

UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IASI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

LUCRARE DE DIPLOMĂ

Coordonator științific:
Ş.l.dr.ing. Cristian-Nicolae Buțincu

Absolvent:
Alexandru-Gicu Melinte

Iași, 2022

UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IASI
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DOMENIUL: Calculatoare și Tehnologia Informației
SPECIALIZAREA: Tehnologia Informației

Platformă de evaluare a studenților în cadrul laboratoarelor

LUCRARE DE DIPLOMĂ

Coordonator științific:
Ş.l.dr.ing. Cristian-Nicolae Buțincu

Absolvent:
Alexandru-Gicu Melinte

Iași, 2022

**DECLARAȚIE DE ASUMARE A AUTENTICITĂȚII
PROIECTULUI DE DIPLOMĂ**

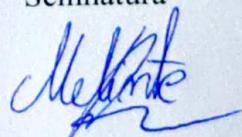
Subsemnatul MELINTE ALEXANDRU - GICU,
legitimat cu Ci seria 1Z nr. 007651, CNP 1991021226758
autorul lucrării PLATFORMĂ DE EVALUARE A
STUDENTILOR ÎN CADRUL LABORATOARELOR

elaborată în vederea susținerii examenului de finalizare a studiilor de licență, programul de studii TEHNOLOGIA INFORMAȚIEI organizat de către Facultatea de Automatică și Calculatoare din cadrul Universității Tehnice „Gheorghe Asachi” din Iași, sesiunea iULIE a anului universitar 2022, luând în considerare conținutul Art. 34 din Codul de etică universitară al Universității Tehnice „Gheorghe Asachi” din Iași (Manualul Procedurilor, UTI.POM.02 - Funcționarea Comisiei de etică universitară), declar pe proprie răspundere, că această lucrare este rezultatul propriei activități intelectuale, nu conține porțiuni plagiate, iar sursele bibliografice au fost folosite cu respectarea legislației române (legea 8/1996) și a convențiilor internaționale privind drepturile de autor.

Data

07.07.2022

Semnătura



Cuprins

Introducere	1
1 Fundamentarea teoretică și documentarea bibliografică	3
1.1 Site-ul Web	3
1.2 Platforma online	3
1.2.1 Serverul back-end	3
1.2.1.1 IntelliJ IDEA	4
1.2.1.2 Java Spring Boot	4
1.2.1.3 Spring Security	4
1.2.1.4 MySQL	4
1.2.1.5 MySQL Workbench	4
1.2.1.6 Java Persistence API	5
1.2.1.7 Hibernate	5
1.2.2 Serverul front-end	5
1.2.2.1 Visual Studio Code	6
1.2.2.2 Angular	6
1.2.2.3 ApexCharts	6
1.2.2.4 HTML	7
1.2.2.5 CSS	7
1.2.2.6 Bootstrap	7
1.2.2.7 JavaScript	7
1.2.2.8 TypeScript	7
1.3 Protocolul HTTP	8
1.3.1 Mesaj de tip cerere HTTP	8
1.3.2 Mesaj de tip răspuns HTTP	9
1.3.3 Politicile CORS	9
1.4 JSON Web Token (JWT)	10
1.5 Serviciile	11
1.5.1 Serviciile RESTful	11
1.6 Distribuția normală	12
2 Proiectarea aplicației	15
2.1 Platforma hardware	15
2.2 Componente software	15
2.2.1 Interacțiunea dintre componente	15
2.2.1.1 Front-end	15
2.2.1.2 Back-end	18
2.2.2 Baza de date	19
2.2.3 Diagrame UML	20
2.2.3.1 Diagrama de clase	21

2.2.3.2	Diagrama de cazuri de utilizare (eng. <i>Use-Case</i>)	24
2.2.3.3	Diagrama de activități	26
2.3	Avantaje și dezavantaje	26
3	Implementarea aplicației	29
3.1	Descrierea implementării pe partea de back-end	29
3.1.1	Descrierea serviciului backend-authentication	29
3.1.2	Descrierea serviciului backend-managefiles	30
3.1.3	Descrierea serviciului backend-evaluation	30
3.2	Descrierea implementării pe partea de front-end	31
3.2.1	Pagina principală	31
3.2.2	Pagina de autentificare	31
3.2.3	Pagina de înregistrare	32
3.2.4	Pagina destinată utilizatorului de tip administrator	33
3.2.5	Pagina destinată utilizatorului de tip profesor	33
3.2.5.1	Componenta responsabilă de catalog	34
3.2.5.2	Componentele responsabile de cursuri și laboratoare	34
3.2.5.3	Componentele responsabile de chestionarul de evaluare	34
3.2.6	Pagina destinată utilizatorului de tip student	35
3.2.6.1	Componentele responsabile de vizualizarea și descărcarea cursurilor și laboratoarelor	36
3.2.6.2	Componenta responsabilă de rezolvarea chestionarului de evaluare	36
3.2.7	Pagina de vizualizare profil	37
3.2.8	Deconectarea din platformă	37
3.3	Dificultăți întâmpinate și modalități de rezolvare	38
3.4	Idei originale, implementări proprii	38
4	Testarea aplicației și rezultate experimentale	39
4.1	Lansarea aplicației, elemente de configurare	39
4.2	Rezultate experimentale	39
4.3	Aspecte legate de fiabilitate/securitate	43
Concluzii		45
Bibliografie		47

Platformă de evaluare a studenților în cadrul laboratoarelor

Alexandru-Gicu Melinte

Rezumat

Plaftorma **EvalPlatform** reprezintă o platformă online de evaluare a studenților în cadrul laboratoarelor.

Evaluarea este realizată în trei etape și este constituită dintr-un chestionar de evaluare: o primă evaluare în săptămâna a cincea, a doua evaluare în săptămâna a noua și ultima evaluare în ultima săptămână.

Platforma poate fi accesată de trei tipuri de utilizatori: administrator, profesor și student. Administratorul gestionează conturile utilizatorilor, profesorul gestionează cursurile, laboratoarele și chestionarele și vizualizează catalogul de studenți ce conține și statistici legate de tipurile de evaluare, iar studentul are acces la cursuri, laboratoare și poate rezolva chestionare.

Toate tipurile de utilizator au posibilitatea să-și vizualizeze profilul și să se deconecteze din platformă.

Lucrarea de față a fost structurată pe partea de back-end cu ajutorul cadrului de dezvoltare *Java Spring Boot*, sistemului de gestiune al bazei de date *MySQL* și mediul de dezvoltare *IntelliJ IDEA*. Partea de front-end a fost dezvoltată cu ajutorul cadrului de dezvoltare *Angular* și al mediului de dezvoltare *Visual Studio Code*.

Ideile originale se regăsesc la nivelul implementării logicii de funcționare a evaluării, atât crearea chestionarului cât și calcularea notei corespunzătoare tipului de evaluare, respectiv nota finală, la nivelul aspectului interfeței cu utilizatorul, dar și la nivelul gestiunii datelor între partea de front-end și partea de back-end (stocare și extragere din baza de date, codificare informații).

Primul capitol conține noțiuni teoretice fundamentale și pune bazele teoretice ale platformei **EvalPlatform**.

Capitolul doi detaliază structura aplicației la nivelul fiecărei componente prin diferite diagrame UML sau diagrame ce scot în evidență interacțiunea dintre componente.

Cel de-al treilea capitol reprezintă o descriere mai amănunțită a capitolului doi, o descriere a implementării funcționalităților.

Ultimul capitol al acestei lucrări scoate în evidență funcționalitatea prezentând testarea interfeței cu utilizatorul, modul de lansare în execuție al platformei și aspecte legate de securitate.

Concluziile acestei lucrări scot în evidență atingerea obiectivelor propuse, soluțiile personale la nivelul implementării și posibilele dezvoltări ale funcționalității.

Introducere

Domeniul și contextul abordării temei

Se poate observa că în urma evenimentelor din ultimii ani, ce au în prim plan educația digitală, ne putem încadra în mijlocul unei revoluții digitale. Fenomenele care se concentrează pe tehnologia informației și comunicațiilor (TIC), inteligența artificială și robotică colaborativă care nu pot fi ignorate deoarece au un impact semnificativ asupra evoluției, necesită tot felul de tehnologii la scară socială, economică, individuală și globală. Prin urmare, adaptarea rapidă la dezvoltarea acestor tehnici este o necesitate pentru învățământul de orice fel, pornind de la cel preșcolar până la cel superior.

Educația digitală presupune, pe deosebit, o acumulare de cunoștințe pe care o persoană trebuie să le aibă în ceea ce privește utilizarea unui sistem digital, adică aceasta să fie capabilă să folosească funcționalitățile minime ale acestuia. Pe de altă parte, educația digitală se referă deseori și la niște metode digitale care să înlocuiască metodele clasice pe care sistemul de învățământ, de orice tip, le-a folosit dintotdeauna.

Obiectivele temei propose

O **platformă de evaluare** a studenților în cadrul laboratoarelor este o metodă digitală de evaluare folosită de cadrul didactic în scopul aprecierii activității unui student pe parcursul semestrului. Un semestru este alcătuit din 14 săptămâni, fiecarei săptămâni corespunzându-i un laborator.

Evaluarea va fi împărțită în trei părți, iar fiecare parte cuprinde materia studiată până la momentul respectiv. Prima evaluare se va desfășura în săptămâna a cincea, cea de a doua evaluare în săptămâna a nouă, iar evaluarea finală se va face în ultima săptămână. La finalul semestrului, pe baza celor trei evaluări efectuate pe parcursul semestrului, fiecare student va avea o medie finală calculată după media aritmetică a evaluărilor.

De asemenea, platforma va oferi posibilitatea vizualizării unor statistici legate de notele studenților și furnizarea unor concluzii legate de veridicitatea evaluării.

Motivarea alegерii temei

Această temă a fost aleasă întrucât o astfel de metodă digitală de evaluare este mult mai avantajoasă decât una clasică. Metoda clasică presupune ca la finalul fiecărui laborator profesorul să corecteze ce a lucrat fiecare student, ca mai apoi la finalul semestrului să calculeze media de laborator pe baza acestor notări. Metoda digitală ar presupune evaluarea sub forma unor chestionare online (*quiz-uri*) ce conțin materia învățată până la momentul respectiv.

Această platformă ar putea să eliminate situațiile în care studentul consideră că profesorul a furnizat o evaluare subiectivă deoarece calculatorul nu are puterea de a nota studentul în funcție de alte motive decât prin cele ce apreciază corectitudinea soluției furnizate pentru evaluarea în curs.

Platforme educative asemănătoare

Platformele care dezvoltă aceeași temă și care au fost analizate drept sursă de inspirație sunt Moodle și Google Classroom. **Moodle** este o platformă de învățare care oferă profesorilor, administratorilor și cursanților de orice fel un sistem integrat unic, robust și sigur pentru a se crea unui mediu de învățare personalizat. **Google Classroom** este o platformă gratuită, utilizată preponderent în domeniul educației ca mijloc de desfășurare a învățării. Multă profesori folosesc această platformă pentru a putea evalua elevii sau studenții și pentru a putea distribui materiale de curs.

Instrumente software utilizate

Printre instrumentele software ce au fost utilizate în dezvoltarea platformei se regăsesc limbajul *Java*, cadrul de dezvoltare *Java Spring Boot*, sistemul de gestiune al bazei de date *MySQL* și mediul de dezvoltare *IntelliJ IDEA* pentru implementarea părții de back-end, iar pentru partea de front-end s-a utilizat cadrul de dezvoltare *Angular* și mediul de dezvoltare *Visual Studio Code*.

Structura lucrării

Structura lucrării este formată dintr-o introducere, patru capitole, fiecare capitol conținând și subcapitole, concluzii și o bibliografie.

Primul capitol este intitulat **Fundamentarea teoretică și documentarea bibliografică** și conține noțiuni de teorie ce susțin lucrarea de față și referințe bibliografice pentru acele paragrafe redate din mediul online sau din cărți, teze de doctorat, articole și aşa mai departe. În cadrul acestui capitol au fost dezvoltate noțiunile de platformă online, de server de back-end și server de front-end, cadrele de dezvoltare folosite, limbajele folosite, a fost descrisă comunicarea prin mesaje de tip cerere-răspuns și care sunt acestea în cadrul protocolului HTTP, ce este un serviciu *RESTful* și ce reprezintă o distribuție normală.

Cel de-al doilea capitol poartă denumirea de **Proiectarea aplicației** și este alcătuit din subcapitole ce prezintă platforma hardware pe care s-a dezvoltat lucrarea de față, interacțiunea dintre componentele platformei, mai exact dintre serverul de front-end și cel de back-end, baza de date ce stochează informațiile și structura tabelelor, diagramele UML precum cele de clasă, cazuri de utilizare și secvențe ce pun în evidență structura serviciilor folosite, precum și avantajele și dezvantajele metodei alese pentru implementarea platformei.

Capitolul trei intitulat și **Implementarea aplicației** este o descriere mai accentuată a capitolului doi. În cadrul acestui capitol s-au detaliat modul în care a fost implementată platforma și descrierea componentelor principale ale platformei, atât pe partea de back-end cât și pe partea de front-end, problemele întâmpinate pe parcursul dezvoltării și soluțiile găsite, descrierile ideilor originale și a implementărilor proprii, dar și descrierea interfeței cu utilizatorul și funcționalitatea acesteia.

Ultimul capitol, numit și **Testarea aplicației și rezultate experimentale**, conține informații legate de testarea implementării descrisă la capitolul trei. Mai exact, acest capitol cuprinde modul de lansare al aplicației și configurațiile necesare, al pornirii serverului de back-end și al serverului de front-end, rezultate experimentale interpretate prin capturi de ecran ce demonstrează funcționalitatea interfeței cu utilizatorul și câteva aspecte legate de securitatea platformei și cum s-a implementat.

Concluziile surprind aspectele legate de gradul în care s-a realizat lucrarea față de cerințele inițiale, o evidențiere a soluțiilor originale implementate în această lucrare și posibile îmbunătățiri sau direcții de dezvoltare, ce se putea face mai bine și cum se mai poate dezvolta platforma.

Capitolul 1. Fundamentarea teoretică și documentarea bibliografică

1.1. Site-ul Web

Un **site Web** este o colecție de pagini Web și conținut asociat, identificate printr-un nume de domeniu comun și publicate pe unul sau mai multe servere Web. Toate site-urile Web ce pot fi accesate printr-o rețea publică formează *World Wide Web* [1]. Există și site-uri Web ce pot fi accesate doar printr-o rețea privată, dar de obicei acestea sunt cele de uz intern pentru angajații unei companii.

Pentru o funcționare optimă a unui site Web acesta trebuie să contină un server de back-end și un server de front-end. Utilizatorul va introduce informația dorită, care va fi preluată și implementată în partea de back-end a site-ului. În final, informația va fi afișată în partea de front-end pentru a putea fi vizualizată de utilizatori.

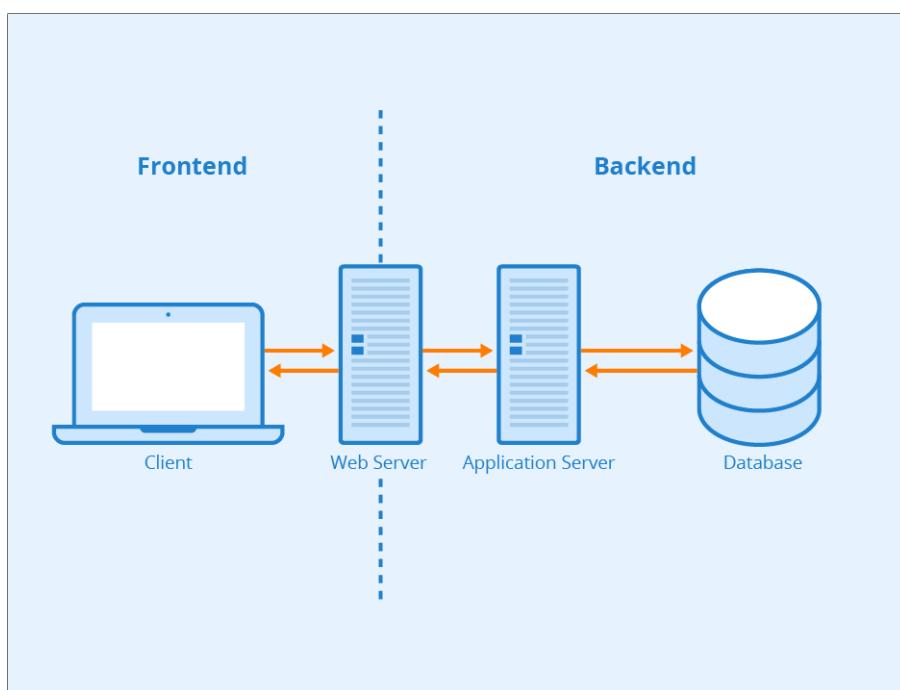


Figura 1.1. Server front-end și server back-end¹

1.2. Platforma online

O **platformă online** este un sistem dinamic, o colecție de cod scris într-unul sau mai multe limbaje de programare specifice și folosită pentru a genera pagini Web dinamice. Cu alte cuvinte, o platformă online conține toate resursele necesare pentru crearea unui site Web, în timp ce un site Web nu dispune de toate resursele pentru a crea o platformă.

Platformele online includ într-o mare măsură relațiile și interacțiunile cu utilizatorii. Câteva exemple de platforme online în ceea ce privește domeniul educației sunt: *Moodle*, *Udemy*, *Google Classroom*, *LinkedIn Learning*.

1.2.1. Serverul back-end

Serverul de back-end este acea componentă dintr-un site Web la care utilizatorul nu are acces, iar responsabilitățile acestuia sunt de a stoca și organiza datele și de a se asigura că tot la

¹https://medium.com/@_nerd.warrior_/front-end-vs-back-end-7e50c7570c0f

ceea ce are acces utilizatorul funcționează corect.

În cadrul unui **server back-end** se pot regăsi următoarele procese și servicii: accesarea datelor din bazele de date cu ajutorul interogărilor SQL, stocarea în bazele de date, criptarea și decriptarea datelor, gestionarea fișierelor, autentificarea și înregistrarea, și multe altele. Aceasta este mult mai complex decât serverul de front-end deoarece sunt utilizate diferite limbi și tehnologii pentru a se realiza toate cerințele utilizatorului: Ruby, PHP, Python, Java, Kotlin și altele. Poate fi asemănăt și cu un spațiu de stocare ce servește unui site Web, unde toate informațiile stocate sunt informații de pe Internet.

1.2.1.1. IntelliJ IDEA

IntelliJ IDEA este un mediu de dezvoltare integrat scris în Java pentru dezvoltarea de cod scris în Java, Kotlin, Groovy și alte limbi bazate pe *JAR*². Este dezvoltat de *JetBrains* și este disponibil într-o ediție comunitară cu licență Apache 2 și într-o ediție comercială proprietară.

1.2.1.2. Java Spring Boot

Java Spring Framework este un cadru de dezvoltare (eng. *framework*) cu sursă deschisă (eng. *open-source*) ce oferă un suport consistent pentru crearea unei infrastructuri de dezvoltare a aplicațiilor Java.

Java Spring Boot este o extensie a cadrului de dezvoltare *Java Spring Framework* cu ajutorul căruia se pot dezvolta aplicații Web și servicii într-o manieră mai rapidă și mai ușoară, eliminând configurațiile de nivel inferior. Aceasta analizează configurațiile setate de către dezvoltator și injectează toate dependențele necesare pentru satisfacerea cerințelor. Principalele avantaje ale Java Spring Boot-ului sunt autoconfigurarea, independența și autonomia acestuia în procesul de dezvoltare a unei aplicații optimizate [2].

1.2.1.3. Spring Security

Spring Security este un cadru de dezvoltare *Java Spring* ce oferă suport pentru autentificare, autorizare și protecție împotriva atacurilor persoanelor rău intenționate, dar care este și usor de personalizat.

1.2.1.4. MySQL

MySQL este cel mai popular sistem de gestiune a bazelor de date relaționale și este dezvoltat și distribuit de Oracle.

O bază de date relațională este o bază de date ce stochează datele în mai multe tabele, nu în una singură. Structurile bazei de date sunt organizate în fișiere fizice optimizate pentru viteza. Modelul logic, cu obiecte precum baze de date, tabele, vederi, rânduri și coloane, oferă un mediu de programare flexibil. Sunt stabilite de asemenea și reguli care stabilesc relațiile dintre diferite tabele, cum ar fi unu-la-unu, unu-la-mulți, mulți-la-mulți, unic, obligatoriu sau optional și adrese către diferite tabele [3].

SQL (*Structured Query Language*) este cel mai comun limbaj standardizat folosit pentru a accesa bazele de date.

1.2.1.5. MySQL Workbench

MySQL Workbench este un instrument vizual de proiectare a bazelor de date care integrează dezvoltarea, managementul, proiectarea, crearea și întreținerea bazei de date SQL într-un singur sistem de dezvoltare integrat pentru sistemul de baze de date MySQL. MySQL Workbench este disponibil pe Windows, Linux și macOS.

²Java ARchive - arhivă de clase Java și metadate

1.2.1.6. Java Persistence API

Maparea obiectelor Java în tabelele unei baze de date, dar și invers, se numește **mapare relațională cu obiecte**.

JPA (*Java Persistence API*) este o abordare a mapării relaționale și este o colecție de clase și de metode folosite pentru a stoca în mod persistent cantități mari de date într-o bază de date furnizată de *Oracle Corporation*. JPA-urile permit programatorului să lucreze direct cu obiecte, fără a lucra indirect prin intermediul instrucțiunilor SQL, iar acest tip de implementare poartă numele de furnizor de persistență. Câteva dintre implementările populare sunt Hibernate, Apache OpenJPA și EclipseLink [4].

Maparea dintre obiectele Java și tabelele bazei de date este definită prin metadatele de persistență. Furnizorul JPA va folosi informațiile despre metadatele de persistență pentru a efectua operațiunile corecte ale bazei de date. Metadatele JPA sunt de obicei definite prin adnotări în clasele din Java. De asemenea, metadatele mai pot fi definite atât prin *XML (Extensible Markup Language)*, cât și printr-o combinație a claselor Java cu XML [4].

1.2.1.7. Hibernate

Hibernate este un cadru de dezvoltare Java ce simplifică interacționarea dintre aplicația Java și baza de date. Este un instrument ORM (*Object Relational Mapping*) cu sursă deschisă (eng. *open-source*). De asemenea, Hibernate implementează specificațiile JPA pentru persistența datelor.

Una dintre cele mai importante caracteristici ale cadrului de dezvoltare Hibernate este maparea de la clase Java la tabelele bazelor de date și maparea de la tipuri de date Java la tipuri de date SQL. Hibernate oferă și posibilități de extragere și interogare a datelor. Acestea generează apeluri SQL și scutește programatorul de manipularea manuală și conversia obiectelor.

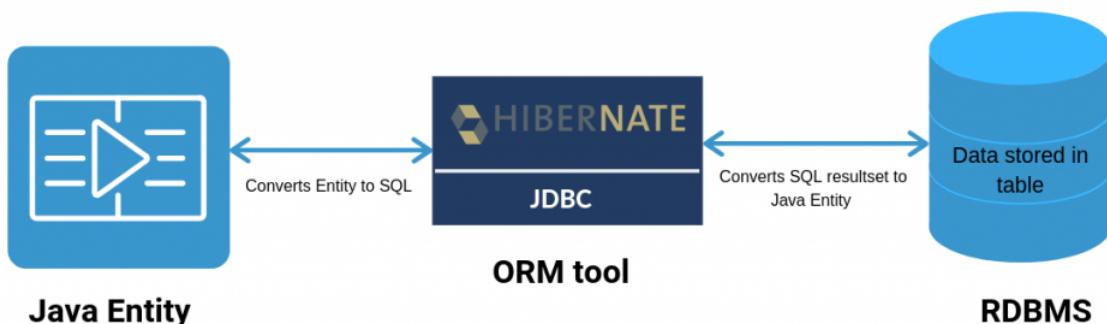


Figura 1.2. Maparea datelor din clase Java în baze de date relaționale folosind Hibernate³

1.2.2. Serverul front-end

Serverul de front-end este acea componentă dintr-un site Web ce interacționează cu utilizatorul și se ocupă de aspectul și funcționalitatea interfeței paginii Web. În serverul de front-end se

³<https://jstobigdata.com/jpa/introduction-to-jpa-and-hibernate/>

implementează tot ce ține de culori, butoane, poze sau videoclipuri, meniuri de căutare, identare, aranjare în pagină și aşa mai departe. Ca și tehnologii și limbaje de programare, pot fi folosite: HTML, CSS și JavaScript, cu librăriile, versiunile și cadrurile de dezvoltare aferente.

1.2.2.1. Visual Studio Code

Visual Studio Code este un mediu de dezvoltare, un editor de cod sursă realizat de cei de la Microsoft pentru Windows, Linux și macOS. Acesta include suport pentru depanare, evidențierea sintaxei, completarea intelligentă a codului, fragmente, refactorizarea codului și *Git* încorporat.

1.2.2.2. Angular

Angular este o platformă și un cadru de dezvoltare pentru construirea de aplicații client cu o singură pagină folosind HTML și TypeScript. Angular este scris în TypeScript și este condus și dezvoltat de echipa de Angular de la *Google*. Funcționalitatea de bază și cea optională sunt implementate ca un set de biblioteci TypeScript pe care mai apoi programatorul le importă în propria aplicație.

Arhitectura unei aplicații **Angular** se bazează pe anumite concepte fundamentale. Elementele de bază ale cadrului de dezvoltare Angular sunt componentele Angular care sunt organizate în *NgModules*. NgModules colectează codul aferent în seturi funcționale deoarece o aplicație Angular este definită de un set de NgModules. O aplicație are întotdeauna cel puțin un modul rădăcină și multe module ce implementează fiecare caracteristică în parte.

Angular Material este o componentă a bibliotecii UI (*User Interface*) dezvoltată de *Google* în 2014. Este special concepută pentru dezvoltatorii de Angular. Ajută la proiectarea aplicației într-o manieră structurată. Componentele sale ajută la construirea de pagini Web și aplicații Web atractive, consecvente și funcționale. Este folosit pentru a crea un site Web receptiv și mai rapid.

1.2.2.3. ApexCharts

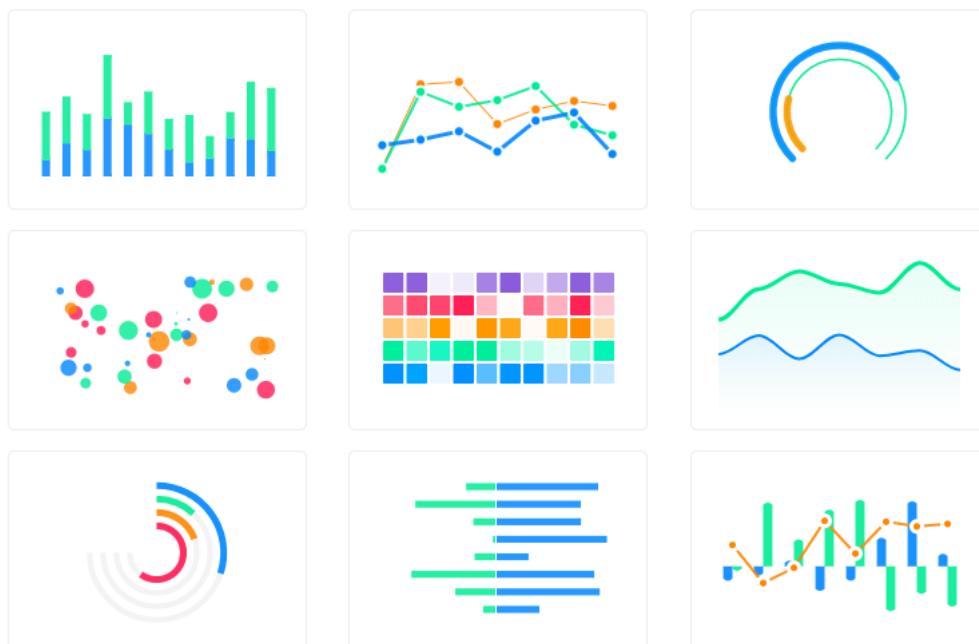


Figura 1.3. Diferite tipuri de diagrame create cu ajutorul ApexCharts⁴

⁴<https://github.com/apexcharts/apexcharts.js>

ApexCharts este o bibliotecă modernă JavaScript de diagrame care ajută dezvoltatorii să creeze vizualizări frumoase și interactive pentru paginile Web. Această bibliotecă pune la dispoziție diagrame precum cele de tip bară, arie, histogramă, radar și multe altele.

1.2.2.4. HTML

Un **limbaj de marcare** este un sistem de adnotare a unui document astfel încât să se distingă din punct de vedere sintactic de text.

HTML (*Hypertext Markup Language*) este un limbaj de marcare, un sistem de formatare a informațiilor preluate de pe Internet. Fiecare informație preluată poate fi considerată ca și o pagină Web.

HTML oferă o modalitate de adnotare a conținutului documentului cu o mare varietate de metadate și instrucțiuni de vizualizare. Instrucțiunile de vizualizare pot varia de la sublinierea sau îngroșarea unui cuvânt până la instrucțiuni de cod avansate și formulare. Metadatele includ informații despre titlul și autorul documentului și informații structurale despre modul în care documentul este împărțit în diferite segmente, paragrafe, liste, titluri.

1.2.2.5. CSS

CSS (*Cascading Style Sheet*) este un **limbaj de stilizare** ce definește modul de afișare a elementelor și documentelor HTML, inclusiv culorile, aspectul și fonturile, modul de așezare în pagină, aliniere și multe altele.

Unul dintre cele mai mari avantaje ale CSS-ului este capacitatea acestuia de a impune consistență aspectului paginilor Web. Spre exemplu, acesta permite programatorului să specifice același aspect pentru mai multe paragrafe prin selectori. [1].

Un **selector CSS** este un model folosit pentru a selecta și găsi elemente HTML în vederea aplicării unor metode de stilizare.

1.2.2.6. Bootstrap

Bootstrap este un cadru de dezvoltare CSS folosit pentru a proiecta pagini Web și aplicații Web, într-o manieră mai ușoară și mai rapidă. Include diverse şablonane de stilizare HTML și CSS pentru butoane, tabele, formulare, panouri și multe altele. În plus, Bootstrap oferă un sistem de rețea care creează aspectul unei pagini printr-un set de rânduri și coloane unde poate fi plasat conținutul.

1.2.2.7. JavaScript

JavaScript este un limbaj de programare orientat-obiect, dinamic, interpretat folosit în programarea Web, a aplicațiilor Web și a jocurilor. Este folosit în primul rând pentru a îmbunătăți paginile Web și pentru a oferi o experiență ușor de utilizat. Printre exemplele de îmbunătățiri ale paginilor Web se regăsesc următoarele: actualizări dinamice ale paginilor Web, îmbunătățiri ale interfeței cu utilizatorul, cum ar fi meniuri și casete de dialog, animații, grafică 2D și 3D, hărți interactive și multe altele.

1.2.2.8. TypeScript

TypeScript este un superset sintactic al limbajului JavaScript, un limbaj compilat ce asigură un sistem de tipuri optional. TypeScript compilează codul și generează erori de compilare, dacă găsește astfel de erori ajutând programatorul să observe eventualele erori de cod, de sintaxă înainte de rularea codului efectiv. JavaScript, fiind un limbaj interpretat, necesită a fi rulat mai întâi pentru a se putea observa eventualele erori.

1.3. Protocolul HTTP

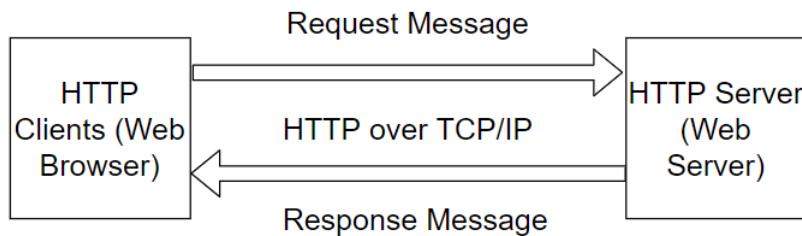


Figura 1.4. Mesaje de tip cerere-răspuns în cadrul protocolului HTTP⁵

HTTP este un **protocol** asimetric cerere-răspuns **client-server**, ilustrat în Figura 1.4. Un client HTTP trimite un mesaj de solicitare către un server HTTP. Serverul, la rândul său, returnează un mesaj de răspuns. Cu alte cuvinte, HTTP este un protocol de extragere, clientul extrage informații de pe server. De asemenea, este un protocol fără stare, ceea ce înseamnă că nici o cerere curentă nu cunoaște ceva despre cererile anterioare [5]. Asfel, pentru persistența informațiilor între accesări este necesar să se utilizeze soluții de tipul: *cookies*⁶, sesiuni, rescrierea URL-urilor, câmpuri ascunse.

1.3.1. Mesaj de tip cerere HTTP

Cererea HTTP este transmisă de la client către serverul Web și semnifică o solicitare pentru obținerea unor informații, resurse (identificate printr-un URL). Aceasta include denumirea metodei utilizate pentru a transfera informații, locația resursei și versiunea protocolului. Metode utilizate în cadrul cererilor HTTP sunt exemplificate precum în [6]:

- **GET** - metodă prin care clientul solicită o resursă de la server;
 - **PUT** - metodă prin care clientul solicită serverului să adauge o nouă resursă;
 - **POST** - metodă prin care clientul solicită și controlează adăugarea unei noi resurse țintă;
 - **DELETE** - metodă prin care clientul solicită serverului o stergere a unei resurse țintă;
 - **PATCH** - metodă prin care clientul solicită actualizarea parțială a unei resurse țintă;
 - **OPTIONS** - metodă prin care i se permite clientului să vizualizeze opțiunile și cerințele unei resurse țintă;
 - **HEAD** - metodă prin care clientul solicită o resursă de la server, dar fără a i se trimite și corpul mesajului în răspuns;
 - **CONNECT** - metodă prin care se stabilește un canal de comunicare cu serverul, identificat printr-o resursă țintă;
 - **TRACE** - metodă prin care se efectuează un test buclă-înapoi de-al lungul căii către resursa țintă.

⁵<https://docs.simuli.co/iot-protocols/hypertext-transfer-protocol-http>

⁶Text special, deseori codificat, trimis de un server unui navigator Web și apoi trimis înapoi de către navigator, de fiecare dată când acceseașă acel server.

1.3.2. Mesaj de tip răspuns HTTP

Răspunsul HTTP este trimis de la serverul Web către client, ca și rezultat al cererii primite, incluzând totodată și o linie de stare precum și alte informații suplimentare. Prima linie a mesajului de răspuns (adică, linia de stare) conține codul de stare al răspunsului, care este generat de server pentru a indica rezultatul cererii. Codul de stare este un număr din 3 cifre:

- **1xx (Informatie)** - cererea este validă, iar serverul procesează cererea primită;
- **2xx (Succes)** - cererea a fost primită, recunoscută și acceptată cu succes;
- **3xx (Redirectare)** - cererea este incompletă; necesită a fi luate măsuri suplimentare pentru a putea fi finalizată cererea cu succes;
- **4xx (Eroare client)** - cererea conține erori de sintaxă sau nu a putut fi recunoscută de către server;
- **5xx (Eroare server)** - cererea este validă, iar serverul nu a putut răspunde din motive interne.

1.3.3. Politicile CORS

O **politică CORS** (*Cross-Origin Resource Sharing*) specifică setările care pot fi aplicate resurselor pentru a permite partajarea resurselor între origini (eng. *cross-origin resource sharing*).

Originea conținutului Web este definită și identificată după schema (protocolul), domeniul și portul URL-ului folosit pentru a fi accesat respectivul conținut.

CORS este un mecanism de securitate care utilizează un antet HTTP suplimentar pentru a informa un navigator Web să permită unei aplicații Web care rulează la o singură origine (domeniu) să aibă permisiunea de a accesa resursele selectate de pe un server de la o altă origine. Fără CORS, paginile Web sunt limitate la accesarea resurselor din aceeași origine prin ceea ce este cunoscut sub numele de politică de aceeași origine (eng. *same-origin policy*).

Politica de aceeași origine (eng. *same-origin policy*) este un mecanism de securitate critic care restricționează modul în care un document sau o secvență de cod încărcată dintr-o origine poate interacționa cu o resursă dintr-o altă origine.

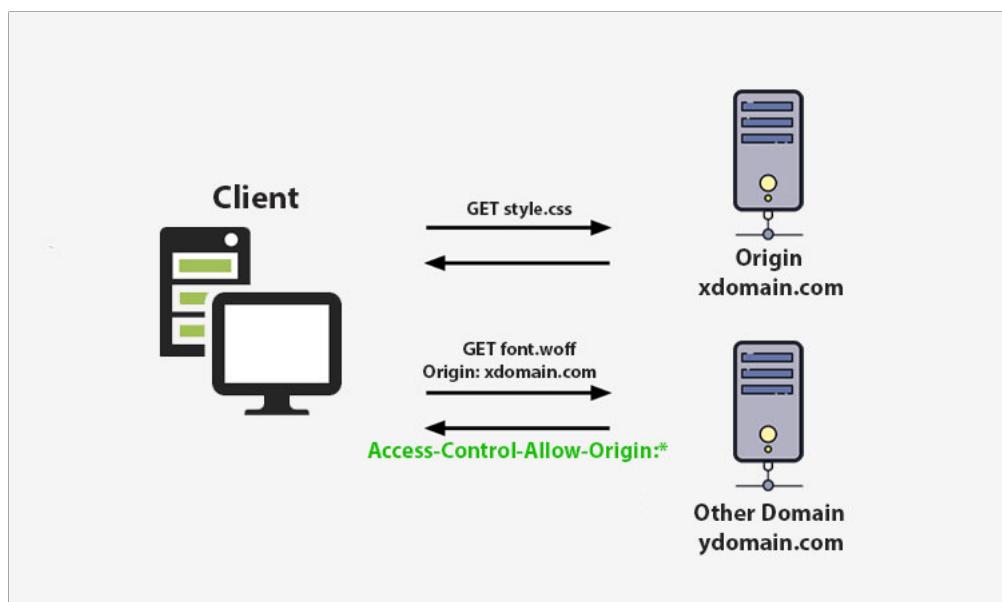


Figura 1.5. Exemplu de utilizare CORS⁷

⁷<https://www.softaox.info/cross-origin-resource-sharing-cors/>

1.4. JSON Web Token (JWT)

JSON Web Token (JWT) este un standard deschis [7] care definește o modalitate compactă și autonomă de transmitere în siguranță a informațiilor între părți ca obiect *JSON*. Aceste informații pot fi verificate și sunt de încredere, deoarece sunt semnate digital. JWT-urile pot fi semnate folosind un algoritm secret (cu algoritmul *HMAC*) sau o pereche de chei publice/private folosind *RSA* sau *ECDSA* [8].

Autorizarea (eng. *Authorization*) este cel mai frecvent scenariu pentru utilizarea JWT. Odată ce utilizatorul este conectat, fiecare cerere ulterioară va include un JWT, permitându-i utilizatorului să acceseze rute, servicii și resurse care sunt permise doar cu acel JWT [8].

Schimbul de informații (eng. *Information Exchange*) este un alt scenariu pentru utilizare a JWT-urilor, deoarece acestea sunt o modalitate bună de a transmite informații în siguranță între două entități. Deoarece acestea sunt semnate, programatorul se poate asigura că utilizatorul care solicită schimbul este cel adevărat [8].

Un **JWT** conține în structura sa **trei părți**, separate prin punct:

- **antet** (eng. *header*) - conține tipul JWT-ului și algoritmul de semnare utilizat, cum ar fi *HMAC SHA256* sau *RSA*; la final, acesta este codificat în format *Base64Url*;
- **informația utilă** (eng. *payload*) - conține corpul mesajului, organizat sub formă de revendicări (eng. *claim*), adică mai exact sub forma unei afirmații despre o entitate ce conține totodată și metadate suplimentare despre simbolul în sine; revendicările pot fi de tip predefinit, public sau privat; la final, toată această informație este codificată în format *Base64Url*;
- **semnătura** (eng. *signature*) - conține antetul și informația utilă reprezentate în formatul *Base64Url*, iar mai apoi semnate cu algoritmul specificat în antet.

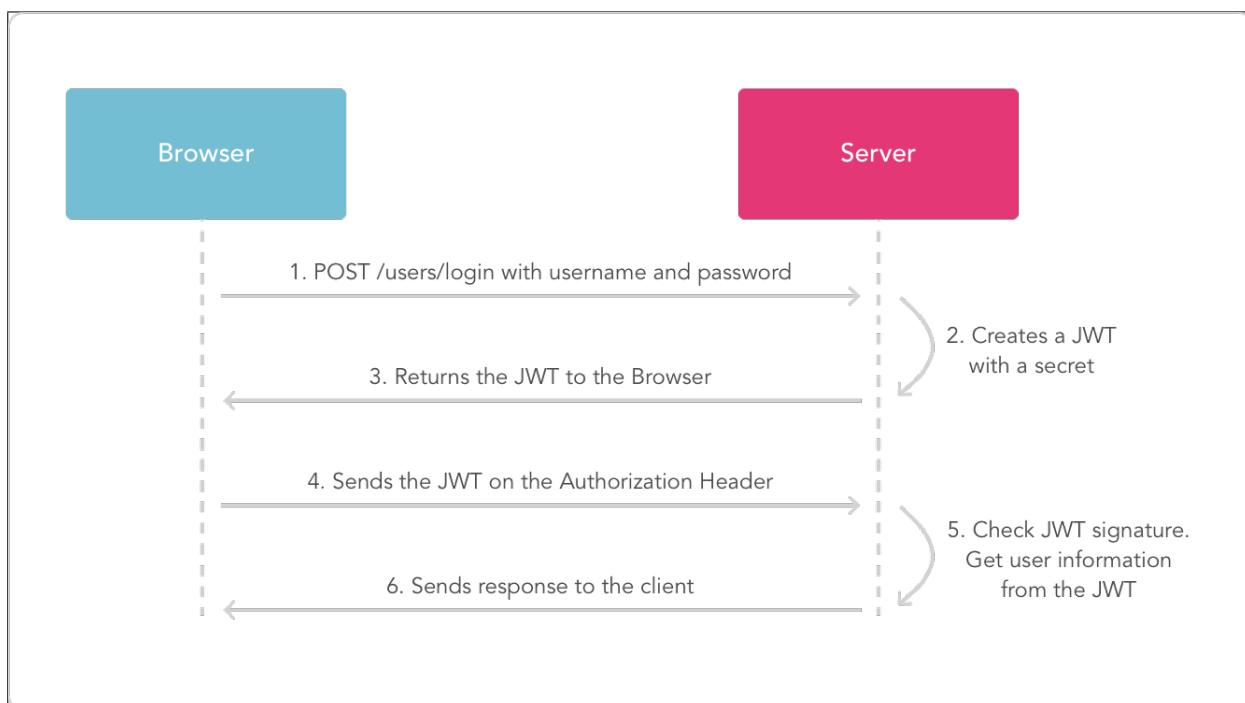


Figura 1.6. Fluxul de date la autorizare folosind JWT⁸

⁸<https://github.com/farbodsalimi/react-redux-jwt-authentication-boilerplate>

1.5. Serviciile

Un **serviciu Web** este orice serviciu disponibil pe Internet, care utilizează un XML standardizat ca și sistem de mesagerie și nu este dependent de niciun sistem de operare sau limbaj de programare [9].

Serviciile Web oferă un mijloc standard de interoperare între aplicațiile software ce rulează pe o varietate de platforme și cadre de dezvoltare. Așadar, ele sunt caracterizate prin interoperabilitatea și extensibilitatea lor mare.

1.5.1. Serviciile RESTful

Conceptul de REST a fost introdus pentru prima dată în anul 2000, de Roy Fielding în [10]. **REST (Representational State Transfer)** reprezintă un model arhitectural de tip client-server folosit pentru a dezvolta servicii Web care sunt axate pe resursele unui sistem și pe interacțiunile dintre resursele unui astfel de sistem.

În arhitectura REST, clienții trimit cereri pentru a prelua sau modifica resurse, iar serverele trimit răspunsuri la aceste solicitări. O cerere este alcătuită dintr-un verb HTTP (*GET, POST, PUT, PATCH, DELETE*) care definește tipul cererii, un antet care permite clientului să transmită informații despre cerere, o cale către resursă și optional un corp de mesaje ce conține date.

Serviciile Web RESTful sunt servicii Web bazate pe arhitectura REST. În Arhitectura REST totul este o resursă. Serviciile Web RESTful sunt foarte scalabile și ușor de întreținut și sunt foarte frecvent utilizate pentru a crea API-uri (*Application Programming Interface*) pentru aplicații bazate pe Web.

Un **API (Application Programming Interface)** este un mecanism ce permite aplicațiilor software să comunice între ele folosind un set de definiții și protocoale.

REST API reprezintă un API care se conformează principiilor de proiectare REST. Din acest motiv, API-urile REST sunt uneori denumite API-uri RESTful. Acestea comunică prin cereri HTTP pentru a efectua funcții standard ale bazei de date, cum ar fi crearea, citirea, actualizarea și ștergerea înregistrărilor (cunoscute ca și *CRUD*). De exemplu, un API REST ar folosi o cerere *GET* pentru a citi/extrage o înregistrare, o cerere *POST* pentru a crea o înregistrare, o cerere *PUT* pentru a actualiza o înregistrare, o cerere *PATCH* pentru a actualiza parțial o înregistrare și o cerere *DELETE* pentru a șterge o înregistrare.

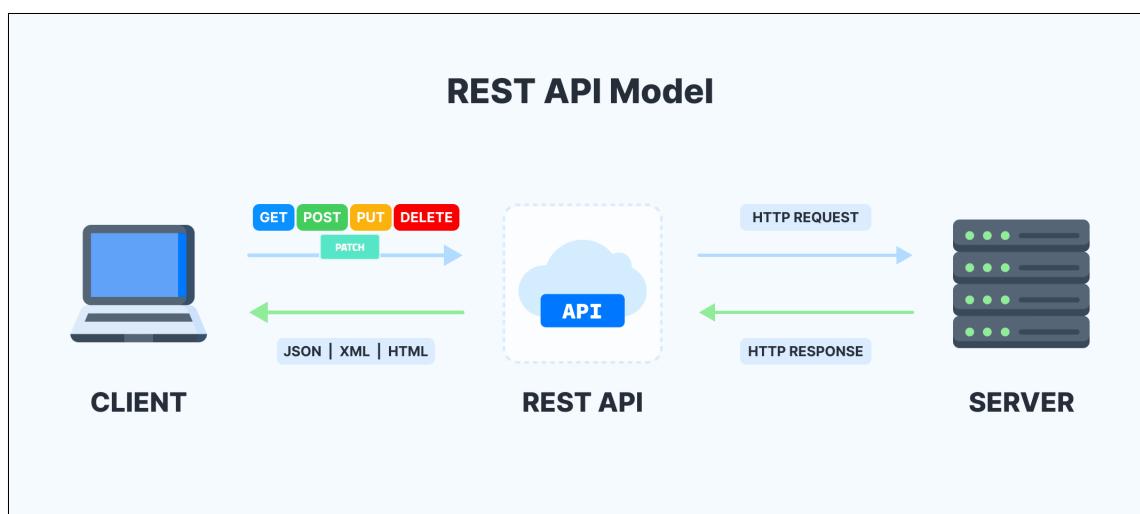


Figura 1.7. Flux de date REST API⁹

⁹<https://appmaster.io/blog/how-create-api-no-code>

1.6. Distribuția normală

Distribuția normală, cunoscută și sub numele de **distribuție Gaussiană**, este o distribuție de probabilitate care este simetrică față de medie, arătând că datele din apropierea mediei apar mai frecvent decât datele aflate la distanță de medie. Sub formă de grafic, distribuția normală va apărea ca o curbă sub forma unui clopot, denumit și **clopotul lui Gauss**. Forma generală a funcției sale de densitate de probabilitate este:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}} \quad (1.1)$$

σ = deviația standard

μ = media

$$\mu = \frac{\sum x_i}{N} \quad (1.2)$$

$$\sigma = \sqrt{\frac{\sum (x_i - \mu)^2}{N}} \quad (1.3)$$

N = numărul datelor

Distribuțiile normale sunt importante în statistică și sunt adesea folosite în științele naturale și sociale pentru a reprezenta variabile aleatoare cu valori reale ale căror distribuții nu sunt cunoscute. Importanța distribuțiilor normale se datorează parțial teoremei limită centrală ce afirmă că suma unui număr mare de variabile aleatoare identic distribuite, standardizate poate fi aproximată la o distribuție normală.

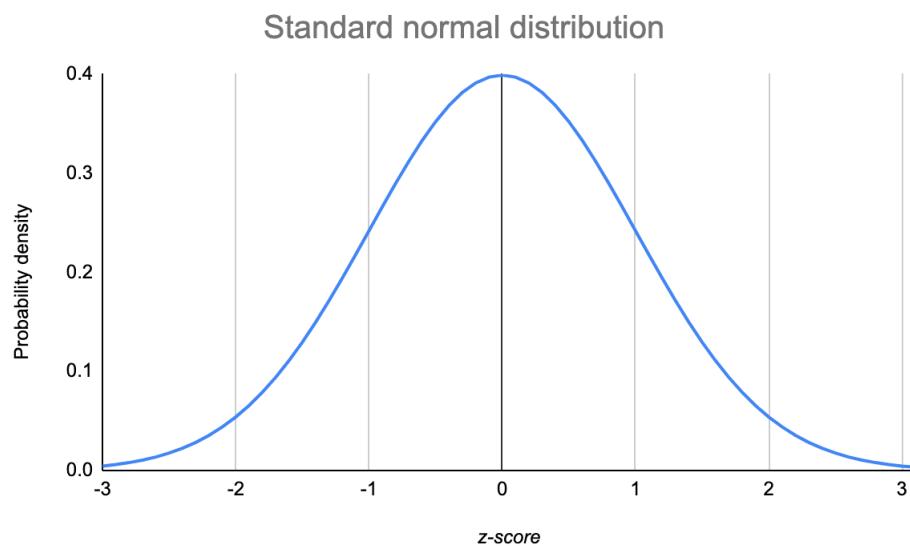


Figura 1.8. Reprezentarea grafică a unei distribuții normale. Clopotul lui Gauss¹⁰

Z test este un test statistic utilizat pentru a determina dacă două medii ale populației sunt diferite atunci când variațiile sunt cunoscute și dimensiunea eșantionului este mare. De asemenea, **Z test** este și un test statistic pentru care distribuția statisticii testului în ipoteza nulă poate fi aproximată printr-o distribuție normală

$$Z = \frac{\bar{X} - \mu_0}{s} \quad (1.4)$$

¹⁰<https://www.scribbr.com/statistics/standard-normal-distribution/>

s = deviația standard

\bar{X} = media aritmetică a eșantioanelor

μ_0 = media

Capitolul 2. Proiectarea aplicației

2.1. Platforma hardware

Platforma de evaluare a studenților în cadrul laboratoarelor a fost dezvoltată pe sistemul de operare *Windows*, folosind mediile de dezvoltare *Visual Studio Code* de la *Microsoft* pentru partea de front-end, iar pentru partea de back-end *IntelliJ IDEA* de la *JetBrains*.

Ambele medii de dezvoltare sunt disponibile atât pe *Windows*, cât și pe sistemul de operare *Linux* și cel de *macOS* dezvoltat de cei de la *Apple*. Cu alte cuvinte, platforma dezvoltată poate fi rulată pe mai multe sisteme de operare deoarece atât mediile de dezvoltare cât și tehnologiile folosite (*Java Spring Boot, Angular*) sunt disponibile de asemenea pe mai multe sisteme de operare.

2.2. Componente software

2.2.1. Interacțiunea dintre componente

Aplicația este construită pe 3 niveluri: nivelul de **front-end** (interfața cu utilizatorul), nivelul de **back-end** (locul unde sunt prelucrate datele) și nivelul de **baze de date** (locul unde sunt stocate datele).

Pentru nivelul de front-end, ca și mediu de dezvoltare s-a ales *Visual Studio Code*, iar ca și cadru de dezvoltare pentru construirea efectivă a interfeței cu utilizatorul s-a ales *Angular*.

În ceea ce privește nivelul de back-end, *IntelliJ IDEA* s-a folosit ca și mediu de dezvoltare, iar ca și limbaj de programare s-a folosit limbajul *Java* împreună cu cadrul de dezvoltare *Java Spring Boot*.

La nivelul bazelor de date, au fost alese baze de date relaționale cu sistemul de gestiune *MySQL*. De asemenea, tabelele din bazele de date sunt mapate din clasele *Java* cu ajutorul cadrului de dezvoltare *Hibernate* și a *JPA*-urilor.

Comunicarea dintre utilizator și partea de front-end este asigurată prin mesaje de tip cerere-răspuns ale protocolului *HTTP*, în timp ce comunicarea dintre partea de front-end și partea de back-end se realizează prin apeluri de *API* ce expun serviciile și livră drept răspuns obiecte de tipul *JSON*.

Maparea claselor *Java* în tabelele *SQL* se realizează cu ajutorul configurațiilor (cele ce țin de *JPA*, *Hibernate* și baza de date în fișierul *application.properties*) și adnotărilor din *Java Spring Boot* (@Entity, @Table, @Id, @Column, @ManyToMany și multe altele).

2.2.1.1. Front-end

Mediul de dezvoltare și editare ce s-a folosit pentru a implementa interfața cu utilizatorul la nivelul front-end-ului a fost *Visual Studio Code*. Astfel, pe partea de front-end se regăsesc 4 pachete ce conțin componente, servicii, modele și metode ajutătoare:

- pachetul de **componente** (eng. *components*) - pachet ce conține toate componentele *Angular* ce construiesc interfața cu utilizatorul;
- pachetul de **servicii** (eng. *services*) - pachet ce conține toate serviciile ce mapează apeluri către nivelul de back-end;
- pachetul de **modele** (eng. *models*) - pachet ce conține entități sub forma unor clase ce încapsulează mai multe informații;
- pachetul de **ajutoare** (eng. *helpers*) - pachet ce conține metode ajutătoare ce nu pot fi considerate servicii.

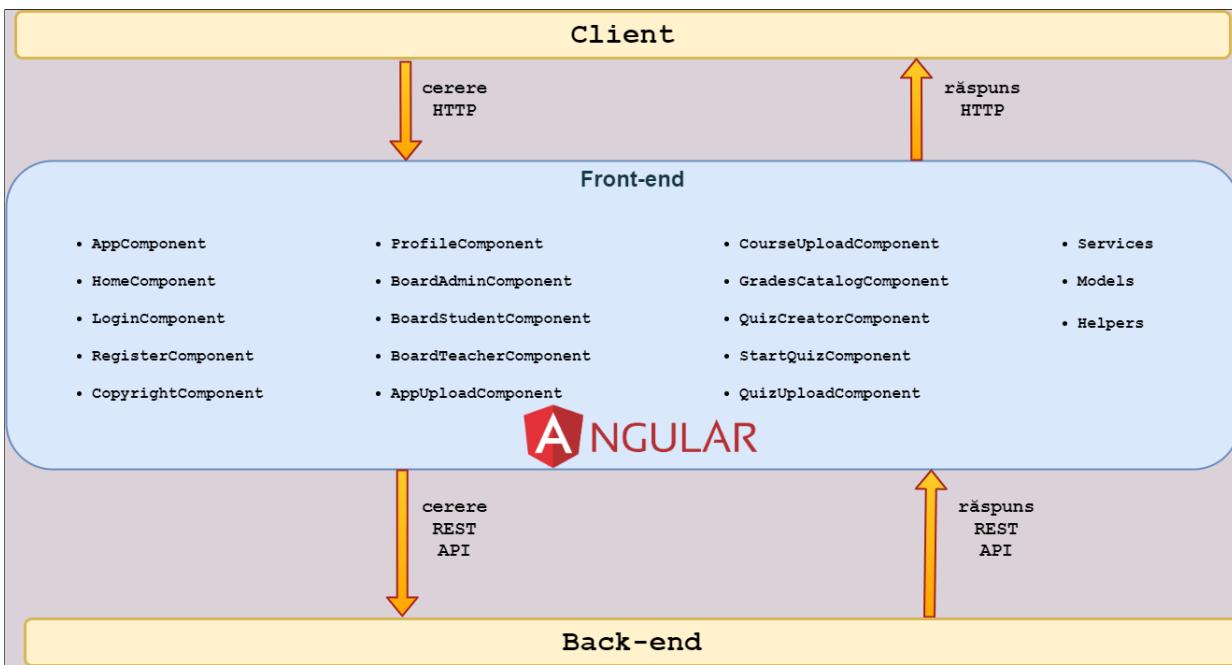


Figura 2.1. Interacțiunea dintre componente¹¹

După cum poate fi observat și în Figura 2.1, componentele Angular ale aplicației sunt:

- **AppComponent** - componentă responsabilă de pagina principală a aplicației împreună cu meniul de accesare a paginii de autentificare, de înregistrare, a paginii acasă, de ieșire din contul autentificat și de accesare a bordului în funcție de tipul de utilizator;
- **HomeComponent** - componentă responsabilă de pagina acasă (eng. *home*) ce conține un mesaj de întâmpinare împreună cu data și ora sistemului;
- **LoginComponent** - componentă responsabilă de pagina de autentificare (eng. *login*) pentru un utilizator; dacă utilizatorul nu are un cont, acesta poate accesa pagina de înregistrare din interiorul acestei componente;
- **RegisterComponent** - componentă responsabilă de pagina de înregistrare (eng. *register*) pentru un utilizator; dacă utilizatorul are un cont deja, acesta poate accesa pagina de autentificare din interiorul acestei componente;
- **CopyrightComponent** - componentă responsabilă cu afișarea unei casete ce conține numele autorului, anul în care s-a realizat și titlul proiectului (eng. *copyright*);
- **ProfileComponent** - componentă responsabilă cu afișarea paginii unde utilizatorul poate vizualiza detalii despre cont (eng. *profile*);
- **BoardAdminComponent** - componentă responsabilă cu afișarea bordului (eng. *board*) pentru utilizatorul administrator; în interiorul acestei componente se regăsește un tabel ce conține utilizatorii ce și-au creat un cont;
- **BoardTeacherComponent** - componentă responsabilă cu afișarea bordului (eng. *board*) pentru utilizatorul profesor; în interiorul acestei componente se regăsesc componente precum *GradesCatalogComponent*, *AppUploadComponent*, *CourseUploadComponent*, *QuizCreatorComponent* și *QuizUploadComponent*;

¹¹Evidențierea componentelor din nivelul de front-end

- **BoardStudentComponent** - componentă responsabilă cu afișarea bordului (eng. *board*) pentru utilizatorul student; în interiorul acestei componente se regăsește și componenta *StartQuizComponent*;
- **AppUploadComponent** - componentă responsabilă cu partea de încărcare a unui laborator de către un utilizator profesor; această componentă este disponibilă doar pentru utilizatorul profesor;
- **CourseUploadComponent** - componentă responsabilă cu partea de încărcare a unui curs (eng. *course*) de către un utilizator profesor; această componentă este disponibilă doar pentru utilizatorul profesor;
- **GradesCatalogComponent** - componentă responsabilă cu afișarea catalogului de note pentru tipurile de evaluare; această componentă este disponibilă doar pentru utilizatorul profesor;
- **QuizCreatorComponent** - componentă responsabilă cu crearea unei metode de evaluare de tip chestionar (eng. *quiz*); această componentă este disponibilă doar pentru utilizatorul profesor;
- **StartQuizComponent** - componentă responsabilă cu rularea unui chestionar (eng. *quiz*) de către un utilizator student; această componentă este disponibilă doar pentru utilizatorul student;
- **QuizUploadComponent** - componentă responsabilă cu partea de încărcare a unui chestionar creat (eng. *quiz*) de către un utilizator profesor; această componentă este disponibilă doar pentru utilizatorul profesor.

Serviciile prezente în aplicația Angular sunt servicii care ajută comunicarea între componentele Angular, gestionează JWT-ul sau expun serviciile din back-end:

- **AuthService** - serviciu care comunică cu partea de back-end responsabilă de autentificarea și înregistrarea pe platformă;
- **EvaluationStudentService** - serviciu care comunică cu partea de back-end responsabilă de situația studenților la toate evaluările;
- **FileUploadService** - serviciu care comunică cu partea de back-end responsabilă de încărcarea și descărcarea fișierelor (cursuri, laboratoare și chestionare);
- **HandleQuestionService** - serviciu folosit pentru a comunica date între componentele Angular;
- **TokenStorageService** - serviciu ce se ocupă de gestionarea JWT-ului.

2.2.1.2. Back-end

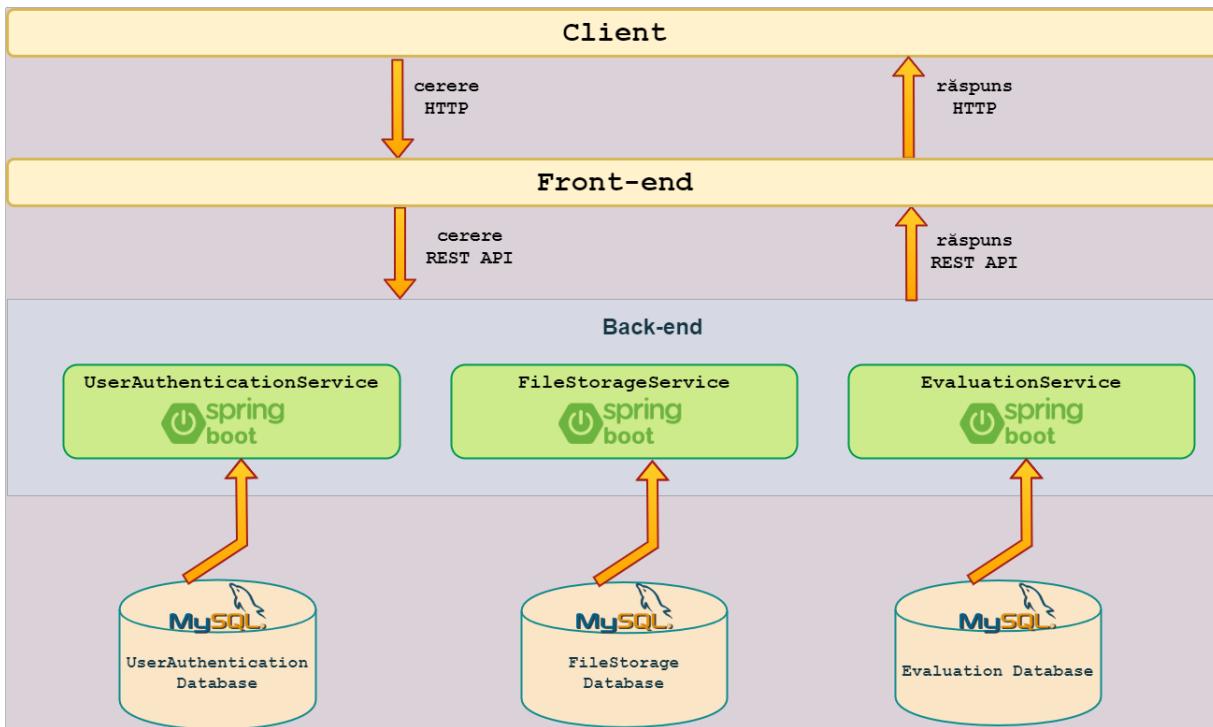


Figura 2.2. Interacțiunea dintre componente¹²

În Figura 2.2 se poate observa că nivelul de back-end este alcătuit din trei servicii implementate în limbajul *Java* și dezvoltate cu ajutorul cadrului de dezvoltare *Java Spring Boot*. Fiecare serviciu îi corespunde o bază de date diferită, gestionată de sistemul de gestiune *MySQL*.

Mediul de dezvoltare și editare ce s-a folosit pentru a implementa serviciile la nivelul back-end-ului a fost *IntelliJ IDEA*.

Primul serviciu din partea de back-end (**UserAuthenticationService**) este responsabil de gestionarea conturilor utilizatorilor de pe platformă. Cu alte cuvinte, la nivelul acestui serviciu sunt implementate funcțiile ce se ocupă de autentificarea utilizatorului în platformă, înregistrarea utilizatorului în platformă, generarea JWT-ului pentru autorizarea accesului utilizatorului la diferite resurse și alte metode adiționale ce extrag date preferențiale din baza de date.

Cel de-al doilea serviciu (**FileStorageService**) se ocupă cu stocarea fișierelor ce conțin notiuni de curs, de laborator și chestionare de evaluare în baza de date, dar și cu extragerea acestora pentru a putea fi descărcate local de către utilizator în partea de front-end. Așadar, fiecare tip de fișier (indiferent de extensie, dar cu o dimensiune maximă de 2 MB¹³) îi corespunde o tabelă SQL din care se pot extrage, stoca, actualiza sau șterge date cu ajutorul cererilor HTTP (*GET, PUT, POST, PATCH, DELETE*).

Ultimul serviciu (**EvaluationService**) este responsabil de gestionarea notei obținute de fiecare student în parte la fiecare tip de evaluare, precum și situația finală. În mare parte, serviciul stochează în baza de date numele, adresa de e-mail a studentului, nota obținută și tipul de evaluare după ce în partea de front-end este rulat un chestionar de evaluare de către un utilizator de tip student.

¹²Evidențierea serviciilor și bazelor de date din nivelul de back-end și de baze de date

¹³Megabyte - unitate de măsură pentru volumul de date; multiplu al byte-ului

2.2.2. Baza de date

La nivelul sistemului de gestiune MySQL se regăsesc trei baze de date, fiecare fiind destinată un serviciu dezvoltat în *Java Spring Boot*. În cazul în care nu sunt deja create, acestea se creează în momentul rulării serviciilor din partea de back-end, iar tabelele din fiecare bază de date sunt de fapt *JPA*-uri construite, descrise și mapate la clasele din Java, în funcție de anotările oferite de *Java Spring Boot*.

Ca și instrument vizual de dezvoltare, creare și întreținere a bazelor de date s-a folosit *MySQL Workbench*.

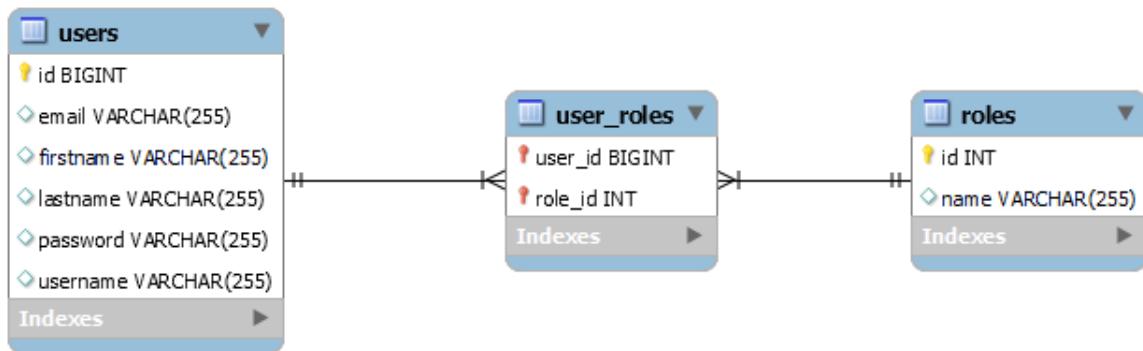


Figura 2.3. Diagrama ER pentru tabelele din baza de date userauthentication

În Figura 2.3 este reprezentată diagrama Entitate-Relație pentru tabelele din baza de date utilizator-autentificare (*userauthentication*). Aceasta diagramă conține următoarele tabele:

- **utilizatori** (eng. *users*) - tabela conține 6 câmpuri: un *id* unic, generat automat consecutiv (eng. *auto-increment*), un *email* ce reprezintă adresa de e-mail a utilizatorului, un câmp *firstname* ce reprezintă prenumele utilizatorului, un câmp *lastname* ce reprezintă numele de familie al utilizatorului, un câmp *password* reprezentat de parola utilizatorului și un câmp *username* ce semnifică numele cu care este identificat utilizatorul pe platformă;
- **roluri** (eng. *roles*) - tabela conține 2 câmpuri: un *id* unic, generat automat consecutiv (eng. *auto-increment*) și un câmp *name* ce reprezintă numele rolului (administrator, profesor sau student);
- **utilizator_roluri** (eng. *user_roles*) - este o tabelă creată în urma relației mulți-la-mulți (eng. *many-to-many*) dintre tabelele utilizatori și roluri și care conține *id*-ul utilizatorului și *id*-ul rolului.

Relația dintre tabelele **utilizatori** și **roluri** este de mulți-la-mulți (eng. *many-to-many*) deoarece un utilizator poate avea mai multe roluri în platformă, precum și același tip de rol poate să fie atribuit mai multor utilizatori. În urma acestei relații, se creează o tabelă intermedieră **utilizator_roluri** (eng. *user_roles*) ce conține identificatorul unui utilizator și identificatorul rolului pe care îl posedă, tabelă ce este într-o relație de mulți-la-unu (eng. *many-to-one*) cu tabela **utilizatori** și, de asemenea, într-o relație de mulți-la-unu (eng. *many-to-one*) și cu tabela **roluri**.

În Figura 2.4 sunt reprezentate tabelele ce gestionează fișierele ce conțin cursuri, laboratoare sau chestionare.

Tabela **laboratoare** (*apps*) este o tabelă în care se stochează fișierele ce conțin laboratoarele la o materie și este descrisă de 4 câmpuri: *id*-ul unic de tip text și generat de UUID¹⁴, câmpul *data* de tipul BLOB (*Binary large object*), adică sub forma un vector de biți ce mapează conținutul

fișierului, câmpul *name* ce reprezintă numele fișierului și câmpul *type* ce reprezintă tipul fișierului (text, audio, imagine și aşa mai departe).

Tabela **cursuri** (eng. *courses*) reprezintă o tabelă unde se stochează fișierele ce conțin cursurile la o materie. Aceasta tabelă este descrisă asemenea tablei **laboratoare**.

Ultima tabelă, cea de **chestionare** (eng. *quizzes*) stochează chestionarele create de utilizatorul profesor. Această tabelă este descrisă asemenea tabelelor de **cursuri** și **laboratoare**, cu mențiunea că mai există un câmp suplimentar denumit *content* ce semnifică conținutul de tip JSON al chestionarului, iar câmpului *name* î se aplică o constrângere unică (eng. *unique*). Astfel, în această tabelă nu pot exista două resurse cu același câmp *name*.

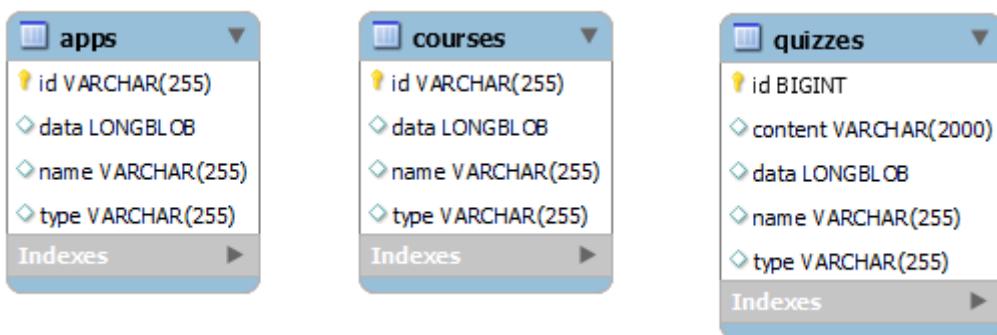


Figura 2.4. Descrierea tabelelor ce se ocupă de gestionarea cursurilor, laboratoarelor și a chestionarelor

Tabela **studenti** (eng. *students*), după cum este descrisă și în Figura 2.5, conține situația unui student în urma unei evaluări. Această tabelă conține astfel 6 câmpuri: un *id* unic, generat automat consecutiv (eng. *auto-increment*), un câmp *email* ce reprezintă adresa de e-mail a utilizatorului de tip student, un câmp *evaluation_type* semnificând tipul de evaluare (prima evaluare, a doua evaluare sau evaluarea finală), un câmp *firstname* ce reprezintă prenumele utilizatorului student, un câmp *grade* ce reprezintă nota obținuta la respectivul tip de evaluare și câmpul *lastname* ce semnifică numele de familie al utilizatorului student. De asemenea, în această tabelă există o constrângere unică (enq. *unique*) și anume că nu poate exista concomitent un utilizator de tip student cu aceeași adresă de e-mail și tip de evaluare.

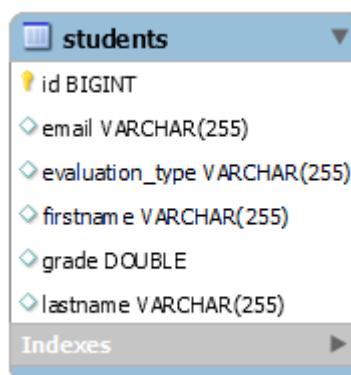


Figura 2.5. Descrierea tabelului ce conține situația unui student după o evaluare

2.2.3. Diagrame UML

¹⁴Universally Unique IDentifier - identificator unic universal

2.2.3.1. Diagrama de clase

În Figura 2.6 este reprezentată diagrama UML¹⁵ de clase a serviciului implementat în *Java Spring Boot* ce se ocupă de gestionarea conturilor utilizatorilor și de autentificarea și înregistrarea în platformă (*backend-authentication*).

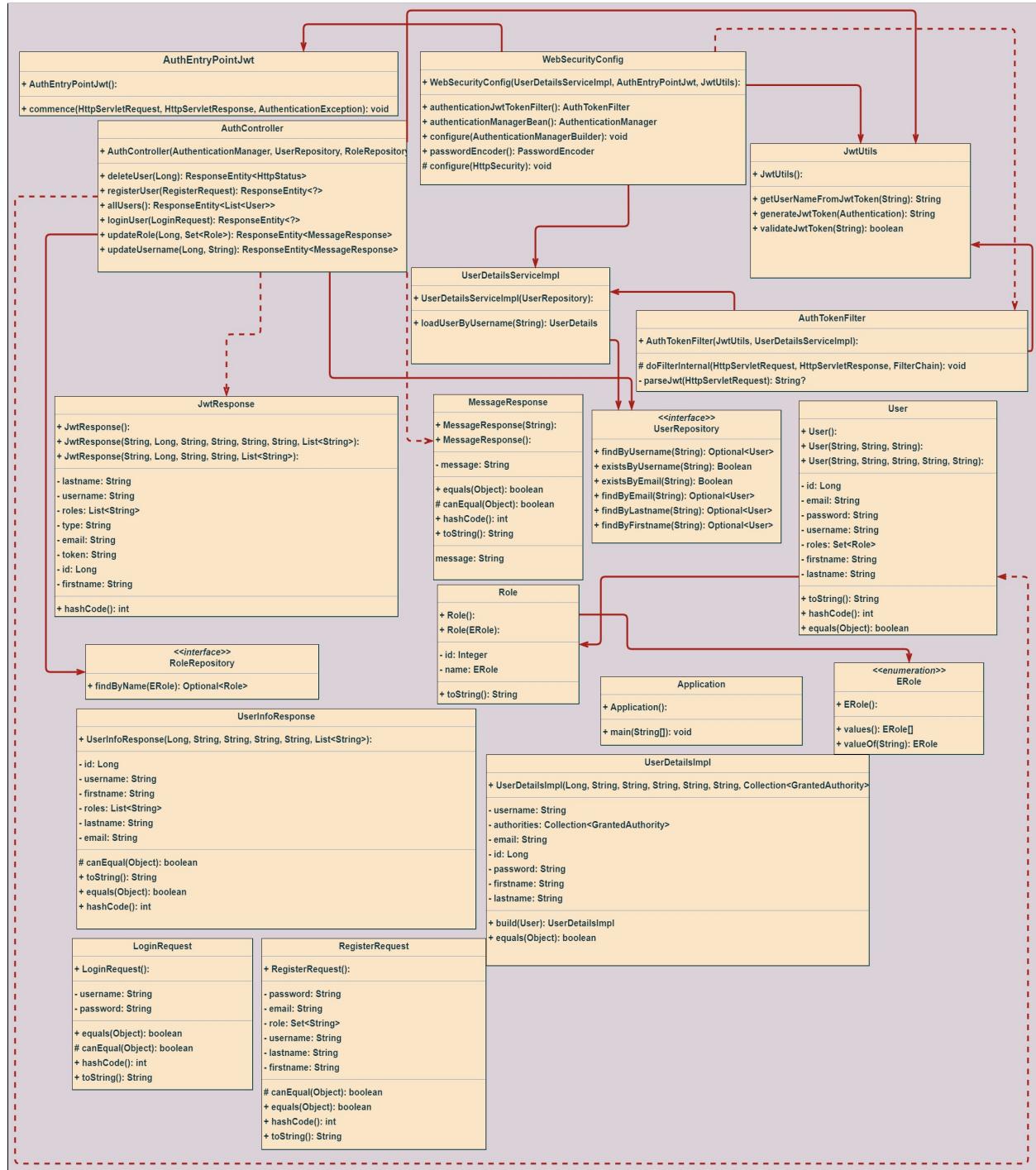


Figura 2.6. Diagrama de clase pentru serviciu ce se ocupă de autentificarea și înregistrarea utilizatorilor

În cadrul acestui serviciu se regăsesc 6 pachete:

- Primul pachet este responsabil pentru procesarea cererilor REST API primite, pregătirea unui răspuns și trimiterea acestuia (eng. *controller*). În interiorul acestui pachet se regăsește clasa **AuthController**.

¹⁵Unified Modeling Language - limbaj de descriere a modelelor software

- Al doilea pachet este responsabil cu structura modelelor (eng. *models*). În interiorul acestui pachet se regăsesc clasele **User**, **Role** și enumerația **ERole**.
- Al treilea pachet se ocupă de informația utilă din cadrul cererilor și răspunsurilor REST API, ce poate fi convertită sau actualizată (eng. *payload*). În interiorul acestui pachet se regăsesc clasele **LoginRequest** și **RegisterRequest** responsabile de cererile de autentificare și de înregistrare în platformă, gestionate într-un singur pachet (*request*), dar și clasele **JwtResponse**, **MessageResponse** și **UserInfoResponse** responsabile de gestionarea mesajelor de răspuns pentru cererile venite de la client. De asemenea, acestea sunt conținute de către un singur pachet (*response*).
- Al patrulea pachet se ocupă de maparea claselor Java în JPA-uri (eng. *repository*). În interiorul acestui pachet se regăsesc interfețele **UserRepository** și **RoleRepository**.
- Al cincilea pachet gestionează securitatea aplicației (eng. *security*). În interiorul acestui pachet se regăsesc clasele **AuthEntryPointJwt**, **AuthTokenFilter** și **JwtUtils** ce se află într-un pachet separat (*jwt*) și sunt responsabile de construirea, filtrarea și validarea JWT-ului. De asemenea, se mai regăsește și clasa **WebSecurityConfig** responsabilă de autorizarea accesului pe diferite resurse din platformă.
- Ultimul pachet este responsabil de gestionarea serviciilor (eng. *services*) din cadrul aplicației Java *Spring Boot*, iar în implementarea curentă, este responsabil de gestionarea privilegiilor fiecărui tip de utilizator (administrator, profesor sau student).

Pe lângă pachetele enumerate mai sus, care la rândul lor sunt gestionate într-un singur pachet (*com.userauthentication*), se regăsește și componenta principală cu ajutorul căreia se poate porni aplicația Java *Spring Boot* și anume clasa **Application**.

Figura 2.7 ilustrează diagrama UML de clase a serviciului ce gestionează situația unui student după fiecare tip de evaluare (*backend-evaluation*). În cadrul acestui serviciu, asemenea celui anterior, se regăsesc aceleși pachete, mai puțin cel de gestionarea securității, incluse într-un pachet principal (*com.evaluation*):

- În pachetul de gestionarea cererilor REST API se regăsește clasa **StudentController**.
- Pachetul de modele conține clasa **Student**.
- În pachetul responsabil de informația utilă se regăsește clasa **ResponseMessage**.
- Pachetul ce mapează clasele Java în JPA-uri conține interfața **StudentRepository**.
- În ultimul pachet, cel de gestionare a serviciului, este inclusă clasa **StudentService**.

Asemenea serviciului responsabil de gestionarea conturilor utilizatorilor din platformă și acest serviciu conține o componentă principală de unde se poate porni aplicația Java *Spring Boot*, mai exact clasa **EvaluationApplication**.

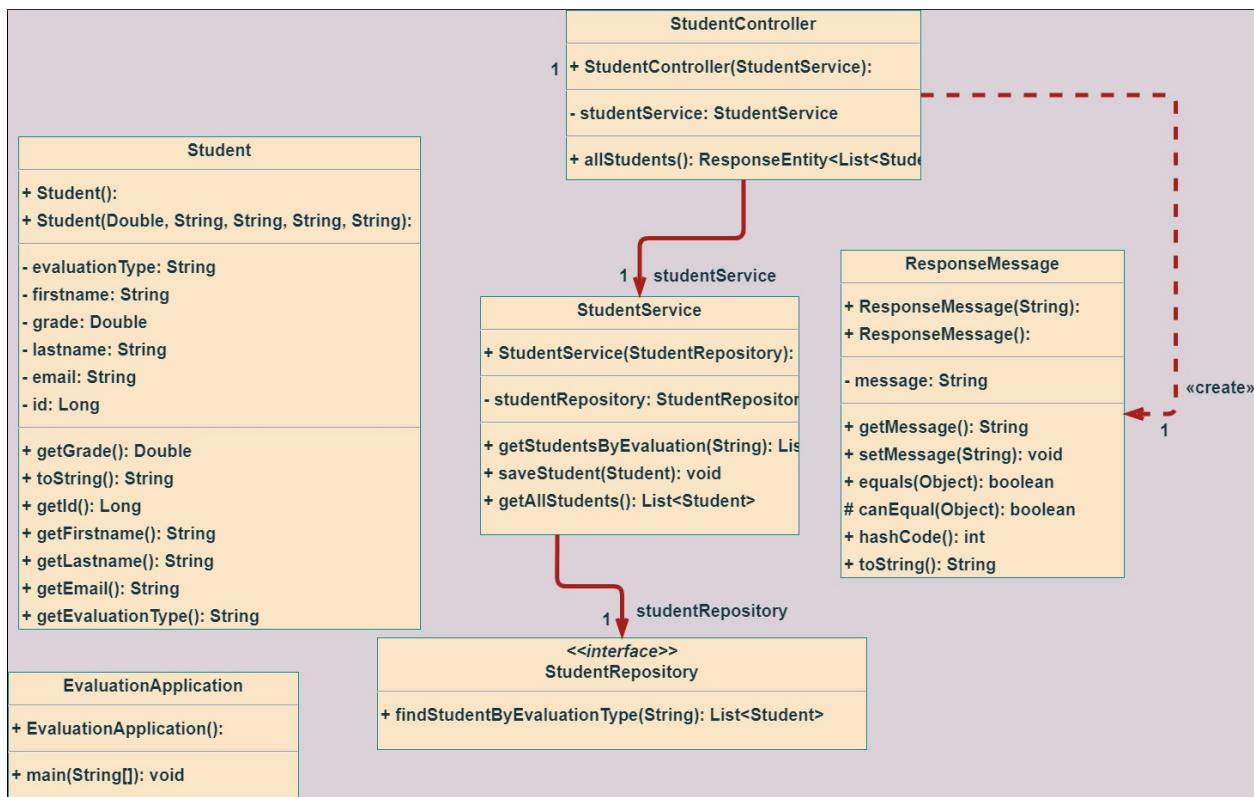


Figura 2.7. Diagrama de clase pentru serviciul ce se ocupă situația studenților după o evaluare

Diagrama din Figura 2.8 reprezintă diagrama de clase UML a serviciului ce gestionează fișierele ce conțin notiuni de cursuri, laboratoare și chestionare de evaluare (*backend-managefiles*). Asemenea celorlalte două servicii dezvoltate în *Java Spring Boot*, acest serviciu conține la rândul lui un pachet principal (*com.managefiles*) ce include alte 6 pachete:

- Pachetul de gestionare a cererilor REST API conține clasa **FileController**.
- În pachetul de modele sunt incluse clasele ce mapează cursurile (**CoursesStorage**), laboratoarele (**AppStorage**) și chestionarele de evaluare (**QuizStorage**).
- Pachetul de excepții (*exception*) este un pachet de gestiune a excepțiilor și include clasa **FileUploadExceptionAdvice**.
- În pachetul responsabil de informația utilă se regăsesc clasele **ResponseFile**, **ResponseFileQuiz** și **ResponseMessage**.
- Pachetul ce mapează clasele Java în JPA-uri conține interfețele **CoursesStorageRepository**, **AppsStorageRepository** și **QuizStorageRepository**.
- Ultimul pachet este acela în care este se gestionează serviciul și care conține clasa **FileStorageService**.

Componenta principală de unde se poate porni aplicația *Java Spring Boot* este reprezentată de clasa **Application**.

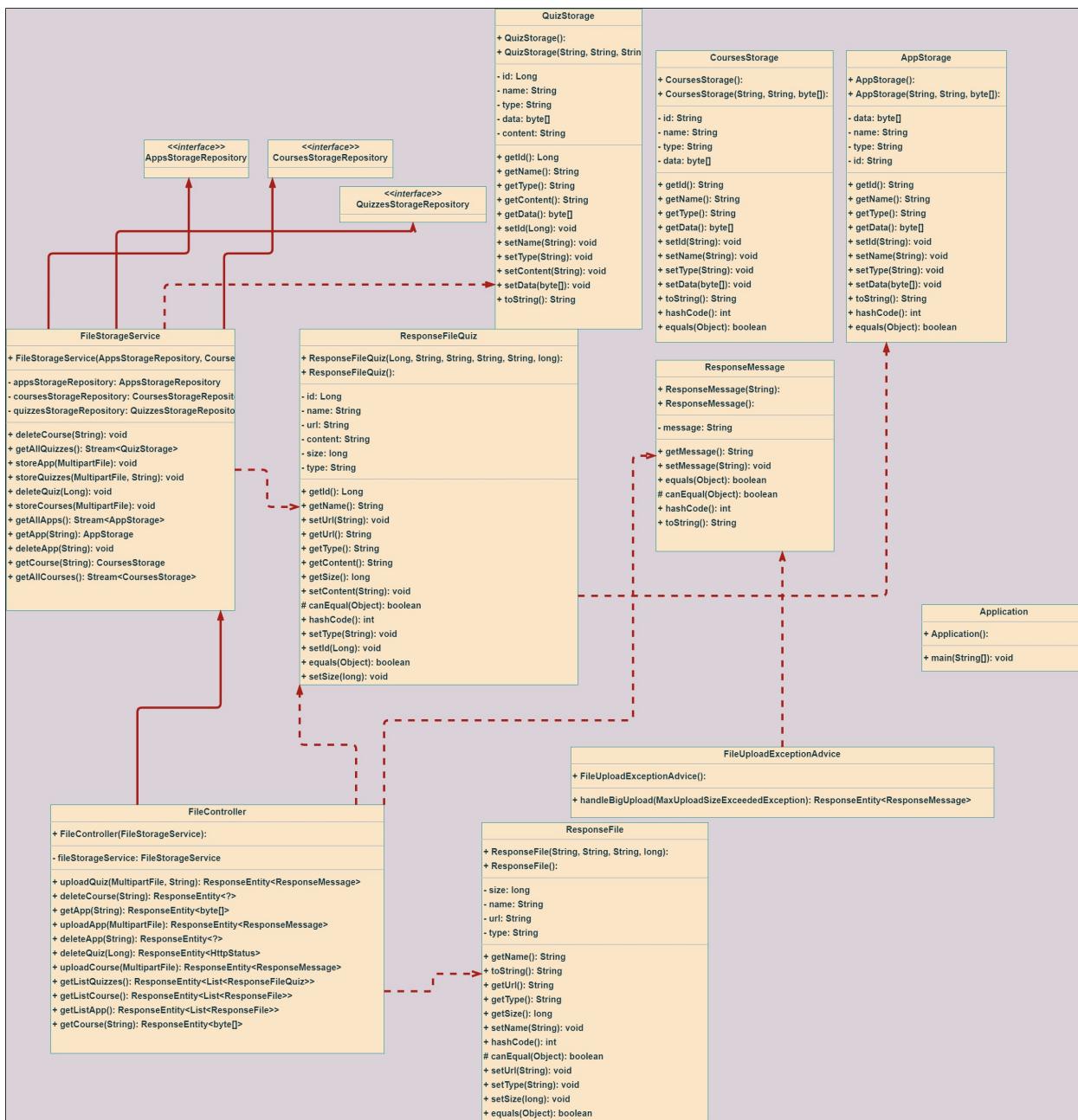


Figura 2.8. Diagrama de clase pentru serviciul gestionarea fișierelor

2.2.3.2. Diagrama de cazuri de utilizare (eng. *Use-Case*)

Figura 2.9 reprezintă diagrama cazuri de utilizare (eng. *Use-Case*) pentru platforma implementată. După cum se poate observa și în diagramă, platforma implementată dispune de trei tipuri de utilizatori:

- Utilizatorul de tip **administrator** (eng. *administrator*) - acest tip de utilizator poate șterge contul unui utilizator de tip student sau profesor, dar nu poate șterge un alt utilizator de tip administrator. De asemenea acesta poate adăuga utilizatori de tip profesor.
- Utilizator de tip **profesor** (eng. *teacher*) - poate încărca laboratoare, cursuri, vizualiza catalogul studentilor, să creeze un chestionar de evaluare și să-l încarce pe platformă și să vizualizeze statistici și rapoarte legate de situația studentilor la evaluări.
- Utilizator de tip **student** (eng. *student*) - are acces la cursuri și laboratoare și le poate descărca, și totodată poate să înceapă rezolvarea unui chestionar de evaluare, o singura dată fiecare tip

de evaluare.

După ce platforma a fost accesată cu succes, indiferent de tipul de utilizator, acesta are posibilitatea să-și vizualizeze profilul și să se deconecteze de pe platformă. Înregistrarea pe platformă poate fi făcută doar de către un utilizator de tip student întrucât după ce se creează un cont cu succes, acesta primește rolul de utilizator student automat. Odată contul creat, autentificarea poate fi făcută cu succes.

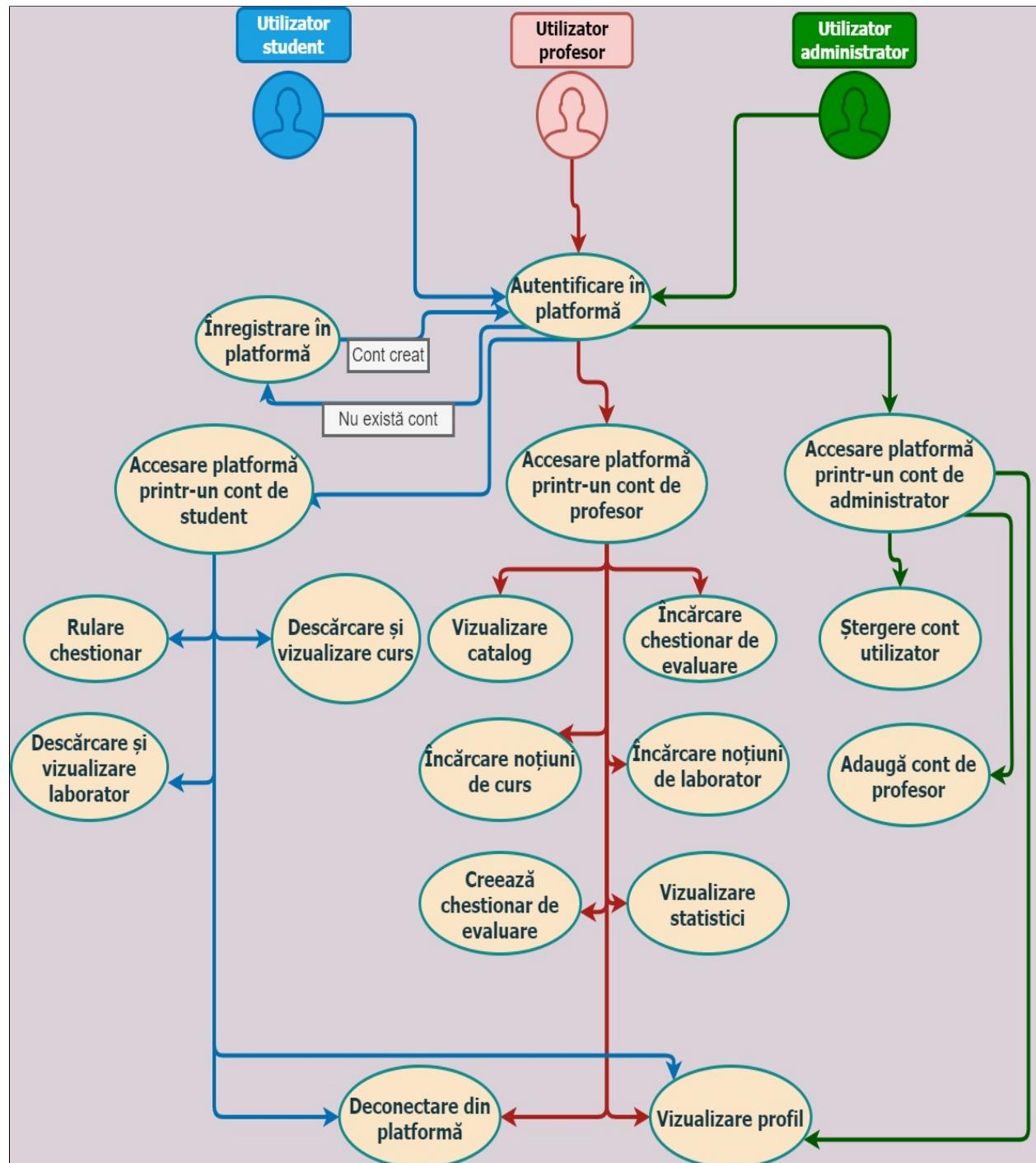


Figura 2.9. Diagrama cazuri de utilizare

2.2.3.3. Diagrama de activități

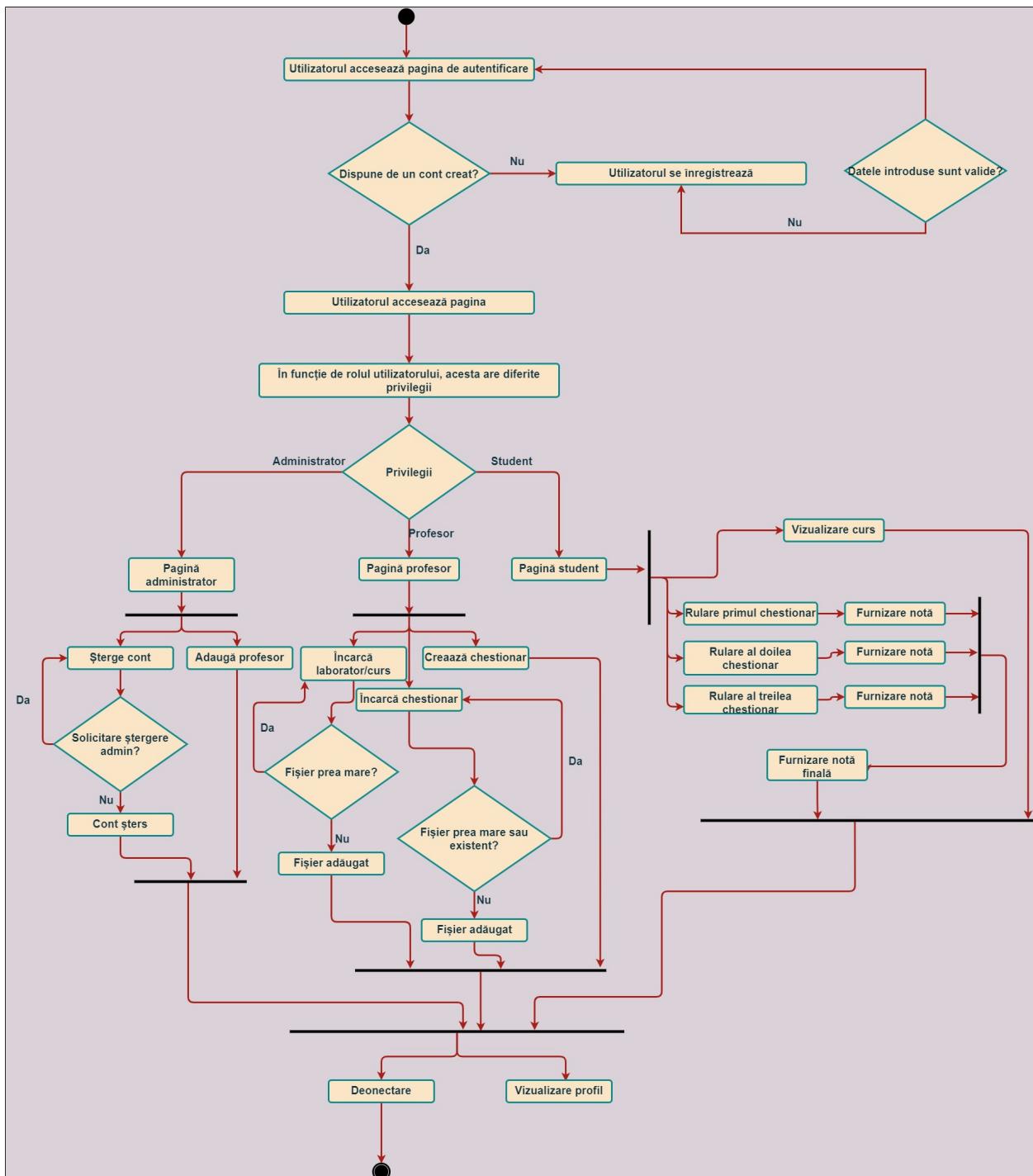


Figura 2.10. Diagrama de activități

2.3. Avantaje și dezavantaje

Unul dintre dezvantajele platformei este legat de autenticitatea utilizatorului de tip student, adică nu dispune de o verificare în ceea ce constă dacă utilizatorul este un student sau un utilizator obișnuit. Platforma în mod automat, la înregistrare, oferă dreptul de student oricărui utilizator care își creează un cont.

Un alt dezavantaj constă în pornirea manuală a serviciilor de back-end și autentificarea manuală în baza de date. O soluție ar fi fost folosirea Docker¹⁶.

¹⁶Instrument software ce utilizează virtualizarea la nivel de sistem de operare pentru a livra software în pachete

Un avantaj al acestei platforme este ușoara accesare a acesteia, întrucât nu necesită cunoștințe avansate pentru a o putea accesa, totul fiind foarte sugestiv, atât din postura unui utilizator de tip student cât și din postura unui utilizator de tip profesor.

Capitolul 3. Implementarea aplicației

3.1. Descrierea implementării pe partea de back-end

La nivelul back-end-ului au fost dezvoltate trei servicii independente cu ajutorul limbajului *Java*, mediului de dezvoltare *IntelliJ IDEA* și cadrului de dezvoltare *Java Spring Boot*.

3.1.1. Descrierea serviciului backend-authentication

Acest serviciu a fost dezvoltat și adaptat conform cerințelor proprii după urmărirea unui articol ajutător, prezentat în [11].

Conform Figurei 2.6 ce pune în evidență clasele din serviciul ce se ocupă de gestionarea conturilor utilizatorilor, serviciul de față este construit din 6 pachete.

Pachetul *model* este un pachet ce conține clase *Java*. Prima clasă (*ERole*) este de fapt un tip de date *enum*¹⁷ ce conține rolurile pe care le poate avea un utilizator în platformă. Rolul de student este mapat într-o constantă **ROLE_STUDENT**, rolul de profesor într-o constantă **ROLE_TEACHER**, iar rolul de administrator în **ROLE_ADMIN**. Cea de-a doua clasă *Java* este reprezentată de o clasă utilizator (eng. *User*) ce conține date despre identificatorul utilizatorului, numele de utilizator, numele de familie, prenumele, parola și rolul pe care îl are utilizatorul. Parola este stocată în baza de date sub formă criptată cu ajutorul *BCryptPasswordEncoder*, clasă ce oferă suport pentru criptarea de tip *BCrypt*¹⁸. Ultima clasă (*Role*) este responsabilă de structura unui rol și conține identificatorul acestuia și denumirea. Toate aceste clase sunt mapate într-o bază de date de tip *MySQL* în trei tabele diferite.

Payload este un pachet ce conține informația utilă ce se trimită ca și răspuns către client, dar și structura unei cereri pentru autentificare (clasa **LoginRequest**) și înregistrare (clasa **RegisterRequest**). Ca și clase ce se ocupă de răspunsul trimis către client se regăsesc **JwtResponse** ce mapează structura de JWT trimisă la client, **MessageResponse** ce gestionează mesajele trimise către client (de exemplu, cele de eroare) și **UserInfoResponse** ce conține informațiile despre utilizator trimise, de asemenea, către client.

Unul dintre cele mai importante pachete din cadrul acestui serviciu este cel ce gestionează securitatea serviciului (*security*). Prin intermediul clasei **WebSecurityConfig** se securizează aplicația deoarece aceasta moștenește clasa **WebSecurityConfigurerAdapter** din *Spring Security*¹⁹ ce oferă suport pentru *CORS*, *CSRF*²⁰, resurse partajate și managementul sesiunii. Clasa **AuthEntryPointJwt** se ocupă de mesajul de eroare ce poate apărea în cazul în care utilizatorul încearcă să acceseze o resursă la care nu are drepturi (spre exemplu, *401 Unauthorized*). Clasele **AuthTokenFilter** și **JwtUtils** sunt responsabile de crearea, semnarea, validarea, parsarea, perioada de valabilitate a JWT-ului și autorizarea accesului utilizatorului.

Pachetul de servicii (*services*) are în componență două clase **UserDetailsImpl** și **UserDetailsServiceImpl**. Cu ajutorul primei clase putem structura datele utilizatorului după autentificare (email, nume de familie, prenume, nume de utilizator, rol și parolă), iar prin cea de a doua clasă putem gestiona un obiect de tip utilizator cu ajutorul unei interfețe din *Spring Security*, **UserDetailsService** ce ușurează accesul la date.

Pachetul de unde se gestionează cererile *HTTP* (*controllers*) conține clasa **AuthController** ce include metodele de tip *POST* (pentru partea de autentificare) și *POST* (pentru partea de înregistrare), dar și metode adiționale de tip *CRUD*²¹, folosindu-se totodată și de mapările claselor

¹⁷ Tip de date ce permite ca o variabilă să fie un set de constante predefinite

¹⁸ funcție de criptare a datelor

¹⁹ Cadrul de dezvoltare Java ce oferă suport pentru autentificare, autorizare și alte funcționalități ce țin de securitate

²⁰ Tip de atac ce permite unui atacator să onlige utilizatorii să efectueze acțiuni pe care nu intenționează să le facă

²¹ Creare, extragere, actualizare și ștergere resurse

Java în JPA-uri, într-o bază de date MySQL. La înregistrare, se verifică în baza de date existența unui utilizator cu același nume de utilizator sau aceeași adresă de e-mail. În cazul în care toate datele au fost validate cu succes, acesta are posibilitatea să se autentifice în platformă. În momentul autentificării utilizatorului, acestuia i se generează un JWT prin intermediul căruia i se gestionează accesul la platformă, mai exact ce pagini poate să acceseze. După ce expiră JWT-ul, acesta este deconectat din platformă în mod automat.

3.1.2. Descrierea serviciului *backend-managefiles*

Figura 2.8 ilustrează clasele Java din componența serviciului care se ocupă de gestionarea fișierelor ce conțin noțiuni legate de cursuri, laboratoare și chestionare de evaluare.

Asemenea serviciului anterior, serviciul *back-end-managefiles* este structurat pe pachete. Diferența se face la nivelul securității, întrucât în cadrul acestui serviciu nu este nevoie de *Spring Security*, fiind vorba doar de salvarea și extragerea din baza de date a unor fișiere cu dimensiunea maximă de 2 MB.

Un curs este mapat în clasa **CoursesStorage**, un laborator în **AppStorage** iar un chestionar de evaluare în **QuizStorage**. Primele două clase au aceeași structură, adică ele conțin un id generat de UUID, o coloană unică ce reține numele fișierului, tipul fișierului (text, imagine, video și aşa mai departe) și o coloană de tip *BLOB* ce mapează conținutul fișierului într-un vector de biți. QuizStorage pe lângă aceste date, mai reține și o coloană sub forma unui text, mai exact conținutul, codificat și stocat în *Base64*²². Aceste clase sunt reținute în baza de date MySQL sub forma unor tabele, cu ajutorul JPA-urilor și instrumentului de mapare *Hibernate*.

Răspunsurile trimise către client sunt gestionate de clasele **ResponseFile**, **ResponseFileQuiz** și **ResponseMessage**, iar mesajul de eroare ce este transmis în cazul în care dimensiunea fișierului este prea mare, prin clasa **FileUploadExceptionAdvice**.

La nivelul pachetului de servicii se regăsește o singură clasă **FileStorageService** ce conține ca și proprietăți, interfețele ce moștenesc interfața pentru JPA, iar ca și funcții, cele ce se ocupă de extragerea, stocarea și ștergerea unui fișier, indiferent de tipul acestuia. Funcțiile de extragere conțin printre parametrii de intrare și un tip de dată *MultipartFile*. Acest tip de dată se folosește pentru încărcarea fișierelor, dar și pentru a transmite mai multe tipuri de date într-o singură cerere (de exemplu, un fișier și un JSON).

Clasa **FileController**, ce gestionează cererile HTTP, are în componență metode ce mapează cererile de tip *GET*, *POST* și *DELETE* și scot în evidență operațiile *CRUD*. Pentru încărcarea unui curs sau a unui laborator, doar se tratează funcționalitatea, unde în caz de succes se trimit ca și răspuns un mesaj corespunzător și statutul cererii HTTP (*200 OK*), iar în caz de eroare un mesaj de eroare și statutul cererii HTTP (*417 Expectation failed*).

3.1.3. Descrierea serviciului *backend-evaluation*

În Figura 2.7 se pot observa clasele ce alcătuiesc serviciul *back-end-evaluation*, serviciu ce se ocupă de gestionarea situației studenților după fiecare tip de evaluare.

Acest serviciu este cel mai puțin complex dintre toate serviciile de back-end implementate, întrucât responsabilitatea acestuia este de a stoca informațiile unui student într-o baza de date MySQL, sub forma unei tabele, și de a calcula media finală.

Clasa **Student** este alcătuită din proprietăți: un id unic generat automat, un nume de familie, un prenume, o adresă de e-mail, nota obținută și tipul de evaluare (prima evaluare, a doua evaluare și evaluare finală). De asemenea, se aplică o constrângere unică astfel încât în baza de date nu se poate stoca un student cu aceeași adresă de e-mail și același tip de evaluare. Această constrângere previne la nivelul back-end-ului stocarea mai multor rezultate ale unui student pentru același tip de evaluare.

²²Schemă de codificare binar-text

Ca și la serviciile anterioare, clasa **ResponseMessage** gestionează mesajul trimis către client în cazul în care cererea a fost sau nu procesată cu succes.

La nivelul pachetului *controller* se regăsește clasa **StudentController** ce gestionează cererile *HTTP* de tip *GET*, *POST* și *DELETE*.

3.2. Descrierea implementării pe partea de front-end

Partea de front-end a fost implementată cu ajutorul cadrului de dezvoltare *Angular* și a mediului de dezvoltare *VisualStudioCode*. De asemenea, aspectul interfeței a fost realizat cu ajutorul librăriei *Angular Material* și a cadrului de dezvoltare *Bootstrap*.

Aplicația *Angular* conține astădat componente *Angular* numite și *ngModules*, servicii, modele și metode ajutătoare necesare în logica aplicației. Comunicarea dintre partea de front-end și partea de back-end se realizează prin mesaje de tip cerere-răspuns *REST API*, gestionate de serviciile din *Angular*. Funcțiile incluse în servicii apelează *API*-uri din partea de back-end și primesc drept răspuns obiecte de tip *JSON*.

3.2.1. Pagina principală

Pagina principală aplicației, mai exact pagina de acasă (*HomeComponent*) conține o bară de instrumente de unde se poate accesa pagina de autentificare și pagina de înregistrare. În momentul în care un utilizator se autentică, aspectul barei de instrumente se modifică și va conține accesul către pagina destinată tipului de utilizator, accesul către vizualizarea profilului și posibilitatea de deconectare din platformă. Se mai poate observa în Figura 3.1, un mesaj de întâmpinare, data și ora curentă.



Figura 3.1. Pagina principală a platformei

3.2.2. Pagina de autentificare

În cadrul paginii de autentificare (Figura 3.2) se regăsește un formular ce conține ca și câmpuri numele de utilizator și parola. La nivel acestor câmpuri s-au aplicat metode de validare astfel:

- **nume de utilizator** (eng. *username*) - câmp obligatoriu, de minim 6 caractere și care respectă modelul impus adică acesta să nu conțină caractere speciale.

- **parolă** (eng. *password*) - câmp obligatoriu, de minim 8 caractere ce trebuie să conține o majusculă, o literă mică, o cifră și un caracter special.

După completarea formularului și apăsarea butonului de autenticare (*Login*), dacă datele introduse sunt valide se realizează autentificarea în platformă cu ajutorul serviciului **AuthService** din *Angular*, ce expune un apel de tip *POST* către back-end. Dacă datele introduse corespund celor din baza de date, atunci autentificarea se realizează cu succes. În caz contrar, va apărea un mesaj de eroare (de exemplu, credențiale greșite).

Dacă utilizatorul este nou și nu dispune de un cont, acesta are posibilitatea să acceze pagina de înregistrare prin apăsarea *hyperlink-ului*²³ *Register here*.

În momentul autentificării cu succes în platformă a utilizatorului, acestuia i se generează un JWT din partea de back-end, iar în momentul expirării acestuia, utilizatorul va fi deconectat în mod automat din platformă. Gestionarea JWT-ului, mai exact stocarea în sesiune a acestuia și ce se întâmplă atunci când acesta expiră sunt implementate la nivelul serviciului *TokenStorageService*, dar și cu ajutorul unei librării externe *angular-jwt*.

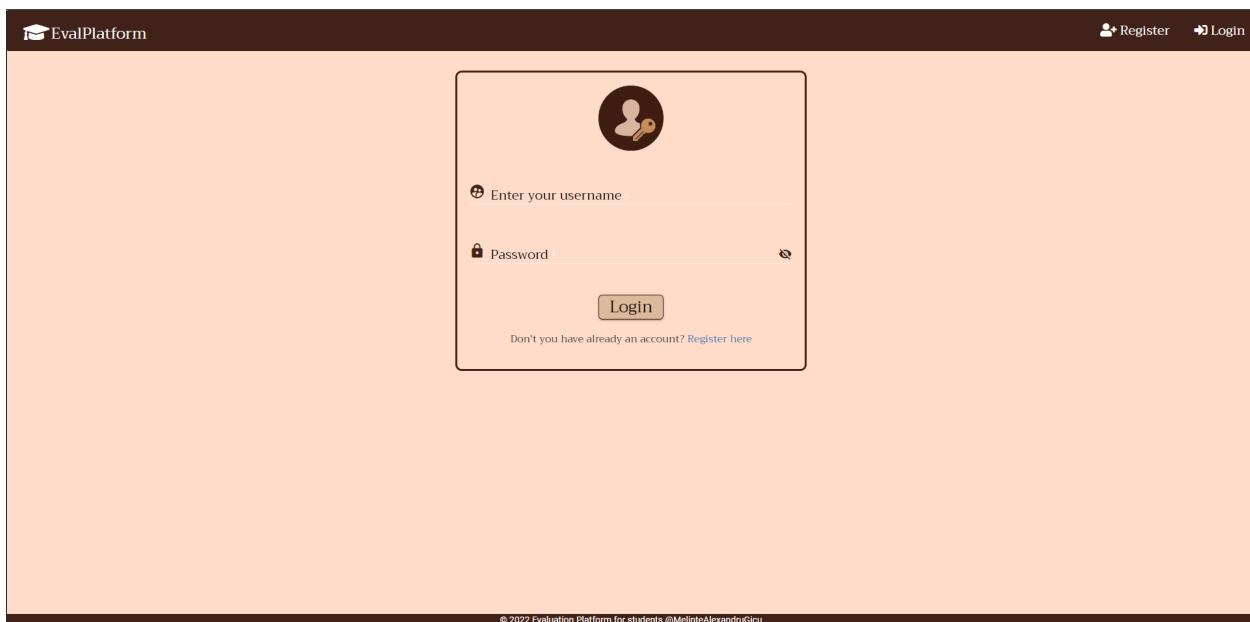


Figura 3.2. Pagina de autentificare a platformei

3.2.3. Pagina de înregistrare

În Figura 3.3 se poate observa pagina de înregistrare, care asemenea paginii de autentificare conține un formular, dar de data aceasta cu 6 câmpuri: adresa de e-mail, numele de utilizator, numele de familie, prenumele, parola și confirmarea parolei. Validarea câmpurilor de nume de utilizator și parolă este asemenea formularului de pe pagina de autentificare, iar cea a câmpului ce conține adresa de e-mail implică obligativitatea utilizatorului de a introduce o adresă de email sub forma [adresa.email@yahoo.com].

Dacă formularul completat este unul valid, în urma apăsării butonului de înregistrare (*Register*) utilizatorul va primi un mesaj de înștiințare cum că înregistrarea a fost efectuată cu succes. Logica de funcționare este gestionată prin serviciul **AuthService** ce expune un apel de tip *POST* către back-end.

De asemenea, în afișajul acestei pagini se mai regăsește posibilitatea accesării paginii de autentificare direct din formular prin apăsarea *hyperlink-ului* *Login here*.

²³element HTML ce conține o adresă URL pe care utilizatorul o poate accesa în urma apăsării

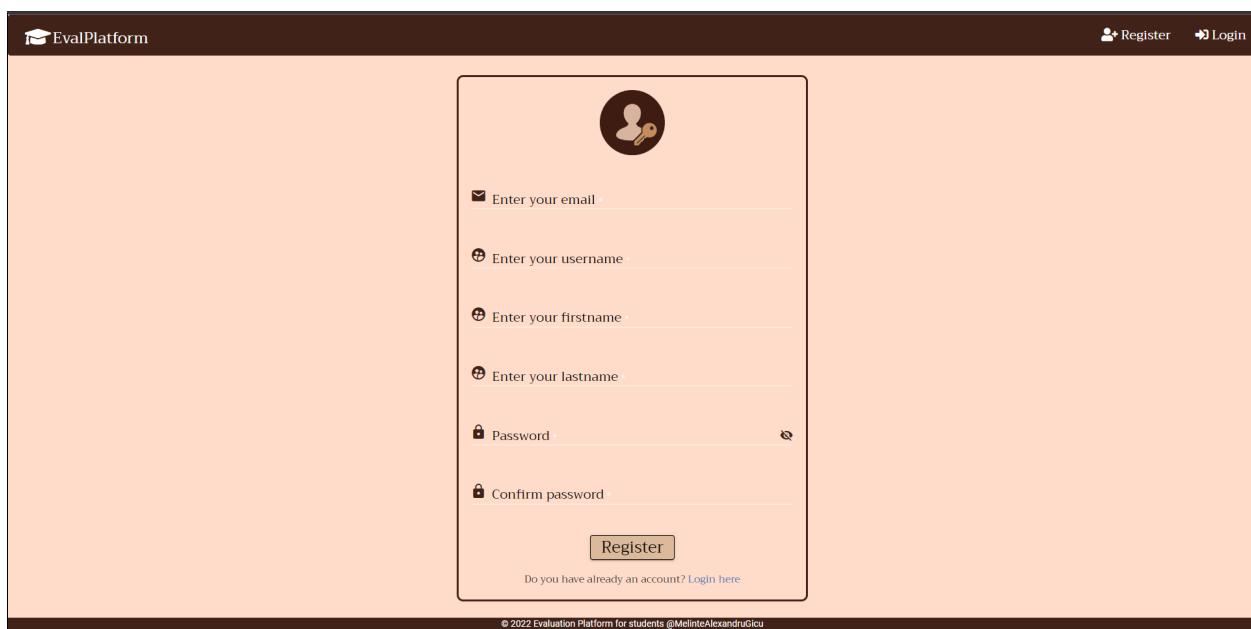


Figura 3.3. Pagina de înregistrare a platformei

3.2.4. Pagina destinată utilizatorului de tip administrator

Utilizatorul administrator, în urma unei autentificări valide, accesează pagina din Figura 3.4. Acesta are posibilitatea să adauge un nou cont de profesor prin apăsarea butonului *Add Teacher* ce expune la nivel de implementare un apel către o metodă POST din cadrul serviciul *backend-authentication*. Pe lângă acest privilegiu, utilizator de tip administrator gestionează conturile din platformă, adică are posibilitatea să steargă anumiți utilizatori, dar doar dacă nu sunt la rândul lor și ei utilizatori de tip administrator.

Nr.	Role	Username	F_firstname	Lastname	Email	Operation
1	ROLE_ADMIN	Administrator	Alexandru	Melinte	alex.melinte@yahoo.com	
3	ROLE_STUDENT	GigelFrone	Gigel	Frone	gigi.frone@yahoo.com	
6	ROLE_STUDENT	DorinelMunteanu	Dorinel	Munteanu	dorinel.munteanu@yahoo.com	
8	ROLE_TEACHER	AlexMelinte	Alexandru-Gicu	Melinte	ag.melinte@yahoo.com	

+ Add teacher

Figura 3.4. Pagină destinată utilizatorului de tip administrator

3.2.5. Pagina destinată utilizatorului de tip profesor

Utilizatorul profesor, în urma unei autentificări valide, accesează pagina corespunzătoare rolului de profesor. Acesta are posibilitatea să acceseze catalogul de studenți, să încarce laboratoare și cursuri, să creeze un chestionar de evaluare și să-l salveze local sub forma unui *JSON* și să

încarce înr-o bază de date chestionarul creat.

3.2.5.1. Componența responsabilă de catalog

În Figura 3.5 este ilustrat catalogul de studenți la care are acces utilizatorul profesor. Acesta are posibilitatea să vizualizeze toți studenții sau să-i filtreze după tipul de evaluare. Prin apăsarea butoanelor *First evaluation*, *Second evaluation*, *Final evaluation*, *All students* se efectuează filtrarea catalogului după tipul de evaluare. Această filtrare este implementată cu ajutorul serviciului **EvaluationStudentService** ce expune apelul unei metode de tip *GET* către serverul de back-end.

Nr.	Firstname	Lastname	Email	Grade	Evaluation
1	Gigel	Frone	gigi.frone@yahoo.com	4	firstEvaluation
3	Student	Model	student.model@yahoo.com	10	firstEvaluation
4	Student	Model	student.model@yahoo.com	1	secondEvaluation
5	Student	Model	student.model@yahoo.com	7	finalEvaluation
6	John	Doe	john.doe@yahoo.com	5.5	firstEvaluation
7	John	Doe	john.doe@yahoo.com	1	secondEvaluation
8	John	Doe	john.doe@yahoo.com	10	finalEvaluation

Applications

Courses

Tests

First - second - final

© 2022 Evaluation Platform for students @MelinteAlexandruGicu

Figura 3.5. Catalogul de studenți

3.2.5.2. Componentele responsabile de cursuri și laboratoare

Utilizatorul de tip profesor dispune de vizualizarea componentelor ce-i oferă posibilitatea să încarce cursuri și laboratoare. Încărcarea se efectuează prin apăsarea butonului *Upload Course* sau *Upload app*, butoane ce devin active în momentul în care este selectat un fișier. Cu ajutorul serviciului **FileUploadService** ce expune apeluri de metode de tip *POST* către serverul de back-end se asigură funcționalitatea acestor butoane.

3.2.5.3. Componentele responsabile de chestionarul de evaluare

Figura 3.6 reprezintă componenta responsabilă de crearea și încărcarea unui tip de evaluare. Prin apăsarea butonului *Create quiz for the first evaluation* se deschide o fereastră de dialog, exact ca în Figura 3.7. Această fereastră conține la rândul ei o componentă responsabilă doar de crearea și salvarea locală a chestionarului. Chestionarul este creat dintr-un număr variabil de întrebări, dar cu restricția să existe minim două. O întrebare este alcătuită din trei răspunsuri gresite și un răspuns corect. Pentru navigarea la următoarea întrebare se apasă butonul *Next*, iar pentru revenirea la o întrebare anterioară butonul *Back*. Se poate adăuga o nouă întrebare la chestionar prin apăsarea butonului *Add question*. După ce utilizatorul profesor termină de creaț chestionarul de evaluare, chestionarul poate fi salvat local sub denumirea tipului de evaluare (spre exemplu, *firstEvaluation.json*).

Odată creat, chestionarul poate fi încărcat în baza de date prin butonul de încărcare ilustrat în Figura 3.6, buton ce devine activ în momentul în care este selectat un fișier. De asemenea, se pot vizualiza toate chestionarele încărcate deja.

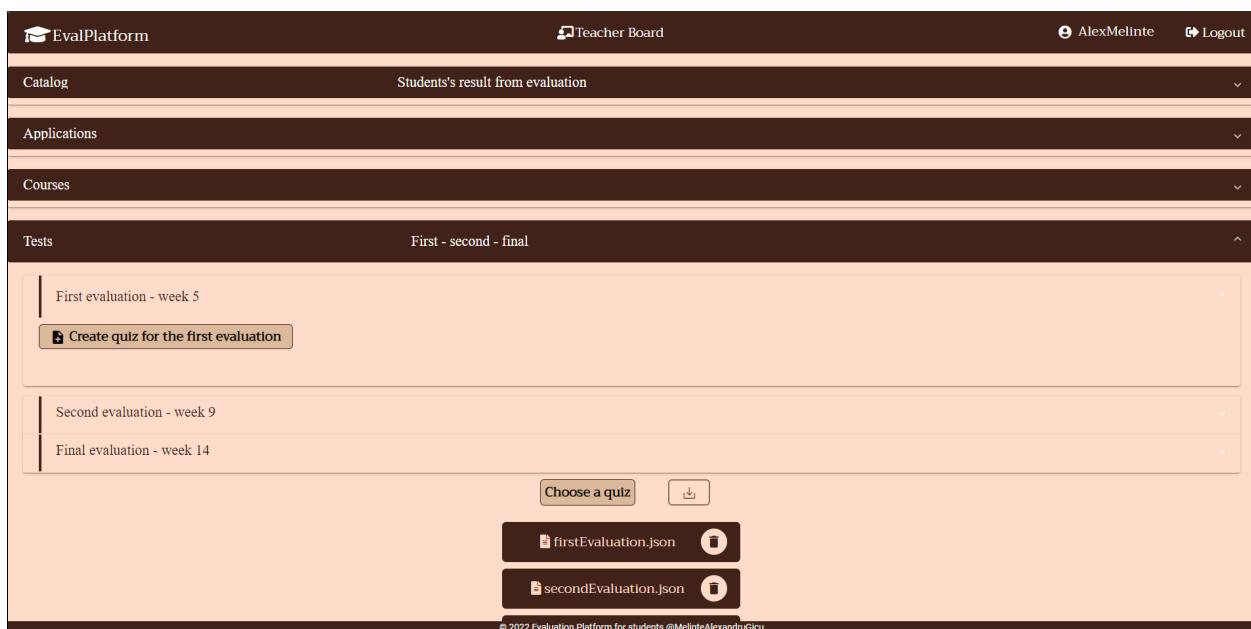


Figura 3.6. Componenta responsabilă gestionarea unui chestionar de evaluare

În funcție de rezultatul încărcării se va afișa un mesaj de eroare sau de succes. Pentru a putea încărca fișierul *firstEvaluation.json*, utilizatorul profesor trebuie să se asigure că în lista de chestionare nu mai este prezent un alt fișier sub această denumire. Dacă există, acesta trebuie șters prin apăsarea butonului de ștergere.

Funcționalitatea butonelor de încărcare și ștergere a chestionarelor de evaluare, dar și vizualizarea chestionarelor deja existente este asigurată de serviciul **FileUploadService**, ce expune cereri te tip *POST*, *GET* și *DELETE* către serverul de back-end.



Figura 3.7. Crearea unui chestionar de evaluare

3.2.6. Pagina destinată utilizatorului de tip student

Utilizatorul de tip student are posibilitatea să descarce și să vizualizeze cursurile și laboratoarele încărcate de utilizatorul profesor și să rezolve un chestionar de evaluare pentru fiecare tip de evaluare.

3.2.6.1. Componentele responsabile de vizualizarea și descărcarea cursurilor și laboratoarelor

În Figura 3.8 se poate observa că, laboratoarele și cursurile sunt dispuse separat, sub forma unor liste. Prin apăsarea *hyperlink*-ului fiecărui fișier, acesta va fi descărcat local de unde poate fi vizualizat. Această funcționalitate este asigurată prin intermediul serviciului **FileUploadService** ce expune o cerere de tip GET către serverul de back-end și se extrage *URL*-ul din răspunsul primit.

The screenshot shows the 'EvalPlatform' application interface. At the top, there's a navigation bar with the platform logo, 'Student Board', user information ('JohnDoe'), and a 'Logout' button. Below the navigation bar, there are two main sections: 'Applications' and 'Courses'. The 'Applications' section contains a list of files: 'Laborator 4.txt', 'Laborator 3.txt', 'Laborator 5.txt', 'Laborator 2.txt', 'Laborator 6.txt', and 'Laborator 1.txt'. The 'Courses' section contains a list of files: 'Curs3.txt', 'Curs1.txt', and 'Curs2.txt'. At the bottom of the page, there's a copyright notice: '© 2022 Evaluation Platform for students @MelinteAlexandruGicu'.

Figura 3.8. Componentele responsabile de vizualizarea și descărcarea cursurilor și laboratoarelor

3.2.6.2. Componența responsabilă de rezolvarea chestionarului de evaluare

Butonul de *Start quiz* din Figura 3.9 deschide o fereastră de dialog ce conține componența *StartQuizComponent*, din Figura 3.10. Înainte de pornirea unui chestionar de evaluare, utilizatorul de tip student este înștiințat că rezultatele la evaluarea respectivă nu sunt disponibile.

The screenshot shows the 'EvalPlatform' application interface. At the top, there's a navigation bar with the platform logo, 'Student Board', user information ('JohnDoe'), and a 'Logout' button. Below the navigation bar, there are three main sections: 'Applications', 'Courses', and 'Evaluation'. The 'Evaluation' section contains a list of evaluations: 'Week 5', 'Week 9', and 'Week 14'. Under 'Week 5', there's a button labeled '→ Start quiz'. A modal window titled 'First Evaluation' is displayed, containing the message 'Results are not available. Run the test first!'. At the bottom of the page, there's a copyright notice: '© 2022 Evaluation Platform for students @MelinteAlexandruGicu'.

Figura 3.9. Pornirea unui chestionar de evaluare

După pornirea unui chestionar de evaluare, se încarcă din baza de date întrebările și răspunsurile. Acestea sunt stocate în forma *Base64* și înainte de a fi afișate în componentă sunt decodeate. Acest lucru previne posibilitatea utilizatorului să vadă pe traficul de rețea cererea și corpul mesajului, unde poate găsi răspunsurile. După ce se răspunde la o întrebare, se trece în mod automat la întrebarea următoare. Dacă se dorește a trece peste întrebare, fără a răspunde momentan, se va apăsa butonul *Next*.

Dacă se dorește terminarea și salvarea chestionarului, se apasă pe butonul *Submit*. Acesta trebuie apăsat de asemenea și în momentul în care au fost rezolvate toate întrebările sau timpul alocat a expirat. Timpul alocat este calculat în funcție de numărul de întrebări. Unei întrebări îi corespunde un minut, aşadar după cum este ilustrat în Figura 3.10, fiind doar două întrebări, timpul alocat va fi de 120 de secunde, adică două minute. Se poate vizualiza numărul întrebării curente în interiorul acestei componente.

După efectuarea celor trei tipuri de evaluare, nota finală va fi afișată, într-un corp de mesaj separat.

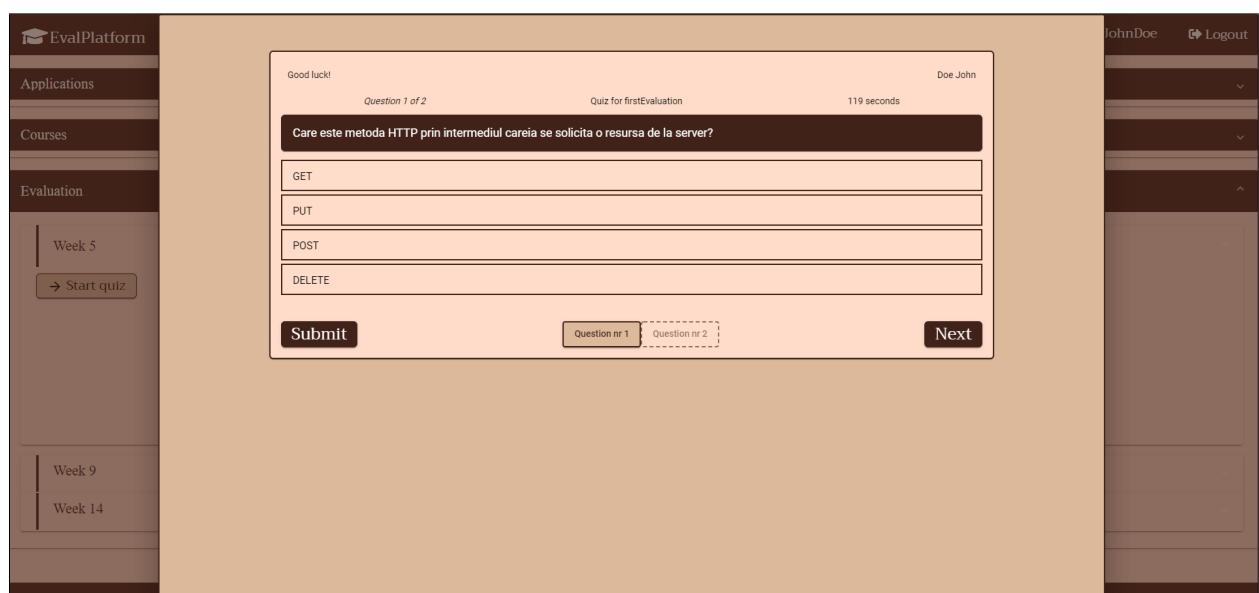


Figura 3.10. Chestionarul de evaluare

După ce s-au salvat răspunsurile, nota este calculată și afișată în dreptul tipului de evaluare, împreună cu numărul de întrebări la care s-a răspuns corect, respectiv greșit.

3.2.7. Pagina de vizualizare profil

Pagina de vizualizare profil conține date despre profilul utilizatorului, mai exact numele de utilizator, adresa de e-mail, numele de familie, prenumele și rolul pe care acesta îl deține în platformă.

Din interiorul acestei pagini, utilizatorul poate să-și editeze profilul, schimbându-și numele de utilizator.

3.2.8. Deconectarea din platformă

Deconectarea din platformă se face prin accesarea butonului *Logout*. Funcționalitatea este asigurată prin intermediul serviciului *TokenStorageService* ce conține o funcție care șterge JWT-ul din cadrul sesiunii. Astfel utilizatorul va rămâne fără un JWT asignat, iar pentru a accesa resurse este nevoie de o nouă autentificare.

3.3. Dificultăți întâmpinate și modalități de rezolvare

Printre dificultățile întâmpinate pe parcursul implementării aplicației se enumeră generația JWT-ului, validarea acestuia și modul în care este folosit atât pe partea de back-end, cât și pe partea de front-end pentru a putea restricționa accesul. Soluția găsită pentru partea de back-end a fost folosirea claselor ce țin de securitatea serviciului *backend-authentication*, mai exact extinderea claselor **OncePerRequestFilter**, **AuthenticationEntryPoint** și **WebSecurityConfigurerAdapter**. Această soluție a fost urmărită și implementată după mai multe parcurgeri ale unor videoclipuri ajutătoare în care s-au prezentat modul de folosire, rolul acestor clase și modul de gestionare a unei aplicații *Java Spring Boot* dezvoltată și cu ajutorul *Spring Security*. Ulterior, dezvoltarea din tutorial a fost implementată și ajustată conform cerințelor platformei.

O altă dificultate întâmpinată a fost legată de transmiterea către serverul de back-end din-spre serverul de front-end a JSON-ului ce conține chestionarul de evaluare. În primă fază am întâmpinat probleme cu stocarea acestuia în baza de date sub forma unui JSON, dar ca și soluție de rezolvare a fost una simplă, scriind practic în baza de date un *String* de lungime mai mare. Dezvoltarea pe partea de front-end nu a întâmpinat probleme, însă doar se transmite un parametru către funcția ce expune cererea *HTTP (POST)* spre serverul de back-end.

Găsirea unei librării ce dispune de afișarea graficelor, statisticilor, clopotului lui Gauss și integrarea acesteia în *Angular* a fost de asemenea o problemă întâmpinată pe parcurs. Soluția în schimb s-a dovedit a fi una ușoară prin folosirea librăriei *ApexCharts*.

O ultimă problemă întâlnită a fost legată de accesul unui utilizator neautentificat la o resursă prin intermediul scrierii manuale a *URL*-ului, dar și accesul unui utilizator autentificat la o resursă la care acesta nu ar trebui să aibă acces, tot prin scriere manuală. Soluția a fost folosirea *AuthGuard*-ului din *Angular* și suprascrierea metodei *CanActivate*.

3.4. Idei originale, implementări proprii

Ideile originale din implementarea platformei constau în:

- Salvarea chestionarelor sub forma unor *JSON*-uri și urcarea acestora în baza de date, cu întregul conținut codificat, pentru a putea fi rulate ulterior, în cadrul evaluării, de către utilizatorul student.
- Compararea clopotului lui Gauss original cu cel obținut în urma furnizării notelor studenților și tragerea concluziilor. Dacă vârful celui obținut este centrat mai la stânga decât clopotul lui Gauss normal, atunci dificultatea testului a fost una ridicată. În cazul în care vârful clopotului obținut este centrat mai la dreapta, atunci testul a fost unul ușor. În cazul în care apar fluctuații, adică mai mult de un vârf, atunci nu se poate trage o concluzie concretă. Cauzele pot fi diferite, de la trișare până la învățare în grupuri de studenți.
- Aspectul interfeței cu utilizatorul este de asemenea implementat într-o manieră originală.
- Logica de salvare și rulare a chestionarelor, dar și furnizarea notelor și salvarea acestora într-o bază de date au fost, de asemenea, idei originale.
- Logica de salvare și descărcare a fișierelor a fost implementată într-o manieră proprie, având ca punct de plecare un videoclip ajutător.

Capitolul 4. Testarea aplicației și rezultate experimentale

4.1. Lansarea aplicației, elemente de configurare

Pentru a putea fi accesată cu succes platforma, necesită a fi pornită manual mai întâi baza de date, apoi serviciile din back-end și în final serverul de *Angular*.

Fiecare serviciu din back-end rulează pe un port separat, are propriul *URL* și pornește cu ajutorul unui server *Apache Tomcat*²⁴ configurat la nivelul aplicației *Java Spring Boot*.

Serviciul de back-end ce se ocupă de gestionarea conturilor utilizatorilor și de autentificare, înregistrare și generarea JWT-ului rulează pe adresa [*localhost:6060/api/auth/<path>*].

Al doilea serviciu de back-end ce se ocupă de gestionarea fișierelor rulează pe adresa [*localhost:6061/api/fileStorage/<path>*].

Ultimul serviciu de back-end ce se ocupă de situația studentului după un tip de evaluare rulează pe adresa [*localhost:6062/api/evaluation/<path>*].

Bazele de date, la nivelul fiecărui serviciu, sunt create dacă ele nu există deja prin furnizarea numelui de utilizator și a parolei bazei de date și adăugarea unor informații adiționale legate de *Hibernate* și *DataSource* în fișierul *application.properties* din Java. Un model este cel din Figura 4.1, unde *filestorage* poate fi înlocuit cu numele dorit pentru baza de date.

```
spring.datasource.url=jdbc:mysql://localhost:3306/<nume_baza_de_date>?useSSL=false&&createDatabaseIfNotExist=true
spring.datasource.username=<nume_de_utilizator>
spring.datasource.password=<parola>
spring.jpa.properties.hibernate.dialect= org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.hibernate.ddl-auto= update
```

Figura 4.1. Exemplu de configurare *application.properties*

În ceea ce privește serverul de front-end, pornirea acestuia este la fel de necesară ca a serviciilor de back-end. Configurarea acestuia se realizează prin instalarea tuturor pachetelor conținute de aplicație prin rularea comenzi *npm install*. Serverul *Angular* se pornește, de exemplu, din terminalul mediului de dezvoltare *Visual Studio Code*, prin comanda *ng serve* rulată din folderul principal al aplicației. Acest server va rula pe adresa [*localhost:4200/<path>*] (exemplu Figura 4.2).

```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **

✓ Compiled successfully.
✓ Browser application bundle generation complete.

5 unchanged chunks

Build at: 2022-07-05T12:09:17.090Z - Hash: e1bd94c697c30983 - Time: 2461ms

✓ Compiled successfully.
```

Figura 4.2. Exemplu de rulare server *Angular* cu succes

4.2. Rezultate experimentale

Figura 4.3 reprezintă ce vede utilizatorul în momentul în care dorește să se înregistreze pe platformă, dar nu completează câmpurile aferente. Câmpurile din formularul de înregistrare sunt obligatorii de completat și mai conțin la rândul lor constrângeri specifice. Spre exemplu, adresa de

²⁴Server web ce permite programatorilor să ruleze aplicații Web

e-mail trebuie să respecte formatul unui email, numele de utilizator nu trebuie să conțină caractere speciale, parolele trebuie să se potrivească și să conțină majusculă, literă mică, cifră, caracter special și minim 8 caractere, iar câmpurile legate de nume și prenume trebuie să conțină doar litere și minim 3 caractere.

The image shows a registration form with the following fields and validation messages:

- Email:** Enter your email *
Email is required
- Username:** Enter your username *
Username is required
- Firstname:** Enter your firstname *
Firstname is required
- Lastname:** Enter your lastname *
Lastname is required
- Password:** Password *
Password is required
- Confirm password:** Confirm password *
Confirm password is required

A large red error message "All fields are required" is displayed at the bottom of the form. A "Register" button is located below the fields, and a link "Do you have already an account? [Login here](#)" is at the bottom right.

Figura 4.3. Validare câmpuri pentru pagina de înregistrare

Dacă toate câmpurile au fost validate, iar comunicarea dintre serverul de back-end și cel de front-end este stabilă, în urma apăsării butonului *Register*, ar trebui să apară un mesaj de însătișare, exact ca în Figura 4.4.



Figura 4.4. Înregistrare cu succes

În Figura 4.5 este ilustrată, asemănător paginii de înregistrare, validarea câmpurilor din pagina de autentificare.

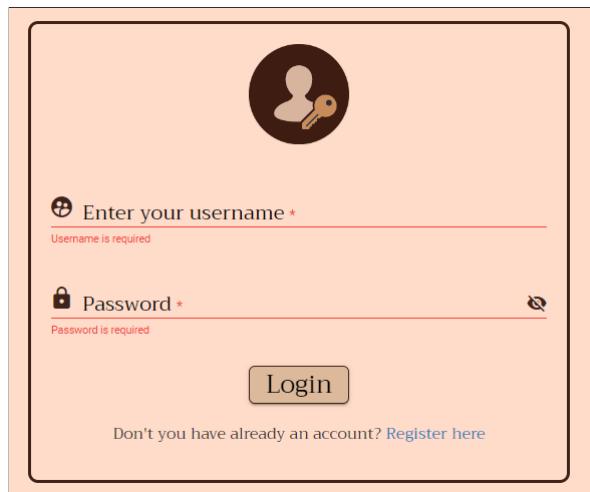


Figura 4.5. Validare câmpuri pentru pagina de autentificare

Dacă numele de utilizator și parola sunt valide, adică există stocat în baza de date un utilizator cu acest nume și această parolă, iar comunicarea dintre serverul de back-end și cel de front-end este stabilă, în urma apăsării butonului *Login*, ar trebui să apară un mesaj de înștiințare, exact ca în Figura 4.4, iar utilizatorul să fie redirectat către resursa specifică rolului, adică în cazul de față, fiind vorba de un utilizator student, acesta să acceze pagina de student.

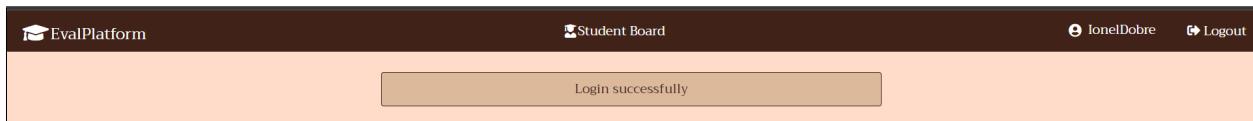


Figura 4.6. Autentificare cu succes

În urma rulării unui chestionar de evaluare de către un utilizator student, acestuia i se va furniza rezultatul evaluării, adică nota obținută, numărul de întrebări la care s-a răspuns corect și numărul de întrebări la care s-a răspuns greșit. O vizualizare a acestor rezultate este ilustrată în Figura 4.7.

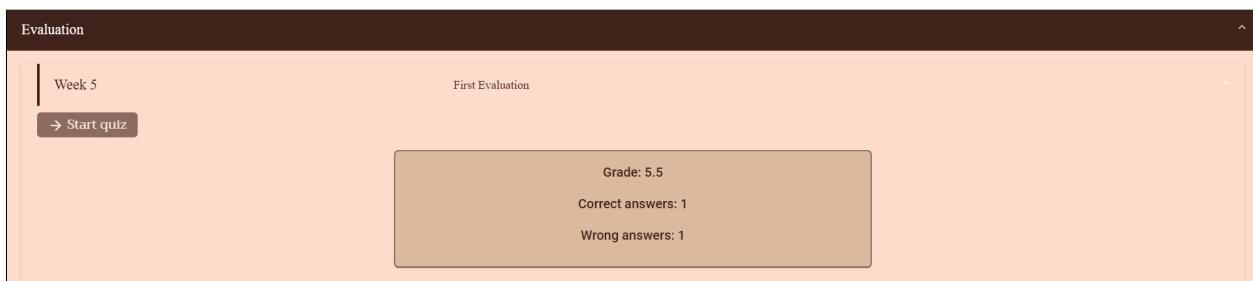


Figura 4.7. Furnizare rezultate după evaluare

După parcurgerea celor trei evaluări, utilizatorului de tip student îi va fi calculată nota finală și furnizată exact ca în Figura 4.8.



Figura 4.8. Furnizare rezultat final după parcurgerea tuturor evaluărilor

Utilizatorul de tip profesor are posibilitatea ca după ce creează un chestionar de evaluare să încarce într-o bază de date respectivul chestionar. Dacă în baza de date nu există un chestionar deja creat cu același nume, rezultatul încărcării va fi cel din Figura 4.9.

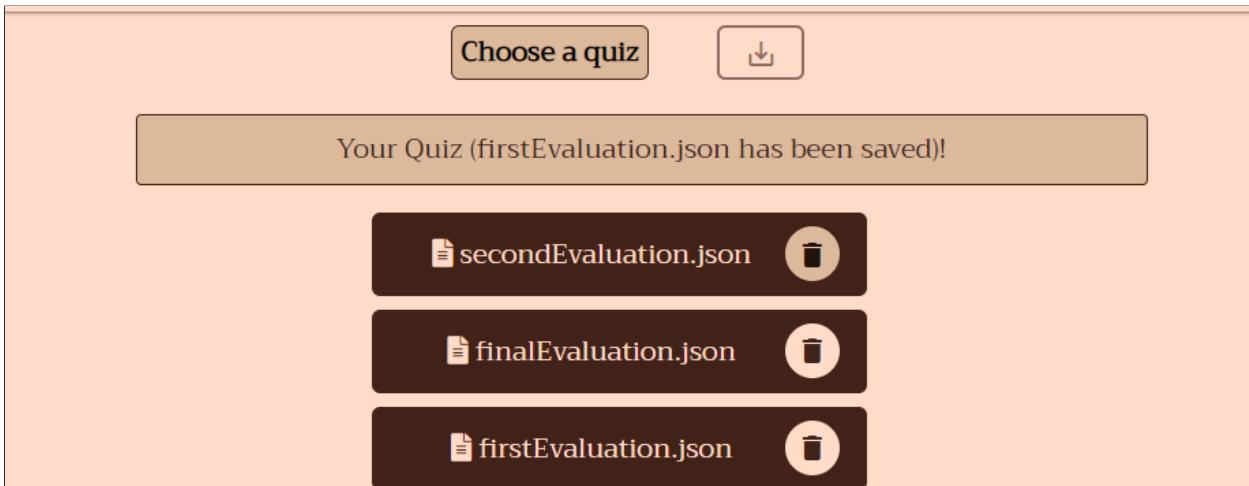


Figura 4.9. Încărcare chestionar de evaluare cu succes

În cazul în care în baza de date există deja un chestionar cu același nume, utilizatorul profesor va fi întărit de acest lucru printr-un mesaj de eroare, reprezentat în Figura 4.10. De asemenea, un alt mesaj de eroare va apărea atunci când utilizatorul încearcă să încarce un fișier cu dimensiunea mai mare de 2 MB.

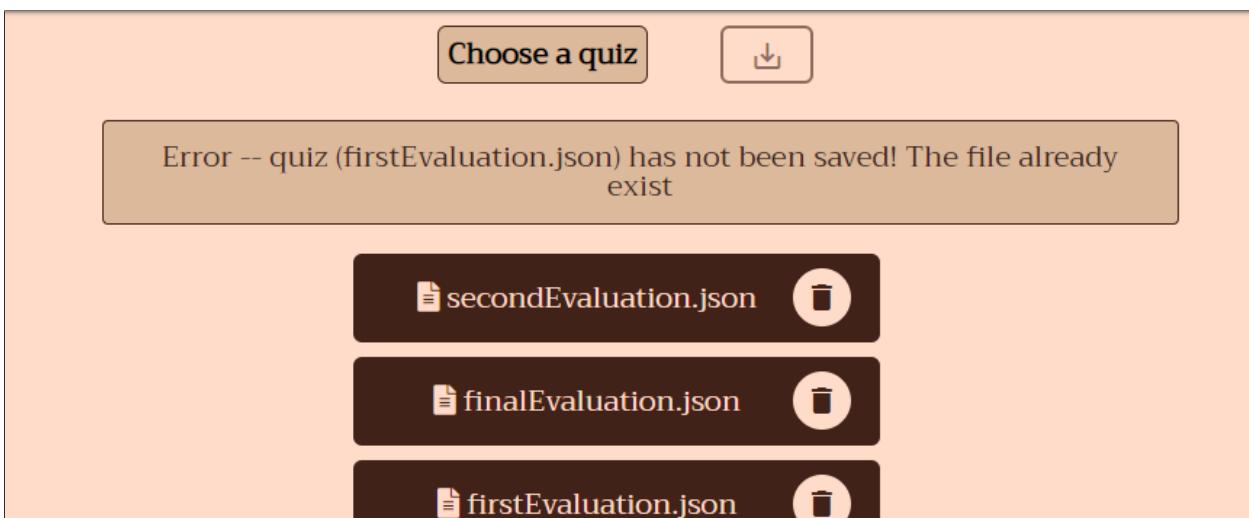


Figura 4.10. Eroare încărcare chestionar de evaluare

Mesajul de eroare legat de dimensiunea fișierului va apărea și atunci când utilizatorul profesor va încerca să încarce un curs sau un laborator cu dimensiunea mai mare de 2 MB. O vizualizare

a acestui mesaj de eroare este reprezentată în Figura 4.11.

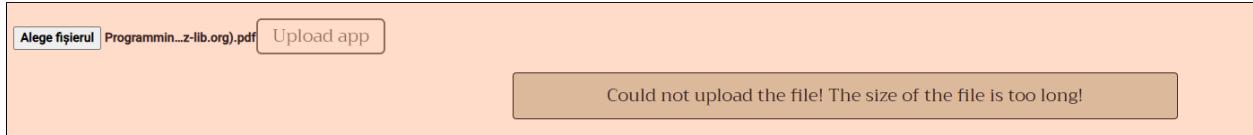


Figura 4.11. Eroare încărcare - dimensiune fișier prea mare

În urma încercării ștergerii unui utilizator ce are rolul de administrator de către un alt utilizator administrator va apărea un mesaj de eroare cum că nu este posibil acest lucru. Această eroare poate fi văzută în Figura 4.12.

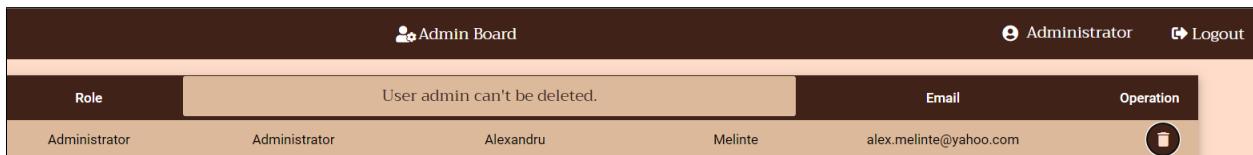


Figura 4.12. Eroare ștergere utilizator de tip administrator

4.3. Aspecte legate de fiabilitate/securitate

Securitatea aplicației este asigurată prin:

- **AuthGuard** - cu ajutorul acestei clase din *Angular* s-a reușit a se restricționa accesul utilizatorilor care nu sunt autentificați. Așadar, orice utilizator care nu s-a autentificat și încearcă să acceseze resurse prin scrierea manuală a URL-ului va fi redirectionat către pagina de autentificare.
- **JWT** - cu ajutorul acestuia s-a reușit a se autoriza accesul în platformă. Cu alte cuvinte, la autentificare, oricărui utilizator îi este asignat un nou JWT, prin intermediul căruia serverul de back-end recunoaște privilegiile acestuia în contextul accesării anumitor resurse. Aceasta este salvat în sesiune pentru a putea fi accesat în partea de front-end. În momentul expirării JWT-ului (setat la 5 minute), utilizatorul este automat deconectat de la platformă și necesită a se autentifica din nou pentru a o putea accesa.
- Utilizatorul poate accesa doar resursa căreia îi corespunde rolului. Adică, dacă un utilizator student scrie manual URL-ul de access pentru pagina de profesor, acesta va fi redirectionat automat către pagina de acasă. Implementarea a fost făcută în constructorul componentei principale al aplicației.
- **Spring Security** - autentificarea și autorizarea au fost personalizate și implementate pe partea de back-end cu ajutorul acestui cadru de dezvoltare.
- Conținutul chestionarului de evaluare este încărcat în baza de date sub forma *Base64*, iar în momentul rulării chestionarului de evaluare de către un utilizator student, acesta este extras din baza de date și decodat (Figura 4.13).
- Parola utilizatorului este stocată într-o formă criptată în baza de date (Figura 4.14).

content
eyJxdWVzdGlvbnMiOlt7InF1ZXN0aW9uIjoiIsImFuc3dcmIiOlt7ImNvcnJY3QjOiyIn0seyJ3cm9uZyI6ijEyIn0seyJ3cm9uZzEiOiyIn0seyJ3cm9uZzIiOiOIn1dfSx7InF1ZXN0aW9uIjoiQyIsImFuc3dI... eyJxdWVzdGlvbnMiOlt7InF1ZXN0aW9uIjoiNSB4IDUiLCjhbnN3ZXJzIpbeyJjb3JyZWNOIjoiMjifSx7Indyb25nIjoiMzifSx7Indyb25nMSI6IjixICJ9LHsId3JvbmcyIjoiMTAiFv19LHsicVlc3Rp... eyJxdWVzdGlvbnMiOlt7InF1ZXN0aW9uIjoiQ2FyZSBlc3RlIGldG9kYSBIVFRQIHByaW4gaW50ZXJtZWRpdWwgY2FyZWlhIHNIiHNVbGljaXRhIG8gcmVzdXJzYSBkZSBsYSBzZXJ2ZXI/iwiYW5zd2Vycy...

Figura 4.13. Conținutul unui chestionar stocat în baza de date

	id	email	firstname	lastname	password	username
▶	1	alex.melinte@yahoo.com	Alexandru	Melinte	\$2a\$10\$yWT.2JQIkZzvXqZRso1v8ugPmca2zhuzjNhzRLlab9AFH7n8jxza	Administrator
	3	gigi.frone@yahoo.com	Gigel	Frone	\$2a\$10\$ffEsEM4rQIn.PguLcPCUmle3v4BgGoRNXl9bfQJWLZnrWSybKhgTQK	GigelFrone
	6	dorinel.munteanu@yahoo.com	Dorinel	Munteanu	\$2a\$10\$YnTwuyxACK0VeLj.AdCt5.gNE7n0f1zj5GnuEuixLZHY2ECSu2GaO	DorinelMunteanu
	8	ag.melinte@yahoo.com	Alexandru-Gicu	Melinte	\$2a\$10\$VzdpQrtMg3M4/YX754/StugNE0ufq5z4glF2/uEAx33/T/GTDa.	AlexMelinte
	11	student.model@yahoo.com	Student	Model	\$2a\$10\$nkPe2CLGpWv0HYQdpZv1FOVKTJQXSa6udPxTsxeIvYe09Th5zq	StudentModel
	12	john.doe@yahoo.com	John	Doe	\$2a\$10\$Iq2NFOHYAqn9DVW/yk8PMeaicJIP/.hEFa8n17mejnFqcTOtqsVO	JohnDoe
	13	ionel.dobre@yahoo.com	Ionel	Dobre	\$2a\$10\$y8n5/32B6gJ0n07ANo5DkuZg32HS6q3WehYQ/fcGo01H.d0QMqY	IonelDobre

Figura 4.14. Conținutul tabelei *users* din baza de date

Concluzii

Gradul în care s-a realizat tema propusă

Tema propusă a fost dezvoltarea unei platforme de evaluare a studenților în cadrul laboratoarelor. Obiectivele acesteia au fost împărțirea celor 14 săptămâni de facultate în 3 tipuri de evaluare, una în săptămâna a cincea, una în săptămâna a noua și în ultima săptămână, furnizarea unei note finale de laborator în urma parcurgerii acestor evaluări, vizualizarea unor statistici legate de notele studenților și concluzii legate de veridicitatea evaluării.

În Capitolul 2 s-a putut observa proiectarea aplicației, mai exact logica de funcționalitate și comunicarea dintre serverul de front-end și cel de back-end. Așadar, platforma a fost proiectată pentru trei tipuri de utilizatori (administrator, profesor și student), unde utilizatorul de tip administrator gestionează conturile utilizatorilor, cel de tip profesor încarcă laboratoare și cursuri și este capabil să creeze tipuri de evaluare sub formă de chestionar corespunzătoare săptămânilor decise la obiective, iar cel de tip student are posibilitatea să rezolve chestionarele de evaluare propuse de utilizatorul profesor, fiindu-i furnizată și nota finală după parcurserea tuturor evaluărilor.

În Capitolul 3 s-a descris ceea ce s-a menționat și structurat în Capitolul 2 legat de ceea ce poate să facă fiecare tip de utilizator punând în evidență funcționalitatea platformei și îndeplinirea obiectivelor.

Capitolul 4 susține funcționalitatea descrisă la Capitolul 3 prin capturi de ecran ce atestă faptul că obiectivele platformei au fost realizate și implementate cu succes.

Consider că lucrarea de față a fost realizată aproape integral conform obiectivelor propuse, singurul minus fiind cel legat de statistici. De asemenea, asta nu înseamnă ca aceasta nu putea fi făcută într-o manieră mai eficientă, mai prietenoasă cu utilizatorul sau dezvoltată cu mai multe funcționalități.

Evidențierea concisă a contribuțiilor/soluțiilor personale

Una dintre soluțiile originale din această platformă este cea legată de salvarea chestionarelor sub forma unor JSON-uri și urcarea acestora în baza de date, cu întregul conținut codificat, pentru a putea fi rulate ulterior, în cadrul evaluării, de către utilizatorul student.

Printre soluțiile personale din implementarea lucrării de față se mai enumeră și logica de salvare și de rulare a chestionarelor, furnizarea notelor și calcularea notei finale pentru fiecare student în parte, dar și aspectul interfeței cu utilizatorul.

Posibile direcții de dezvoltare

O posibilă dezvoltare a lucrării prezentate ar fi adăugarea mai multor materii utilizatorului de tip student, ca acesta să nu fie constrâns la accesul către o singură materie.

Platforma mai poate fi îmbunătățită și la nivelul pornirii acesteia, întrucât la momentul actual, serviciile din back-end împreună cu serverul de front-end și baza de date se pornesc manual.

O altă posibilă dezvoltare ar fi reprezentarea de posibilitatea profesorului de a alege numărul de întrebări corecte și gresite din fiecare întrebare în cadrul realizării chestionarului de evaluare. De asemenea, ar putea fi adăugate și posibilitatea studentului de răspunde în scris sau a alege dintr-un meniu răspunsul final la întrebarea din chestionar.

De asemenea, o altă extindere ar putea fi reprezentată de posibilitatea vizualizării întrebărilor la care s-a răspuns greșit de către utilizatorul student.

Bibliografie

- [1] R. W. Sebesta, *Programming the WORLD WIDE WEB*, 8th ed. Pearson, 2014.
- [2] I. C. Education, “Java spring boot,” <https://www.ibm.com/cloud/learn/java-spring-boot>, 2020, Ultima accesare: 07.05.2022.
- [3] Oracle, “Mysql 8.0 reference manual,” <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>, 2022, Ultima accesare: 01.06.2022.
- [4] S. S. Lars Vogel, “Java persistence api - tutorial,” <https://www.vogella.com/tutorials/JavaPersistenceAPI/article.html>, 2017, Ultima accesare: 01.06.2022.
- [5] Simuli, “Hypertext transfer protocol (http),” <https://docs.simuli.co/iot-protocols/hypertext-transfer-protocol-http>, 2021, Ultima accesare: 10.06.2022.
- [6] Mozilla, “Http | mdn,” <https://developer.mozilla.org/en-US/docs/Web/HTTP>, 2022, Ultima accesare: 20.06.2022.
- [7] M. Jones, “JSON Web Token (JWT),” RFC 7519, 2015. [Online]. Available: <https://datatracker.ietf.org/doc/html/rfc7519>
- [8] Auth0, “Introduction to json web tokens,” <https://jwt.io/introduction>, 2022, Ultima accesare: 22.06.2022.
- [9] E. Cerami, “Introduction to Web Services,” in *Web Services Essentials*. O'Reilly, 2002, ch. 1, pp. 6–7, <http://ommolketab.ir/aaf-lib/ooz6p9fd46m2w11y90n5jhuhp1s98f.pdf>.
- [10] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” Ph.D. dissertation, University of California, Irvine, 2000.
- [11] bezkoder, “Spring boot token based authentication with spring security, jwt,” <https://www.bezkoder.com/spring-boot-jwt-authentication/>, 2022, Ultima accesare: 06.07.2022.

