPLU O COLECTIE DE-A MEA ŞI SA FAC TOATE EXEMPLIFICARILE PE ACEA COLECTIE

4.1.4 EXPRESS JS

Express.js este un framework pentru aplicații web, construit pentru a crea aplicații web cu pagini multiple sau cu o singură pagină. Acest framework este cross-platform, adică poate rula pe diverse sisteme de operare. Este folosit pe platforma Node.js şi a fost scris în JavaScript.

Acest framework oferă un set robust de funcționalități pentru aplicații web şi aplicații pentru mobil. Simplifică mult crearea unui API prin punerea la dispoziție a unei multitudini de metode pentru HTTP şi prin middleware. Middleware-ul poate fi definit ca un nivel de abstractizare în aplicații. Într-o altă exprimare se poate spune că middleware-ul reprezintă linia "-" dîntre client-server.

Avantajul Express.js este dat de faptul că nu ascunde funcționalități din Node.js ci permite încă folosirea lor, prin layer-ul subțire al funcționalităților de bază pentru aplicațiile web.

Express.js la fel ca şi MongoDB se poate instala cu ajutorul managerului de pachete, npm, utilizând comanda "npm install express".

Dacă mai devreme a fost exemplificat modul in care se poate afişa un simplu mesaj "Hello world !" doar cu Node.js, acum vom face acelaşi lucru folosind Express.

Odată ce a fost instalat Express.js, se poate folosi acest framework şi construi o aplicație :

```
var express = require("express");
var app = express();
app.get("/", function (req, res) {
      res.send("Hello World!");
});
var server = app.listen(3000, function () {
      var host = server.address().address;
      var port = server.address().port;
      console.log("Example app listening at http://%s:%s", host, port);
});
```

Aplicația porneşte un server care va aştepta conexiuni pe portul 3000. Dacă utilizatorul va intra pe prima pagină, aceasta fiind semnificată de rută "/", el va fi întâmpinat de mesajul "Hello World!". Pentru oricare altă rută, serverul va răspunde cu un cod 404 Not Found pentru că nu au fost definite alte rute.

Express.js nu blochează accesul la funcționalitățile de bază din Node.js. Obiectele “req” şi “res” sunt exact aceleaşi obiecte pe care Node.js le oferă aşa că se poate apela req.pipe(), req.on(“data”, callback) sau orice altă metodă ca şi cum nu ar fi implicat şi Express.js.

Se poate salva fişierul cu numele următor “app.js” iar aplicația va fi rulată cu ajutorul comenzii “node app.js”. Se va accesa într-un browser “http://localhost:3000” şi vom vedea mesajul.

4.1.5 ANGULAR JS

AngularJS este un framework pentru aplicații web de tipul open-source lansat prima dată în 2009, scris în JavaScript, putând rula pe diferite sisteme de operare, fiind creat de Google şi întreținut de către aceştia şi de către diferiți dezvoltatori şi corporații.
Este folosit în principal pentru aplicații de tipul single-page. În cadrul acestui tip de aplicații, fie tot codul necesar - HTML, JavaScript, CSS, fie toate resursele necesare sunt obținute dinamic la prima încărcare a paginii şi sunt apoi afişate după cerințe, cel mai des în funcție de interacțiunea utilizatorului cu interfața.

Acest framework îşi propune să simplifice procesul de dezvoltare şi testare al aplicațiilor punând la dispoziția dezvoltatorului un framework adaptat pentru o arhitectură de tipul MVC - Model-View-Controller.

În câteva rânduri, librăria AngularJS funcționează în modul următor : citeşte pagina HTML în care au fost introduse anumite tag-uri speciale sau atribute speciale în tag-uri deja cunoscute, le interpretează ca directive către care să conecteze părți de input sau output ale paginii la un model care este reprezentat de variabile JavaScript standard. Valorile pentru aceste variabile pot fi setate din cod sau pot fi obținute în mod static sau dinamic în format JSON.

Această librărie este distribuită sub forma unui fişier JavaScript şi poate fi adăugat la pagina web folosind tag-ul <script></script>.
Posibilitatea de extindere a vocabularului HTML, crearea de noi tag-uri adaptate cerințelor aplicației a făcut ca acest framework să fie foarte îndrăgit de comunitatea dezvoltatorilor.
Utilitatea este dată de faptul că fiecare funcționalitate poate fi modificată sau înlocuită pentru a se adapta cu procesul unic de dezvoltare al celui care foloseşte această librărie.

Design-ul AngularJS include următoarele :

- decuplarea manipulării DOM de logica aplicației. DOM (Document Object Model) este o convenție independentă de limbajul de programare folosit, fiind utilizat pe diverse sisteme de operare pentru a reprezenta şi a interacționa cu obiecte din documentele HTML, XHTML sau XML. Nodurile fiecărui document sunt organizate într-o structură tip arbore, denumită "arborele DOM". Obiectele din acest arbore pot fi selectate şi manipulate folosind metode asupra obiectelor.
- decuplarea părții de client a aplicației de partea de server. Aceasta permite ca dezvoltarea să progreseze în paralel, şi codul scris pentru ambele părți poate fi apoi reutilizat
- oferă structurarea aplicației : de la design-ul interfeței cu utilizatorul până la scrierea logicii aplicației şi până la testarea efectivă a aplicației

AngularJS implementează şablonul MVC pentru a separa componentele folosite pentru logică de cele pentru prezentare sau pentru datele utilizate.

El introduce conceptul de directivă, aşa cum am menționat anterior, directiva fiind un tag html sau un atribut al unui tag. Pentru a folosi o directivă, trebuie să folosim cuvântul cheie "ng". De exemplu, pentru a adăuga un Controller într-o aplicație vom folosi "ng-controller", pentru a adăuga un eveniment de tip click pe un tag vom folosi "ng-click", etc.

Modulele AngularJS pot fi văzute ca părți independente ale unei aplicații

Termenul de data-binding se referă la sincronizarea datelor dîntre view şi model (MV din modelul MVC). Modelul influențează datele view-ului, astfel propagarea datelor se face de la model la view. Conceptul nou adus de cei de la Google presupune că orice schimbare a modelului va afecta view-ul iar orice modificare a view-ului va fi propagată şi în model.

Unele din trăsăturile care ies în evidență atunci când vorbim despre Angular sunt : RESTEasy - acțiunile REST despre care am vorbit anterior devin standardul comunicării de la server la client. Cu o singură linie JavaScript se poate comunica cu serverul în vederea obținerii informației necesare pentru a interacționa cu pagina web creată.

MVVM - modelele comunică cu obiectele ViewModel prin obiectul numit "$scope". Oferă Controllere fără stare (stateless) care inițializează şi controlează obiectul $scope. Totodată, ANGULAR JS pune la dispoziția clienților săi o arhitectură opțională care furnizează un stil de organizare a codului după tipul Model-Vizualizare-Control (Model-View-Controller). Acesta este un model arhitectural utilizat în ingineria software, pentru crearea interfețelor utilizator, care împarte o aplicație în 3 arii de responsabilitate:

- modelul (model): structurile de date care reprezintă starea aplicației
- vizualizarea (view): care observă starea şi generează un produs de ieşire pentru utilizator
- control (controller): care traduce intrarea utilizatorului în operații asupra modelului

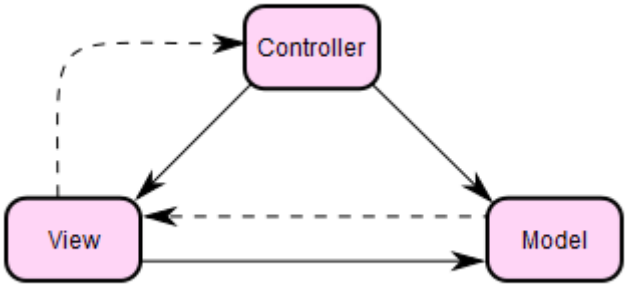

Figura 4.1.5.1 Model-Vizualizare-Control

4.1.6 GITHUB

GitHub este un serviciu de găzduire web utilizat în proiecte de dezvolare a produselor software şi utilizează un sistem de control al versiunilor denumit Git.
GitHub este un proiect inițiat de compania Google, lansat inițial în 2008. Acest serviciu oferă posibilitatea de a oferi codul în mod open-source şi astfel oferă posibilitatea utilizatorilor de a contribui la dezvoltarea unui anumit proiect.

GitHub este bazat pe un sistem de versionare denumit Git. Pe lângă viteză, integritatea datelor şi suportul pentru fluxul de lucru distribuit, GitHub vine cu funcționalități noi şi anume interfața grafică web, desktop sau pentru mobil, permite sau restricționează accesul utilizatorilor, poate urmări evoluția unei funcționalități din cadrul unui proiect şi în cazul apariției unui “bug” evoluția acestuia poate fi urmărită.

În acest proiect, pe parcursul dezvoltării lui a fost folosit GitHub pentru a se salva modificările şi îmbunătățirile aduse în fiecare fază a proiectului.

Stocarea se realizează remote aşa că dacă dintr-o greşeală suportul de stocare al proiectului suferă defecțiuni, munca nu este pierdută, ci ea este salvată în contul de GitHub. Astfel acest utilitar devine şi o resursă foarte bună de back-up. Dezvoltarea diferitelor funcționalități se poate face în paralel pe diferite ramuri ale proiectului, denumite în continuare “branch-uri”, care vor putea fi unite într-un anumit moment pe principalul branch de dezvoltare.

În primul rând se creează un repository (locație de stocare) într-un terminal cu ajutorul comenzii “git init”. În acest moment, în folderul respectiv se va crea un sub-folder denumit “.git” care conține mai multe fișiere de configurare a repository-ului.
Pentru a vedea statutul repository-ului se utilizează comandă “git status”. Dacă există fișiere care nu au fost adăugate la lista internă de fișiere care trebuie versionate ele vor apărea în acest moment. Pentru a adăuga aceste fișiere se va folosi una din următoarele comenzi :
“git add .” - pentru a adăuga toate fișierele care nu au fost încă adăugate la lista internă de fișiere
“git add <file>“ - <file> reprezintă calea către fișierul pe care dorim să îl adăugăm

Comanda “git add” are o lista lungă de opțiuni ce pot fi folosite pentru a adăuga fișiere.

După ce au fost adăugate fișierele, este timpul pentru primul commit, care se realizează cu următoarea comandă : “git commit [-m "<mesaj>"]”.
Parametrul -m este opțional, dar dacă este folosit trebuie să introducem și un mesaj între ghilimele duble care va reprezenta mesajul commit-ului. De obicei, în acest mesaj sunt menționate diferite modificări care au fost aduse la proiectul inițial în aceste fișiere modificate.

Pentru a vedea toate commit-urile făcute până în acel moment se poate utiliza comanda “git log”. Pentru a face upload la aceste fișiere în spațiul de stocare de pe GitHub se rulează comanda “git push”. Din terminal se vor cere datele de autentificare de la contul de GitHub și în urma unei autentificări corecte se va avea prima versiune a fișierelor în Git. Aceşti paşi se vor urma și la modificările ulterioare.

Dacă se dorește să ștergerea fișierelor deja stocate se folosește comanda “git rm <file>“.

Dacă altcineva a făcut un commit pe proiect, se poate utiliza comanda “git pull” și astfel toate datele din computerul pe care se lucrează, referitoare la proiectul respectiv, vor fi actualizate cu ceea ce este stocat în Git.

Dar, pentru ca altcineva să lucreze la același proiect, va trebui mai întâi să “cloneze” repository-ul cu ajutorul comenzii “git clone”

La toate comenzile prezentate mai sus se pot folosi diverși parametri pentru a simplifica acest proces.

```
# On branch master
#
# Initial commit
```

```
#
# Changes to be committed:
#   (use "git rm --cached ..." to unstage)
#
#   new file:   .gitignore
#   new file:   release.txt
```

4.1.7 BOOTSTRAP

Bootstrap este o colecție de instrumente utilizată pentru crearea site-urilor web şi a aplicațiilor web. Ajuns la versiunea 3, scris în HTML, CSS, LESS, Sass şi JavaScript acest framework conține template-uri bazate pe HTML şi CSS ce includ componente pentru o interfață grafică cum ar fi formulare, butoane sau elemente de navigație.
Bootstrap este compatibil cu ultimele versiuni de Google Chrome, Firefox, Internet Explorer, Opera şi Safari.

Bootstrap pune la dispoziție un set de fişiere de stil care oferă stilizare pentru toate componentele de bază ale HTML. Pe lângă aceste elemente esențiale din HTML, conține şi alte elemente des folosite într-o pagină web, cum ar fi butoanele drop-down, paginarea, tab-urile verticale şi orizontale, mesajele de avertizare sau barele de progres. Aceste componente sunt implementate ca nişte clase CSS care pot fi aplicate diferitelor elemente HTML.

A fost creat de cei de la Twitter şi poate fi utilizat pentru a gestiona cât mai bine faza inițială a unui proiect deoarece se pot folosi o serie de componente de bază pentru a nu fi nevoiți să începem de la zero.

Bootstrap propune un sistem de ghidaj, “grid system”, pentru o mai bună şi mai eficientă structurare a design-ului, oferind şi conceptul de pagini web “responsive”. Asta înseamnă că tot conținutul paginilor va putea fi vizualizat optim indiferent de dispozitivul folosit. Design-ul web va răspunde comportamentului şi mediului utilizatorului în funcție de rezoluția ecranului, platformă şi orientare.

În sistemul de ghidaj oferit de Twitter Bootstrap tot layout-ul este împărțit în doisprezece coloane. În funcție de dimensiunea ecranului vom putea folosi clasele col-xs-, col-sm-, col-md-, col-lg-.

Dacă ecranul dispozitivului este < 768px, telefoanele mobile incluzându-se în această categorie, vom folosi clasa .col-xs-.
Dacă ecranul dispozitivului este >= 768px, în general tablete, vom folosi clasele .col-sm-. O coloană va avea o lățime de aproximativ 62px.

Dacă ecranul dispozitivului este >= 992px, în general monitoarele computerelor, vom folosi .col-md-, caz în care o coloană va avea o lățime aproximativă de 81px.
Pentru dispozitive ce au dimensiunea ecranelor >=1200px, vom folosi .col-lg-, o coloană având lățimea de aproximativ 97px.

Aceste coloane pot fi poziționate şi utilizând offset-uri, având posibilitatea de a aranja conținutul pe mijlocul paginii. De exemplu, pentru a afişa o porțiune din conținut având o margine în partea stângă se poate folosi o combinație de două clase : “col-md-4 col-md-offset-4”.
AICI AR TREBUI UN TABEL. CEL DE PE SITEUL LOR E FOARTE BUN
http://getBootstrap.com/css/#grid-options
TOT AICI AR TREBUI SA VORBESC DESPRE INTERFATA ŞI SA NUANTEZ DESPRE BOOTSTRAP ŞI CLASELE FOLOSITE, FIŞIERELE CSS

CAPITOLUL V. MANUALUL UTILIZATORULUI

Realizarea aplicației a fost mult simplificată datorită utilizării AngularJS. Pe lângă îmbunătățirea performanței datorită faptului că fișierele sunt încărcate la prima rulare a website-ului, acest framework mi-a permis să dezvolt interfața grafică pe fragmente.

De exemplu, meniul a fost încărcat o singură dată, au fost create legături pe link-urile din meniu pentru a şti ce fragment al aplicației trebuie arătat, iar apoi toate aceste fragmente au fost dezvoltate separat, nefiind nevoie să adaug la fiecare pagină meniul, eliminând încărcarea sistemului.

În AngularJS, diferitele fragmente au fost afişate folosind tag-ul specific <ui-view></ui-view>. Conținutul paginii html care va trebuie afişat, va fi introdus în acest tag pereche.

```
<nav class="navbar navbar-default navbar-fixed-top">
        <div class="container">
            <div class="navbar-header page-scroll">
                <button type="button" class="navbar-toggle" data-toggle="collapse" data-target="#bs-example-navbar-collapse-1">
                    <span class="sr-only">Toggle navigation</span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                    <span class="icon-bar"></span>
                </button>
                <a class="navbar-brand" ui-sref="root.mainpage">Teach Me</a>
            </div>

            <div class="collapse navbar-collapse" id="bs-example-navbar-collapse-1">
                <ul class="nav navbar-nav navbar-right">
                    <li class="hidden">
                        <a href="#page-top"></a>
                    </li>
                    <li class="page-scroll">
                        <a href="#despre">Despre</a>
                    </li>
                    <li class="page-scroll">
                        <a href="#contact">Contact</a>
                    </li>
                    <li >
                     <a ui-sref="root.login">Autentificare</a>
                   </li>
                </ul>
            </div>
        </div>
    </nav>

<div ui-view></div>
```

Figura 6.1 Ilustrarea tag-ului pereche "<ui-view></ui-view>"

```
.state('root.login', {
    url: "login",
    templateUrl: "modules/Authentication/Templates/SignIn.html",
    controller: loginController,
    resolve: {
        userLoggedIn: function($http, $state, $rootScope) {
            return $http.get("/cookie").success(function(data) {
                if (data != null && data != "") {
                    userCredentials = data;
                    $rootScope.userCredentials = userCredentials;
                    if (data.tipUser == "teacher") {
                        $state.go('account_prof.dashboard');
                    } else {

                        $state.go('account_elev.dashboard');
                    }
                } else {
                    userCredentials = null;
                    $rootScope.userCredentials = null;
                }
            })
        }
    }
})
.state('root.signup', {
    url: "signup",
    templateUrl: "modules/Authentication/Templates/SignUp.html",
    controller: signUpController,
    resolve: {
        userLoggedIn: function($http, $state, $rootScope) {
            return $http.get("/cookie").success(function(data) {
                if (data != null && data != "") {
                    userCredentials = data;
                    $rootScope.userCredentials = userCredentials;
                    if (data.tipUser == "teacher") {
                        $state.go('account_prof.dashboard');
                    } else {

                        $state.go('account_elev.dashboard');
                    }
                } else {
                    userCredentials = null;
                    $rootScope.userCredentials = null;
                }
            })
        }
    }
});
```

Figură 6.2 Calea aleasă în funcție de interacțiunea utilizatorului cu interfața grafica

Fragmentul care trebuie afişat, este ales în momentul interacțiunii utilizatorului cu interfața prin click pe unul din elementele meniului.

Pentru a exemplifica acest concept, se observă cum tot conținutul paginii rămâne neschimbat în momentul în care se doreşte autentificarea - doar formularul specific va fi arătat.

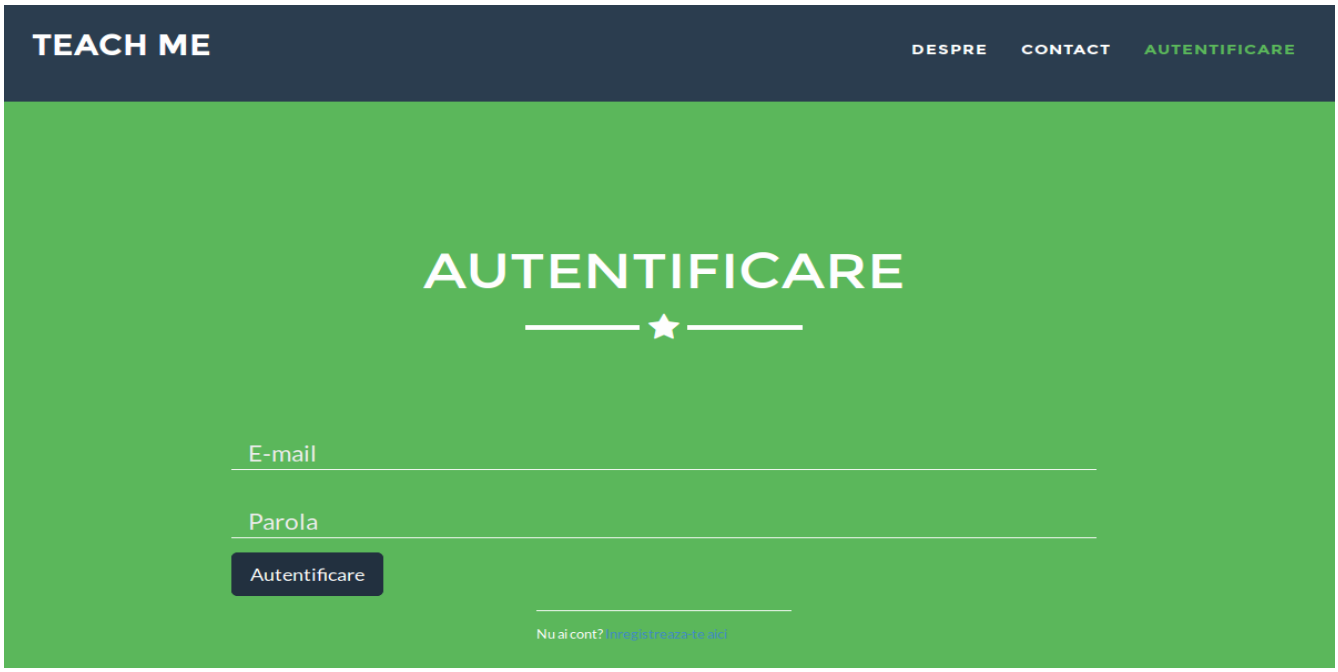

Figura 6.3 Fragmentul paginii de autentificare

Figura 6.4 Fragmentul paginii de înregistrare

În momentul în care utilizatorul doreşte să îşi creeze un cont, pe baza datelor introduse, se va face o căutare în baza de date pentru a elimina problema introducerii a două conturi cu aceleaşi date în baza de date. În cazul în care datele introduse de utilizator corespund cu ale altui cont deja înregistrat el va primi mesaje de eroare.

Diferențierea între contul profesorului şi contul studentului se va face printr-un checkbox, astfel accesul profesorului nu va fi restricționat de către un administrator de sistem.

Când autentificarea se realizează cu succes, utilizatorul va fi trimis către pagina lui de administrare. În acest moment, meniul studentului diferă față de meniul profesorului datorită diferenței de facilități oferite pentru fiecare tip de cont.

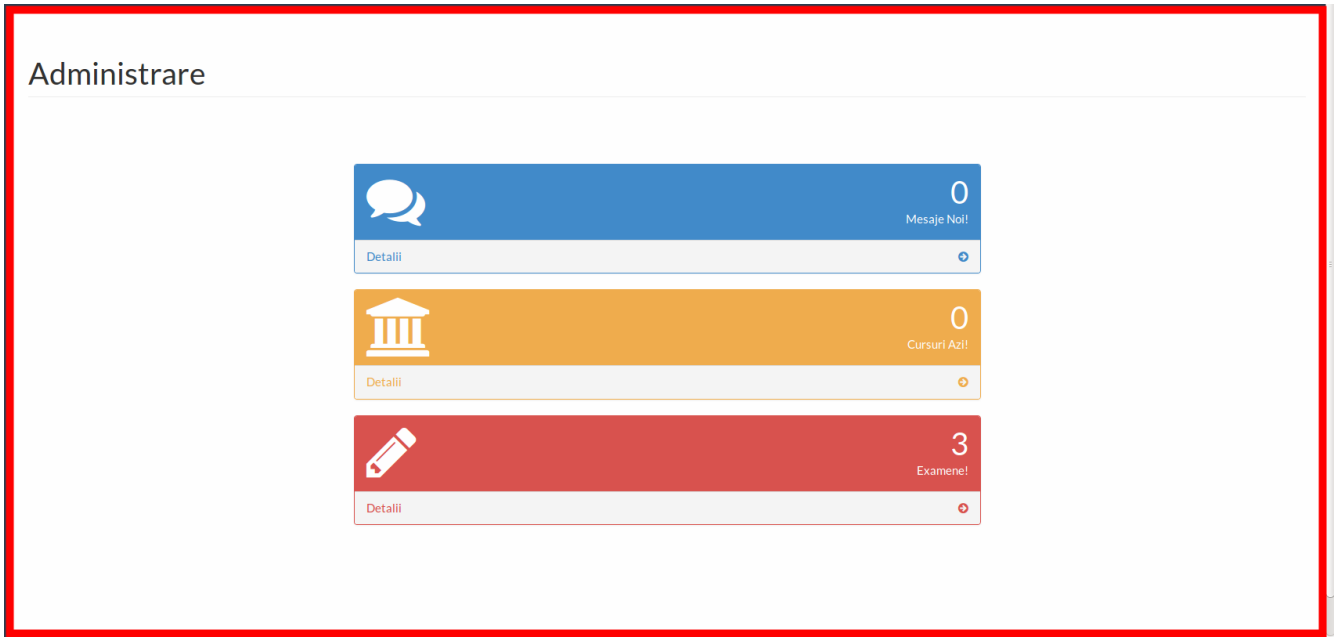

Figura 6.5 Conținutul care se va schimba

De asemenea, aceste meniuri au fost create o singură dată, singura porțiune care se va schimba pe tot parcursul conectării fiind zona evidențiată în captura de ecran prezentată mai jos :

AICI CRED CA AR TREBUI ALT PRINTSCREEN CU TOT CU MENIUL DE SUS ŞI DIN STANGA ŞI CU ZONA ASTA EVIDENTIATA

Meniul contului de student conține următoarele elemente : ACASA, CALENDAR, NOTE, CURSURI, CHAT, TASK – cu sub-meniurile TEME/EXAMENE

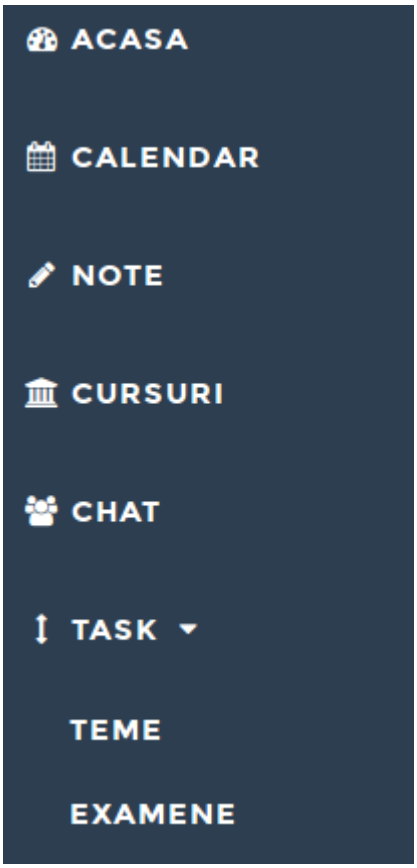

Figura 6.6 Meniu cont "student"

În continuare vom prezenta punctual ce facilități există în cadrul fiecăreia dîntre aceste opțiuni :

NOTĂ : FIECARE DENUMIRE DE SUBMENIU TREBUIE BOLDAT

FIECARE PRINTSCREEN SĂ NU CONȚINĂ ŞI MENIUL DIN STÂNGA

- În momentul în care utilizatorul apasă pe butonul "**ACASA**" el va fi redirecționat către pagina principală de administrare. Pe această pagină utilizatorului îi sunt prezentate informații sumare legate de cursurile care au loc în ziua curentă, mesajele noi recepționate cât utilizatorul a fost deconectat dar şi a numărului de "EXAMENE" din categoria "TASK" la care are acces.
- Accesarea submeniului "**CALENDAR**" va afişa utilizatorului un calendar sub formă de tabel în care el poate introduce diferite evenimente pentru o mai bună organizare a timpului. Atunci când utilizatorul efectuează click pe unul din evenimentele deja introduse, acesta va fi afişat într-o casetă popup. Pentru a adăuga un eveniment nou este de ajuns să se efectueze click pe una din zilele lunii şi un formular se va afişa pentru a introduce aceste date.

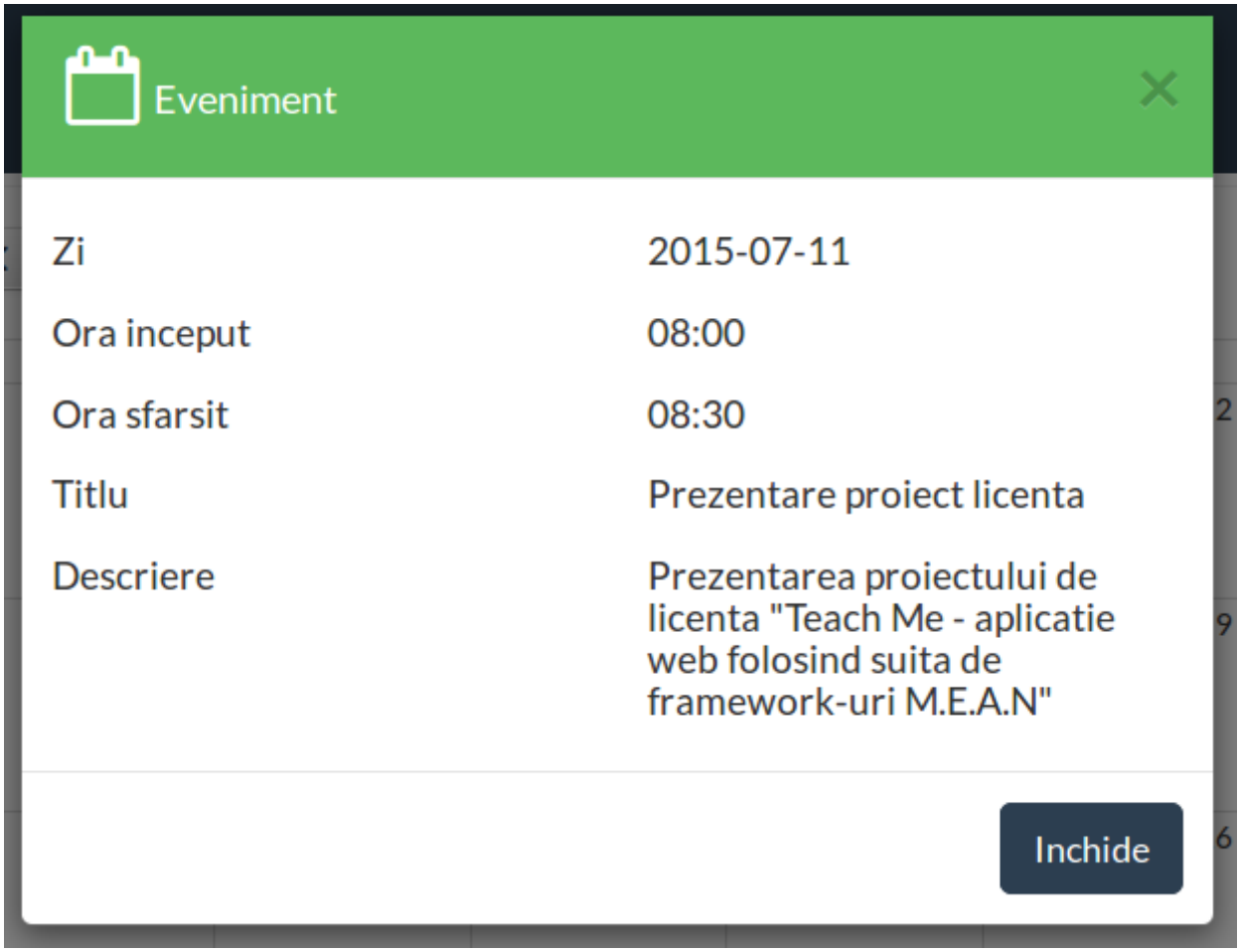

Figura 6.7 Prezentare eveniment din baza de date

- Vizualizarea notelor din catalog se va face accesând sub-meniul "**NOTE**". Informațiile sunt dispuse sub forma unui tabel în care sunt prezentate denumirea materiei şi nota. O funcționalitate extrem de ajutătoare pentru utilizator este reprezentată de caseta de deasupra tabelului ce permite filtrarea notelor în funcție de numele materiei.

Figura 6.8 Tabel note + bara de filtrare

- Aşa cum am menționat, utilizatorul se poate înscrie la ce cursuri dorește accesând sub-meniul "**CURSURI**". Aici, cursurile la care utilizatorul s-a înscris se vor diferenția de cele la care el nu s-a înscris prin culoarea de fundal a rândului din tabel şi a mesajului

"Eşti înregistrat"/"Nu eşti înregistrat".

- Dacă utilizatorul doreşte mai multe informații legate de un curs, el poate efectua click pe un rând din tabel şi toate informațiile îi vor fi prezentate printr-o casetă tip popup. Dacă el nu este încă înscris la cursul respectiv, fereastra de pop-up va avea butonul de "Înregistrare" activ.

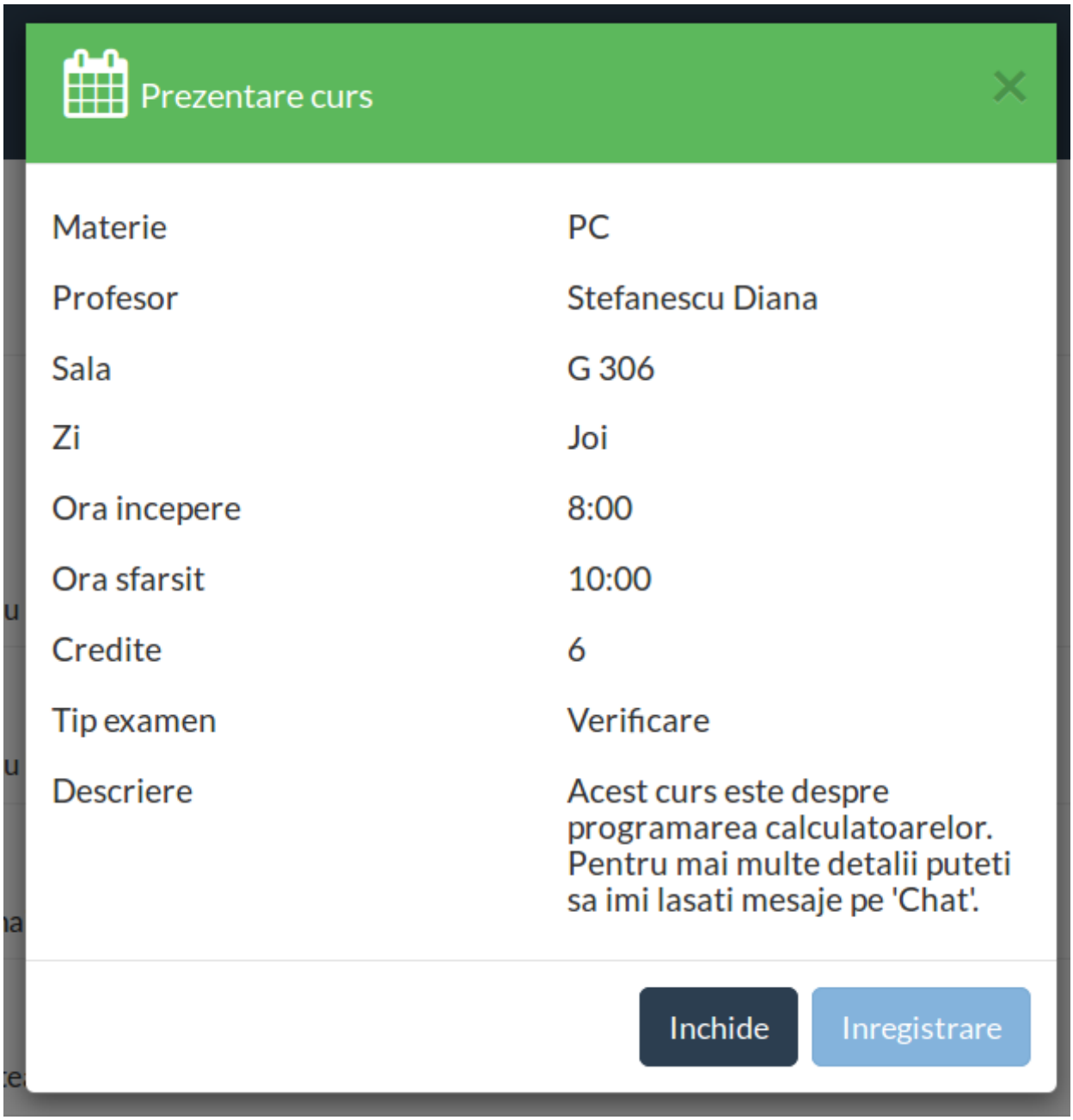

Figura 6.9 Prezentarea unui curs la care utilizatorul s-a înregistrat

- Sub-meniul "**CHAT**" va realiza conexiunea la chat la prima accesare a lui. Acest lucru va fi evidențiat prin afişarea temporară a mesajului "AI FOST CONECTAT LA CHAT". Dacă utilizatorul a recepționat mesaje în perioada cât a fost deconectat, o fereastră cu aceste mesaje va fi prezentă. Pe lângă aceasta, o fereastră cu lista

utilizatorilor va fi afișată.

- Disponibilitatea utilizatorilor va fi afișată printr-o bulină roșie în cazul în care utilizatorul nu este conectat şi printr-o bulină verde în cazul în care utilizatorul este conectat.
- Dacă utilizatorul primește mai multe mesaje de la utilizatori diferiți, doar o fereastră va putea fi activă la un moment dat. Celelalte ferestre vor fi dispuse sub forma unor tab-uri, şi dacă vor conține mesaje care nu au fost citite încă de utilizator, numele expeditorului din titlul tab-ului va avea culoarea roşie.

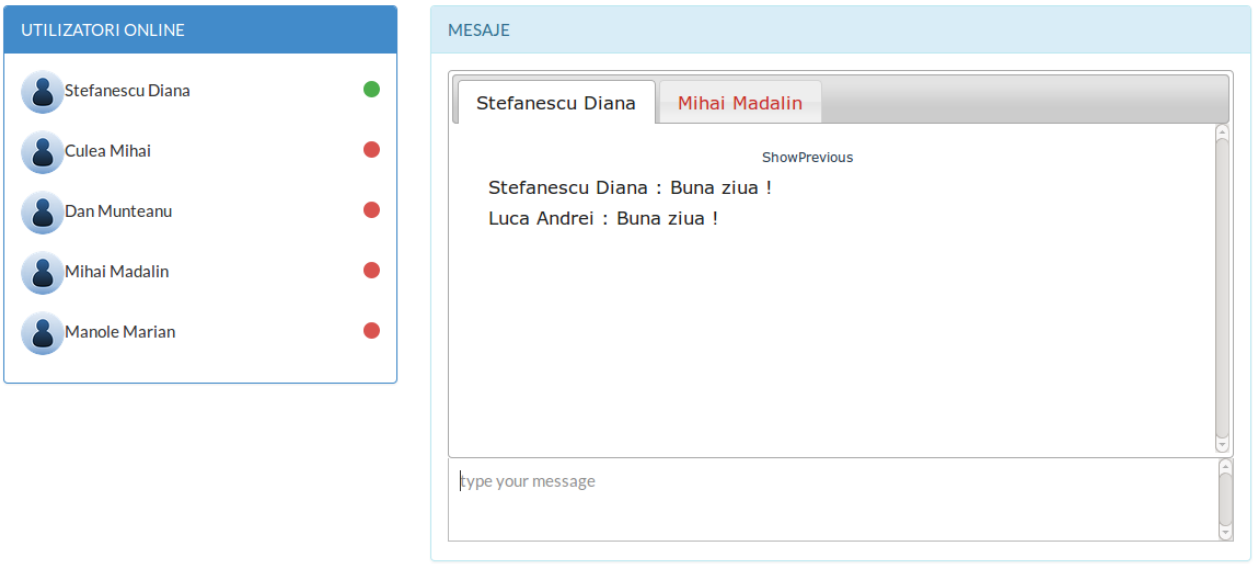

Figura 6.10 Chat

- Sub-meniul "**TASK**" are la rândul lui alte 2 opțiuni :
  - "**TEME**" va conține acele teste la care utilizatorul are un număr nelimitat de încercări
  - "**EXAMENE**" are teste la care utilizatorul va primi la final o notă şi are un număr finit de încercări : setat în mod predefinit la valoarea 3. Aceste teste conțin diverse interactivități : teste radio / checkbox sau teste "Drag and Drop" în care utilizatorul va trebui să tragă cu mouse-ul diferite elemente pe ecran pentru a le pune în poziția corespunzătoare. Diferența între testele tip "Tema" şi testele tip "Examen" este făcută prin prezența butonului "Verifica" pe care utilizatorul va trebuie să efectueze click pentru a-i fi validate răspunsurile.

Meniului contului de profesor nu diferă foarte mult de cel al studentului, în schimb ceea ce diferă e funcționalitatea fiecărei pagini. Aceste sub-meniuri vor fi prezentate în comparație cu meniul studentului. Astfel, profesorul dispune de următoarele : “ACASA”, “CALENDAR”, “CATALOG”, “CURSURI”, “CHAT”, “TASK” -> “TEME”,”EXAMENE”.

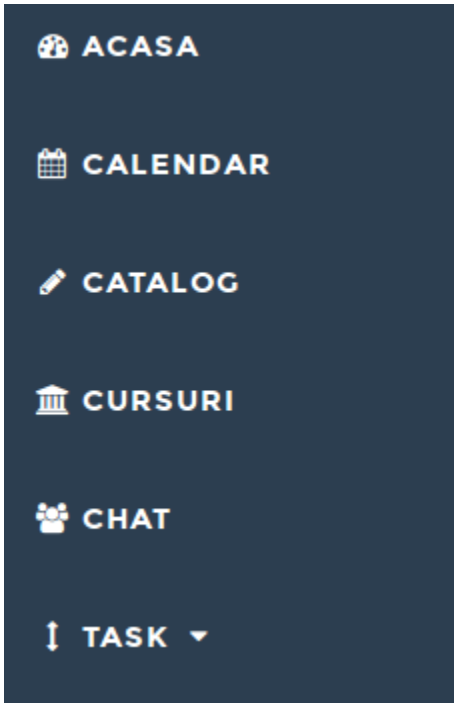

Figura 6.11 Meniu cont “profesor”

Facilitățile din cadrul fiecăruia sub-meniu sunt :

- La fel ca în cazul studentului, sub-meniul “**ACASA**” prezintă informații legate de mesajele recepționate cât timp a fost deconectat, a cursurilor ce au loc în ziua curentă dar, spre deosebire de contul studentului, ultima coloană corespunde numărului de utilizatori înregistrați la cursurile introduse de profesor.
- Sub-meniul “**CALENDAR**” prezintă aceleaşi facilități ca cele din cadrul profilului de student. De fapt, această componentă este comună pentru ambele tipuri de profil. În momentul în care acest sub-meniu este accesat de un student sau de către un profesor, se realizează un apel către baza de date care aduce informațiile necesare populării tabelului. Căutarea datelor în baza de date se face după id-ul utilizatorului, salvat în cookie în momentul conectării în aplicație.
- “**CATALOGUL**” va afişa lista utilizatorilor la accesarea lui. În momentul în care se efectuează click pe butonul “Note” din dreptul fiecărui student, va fi afişat un tabel cu informațiile legate de notele utilizatorului selectat. Denumirea materiei va fi pe prima coloană, nota pe cea de a două coloană, apoi data la care a fost introdusă nota şi două

butoane “EDIT” şi “DELETE”. Sub acest tabel există şi butonul “Adauga” care va introduce un nou rând în momentul în care este apăsat.

- Când se realizează click pe butonul “DELETE” ilustrat printr-un buton cu fundal roşu având un “X” ca şi pictogramă, nota aceea va fi ştearsă din baza de date.
- Când se realizează click pe butonul “EDIT” ilustrat printr-un buton cu fundal albastru şi o pictogramă sugestivă, toate celelalte butoane vor deveni inactive, butonul “EDIT” se va schimba într-un buton tip “SAVE” ilustrat tot printr-un buton cu fundal albastru dar care va avea binecunoscuta dischetă ca şi pictogramă, iar funcționalitatea butonului “DELETE” va avea acum rolul de a anula încercarea de modificare a notei. Câmpul unde este afişată nota se va transforma într-o listă tip drop-down de unde profesorul va putea alege nota dorită, pe o scară de la 1 la 10. De asemenea, data va fi schimbată cu data curentă.
- Când se realizează click pe butonul “Adauga”, un rând nou va fi introdus, în care câmpurile cu denumirea materiei şi nota vor fi liste ce se desfăşoară. Profesorul va putea alege oricare din materiile adăugate de el şi o notă de la 1 la 10. Data va fi ziua curentă, iar butoanele vor fi “SAVE” şi “CANCEL”, aşa cum au fost prezentate mai sus.

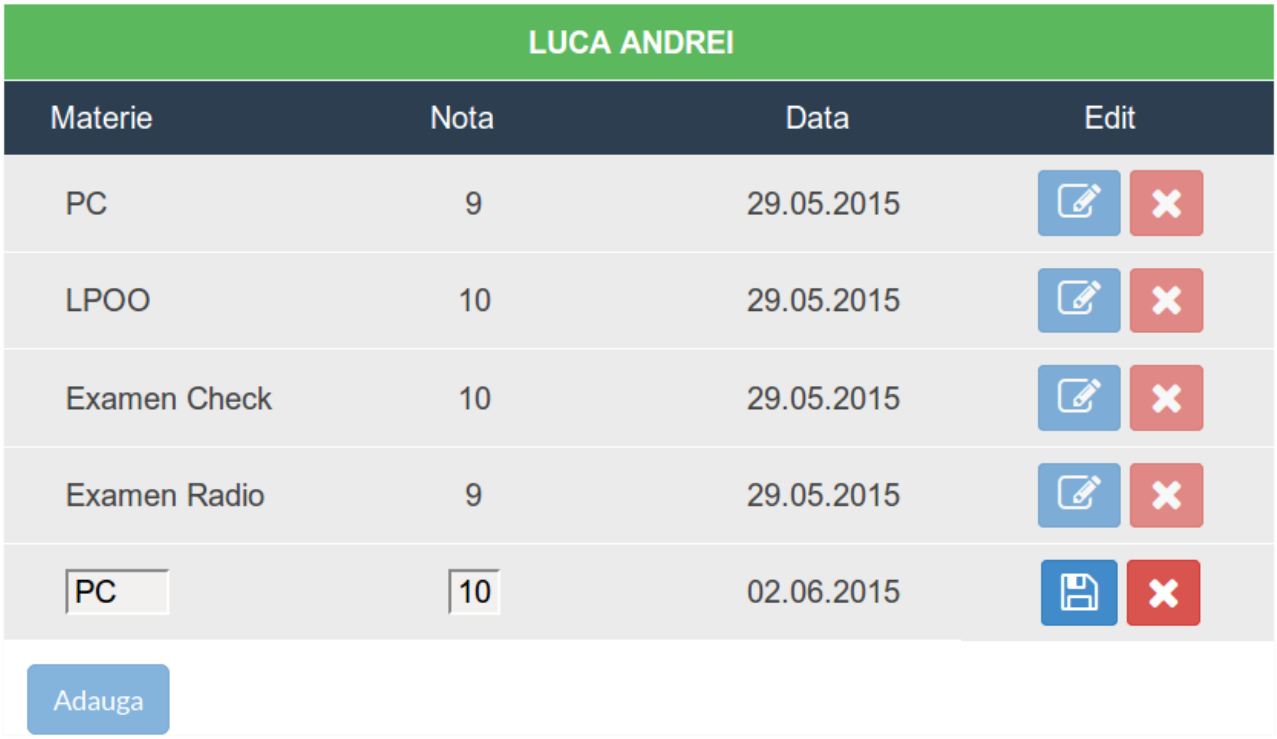

Figura 6.12 Adaugare nota de catre profesor

- Profesorul va putea vizualiza cursurile introduse de el şi va putea introduce un curs nou accesând sub-meniul "**CURSURI**". Această pagină va fi împărțită în două, iar ele vor lucra independent. În jumătatea din stânga vor fi afişate cursurile deja introduse, iar în partea din dreapta va fi afişat butonul "Adauga Curs". În momentul în care se apasă pe acest buton, în locul lui va apărea un formular ce simplifică introducerea unui curs nou.
    - Primul câmp al formularului va conține denumirea materiei. Lângă acest câmp, un buton având ca pictogramă semnul "+" va fi afişat. La apăsarea acestui buton, va apărea o fereastră tip popup ce va permite introducerea unei materii noi în baza de date. Dacă utilizatorul vrea să introducă un curs cu aceeaşi denumire a materiei dar poate în altă zi, el va putea face acest lucru. Numele materiei va fi selectat de asemenea dintr-o listă tip dropdown.
    - Trebuie menționat faptul că în cazul în care datele introduse de utilizator sunt invalide, el va fi atenționat prin mesaje specifice.
    - Profesorul va putea selecta numărul de credite şi tipul examenului - "Examen" sau "Verificare".
    - Într-o casetă de text va fi introdusă sala, ziua va fi selectată dintr-o listă dropdown, iar orele de început şi de sfârşit vor fi introduse în alte casete de text.
    - Câmpul profesor va fi completat automat cu numele profesorului şi acest câmp nu va putea fi editat.
    - O zonă de text va permite introducerea unei descrieri sumare a cursului. După acest formular sunt prezente butoanele "Adauga" şi "Inchide".
    - Dacă utilizatorul apasă pe butonul "Inchide" formularul cu toate datele introduse va dispărea, iar butonul cu "Adauga Curs" va fi afişat iar.
    - În cazul apăsării butonului "Adauga", acest curs va fi introdus în baza de date, urmând ca el să poată fi vizualizat de către studenți în sub-meniul "CURSURI" specific studenților, dar şi în jumătatea din stânga acolo unde au fost introduse deja anumite cursuri.

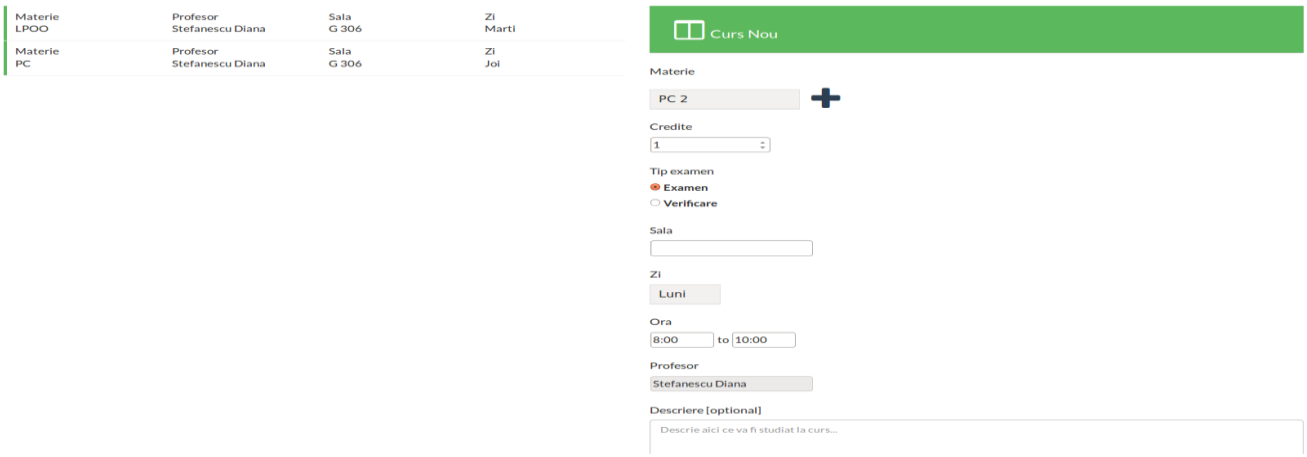

Figura 6.13 Adaugare curs de catre profesor

- Sub-meniul "**CHAT**" va avea aceeaşi funcționalitate ca şi în cazul celuilalt tip de profil
- Sub-meniul "**TASK**", împreună cu elementele componente "TEME" şi "EXAMENE" vor afişa lista utilizatorilor la accesarea lor. Ceea ce diferă între aceste 2 componente este doar tipul testului la care i se acordă acces unui utilizator, de aceea ele vor fi prezentate împreună.
    - În momentul în care este selectat un anumit utilizator, va fi afişat un tabel cu cele trei teste disponibile. Pe prima coloană va fi numele testului, pe cea de-a două coloană numărul de încercări pe care utilizatorul le-a efectuat, pe cea de-a treia coloană este prezentă data ultimei accesări, iar pe ultima coloană un buton a cărui pictogramă şi funcționalitate se va schimba în funcție de accesul utilizatorului la acel test.
    - Dacă utilizatorul are acces, fundalul butonului va fi albastru iar pictograma va fi a unui lacăt deschis, iar în cazul în care utilizatorul nu are acces la test, fundalul butonului va fi roşu, iar pictograma va avea un lacăt închis.
    - Dacă profesorul blochează accesul utilizatorului la un anumit test, câmpurile "Incercari" şi "Ultima accesare" vor fi resetate.

BIBLIOGRAFIE

- Goodman, Danny; Eich, Brendan (2001). *JavaScript Bible*. John Wiley & Sons. ISBN 0-7645-3342-8.
- http://www.w3schools.com/dom/dom_intro.asp
- http://en.wikipedia.org/wiki/KML
- http://en.wikipedia.org/wiki/JavaScript
- ("Flanagan, David; Ferguson, Paula (2006). JavaScript: The Definitive Guide (5th ed.). O'Reilly & Associates. ISBN 0-596-10199-6.")
- (" "ECMAScript Language Specification"")
- (" Douglas Crockford. Douglas Crockford on Functional JavaScript").
- (" Flanagan, David (2006). JavaScript: The definitive Guide p.16 ISBN 978-0-596-10199-2", "Elemente de stil JavaScript" . Douglas Crockford, http://JavaScript.crockford.com/style2.html)
- ("Lenssen, Philipp (1 September 2008). "Google on Google Chrome")
- (""Introduction - Chrome V8" - Pagina de dezolvate Google")
- ("Jolie O'Dell (March 10, 2011). "Why Everyone Is Talking About Node"")
- (" "Here's why you should be happy that Microsoft is embracing Node.js". The Guardian. November 9, 2011. "),
- Yahoo!, Walmart (""Why Walmart is using Node.js". VentureBeat. January 24, 2012"), Groupon,
- LinkedIn(""You'll never believe how LinkedIn built its new iPad app". VentureBeat. May 2, 2012") sau PayPal(" "Clash of the Titans: Releasing the Kraken, Node.js @paypal". fluentconf.com. May 28, 2013").
- ("Professional Node.js: Building JavaScript Based Scalable Software, John Wiley & Sons, 01-Oct-2012")
- Hughes-Croucher, Tom; Wilson, Mike (April 2012), Up and Running with Node.js (First ed.), O'Reilly Media, p. 204, ISBN 978-1-4493-9858-3
- Ornbo, George (September 2012), Sams Teach Yourself Node.js in 24 Hours (First ed.), SAMS Publishing, p. 440, ISBN 978-0-672-33595-2
- Teixeira, Pedro (October 2012), Professional Node.js (First ed.), John Wiley & Sons, p. 408, ISBN 978-1-1182-2754-1
- Rândal L. Schwartz and Aaron Newcomb (9 January 2013). "Episode 237: Node.js".

http://twit.tv/show/floss-weekly (Podcast). TWiT.tv. Event occurs at 1:08:13. Retrieved 9 January 2013.

- Ribeiro Pereira, Caio (July 2013), Aplicações web real-time com Node.js (First ed.), Casa do Código, p. 143, ISBN 978-85-66250-14-5
- Kurniawan, Agus (July 2012), Node.js Programming By Example (First ed.), PE Press, p. 67
- Gackenheimer, Cory (October 2013), Node.js Recipes: A Problem-Solution Approach (First ed.), Apress, p. 376, ISBN 978-14-30260-58-5