



**UNIVERSITATEA ALEXANDRU IOAN CUZA DIN IAȘI
FACULTATEA DE INFORMATICĂ**

LUCRARE DE LICENȚĂ

COORDONATOR:
CRISTIAN

ABSOLVENT: Lect. Dr. FRĂȘINARU
PÎRLOG MARCEL IONUȚ

IAȘI 2020

UNIVERSITATEA ALEXANDRU IOAN CUZA DIN IAȘI
FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

COORDONATOR:
CRISTIAN

ABSOLVENT: Lect. Dr. FRĂȘINARU
PÎRLOG MARCEL IONUȚ

IAȘI 2020

Cuprins

1	Introducere	4
1.1	Prezentarea aplicației	4
1.2	Motivația	4
1.3	Aplicații similare	5
1.4	O scurtă trecere în revistă a tehnologiilor folosite	5
2	Specificatii functionale	6
2.1	Aplicația desktop	6
2.2	Aplicația Web	7
2.3	Serverul	7
3	Arhitectura sistemului	8
3.1	Schema bazei de date	8
3.1.1	Rolul tabelelor din baza de date	9
3.2	C4 Model	13
3.2.1	Level 1	13
3.2.2	Level 2	14
3.2.3	Level 3	15
3.3	Diagrame Use-case	18
4	Implementare si testare	20
4.1	Tehnologii folosite	20
4.1.1	Java	20
4.1.2	Java Fx	22
4.1.3	Spring boot	23
4.1.4	Java server faces	24
4.1.5	PostgreSQL	25
4.2	Detalii tehnice	26
5	Manual de utilizare	39
5.1	Modul de utilizare	39
5.1.1	Aplicația desktop	39
5.1.2	Aplicația profesorului	44
6	Concluzii și direcții viitoare	50

Capitolul 1

Introducere

1.1 Prezentarea aplicației

Aplicația este un soft educațional menit să îmbunătățească comunicarea dintre studenți și profesor, oferindu-i profesorului un mai bun control asupra activităților studentului cât și o posibilitate de a vedea evoluția acestuia în fiecare laborator și proiect.

Proiectul își propune să rezolve lipsa de control asupra modului în care un student rezolvă laboratoarele de programare cât și când îl rezolvă. Utilizând această aplicație profesorul poate vedea câteva etape intermediare din evoluția codului nu doar varianta finală care este și notată. Existând un sistem de versionare unde se poate vedea cu ușurință modificările făcute de student, dacă a rezolvat treptat laboratorul sau a luat totul dintr-o sursă externă. Aplicația vine echipată cu un sistem de plagiere care verifică procentul de unicitate în rezolvarea problemei/problemelor cerute de profesor.

1.2 Motivația

Motivația a venit din imposibilitatea profesorilor de a urmări evoluția activităților de programare ale studenților în cadrul laboratoarelor. O altă motivație a fost faptul că profesorul putea vedea doar varianta finală nu și evoluția acesteia (de la ce a pornit și pașii prin care a ajuns la versiunea finală). Aplicația venind în ajutorul profesorilor prin funcționalitățile sale, oferindu-le o evoluție în timp a proiectelor/laboratoarelor studenților săi.

Utilizând aplicațiile moderne de versionare este destul de greu pentru a vedea fiecare proiect al fiecărui student. Acest lucru m-a motivat să grupez studenții în grupuri create de profesori unde au posibilitatea de a accesa foarte ușor fiecare proiect care îi este destinat. Acest lucru eliminând munca studenților de a trimite prin email/discord/etc... proiecte/teme și mai ales a profesorilor de a descărca stoca aceste fișiere. Trebuie pornită o aplicație și sunt necesare câteva secunde după care studentul poate lucra liniștit. La final apăsând un buton pentru a "notifica" finalizarea proiectului/temei.

1.3 Aplicații similare

O primă aplicație similară este **Fiieval** folosită pentru evaluarea activității de laborator la **Programare in Python**, unde profesorul poate vedea nu doar ultima varianta a testului si variante intermediare.

Sisteme de versionare populare la momentul actual (GitHub, Gitlab) acestea având o idee de versionare bună în care se poate vedea cât de cât progresul studentului dar necesită puțin efort din partea studentului.

1.4 O scurtă trecere în revistă a tehnologiilor folosite

În această secțiune vor fi prezentate pe scurt tehnologiile folosite. Se va insista pe descrierea si motivarea folosirii acestora în capitolul ”Implementare și testare”. Aici doar fiind menționate și locul unde vor fi folosite.

Pentru implemetarea acestui proiect au fost folosite urmatoarele tehnologii:

- Pentru componenta Desktop am folosit:
 - **Java 11** pentru colectarea datelor din front-end stocarea acestora si interacțiunea cu serverul web pentru a beneficia de funcționalitățile acestuia
 - **Java Fx** pentru realizarea interfeței grafice folosită de catre student
- Pentru componenta Web:
 - **Java Server Face (JSF)** pentru interfata web folosită de profesor
- Pentru server:
 - **Spring Boot** pentru crearea serverului și serviciilor acestuia

Capitolul 2

Specificatii functionale

Proiectul este compus din 3 mari componente:

- O aplicație desktop destinată studentului
- O aplicație web destinată profesorului
- Un server format din 4 submodule:
 - version management
 - plagiator
 - compiling
 - user management

În cele ce urmează vor fi prezentate funcționalitățile oferite de fiecare componentă și modul în parte.

2.1 Aplicația desktop

Studentul se va conecta în aplicație cu un cont creat implicit de către administrator, fiind de forma conturilor create de Facultatea de Informatică (prenume.nume) având o parolă generată aleator.

După conectare studentul va putea să creeze un nou "proiect" reprezentând un laborator la o materie de programare sau la o activitate la care face parte la care dorește să lucreze. Dacă este vorba de un laborator va trebui să selecteze materia respectivă, dacă este o activitate coordonată de un profesor aplicația va introduce automat datele necesare. Pe lângă acestea el mai trebuie să introducă și o cale către directorul în care se vor afla fișierele sursă care vor fi evaluate. Imediat ce va introduce o cale către un director valid și apasă pe butonul de creează programul va încărca pe server o versiune inițială a proiectului. În același timp îi vor fi afișate și studentului fișierele care au fost încărcate inițial.

Pe măsura ce studentul scrie cod aplicația va colecta la anumite intervale de timp sau când vor fi suficiente modificări noua versiune și o va încărca pe server. Când studentul decide că are o formă finală a proiectului sau forma la care dorește să se oprească va apăsa pe butonul de "finish". La apăsarea acestui buton aplicația desktop va trimite pe server ultima variantă a fișierelor specificând că este varianta care va fi compilată și pe care va fi testat plagiatul, iar studentul va fi redirectionat către ecranul inițial.

2.2 Aplicatia Web

Profesorul se va loga si va vedea toti studentii carora le preda si ce materie le preda. Va avea acces la un buton de testare a plagiatului. Aceasta testare a plagiatului se va efectua pentru toti studentii care au respectiva materie nu doar pentru studentii grupei/grupelor sale. Daca un alt profesor activeaza testarea plagiatului un alt profesor nu o poate face pana la finalizarea procesului respectiv.

Profesorul va avea o pagina cu statistici pentru fiecare student in care poate vedea laboratoarele la care a lucrat cat si procentul in care acestea sunt plagiate. Profesorul va avea dreptul sa stearga sursele trimise de studenti, dar actiunile acestuia vor ramane salvate.

Pe langa acestea profesorul poate vedea daca varianta finala a studentului compileaza si versiunile proiectului (evolutia acestuia in timp) cat si data, ora la care au fost incarcate pe server.

Acesta poate crea si grupe de studenti pentru diferite activitati/proiecte acestea for fi tratate diferit fata de grupele din cadrul facultatii.

2.3 Serverul

Serverul va fi compus din 4 servicii web (user management, subversion, compilare, testarea plagiatului) fiecare fiind specializat pe o sarcina bine definita.

- User management: Acest serviciu se va ocupa de actorul User din aplicatie (login, oferire de informatii, etc..). Serviciul va fi folosit in principal in momentul loginului si cand va fi nevoie de informatii legate de utilizatori.
- Subversion: Acest serviciu va colecta fisiere de cod de la studenti si le va ordona intr-o baza de date in functie de timpul aparitiei pe server creind astfel o imagine de ansamblu asupra evolutiei in timp a proiectului. Serviciul acesta se va ocupa strict de versionarea fisierelor si de salvarea acestora pentru a putea fi incarcate ulterior in interfata profesorului. Acest serviciu se mai ocupa si de salvarea variantei finale ale proiectului si de notificare modulului de compilare.
- Compiler: Acest serviciu compilează o versiune la cererea profesorului indiferent dacă este finală sau nu. Serviciul nu expune funcționalități studenților fiind strict pentru profesor. La fiecare cerere este verificată tipul utilizatorului astfel oferă servicii doar profesorilor. Când termină sarcina pe care o primește actualizează coloana "compiling" din tabela "Versiuni".
- Plagiator: Va fi activat la cerere si va testa plagiatul tuturor versiunilor finale din baza de date pentru o anumita materie. Materia va fi stabilita de profesorul care o preda. Doar profesorul va avea acces la acest modul verificandu-se identitatea celui care face cerere de testare plagiatului.

Capitolul 3

Arhitectura sistemului

3.1 Schema bazei de date

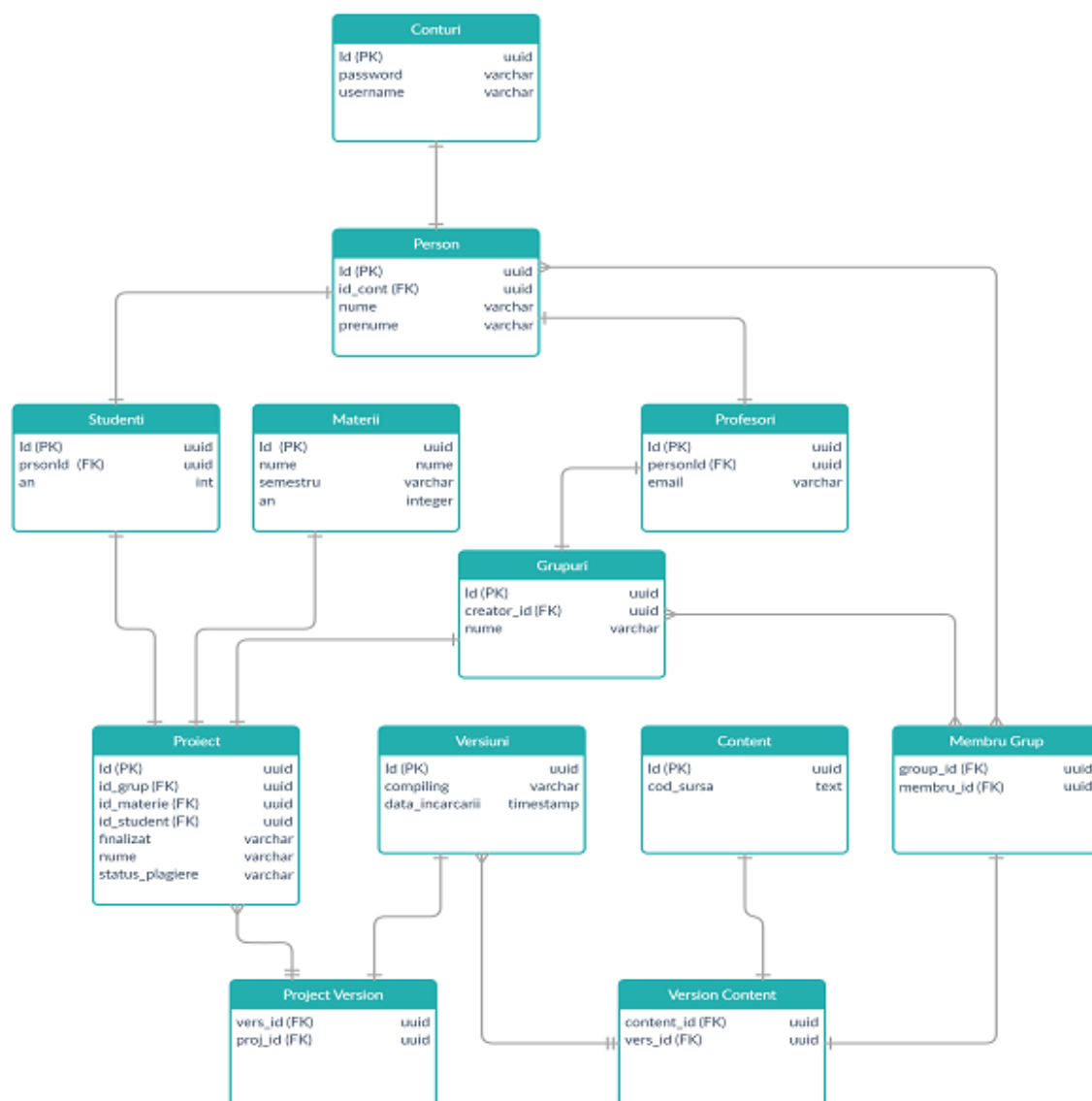


Figura 3.1: Schema bazei de date a aplicatiei

3.1.1 Rolul tabelelor din baza de date

Tabela Conturi

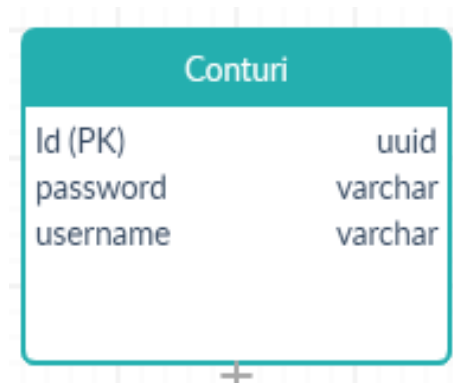


Figura 3.2: Tabela Conturi

Această tabelă stchează conturile persoanelor care au dreptul sa folosească aplicația. Usernamul are forma prenume.num. Parola este memorată sub forma unu hash sha256 pentru a proteja identitatea și datele utilizatorilor.

Relatii cu alte tabele:

- OneToOne cu tabela Person

Tabela Person

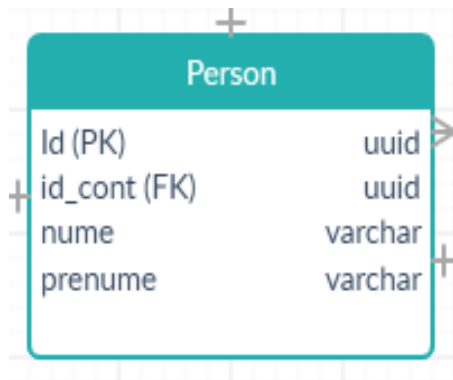


Figura 3.3: Tabela Person

Aici vor fi stocate informațiile persoanelor Nume, Prenume și id-ul contului.

Relatii cu alte tabele:

- OneToOne cu tabela onturi
- OneToOne cu tabla Studenti
- OneToOne cu tabela Profesori
- ManyToMany cu tabela Membri Grup

Tabelele Studenti si Profesori

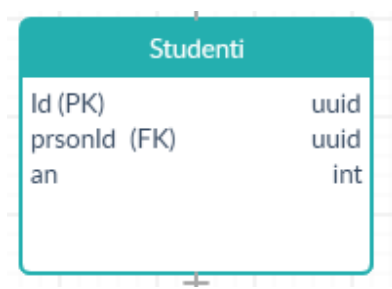


Figura 3.4: Tabela Studenti



Figura 3.5: Tabela Profesori

Aceste 2 tabele sunt folosite pentru a diferenția cele două tipuri de persoane care au acces în aplicație. Este nevoie de această diferențiere deoarece profesorul și studentul nu au aceleași funcționalități și drepturi în aplicație.

Tabela Materii

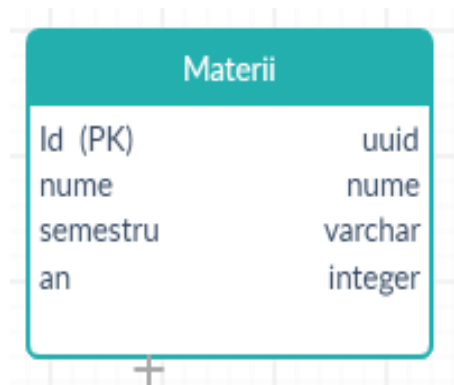


Figura 3.6: Tabela Materii

Tabela Grupuri

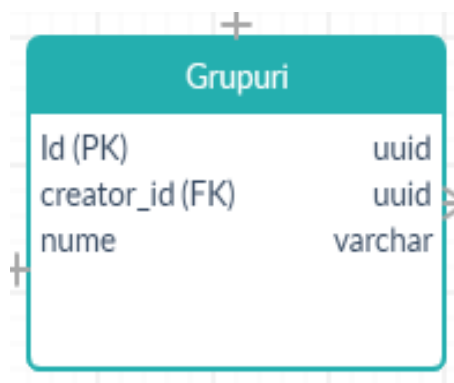


Figura 3.7: Tabela Grupuri

Aici sunt memorate grupurile create de către profesorii facultății. Aceste grupuri pot reprezenta grupele din cadrul facultății (A1, B2, B4...), dar și grupuri pentru diferite activități din cadrul facultății (optionale, programare competitivă, etc...)

Tabela Membri Grup

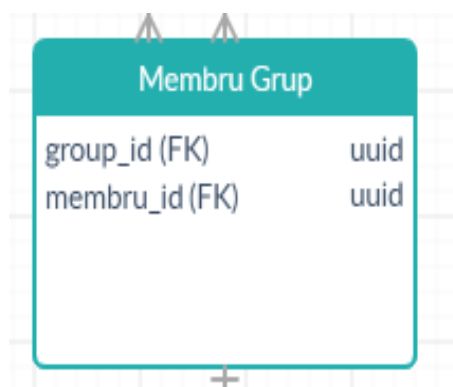


Figura 3.8: Tabela Membri Grup

În această tabelă stocăm membrii fiecărui grup (fiecare student căror grupuri aparține)

Tabela Proiect



Figura 3.9: Tabela Proiect



Figura 3.10: Tabela ProjectVersion

În tabela "Proiect" stocăm toate informațiile necesare despre un proiect:

- Studentul care a creat proiectul
- Pentru ce grup a creat acest proiect, este necesar pentru a filtra proiectele după anumite criterii de grupă/materie/profesor. Acest lucru este necesar în timpul testării antiplagiat.
- Plagiere, rezultat care este oferit de plagiator

- Faptul dacă este terminat sau nu

Tabela "Project Content" asigură îndeplinirea relației OneToMany dintre tablele "Proiect" și "Versiuni".

Tabela Versiuni

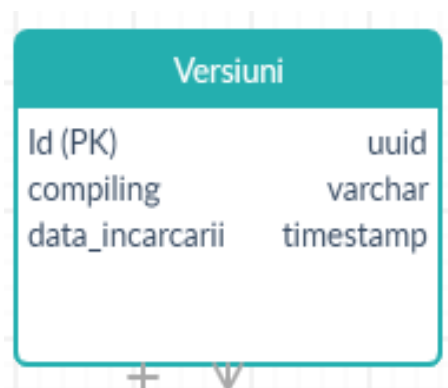


Figura 3.11: Tabela Versiuni

În această tabelă stocăm toate versiunile unui proiect, pentru fiecare versiune stocăm aici informațiile comune tuturor (rezultatul compilării și data când a fost încărcată).

Id-ul acestei entități este folosit mai jos pentru a stoca și conținutul fișierelor cod.

Tabela Content



Figura 3.12: Tabela Content

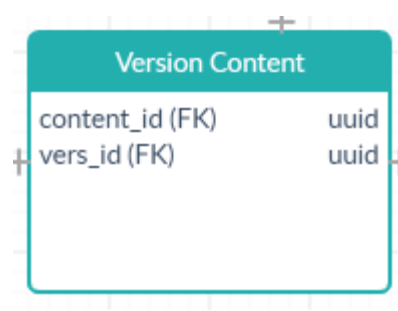


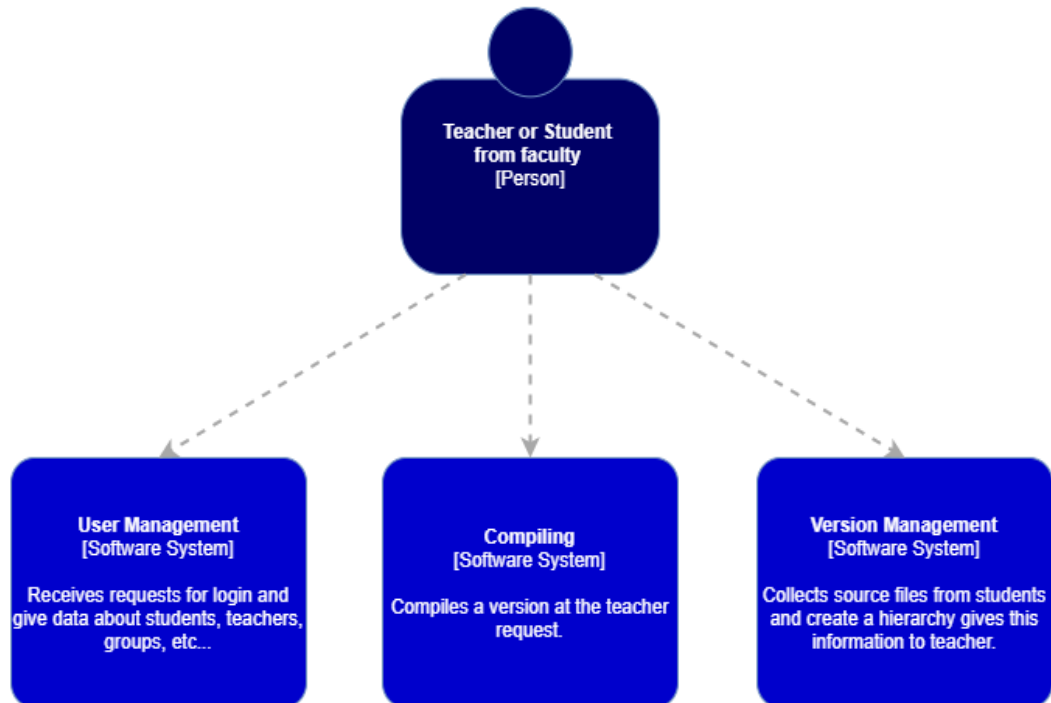
Figura 3.13: Tabela VersionContent

În tabela "Content" stocăm conținutul fiecărui fișier sursă encodat in baza64 cât și numele fișierului sursă.

Tabela "Version Content" este folosită pentru a satisface relația OneToMany dintre tablele "Versiuni" și "Content".

3.2 C4 Model

3.2.1 Level 1



System Context diagram for Application

Figura 3.14: Structura generală a aplicației

La nivelul 1 din C4 putem vedea principalele mari sisteme din aplicație. La acest nivel nu vom detalia componența sistemelor.

3.2.2 Level 2

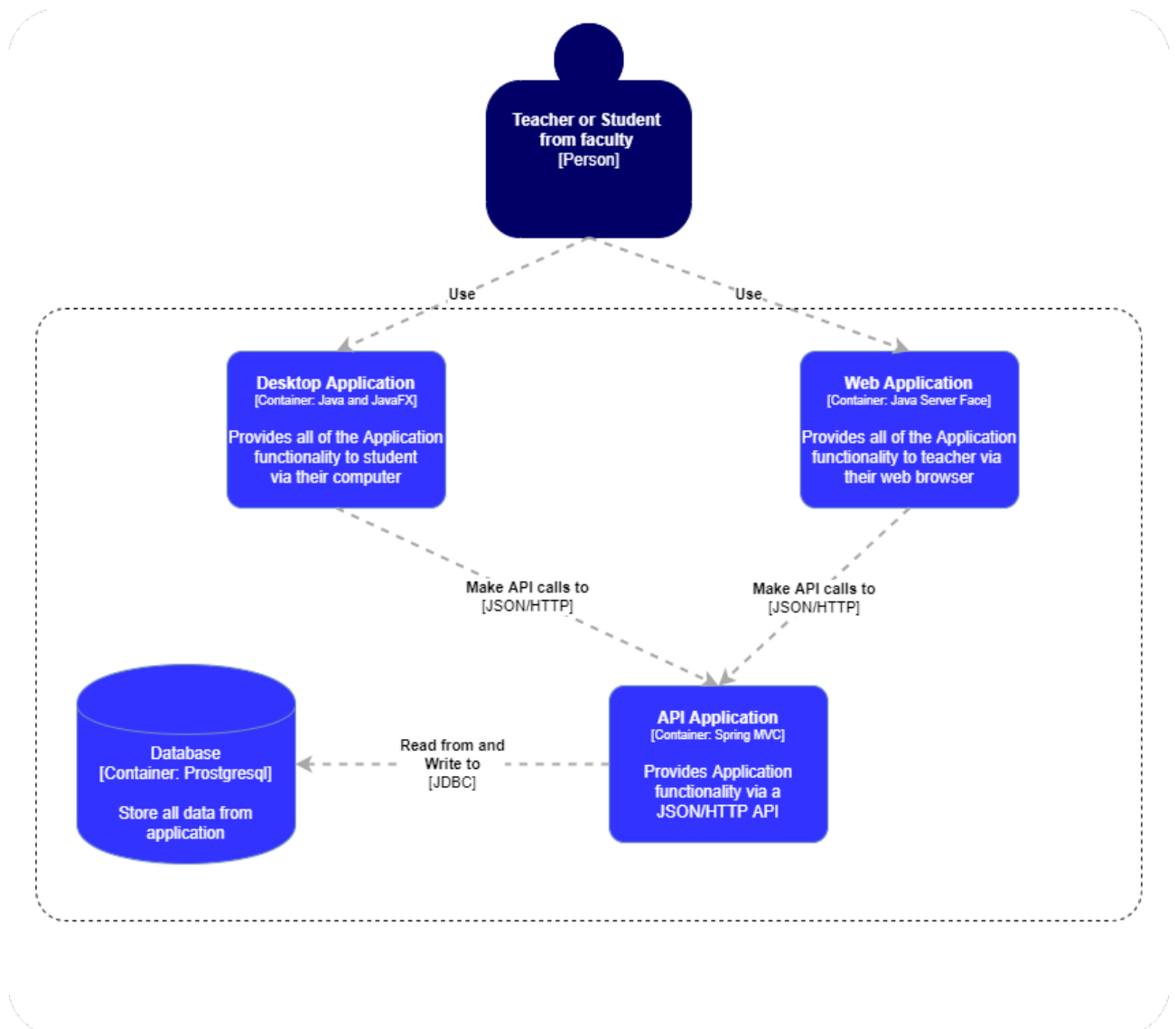


Figura 3.15: Structura generală a unui serviciu

La acest nivel putem vedea din ce este compus proiectul.

- Desktop Application: Oferă acces la un număr restrâns de funcționalități ale serverului prin intermediul unor requesturi GET si POST. Această aplicație nu poate solicita informații detaliate despre anumite entități (grupuri, membri...).
- Single-Page Application: Oferă mai multe funcționalități profesorului. Acesta putând să folosească toate serviciile oferite de server prin intermediu protocolului HTTP informațiile fiind afisate în browserul web al acestuia.
- API Application: Oferă endpointuri pentru funcționalitățile serverului, această componentă este detaliată la nivelul 3

3.2.3 Level 3

Level 3 - User Management

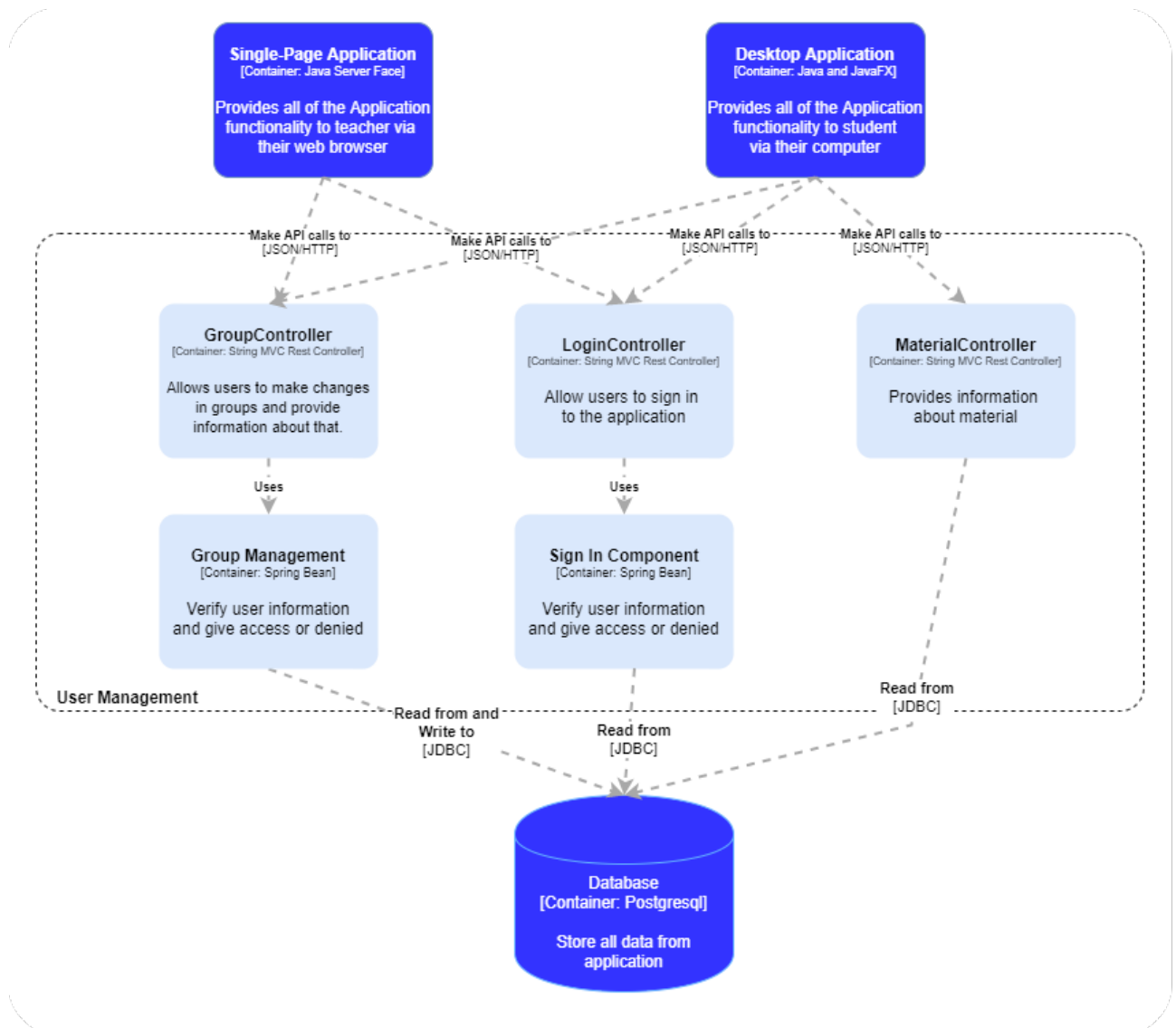


Figura 3.16: Structura sisemului de user management

În această diagramă sunt prezentate componentele care alcătuiesc API-ul Sistemului de user-management.

- Componentele **GroupController** și **Group Management** se ocupă de grupuri și membrii acestuia.
- **LoginController** și **Sign In Component** au sarcina de a aproba autentificare utilizatorilor pe baza unui cont
- **MaterialController** oferă informații despre materii

Level 3 - Version Management

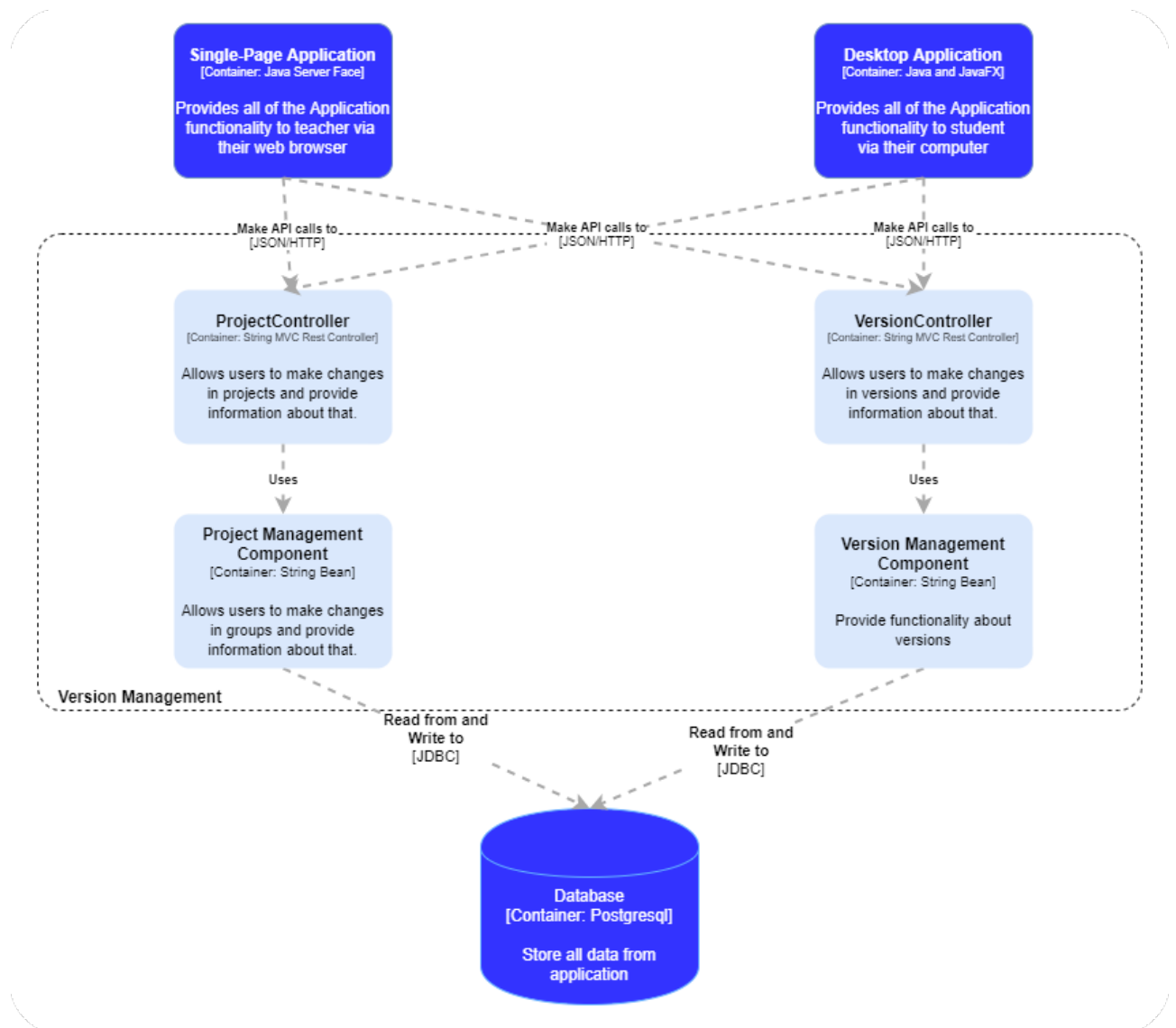


Figura 3.17: Structura sistemului de versionare

În această diagramă sunt prezentate componentele care alcătuiesc API-ul Sistemului de version-management.

- Componentele **ProjectController** și **Project Management Component** au sarcina de a crea și de a oferi informații despre proiecte în funcție de atributul dorit
- Componentele **VersionController** și **Version Management Component** au sarcina de a salva în baza de date versiunile primite de la studenți și să ofere informații despre acestea profesorului

Level 3 - Compiler

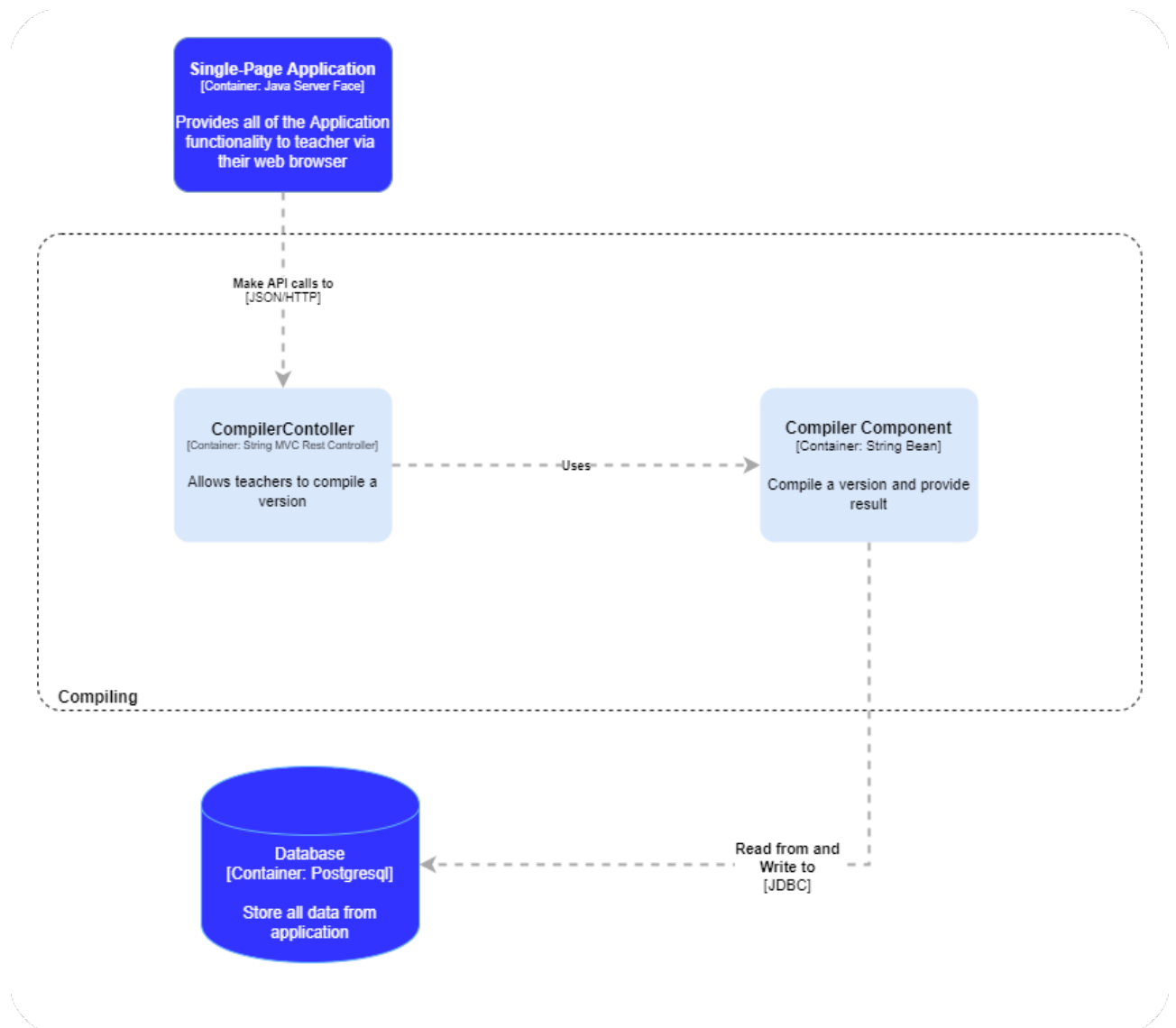


Figura 3.18: Structura sistemului de compilare

În această diagramă sunt prezentate componentele care alcătuiesc API-ul Sistemului de compilare.

- Componenta **CompilerController** are sarcina de a primi requesturile către acest serviciu și de a verifica identitatea persoanei care face această solicitare
- Componenta **"Compiler Component"** are sarcina de a compila fișierele și de a actualiza statusul de compilare pentru acea versiune.

3.3 Diagrame Use-case

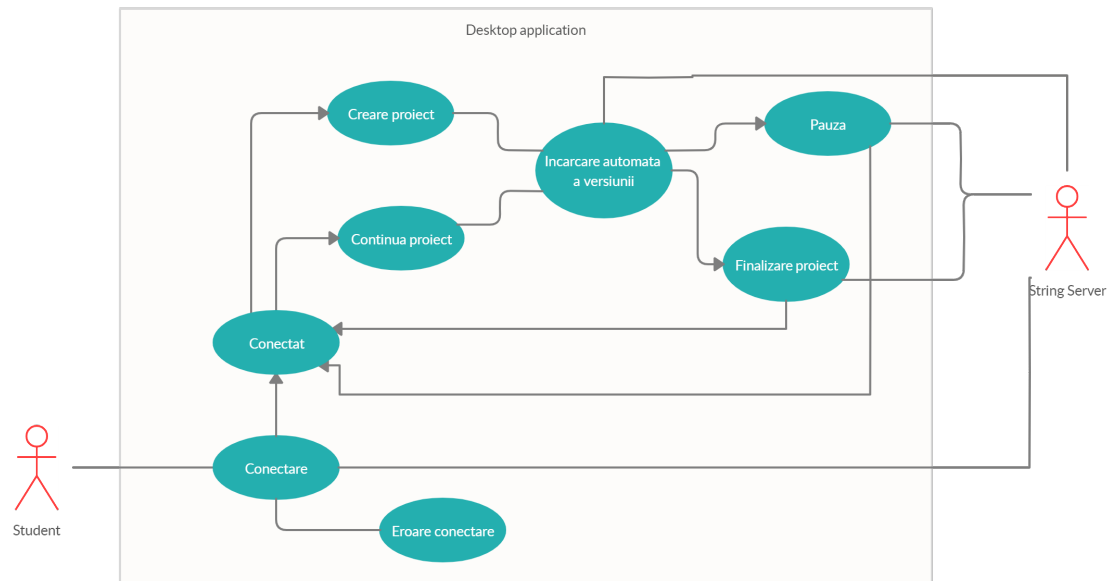


Figura 3.19: Desktop application Use case diagram

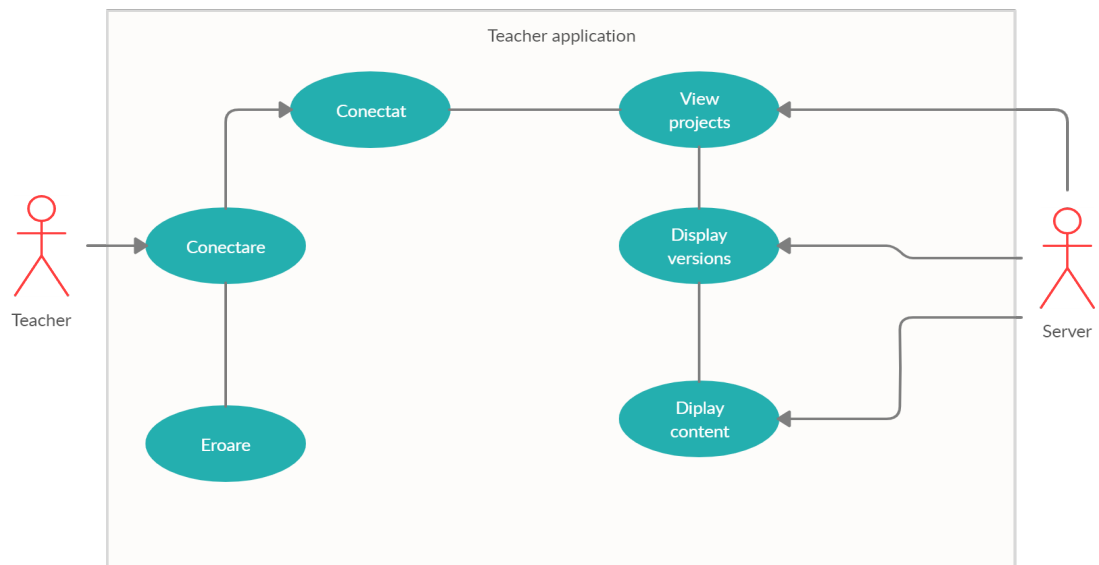


Figura 3.20: Teacher application part 1

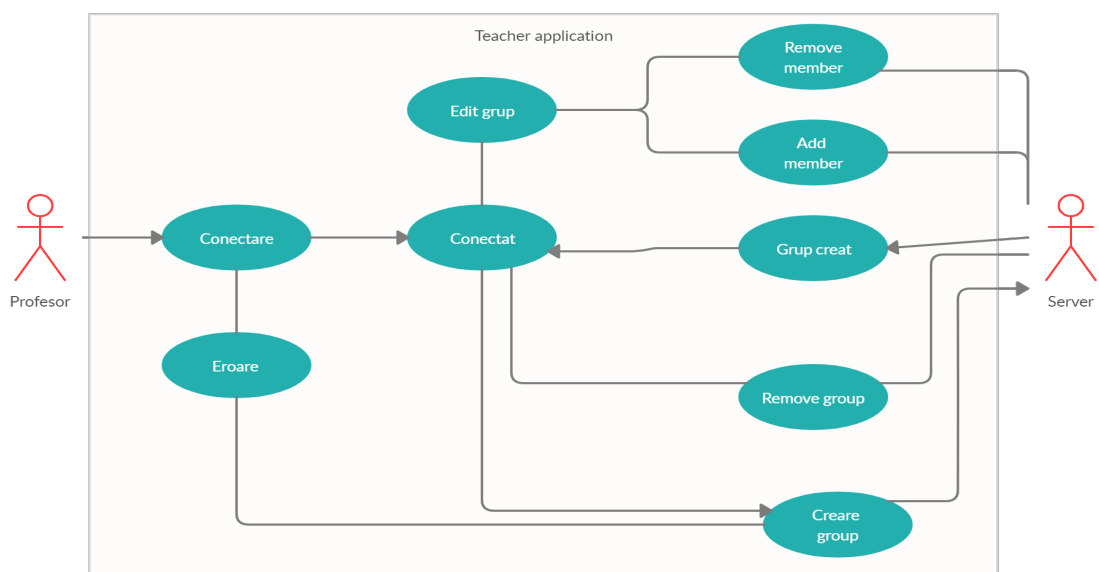


Figura 3.21: Teacher application part 2

Capitolul 4

Implementare si testare

4.1 Tehnologii folosite

4.1.1 Java

Java este un limbaj de programare orientat-obiect de nivel înalt dezvoltat de Sun Microsystems (acum deținută de Oracle) la începutul anilor '90, fiind lansat oficial în anul 1995 o versiune "JDK Beta" urmând ca în ianuarie 1996 să fie lansat "JDK 1.0". De la versiunea "Java SE 8 (LTS)" Oracle a început un trend de a publica câte o nouă versiune de java la fiecare 6 luni. În prezent suntem la versiunea Java SE 14 urmând ca în septembrie să apară "Java SE 15".

A fost inițial conceput pentru dezvoltarea de programe pentru set-top box-uri și dispozitive portabile, dar ulterior a devenit o alegere populară pentru crearea de aplicații web. Oracle a achiziționat Sun Microsystems în ianuarie 2010. Prin urmare, Java este acum întreținut și distribuit de Oracle. Există o mulțime de aplicații și site-uri web care nu vor funcționa decât dacă aveți Java instalat și mai multe sunt create în fiecare zi. Java poate fi întâlnit pe laptopuri, centrele de date, console de jocuri la supercomputere științifice, telefoane mobile.

Sintaxa Java este similară cu cea din C++, dar este strict orientat pe obiect. Java este cunoscut și pentru faptul că este mai stric decât C++, ceea ce înseamnă că variabilele și funcțiile trebuie definite în mod explicit.

Programele java sunt interpretate de Java Virtual Machine (JVM) care rulează pe mai multe platforme. Asta înseamnă că toate programele Java sunt multiplatform și pot fi rulate pe diferite platforme, inclusiv computere Macintosh, Windows și Unix.

Principalele extensii de fișiere folosite de Java sunt:

- .java
- .class
- .jar

În Java, tot codul sursă este scris inițial în fișiere plain text a căror denumire se încheie cu extensia `".java"`. Apoi aceste fișiere sunt compilate în cod binar cu ajutorul compilatorului Java, iar codul de byte este executat cu ajutorul Java Virtual Machine (JVM). Compilatorul și JVM-ul fac parte din kitul de dezvoltare Java. Codul binar este salvat în fișiere cu extensia `".class"` (cum poate fi observat și în următoarea schemă).

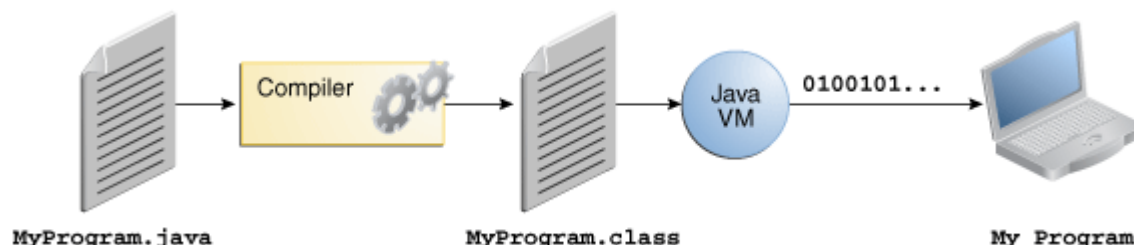


Figura 4.1: Procesul de compilare în Java

Pe lângă versiunile care sunt lansate la anumite intervale de timp, Java este oferită și în două ediții:

- Java Standard Edition (JSE) care conține API-urile de bază Java pentru aplicații desktop independente și pentru linia de comandă. Există atât un JRE, cât și un JDK pentru Java Standard Edition
- Java Enterprise Edition (JEE) conține o mulțime de instrumente și API-uri suplimentare pentru executarea componentelor Java în cadrul unui server Java Enterprise: Servlet, Java Server Pages (JSP), Java Server Faces (JSF), Enterprise Java Beans (EJB)
- Java Platform, Micro Edition (JAVA ME) este o platformă pentru dezvoltarea și rularea codului portabil pentru dispozitive încorporate și mobile (micro-controlere, senzori, gateway-uri, telefoane mobile, asistenți digitali personali)
- JavaFX despre aceasta vom discuta puțin mai jos.

Pentru a scrie cod în acest limbaj de programare se recomandă folosirea unui IDE profesionist printre cele mai populare fiind: Eclipse, IntelliJ, NetBeans, JDeveloper, MyEclipse, BlueJ... lista poate continua cu multe alte exemple, fiecare având funcționalități și scop diferit.

4.1.2 Java Fx

JavaFX este un set de pachete grafice și media care le permite dezvoltatorilor să proiecteze, să creeze, să testeze și să implementeze aplicații client complexe care operează constant pe diverse platforme. Aplicațiile JavaFX pot utiliza bibliotecile API Java pentru a accesa funcțiile sistemului nativ și pentru a se conecta la server-based middleware applications.

Următoarea imagine prezintă arhitectura API JavaFX. Aici puteți vedea componentele care acceptă API-ul JavaFX:

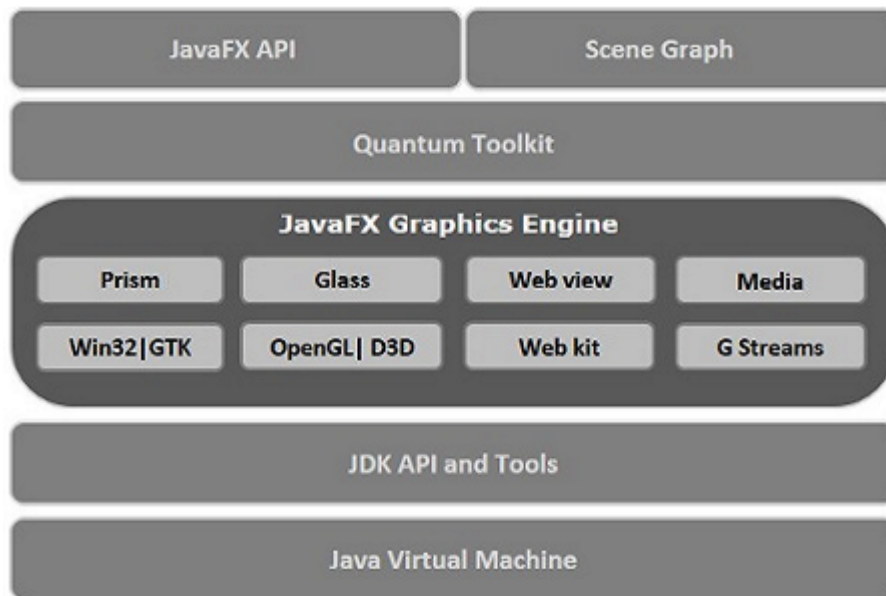


Figura 4.2: Arhitectura JavaFX API

JavaFX vine cu următoarele funcționalități:

- Un set mare de componente GUI încorporate, cum ar fi butoane, câmpuri de text, tabele, arbori, meniuri, diagrame și multe altele. Acest lucru vă economisește mult timp atunci când construim o aplicație desktop.
- Componentele pot fi proiectate folosind CSS și putem utiliza FXML pentru a crea un GUI în loc să o scriem în cod Java. Acest lucru face mai ușor și mai rapid crearea unui GUI.
- Vine cu suport pentru grafică 2D și 3D, precum și asistență audio și video. Acest lucru este util dacă dorim să dezvoltăm un joc sau o aplicație media.
- Conține un WebView bazat pe popularul browser WebKit, astfel încât să putem încorpora pagini web sau aplicații web în JavaFX.

JavaFX poate fi rulat pe laptopuri, dispozitive mobile și Raspberry Pi. Tot odată prezintă și un Game Engine creat de către Almans Bain numit "FXGL".

4.1.3 Spring boot

Spring Boot este framework bazat pe Java, folosit pentru a crea un microserviciu. Este dezvoltat de Pivotal Team și este utilizat pentru a construi aplicații de Spring de sine stătătoare și pentru producție.

Ce este un microserviciu ?

Un microserviciu este o arhitectură care permite dezvoltatorilor să creeze și să desfășoare servicii independente. Fiecare serviciu care rulează are propriul proces, iar acest lucru reprezintă un model ușor de suport pentru aplicațiile de afaceri.

Schema de mai jos arată flow-ul clasic al unui serviciu scris folosind Spring-boot, bineînțeles că acesta poate fi modificat în funcție de aplicație și nevoi:

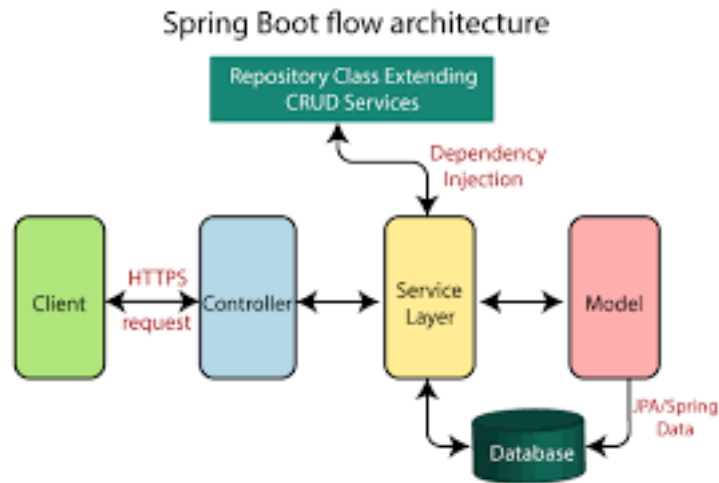


Figura 4.3: Spntring-boot flow

Iată doar câteva dintre caracteristicile Spring Boot:

- Dependințe de început optimizate pentru a simplifica construirea și configurarea aplicației
- Server încorporat pentru a evita complexitatea implementării aplicațiilor
- Configurare automată pentru funcționalitatea Spring ori de câte ori este posibil.

Pentru a crea o aplicație spring su ușurință putem folosi "spring-boot initializer". Acesta poate fi găsit într-un IDE sau la linkul următor: <https://start.spring.io/>. Această unealtă ne va crea ierarhia de fișiere minimă împreună cu dependențele maven necesare.

Spring Boot configurează automat aplicația pe baza dependențelor pe care le-am adăugat la proiect folosind adnotarea **@EnableAutoConfiguration**. De exemplu, dacă baza de date MySQL se află pe calea de clasă, dar nu am configurat nicio conexiune la baza de date, atunci Spring Boot configurează automat o bază de date în memorie.

Punctul de intrare al aplicației de pornire cu arc este clasa care conține adnotarea **@SpringBootApplication** și metoda principală. Spring Boot scanează automat toate componentele incluse în proiect folosind adnotarea **@ComponentScan**.

4.1.4 Java server faces

Java Server Faces (JSF) este un cadru de aplicații web bazat pe Java destinat simplificării integrării de dezvoltare a interfețelor de utilizator bazate pe web. Java-Server Faces este o tehnologie de afișare standardizată, care a fost oficializată într-o specificație prin Procesul Comunității Java. JSF 2 folosește Facelets ca sistem implicit de șabloane. De asemenea, pot fi utilizate alte tehnologii de vizualizare, cum ar fi XUL sau Java simplu. În schimb, JSF 1.x folosește JavaServer Pages (JSP) ca sistem de modelare implicit.

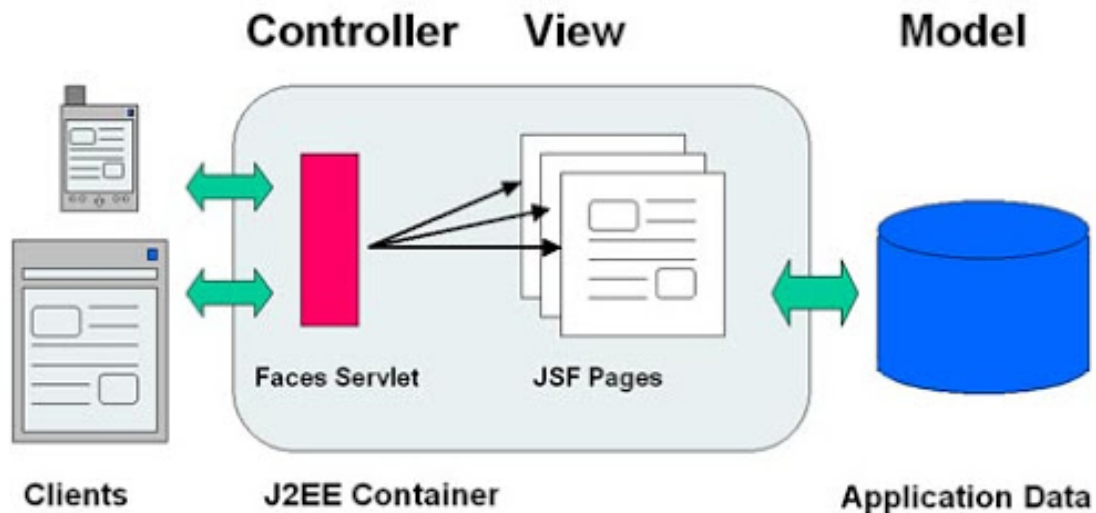


Figura 4.4: Java Server Faces

Proiectată pentru a fi flexibilă, tehnologia JavaServer Faces încurajează concep-tele existente, standard UI și web-tier, fără a limita dezvoltatorii la un anumit limbaj de marcaj, protocol sau dispozitiv client. Clasele de componente UI incluse în tehnologia JavaServer Faces încapsulează funcționalitatea componentelor, nu prezentarea specifică clientului, permițând astfel componentele UI JavaServer Faces să fie redade pe diverse dispozitive client. Ca comoditate, tehnologia JavaServer Faces oferă un randator personalizat și o bibliotecă de etichete JSP personalizate pentru redarea către un client HTML, permițând dezvoltatorilor aplicațiilor Java Platform, Enterprise Edition (Java EE) să utilizeze tehnologia JavaServer Faces în aplicațiile lor.

JSF reduce efortul de creare și întreținere a aplicațiilor, care va rula pe un server de aplicații Java și va reda UI-ul aplicației către un client țintă. JSF facilitează dezvoltarea aplicațiilor web prin:

- Furnizarea de componente UI reutilizabile
- Eficientizarea transferului de date între componentele UI
- Gestionarea stării UI pentru mai multe solicitări de server
- Permite implementarea componentelor personalizate

4.1.5 PostgreSQL

PostgreSQL este un sistem de baze de date relațional de obiecte, open source, care utilizează și extinde limbajul SQL combinat cu multe caracteristici care stochează și scalează în siguranță cele mai complicate sarcini de date. Originile PostgreSQL datează din 1986 ca parte a proiectului POSTGRES de la Universitatea California din Berkeley și are mai mult de 30 de ani de dezvoltare activă pe platforma de bază.

PostgreSQL vine cu multe funcții menite să ajute dezvoltatorii să creeze aplicații, administratori pentru a proteja integritatea datelor și pentru a construi medii tolerante la erori și pentru a ne ajuta să gestionăm datele oricât de mare sau mic este setul de date. Pe lângă faptul că este open source, PostgreSQL este extrem de extensibil. De exemplu, puteți să definim propriile tipuri de date, să creăm funcții personalizate, chiar să scriem cod din diferite limbaje de programare fără să vă recompilăm baza de date!

Caracteristici găsite în PostgreSQL:

- Tipuri de date:
 - Primitive: Integer, Numeric, String, Boolean
 - Structurate: Date/Time, Array, Range, UUID
 - Document: JSON/JSONB, XML, Key-value (Hstore)
 - Geometrice: Point, Line, Circle, Polygon
 - Personalizate: Composite, Custom Types
- Integritatea datelor, asigurată prin:
 - Constrângeri
 - Primary keys
 - Foreign keys
 - Explicit Locks
 - Advisory Locks
- Securitatea:
 - Autentificare: GSSAPI, SSPI, LDAP, SCRAM-SHA-256
 - Sistem de control acces rapid
 - Securitate la nivel de coloană și rând
 - Autentificare multi-factor cu certificate și o metodă suplimentară
- Extensibilitate:
 - Funcții și proceduri stocate
 - Limbi de procedură: PL / PGSQL, Perl, Python
 - Expresii de cale SQL / JSON
 - Interfață de stocare personalizabilă pentru tabele
 - Conectare la alte baze de date sau fluxuri cu o interfață SQL standard

4.2 Detalii tehnice

Dicțiua despre detaliile tehnice o voi începe cu aplicația desktop continuând cu serviciile scrise în spring, iar la final voi vorbi despre aplicația web.

Pentru aplicația desktop am folosit java 11 împreună cu javaFX 11. Pentru a avea pachetele de java fx 11 am folosit următoarele dependențe maven:

```
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-controls</artifactId>
  <version>11.0.2</version>
</dependency>
```

Figura 4.5: javafx-controls

```
<dependency>
  <groupId>org.openjfx</groupId>
  <artifactId>javafx-fxml</artifactId>
  <version>11.0.2</version>
</dependency>
```

Figura 4.6: javafx-fxml

”javafx-controls” oferă elementele grafice din javaFX (butoane, label-uri, tabele, etc) acest pachet le oferă, dar este puțin dificil de lucrat cu acestea de aceea intevine cea-laltă depentență.

”javafx-fxml” acest pachet oferă un set de clase care ne permite sa creem elemente grafice in format FXML, acesta este un format XML inbogățit care este interpretat de acest pachet și transformat în elemente grafice din javaFX. Astfel noi nu trebuie sa mai scriem cod java pentru a creat inferfața, scriem în acest FXML.

Codul pentru interfața de logare este cel din imaginea de mai jos:

```
-
7  <AnchorPane prefHeight="400.0" prefWidth="400.0" xmlns="http://javafx.com/javafx/10
8    <children>
9      <AnchorPane prefHeight="400.0" prefWidth="400.0" style="-fx-background-color:
10        <children>
11          <Text fill="#f77604" fontSmoothingType="LCD" layoutX="125.0" layoutY="8
12            <font>
13              <Font name="Ink Free" size="39.0" />
14            </font>
15          </Text>
16          <Button fx:id="loginButton" layoutX="75.0" layoutY="321.0" mnemonicPars
17          <TextField fx:id="usernameField" layoutX="63.0" layoutY="175.0" prefHei
18          <PasswordField fx:id="passwordField" layoutX="62.0" layoutY="241.0" pre
19          <Label fx:id="errorLabel" alignment="CENTER" layoutX="25.0" layoutY="11
20            <font>
21              <Font name="System Bold" size="20.0" />
22            </font>
23          </Label>
```

Figura 4.7: Pagina de login în FMXL

Acest od este salvat intr-un fișiere cu extensi .fxml. Pentru al afișa utilizez clasa ”javafx.fxml.FXMLLoader”. Aceasta interpretează codul din fișier și crează obiecc-tele necesare in memorie. Pentru a putea vizualiza interfața aceasta trebuie pusă intr-o scenă ulterior fiind desenată pe ecran.

```

public Parent loadFXML(String fxml) throws IOException {
    FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource( name: fxml + ".fxml"));
    return fxmlLoader.load();
}

```

Figura 4.8: Funcția care interpretează fxml-ul și construiește obiectele

Acum trebuie să facem desenarea interfeței acest lucru se face prin punerea obiectului obținut într-o scenă

```

@Override
public void start(Stage stage) throws IOException {
    FxmlController.scene = new Scene(new FxmlController().loadFXML("/Marcel/LoginScreen"));
    stage.setScene(FxmlController.scene);
    stage.setTitle("Version software");
    App.stage = stage;
    App.stage.resizableProperty().setValue(Boolean.FALSE);
    App.stage.show();
}

```

Figura 4.9: Desenarea interfeței

După ce se execută linia **App.stage.show()** pe ecran se va vedea următoarea fereastră:

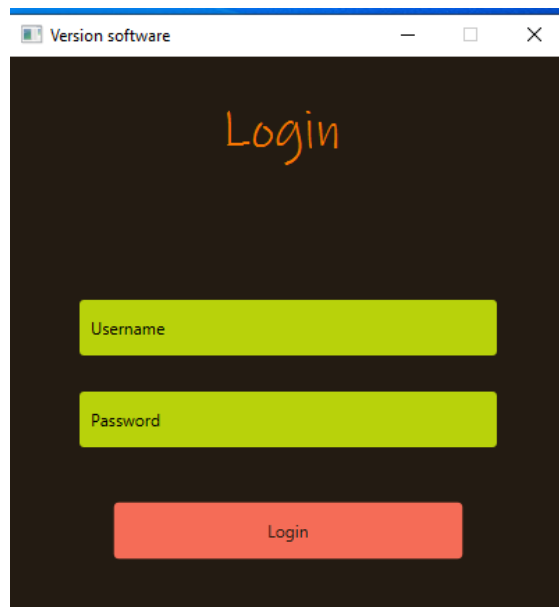


Figura 4.10: Login screen

Pentru a desena mai rapid, eficient și frumos interfața am folosit un tool integrat cu IntelliJ IDEA și anume "SceneBuilder".

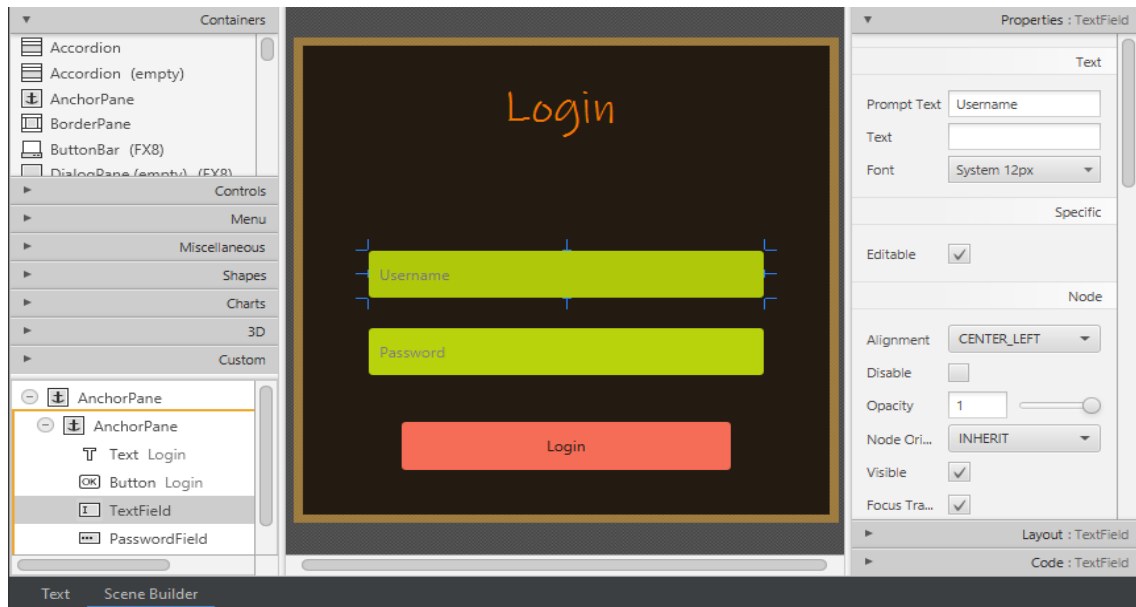


Figura 4.11: Scene Builder

Cu acest tool poziționăm elementele cum dorim. Ca totul să funcționeze cum trebuie și să pot controla interfața am avut nevoie de o clasă "Controller" în cazul acesta LoginScreenController. Controllerul în setăm fie din SceneBuilder (Controller-*j* coltroller class) sau din FXML completând atributul "**fx:controller**".

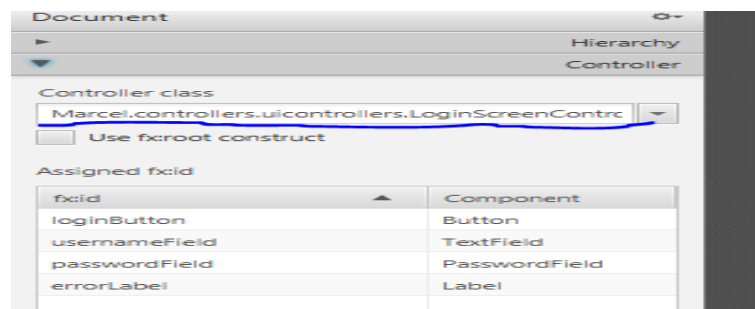


Figura 4.12: FXController în Scene Builder

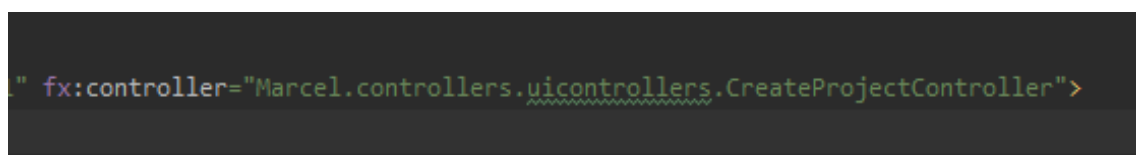


Figura 4.13: FXController în fișier fxml

Pentru a trimite requesturi HTTP către server am optat pentru pachetul ”**org.apache.httpcomponents**” care este disponibil la dependența de mai jos

```
<dependency>
  <groupId>org.apache.httpcomponents</groupId>
  <artifactId>httpmime</artifactId>
  <version>4.5.7</version>
</dependency>
```

Figura 4.14: Pachet HTTP

Aceasta furnizează funcționalități eficiente și ușor de folosit când vine vorba de comunicare HTTP cu un server web. În poza de mai jos prezint modul prin care creez un request POST și îl trimit către server.

```
public static HttpResponse POSTMethod(String url, String body) throws IOException, InterruptedException {
    HttpClient httpClient = HttpClient.newHttpClient();
    HttpRequest httpRequest = HttpRequest.newBuilder()
        .uri(URI.create(url))
        .header("Content-Type", "application/json")
        .POST(HttpRequest.BodyPublishers.ofString(
            body
        ))
        .build();
    return httpClient.send(httpRequest, HttpResponse.BodyHandlers.ofString());
}
```

Figura 4.15: POST request

Acest pachet este vital pentru aplicația mea el fiind găsit în toate paginile aplicației aducând funcționalitățile serviciilor spring în aplicație.

Un alt modul foarte important în aplicația mea este oferit de către Google și anume ”**com.google.code.gson**”. Fiind open-source acesta este disponibil la următoarea dependență:

```
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.8.6</version>
</dependency>
```

Figura 4.16: com.google.code.gson

Acest pachet transformă instanțe ale claselor din Java în string respencând formatul JSON pentru a putea fi puse cu ușurință în body-ul requesturilor HTTP. Gson furnizează și funcționalitățile inverse un string în format json poate fi convertit într-un obiect java.

Pentru a folosi acest modul am facut următoarele lucruri:

1) Am creat o instanță a unui obiect Gson:

```
Gson g = new Gson();
```

2) Am creat un obiect și l-am dat ca argument la metoda "toJson(Object)"

```
Gson gson = new Gson();  
String bodyForRequest = gson.toJson(versionModel);
```

3) Am salvat răspunsul de la server într-un string pe care l-am dat ca argument funcției "fromJson(String, Type)"

```
(g.fromJson(response.body().toString(), GroupEntity[].class))
```

Acest modul nu este pretențios la tipurile de date sau la containere poate converti orice obiect. Pentru o deserializare corectă trebuie dat exact tipul de dată sau containerul în care dorim să salvăm.

Când aplicația este lansată în execuție creez un fir de execuție fiind firul principal de execuție al aplicației. Când studentul începe un proiect un al doilea fir de execuție sub forma unei animații este pornit pentru a putea colecta fișiere de la student și pentru a avea în continuare acces la interfață în cazul în care dorește să facă o pauză sau să termine.

```
public static void main(String[] args) {  
    App app = new App();  
    primaryThread = new Thread(app, name: "Primary thread");  
    primaryThread.start();  
}  
  
private AnimationTimer animationTimer = new AnimationTimer() {  
    @Override  
    public void handle(long l) {  
        if(System.currentTimeMillis() - startTime > 120_000){...}  
    }  
};
```

Figura 4.17: Fir de execuție principal

Figura 4.18: Fir secundar pentru colectare

Firul secundar de execuție este oprit la una din acțiunile studenților și repornit când se revine în ecranul de colectare.

Fișierele le colectez pe baza extensiei (**.java**). Din directorul dat de student caut recursiv în toate directoarele și subdirectoarele. Funcția de mai jos returnează o lista cu aceste fișiere din care este extras codul sursă și îl encodează în baza64 ca o măsură de siguranță.

```
public class SearchInDirectory {

    public static List<File> searchInDirectoryAndSubDirectory(String directoryName) {
        List<File> fileCodes = new LinkedList<File>();
        File directory = new File(directoryName);
        File[] allFileFromDirector = directory.listFiles();
        if(allFileFromDirector != null){
            for (File file : allFileFromDirector) {
                if (file.isFile() && file.getName().endsWith(".java")) {
                    fileCodes.add(file);
                } else if (file.isDirectory()) {
                    fileCodes.addAll(searchInDirectoryAndSubDirectory(file.getAbsolutePath()));
                }
            }
        }
        return fileCodes;
    }
}
```

Figura 4.19: Colectarea de fișiere

După ce am prezentat cum sunt colectate fișierele este momentul să prezint cum sunt încărcate pe server folosind HTTP. În poza de mai jos se poate vedea cum sunt combinate pachetele GSON și HTTP împreună cu rezultatul funcției care folosește fișiere.

```
private void sendFileToServer(String string){
    if(contentList.size() > 0){
        String[] strings = new String[contentList.size()];
        versionModel = new VersionModel(UUID.fromString(appConfiguration.getProjectId()), contentList.toArray(new ContentModel[0]), string);
        Gson gson = new Gson();
        String bodyForRequest = gson.toJson(versionModel);
        try {
            HttpResponse response = HttpRequestAPI.POSTMethod( url: "http://localhost:9093/version/", bodyForRequest);

            if(response.statusCode() != HttpURLConnection.HTTP_CREATED){
                Thread.sleep( 5000);
                sendFileToServer(string);
            }
        } catch (IOException e) {
        } catch (InterruptedException e) {
        }
        contentList = new LinkedList<>();
    }
}
```

Figura 4.20: Colectarea de fișiere

Acum că am trecut prin aplicația desktop este timpul să prezint detalii despre serviciile spring. Pe astea le voi prezenta pe toate la un loc, diferența între acestea sunt la nivel de funcții și funcționalități. Toate sunt construite pe o arhitectura cu 3 nivele:

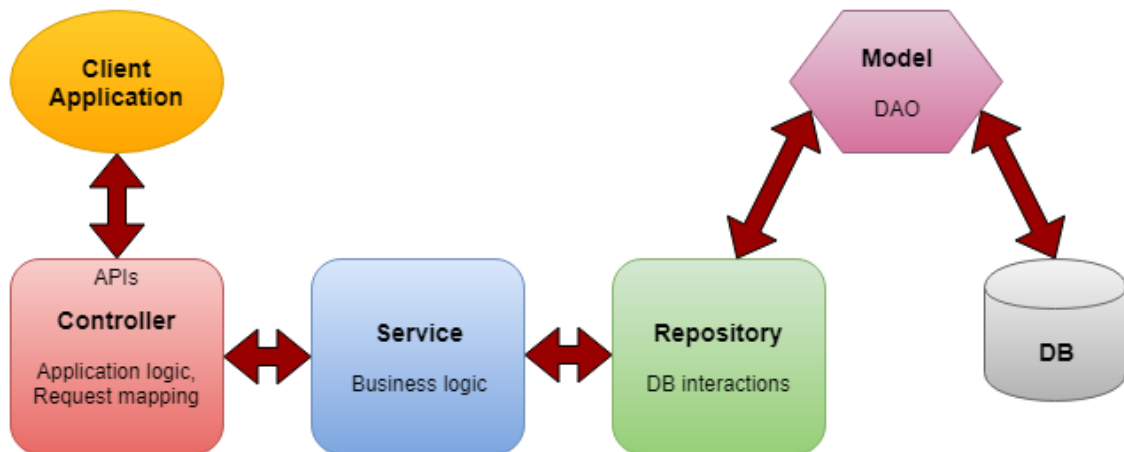


Figura 4.21: Arhitectura serviciilor

- **Controller:** este logica de nivel înalt care este expusă către exterior. La acest nivel sunt recepționate toate cererile din exterior umând să colecteze datele importante din request după care sunt trimise către nivelul următor pentru a fi analizate. Nivelul controller împachetează datele primite de la nivelul service și le trimite către aplicația client. Tot la acest nivel sunt trimise și erori către client împachetate special pentru a putea fi despachetate ușor
- **Service:** este nivelul unde datele primite de pe nivelul Controller sunt analizate. Dacă datele îndeplinesc cerințele cerute sunt acceptate și redirectionate către nivelul următor dacă nu sunt generate diferite erori. Acest nivel mai are rolul de a trimite către Controller și analizează rezultatele de la nivelul Repository
- **Repository:** este nivelul cel mai de jos care are acces la baza de date și poate citi/scrie în și din aceasta. După ce termină operațiile acest nivel trimite rezultatele la nivelul Service.

Pentru a trata erorile din servicii am utilizat **@RestControllerAdvice** care poate prinde orice erori care apar în logică inclusiv erori apărute din greșeli ex: `NullPointerException`, `IllegalArgumentException`...

Această secțiune este destinată aplicației studentului unde va fi prezentat tot ceea ce trebuie să știe studentul pentru a putea utiliza aplicația. Diferența dintre **@ControllerAdvice** și **@RestControllerAdvice** este următoarea:

@RestControllerAdvice este o componentă între **@ControllerAdvice** și **@ResponseBody**

Pentru a putea prinde erorile care apar în server am procedat în felul următor:

```
@ExceptionHandler(NotFoundException.class)
@ResponseStatus(value = HttpStatus.NOT_FOUND)
public ErrorModel handleNotFoundException(NotFoundException nfe){
    return new ErrorModel(HttpStatus.NOT_FOUND.value(), nfe.getMessage());
}

@ExceptionHandler(UnauthorizedException.class)
@ResponseStatus(value = HttpStatus.UNAUTHORIZED)
public ErrorModel handleUnauthorizedException(UnauthorizedException ue){
    return new ErrorModel(HttpStatus.UNAUTHORIZED.value(), ue.getMessage());
}

@ExceptionHandler(BadRequestException.class)
@ResponseStatus(value = HttpStatus.BAD_REQUEST)
public ErrorModel handleBadRequestException(BadRequestException bre){
    return new ErrorModel(HttpStatus.BAD_REQUEST.value(), bre.getMessage());
}
```

Figura 4.22: Error handling controller

- Am creat o clasă care am decorato cu **@RestControllerAdvice**
- Fiecare metodă am marcat cu adnotarea **@ExceptionHandler** care primește ca argument tip de excepție creată de mine sau una existentă
- În final am adăugat și adnotarea **@ResponseStatus** aceasta pune în răspunsul care ajunge la client codul specificat de mine

Am creat o clasa tip pentru a salva codul și un mesaj care să îl ajute pe client să vadă ce este greșit în request

```
public class ErrorModel {
    private int statusCode;

    private String message;

    public ErrorModel(int statusCode, String message) {
        this.statusCode = statusCode;
        this.message = message;
    }
}
```

Figura 4.23: Error class

De multe ori pot primi un input invalid de la utilizator pentru a nu permite acestor erori să ajungă la nivelul de Service sau mai rău la Repository am folosit AOP (Aspect Oriented Programming) via maven cu următoarea dependență:

```
<dependency>
  <groupId>org.assertj</groupId>
  <artifactId>assertj-core</artifactId>
  <version>3.12.2</version>
  <scope>test</scope>
</dependency>
```

Figura 4.24: AOP dependență

Cu ajutorul acestei paradigme oferite de spring prin pachetul disponibil mai sus am asigurat corectitudinea datelor primite prin request înainte ca acestea să fie procesate de controller.

```
@Aspect
@Component
public class ControllerAspects {

    private Logger logger = LoggerFactory.getLogger(this.getClass());

    @Around("@annotation(Timed)")
    public Object thread(ProceedingJoinPoint joinPoint) throws Throwable {...}

    @Pointcut("execution(public * *(..))")
    public void publicMethods() {
    }
}
```

Figura 4.25: ControllerAspects

Adnotare **@Around** va executa codul funcției "thread" pentru fiecare funcție care folosește adnotarea "@Timed"

@Pointcut ajută la potrivirea unui sfat care trebuie aplicat de un aspect la un anumit JoinPoint. Sfatul este adesea asociat cu o expresie Pointcut și se execută la orice Joinpoint asortat de Pointcut.

Pentru a utiliza această paradigmă am creat o clasă pe care am adnotat cu **@Aspect**. Odată pusă această adnotare Spring-ul va ști să o trateze ca o componentă nu ca o clasă oarecare.

```
@Pointcut("execution(* marcel.versionmanagement.controllers.ProjectController.getFinishedProject(..))")
public void getFinished() {}

@Pointcut("publicMethods()&&controllers()")
public void publicMethodsWithinControllers() {}

@Before("addProject()")
public void beforeAddProject(JoinPoint joinPoint) throws InvalidPathVariableException, InvalidObjectsException {...}

@Before("getAllGroupProject()")
public void beforeGetAllGroupProject(JoinPoint joinPoint) throws InvalidPathVariableException, InvalidObjectsException {...}

@Before("addVersion()")
public void beforeAddVersion(JoinPoint joinPoint) throws InvalidPathVariableException, InvalidObjectsException {...}
```

Figura 4.26: Before function call

Adnotarea **@Before** asociată unei funcții face legătura între funcția curentă și o funcție marcată cu **@Pointcut**. În momentul când se va apela funcția din controller înainte să fie executat codul din aceasta se execută funcția marcată cu **@Before**. Eu utilizez acest mecanism pentru a valida inputul înainte ca acesta să ajungă în controller sau la nivelele inferioare unde poate distruge integritatea datelor sau poate genera excepții.

Cu ajutorul funcțiilor marcate cu **@Before** validez atât variabilele cât și conținutul cereri primite.

Pentru baza de date și accesul la aceasta am folosit JPA. Entitățile fiind create inițial în java apoi prin intermediul interfețelor din JPA acestea au fost create ca tabele în baza de date. Figura 4.27 de mai jos prezintă tabela care memorează conturile ca obiect java, iar Figura 4.28 prezintă tabela conturi cum este cu adevărat în baza de date.

```
@Entity
@Table(name = "Conturi")
public class AccountEntity implements Serializable {
    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    @Column(name = "Id")
    private UUID id;

    @Column(name = "Username")
    private String username;

    @Column(name = "Password")
    private String password;
```

Figura 4.27: Tabela conturi în java

conturi	
id	uuid
password	varchar(255)
username	varchar(255)

Figura 4.28: Tabela conturi în baza de date

Sistemul de plagiere are la bază distanța Levenshtein. Aceasta este o metrică de șir pentru măsurarea diferenței dintre două secvențe. În mod informal, distanța Levenshtein între două cuvinte este numărul minim de editări cu un singur caracter (inserții, ștergeri sau substituții) necesare pentru a schimba un cuvânt în celălalt. Aceasta se calculează după următoarea formulă:

$$\text{lev}_{a,b}(i, j) = \begin{cases} \max(i, j) & \text{if } \min(i, j) = 0, \\ \min \begin{cases} \text{lev}_{a,b}(i-1, j) + 1 \\ \text{lev}_{a,b}(i, j-1) + 1 \\ \text{lev}_{a,b}(i-1, j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{altfel} \end{cases}$$

În acest proiect am implementat această distanță cu o matrice completă. Acest lucru presupune:

Fie s1, s2 două stringuri, $m, n \in \mathbb{N}$ astfel încât $m = \text{len}(s1)$ & $n = \text{len}(s2)$

$D \in M_{m \times n}$ matricea distanțelor, rezultând o matrice asemănătoare cu cea de mai jos.

		s	a	t	u	r	d	a	y
	0	1	2	3	4	5	6	7	8
s	1	0	1	2	3	4	5	6	7
u	2	1	1	2	2	3	4	5	6
n	3	2	2	2	3	3	4	5	6
d	4	3	3	3	3	4	3	4	5
a	5	4	3	4	4	4	4	3	4
y	6	5	4	4	5	5	5	4	3

Distanța dintre cele două stringuri este celula $D[m][n]$. Peste acest număr aplic următoarea formulă pentru a obține o valoare procentuală a plagiatului pentru a o putea folosi informația

$$P = \left(1 - \frac{D[m][n]}{\text{len}(s1) + \text{len}(s2)}\right) * 100 \quad (4.1)$$

Din formula 4.1 reies următoarele: cu cât distanța dintre cele două stringuri este mai mică cu atât valoare fracției va fi mai mică, iar rezultatul final mai mare.

Exemplu:

s1 = Saturday s2=sunday $D[m][n] = 3$

$$P = \left(1 - \frac{3}{8+6}\right) * 100 = (1 - 0,2142) * 100 = 0,7858 * 100 = 78,58\% \quad (4.2)$$

Compilerul este creat utilizând pachetul ”**javax.tools**” disponibil în Java SDK 11. Acesta accesează compilerul java de pe sistem și crează un punct de acces către acesta. Toată activitatea de compilare se petrece în memorie nefiind scris nimic pe disc decât rezultatul final. În figura 4.29 prezint funcția care compilează un fișier deja creat în memorie returnând răspunsul compilerului.

```
public boolean compile(String className,
                      CharSequence source,
                      DiagnosticListener<JavaFileObject> diagnosticListener){
    JavaCompiler javaCompiler = ToolProvider.getSystemJavaCompiler();
    List<String> options = new LinkedList<String>();
    JavaFileObject javaFile = new MemoryJavaFileObject(className, source);
    MemoryFileManager fileManager = new MemoryFileManager(javaCompiler
        .getStandardFileManager( diagnosticListener: null, locale: null, charset: null),
        byteCodeForClasses.get(className));
    JavaCompiler.CompilationTask compilationTask = javaCompiler
        .getTask( out: null,
                  fileManager,
                  diagnosticListener,
                  options, classes: null,
                  singleton(javaFile));
    return compilationTask.call();
}
```

Figura 4.29: Funcția care compilează un fișier

În finalul acestei secțiuni voi discuta și despre aplicația web destinată profesorului. Aceasta formată din 2 componente:

- Partea de cod XHTML pentru interfața web(front-end)
- Partea de cod Java dedicată funcționalității și stocării de date.

Codul xhtml este asemănător cu HTML însă este mult mai îmbogățit cu atribute și taguri. Prin conectarea sa cu Java oferă un acces pult mai ușor și rapid la informațiile din elemente.

```
<div class="col-sm-12 col-sm-offset-0 top-margin-50" style="float: left;">
  <div class="col-sm-3">
    <h:commandButton value="Back" styleClass="btn btn-primary" style="margin: 5px; color: white;" action="#{plagiatorManagement.back()}" />
  </div>
  <div class="col-sm-3">
    <h:commandButton value="test" styleClass="btn btn-primary" style="margin: 5px; color: white;" action="#{plagiatorManagement.startTest()}" />
  </div>
  <div id="version-manage" class="col-sm-offset-0 col-sm-12 disable-select" style="margin-top: 100px;">
    <p:dataTable var="cProject" value="#{plagiatorManagement.selected}" scrollable="true" scrollHeight="700" emptyMessage="No projects found">
      <p:column footerText="Project Name" headerText="Project Name" style="text-align: center;" styleClass="disable-select text-overflow-custom">
        <h:outputText value="#{cProject.name}" />
      </p:column>
    </p:dataTable>
  </div>
```

Figura 4.30: Cod XHTML

Pentru a controla elementele de pe o pagină XHTML trebuie să ne vrem o clasă și să o adnotăm cu ”@ManagedBean” cum se poate vedea în figura 4.31. Aceasta va fi indexat cu prima literă mică continuând cu denumirea clasei. Când doresc să folosesc elemente din această clasă în pagina XHTML trebuie să respect următorul format ”#{numeClasă.atribut}” sau ”#{numeClasă.metodă}” cum se poate vedea în figura 4.30.

```
17
18  @ManagedBean
19  @RequestScoped
20  public class PlagiatorManagement {
21      public Map<ProjectEntity, Boolean> projectEntities;
22      public ProjectEntity[] selected;
23      private Map<String, Object> sessionVar;
24      private GroupEntity groupEntity;
25      private Gson gson = new Gson();
26      public boolean b;
27
```

Figura 4.31: Clasa adnotată

Pentru navigarea prin pagini a fost nevoie de o regulă de navigare definită în format XML în fișierul ”faces-config.xml”, aceasta având următoarea formă:

```
<navigation-rule>
  <from-view-id>/faces/</from-view-id>
  <navigation-case>
    <from-action>#{login.loginListener}</from-action>
    <to-view-id>#{login.loginListener}</to-view-id>
  </navigation-case>
</navigation-rule>
```

Figura 4.32: Regula de navigare

Pentru un design mai frumos al aplicației am folosit bootstrap și jquery prin intermediul unor linkuri HTTPS:

```
<script type="text/javascript" src="https://code.jquery.com/jquery-2.1.1.min.js"></script>
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css"/>
<script type="text/javascript" src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
```

Aceste linkuri îmi permit să accesez și să utilizez bibliotecile respective fără să le descarc local.

Capitolul 5

Manual de utilizare

5.1 Modul de utilizare

5.1.1 Aplicația desktop

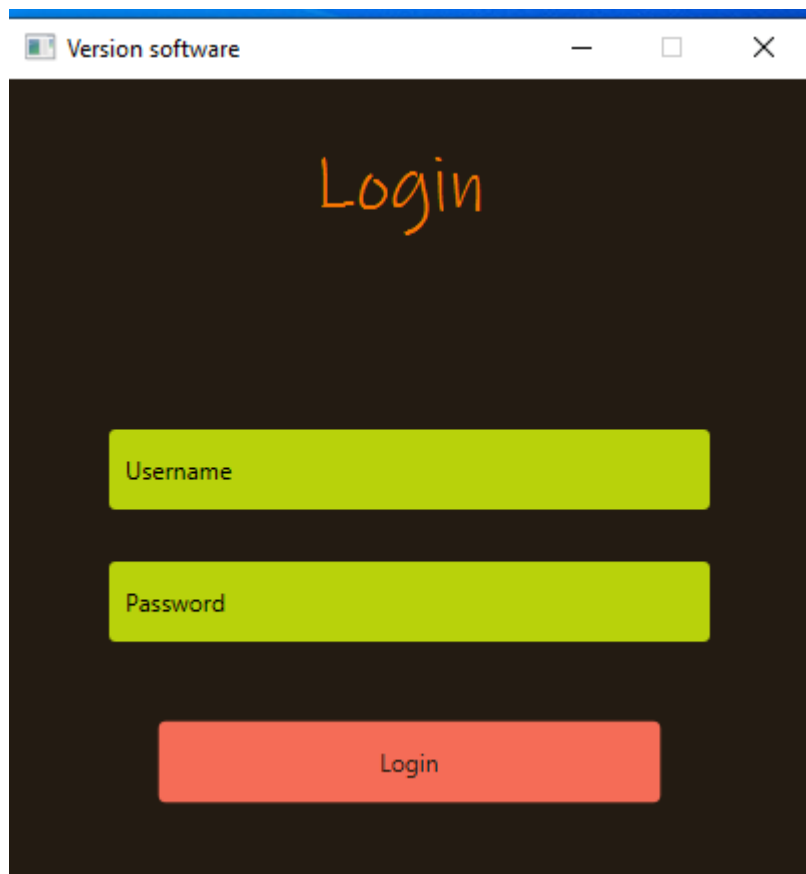


Figura 5.1: Login screen

Acesta este ecranul pentru conectarea la aplicație, studentul trebuie sa introducă un nume de utilizator și o parolă dacă acestea sunt corecte va fi trimis catre "Profilul" acestuia. Dacă nu va apărea un mesaj de eroare.

Aceasta este pagina profilului în care studentul își vede numele și anul de studiu. Aici sunt prezente 2 butoane:

- Creeza un nou proiect: apăsând acest buton studentul este redirecționat către ecranul de creare al proiectului
- Finalizeaza un proiect inceput: apăsând acest buton studentul este redirecționat către ecranul de unde poate relua lucrul la un proiect neterminat

În cazul în care studentul nu are nici un proiect inceput sau nu este înscris în nici un grup se va afișa un mesaj de eroare urmând să se creeze un proiect sau să vorbească cu un profesor să fie adăugat într-un grup pentru a putea lucra.

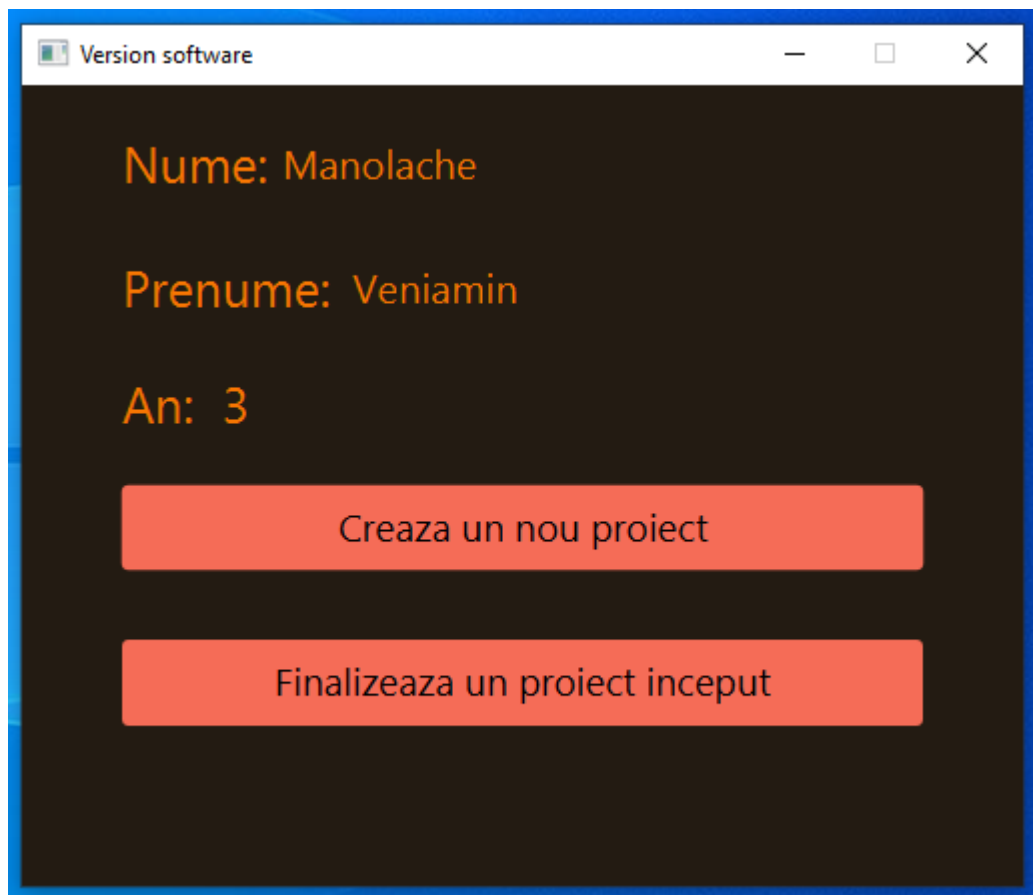
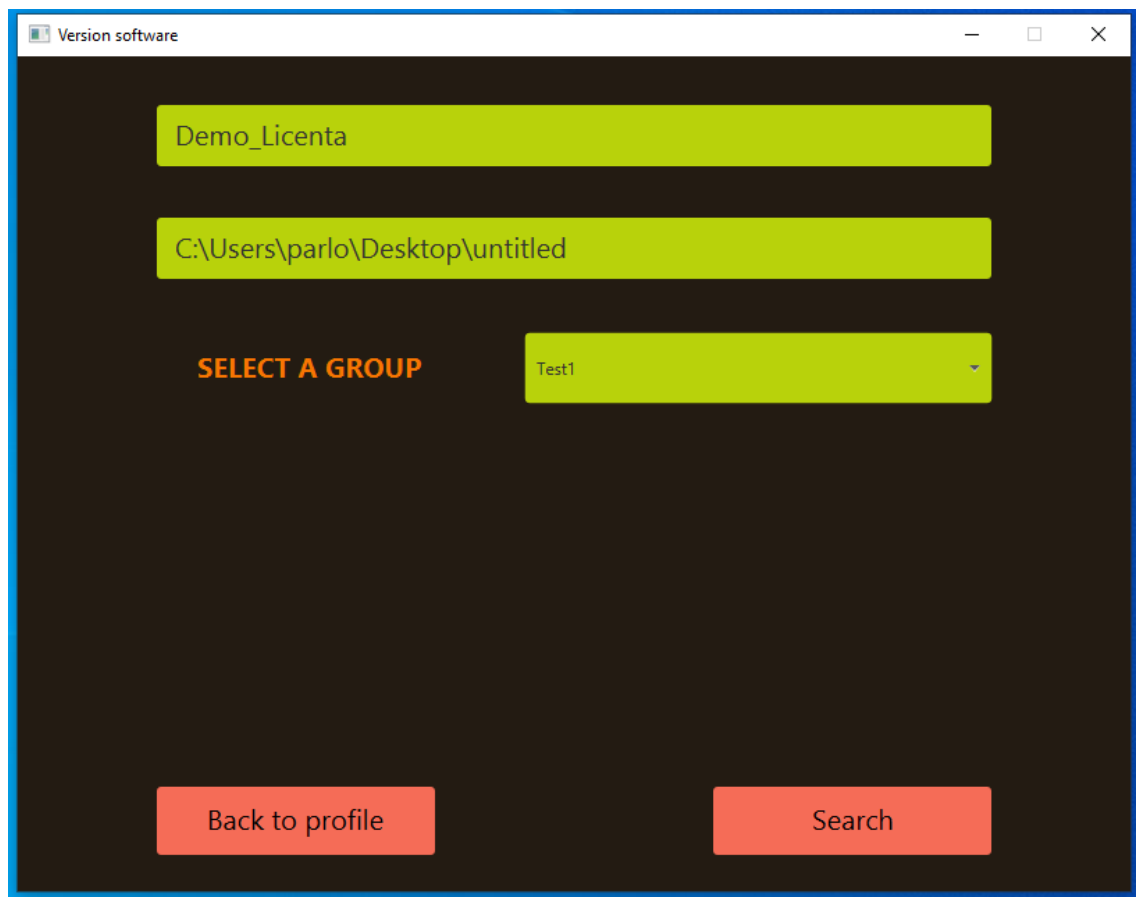


Figura 5.2: Profile screen

În această fereastră studentul trebuie să completeze cu următoarele informații OBLIGATORII pentru crearea unui proiect și anume:

- Nume pentru acesta - Demo_Licenta. Numele este necesar pentru a putea diferenția proiectele.
- Calea către fișierele sursă - C:\Users\parlo\Desktop\untitled. Ruta este utilizată pentru a colecta fișierele sursă la care lucrează. Această sursă este salvată local ne fiind încărcat pe server.
- Numele grupului- Test1. Folosit pentru a specifica faptul că acest proiect este exclusiv pentru acel grup.
- Back to profile întoarce studentul către ecranul anterior
- Search- Apăsând acest buton se trimite cererea de creare a proiectului apoi este încărcată versiunea inițială și direcționează studentul către pagina unde poate vedea în timp real ce fișiere au fost trimise către server.

În cazul în care orice informație este invalidă sau lipsește va fi afișată un mesaj de eroare.



The screenshot shows a web application window titled "Version software". The form has a dark background with yellow input fields and orange buttons. The fields are labeled with the following text:

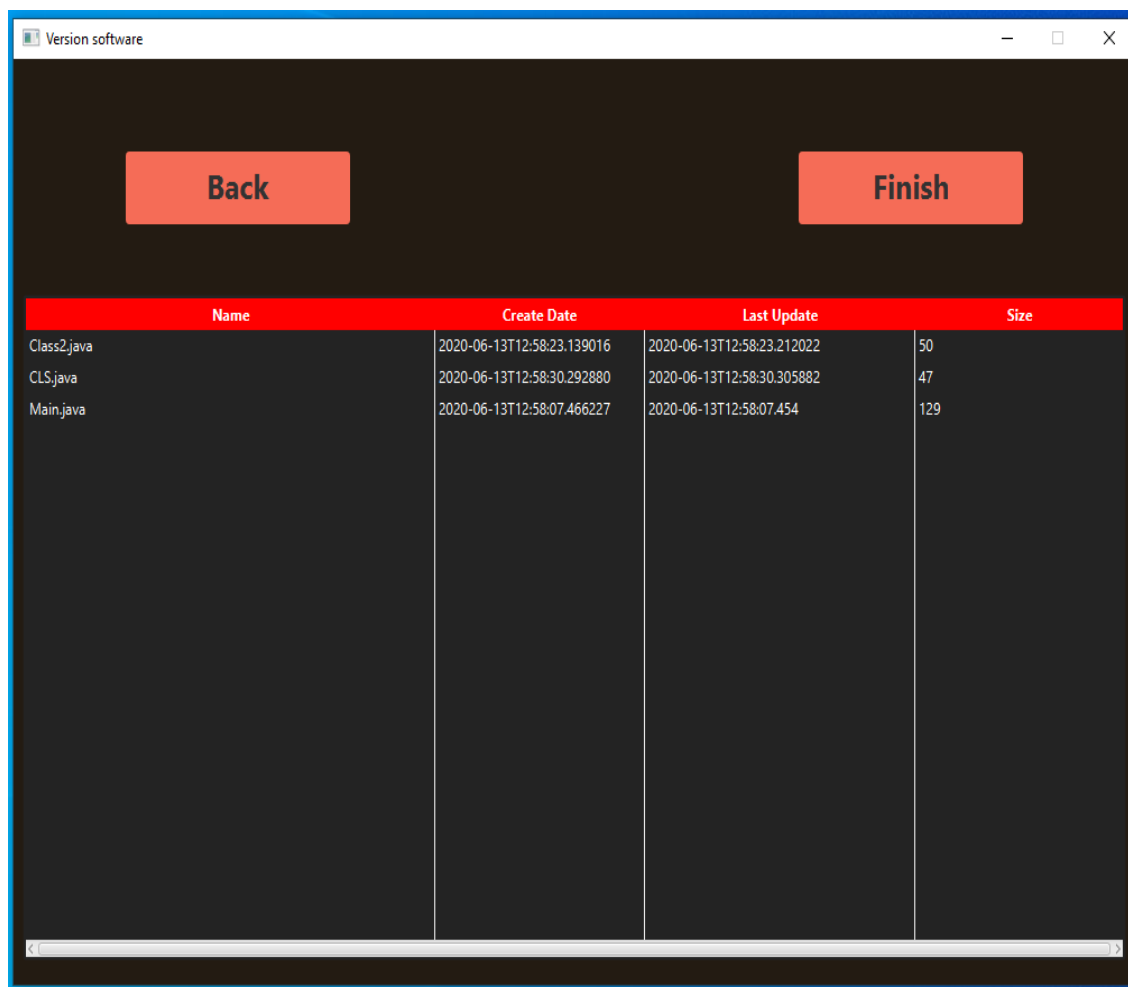
- Top field: Demo_Licenta
- Second field: C:\Users\parlo\Desktop\untitled
- Third field: SELECT A GROUP (with a dropdown menu showing Test1)
- Bottom left button: Back to profile
- Bottom right button: Search

Figura 5.3: Create project screen

Pe ecest ecran studentul va pentrece cel mai mult timp. Unde poate vedea fişierele care au fost încărcate pe server plus câteva invormăţii suplimentare (Data de creare, Data când a fost ultima dată actualizat şi dimensiunea respectivului fişier) aceste informaţii sunt vizibile in tabelul din imagine.

Butonul "Back" intorce studentul la pagina prezentată mai sus dar nu marchează proiectul ca finalizat urmând ca studentul să îl poată termina in viitor.

Butonul "Finish" în momentul în care studnetul consideră că termină proiectul în apasă încheind acest proiect fără a putea mai face alte modificări în viitor.



Name	Create Date	Last Update	Size
Class2.java	2020-06-13T12:58:23.139016	2020-06-13T12:58:23.212022	50
CLS.java	2020-06-13T12:58:30.292880	2020-06-13T12:58:30.305882	47
Main.java	2020-06-13T12:58:07.466227	2020-06-13T12:58:07.454	129

Figura 5.4: View files screen

În această ultimă fereastră studentul poate alege un proiect la care a lucrat în trecut pentru a-l continua. În câmpul "Project folder location" trebuie introdus obligatoriu calea către un folder de unde aplicația poate continua să colecteze fișiere java.

Pentru a relua activitatea la proiect trebuie selectat un proiect început și introdusă calea către folderul cu surse apoi apăsat butonul "Continue". Odată apăsat acest buton se va face o redirectionare catre pagina prezentată în Figura 5.4.

Dacă dorim sa revenim la pagina de profil trebuie apăsat butonul "Back".

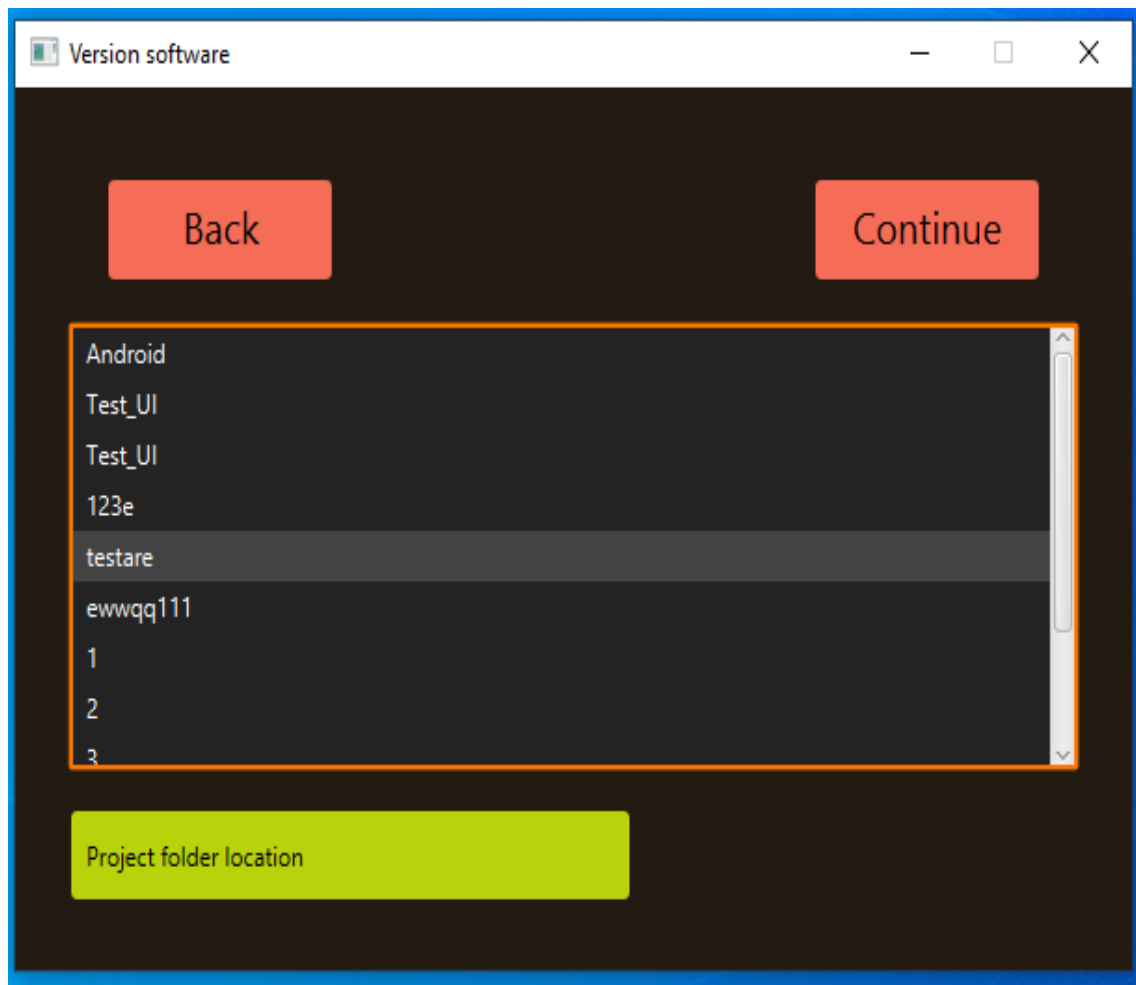


Figura 5.5: Continue project screen

5.1.2 Aplicația profesorului

Această secțiune este dedicată modului de utilizare al aplicației web destinată strict profesorului. Funcționalitățile care se regăsesc aici sunt disponibile doar profesorului.

Pentru început profesorul trebuie să se conecteze cu un cont special creat pentru acesta. În cazul în care un student încearcă să acceseze aplicația cu contul său va primi o eroare, același lucru se întâmplă cu profesorul dacă introduce un cont greșit.

După ce a introdus un cont corect profesorul este trimis către pagina în care vede grupurile sale.

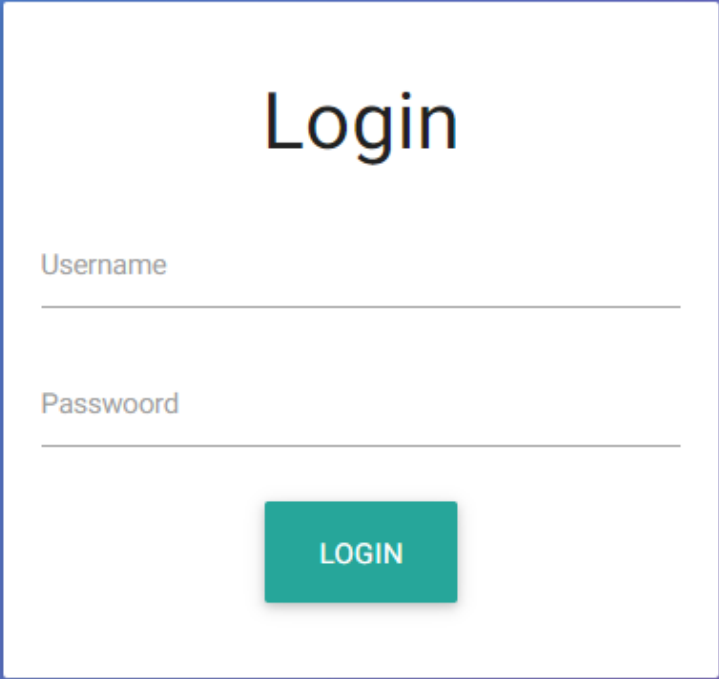
The image shows a login form centered on a blue-to-purple gradient background. The form is a white rectangle with the word "Login" in a large, black, sans-serif font at the top. Below the title, there are two input fields. The first is labeled "Username" in a light gray font, followed by a horizontal line. The second is labeled "Password" in a light gray font, followed by a horizontal line. At the bottom of the form is a teal-colored button with the word "LOGIN" in white, uppercase, sans-serif font.

Figura 5.6: Pagina de logare

Pe această pagină ajunge profesorul după ce s-a conectat cu succes. Aici sunt vizibile următoarele informații:

- Un box în care profesorul poate crea un nou grup
- Un tabel unde sunt disponibile grupurile create de profesor alături de 3 butoane pentru fiecare grup:
 - "Edit Group" - Acest buton redirecționează profesorul către pagina de unde poate modifica membri (poate adăuga sau șterge) în funcție de preferințele acestuia
 - "View project" - Acest buton redirecționează profesorul către pagina unde sunt vizibile toate proiectele create de studenți pentru acel grup
 - "Remove" - Acest buton va șterge grupul și tot ceea ce conține membri, proiecte, versiuni. Odată apăsat toate informațiile sunt pierdute.

Group Name	Manage Group	Group Content	Remove Group
Test1	Edit Group	View projects	Remove

Figura 5.7: Pagina grupurilor profesorului

Pe această fereastră sunt afișate următoarele informații:

- În partea stângă studenții care fac parte din grup
- În partea dreaptă studenții care pot fi adăugați în grup

Pentru a face diferite acțiuni asupra studenților în fiecare tabel este un buton în dreptul fiecărui student care adaugă sau după caz șterge un student din grup.

Dacă se dorește filtrarea studenților după Nume, Prenume sau an în fiecare coloană este prezent un element care facilitează acest lucru. Studenții pot fi filtrați după un singur criteriu la un moment de timp ex: dacă îi filtrăm după nume nu putem să îi mai filtrăm după prenume/an.

Butonul "Back to Home" redirecționează profesorul către pagina anterioară.

Back to Home

First Name	Last Name	Year	Action
<input type="text"/>	<input type="text"/>	Select One	
Stefanescu	Elena	3	Remove member
Tudose	Laurentiu	1	Remove member
Popescu	Iustinian	2	Remove member
Alexa	Elvira	2	Remove member
Manole	Laurentiu	1	Remove member
Manolache	Veniamin	3	Remove member
First Name	Last Name	Year	Action

First Name	Last Name	Year	Action
<input type="text"/>	<input type="text"/>	Select One	
Ursu	Roxana	3	Add member
Chis	Antonela	1	Add member
Petre	Ecaterina	3	Add member
Negrea	Aristita	3	Add member
Paraschiv	Speranta	2	Add member
Sima	Adelaida	1	Add member
Fratila	Octaviu	2	Add member
Crisan	Codrina	3	Add member
Oltean	Henrieta	1	Add member
First Name	Last Name	Year	Action

Figura 5.8: Membri grupului

Pe această pagină sunt disponibile toate proiectele create de studenți destinate acestui grup. Tabelul de mai jos conține următoarele informații:

- Numele proiectului în coloana "Project Name"
- Numele prenumele și anul studentului care a creat proiectul în coloana "Student Information"
- Statusul plagatorului în coloana "Status Plagiator". În momentul creării proiectului este setat default textul "unknown" după ce se testează plagiatul această valoare este actualizată.
- În coloana "Is ended" se specifică dacă proiectul este finalizat sau nu. Această informație este folosită de către plagiator.
- În ultima coloană este butonul cu care se trece la fereastra de afișare a versiunilor.

[Back to Home](#)

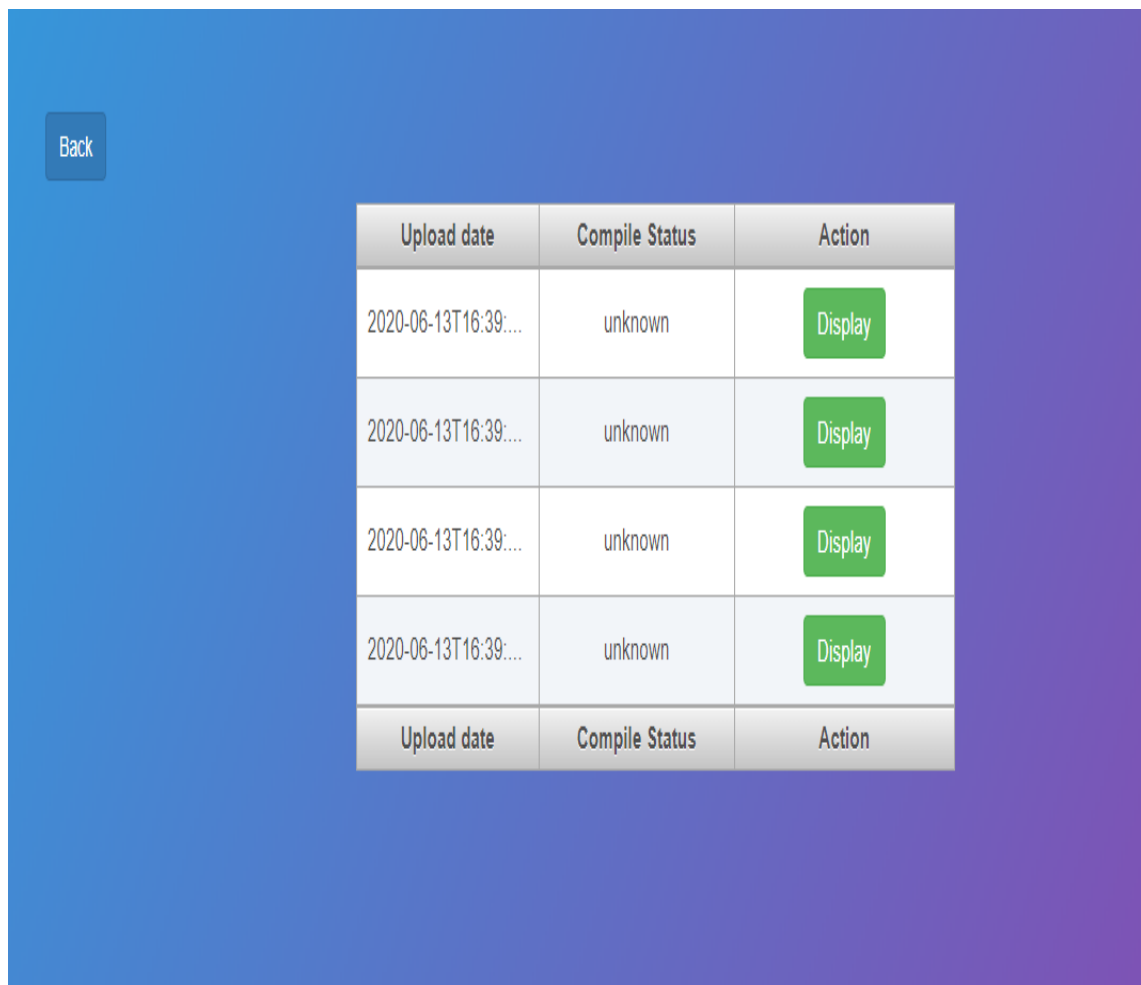
Project Name	Student Information	Status Plagiator	Is ended	Display Versions
veabadvqe	Veniamin + Manolache + 3	unknown	true	Display
numele	Veniamin + Manolache + 3	unknown	true	Display
Test_UI	Veniamin + Manolache + 3	unknown	false	Display
Test_UI	Veniamin + Manolache + 3	unknown	false	Display
123e	Veniamin + Manolache + 3	unknown	false	Display
testare	Veniamin + Manolache + 3	unknown	false	Display
ewwqq111	Veniamin + Manolache + 3	unknown	false	Display
Panarama_totala	Veniamin + Manolache + 3	unknown	true	Display
1	Veniamin + Manolache + 3	unknown	false	Display
Project Name	Student Information	Status Plagiator	Is ended	Display Versions

Figura 5.9: Proiectele grupului

Pe această pagină sunt disponibile toate versiunile unui proiect create colectate de la student. Tabelul conține următoarele informații:

- "Upload date" data și ora când au fost încărcate fișierele
- "Compile Status" rezultatul serviciului de compilare. Inițial acesta este "unknown" după este trimisă cererea de compilare se va schimba în "success" sau "failed".
- "Action" acest buton va redirecționa profesorul către fereastra unde poate vedea conținutul fișierelor sursă.

Butonul "Back" întoarce profesorul către pagina anterio



Upload date	Compile Status	Action
2020-06-13T16:39:...	unknown	Display
2020-06-13T16:39:...	unknown	Display
2020-06-13T16:39:...	unknown	Display
2020-06-13T16:39:...	unknown	Display
Upload date	Compile Status	Action

Figura 5.10: Versiunile proiectului

Această pagină afisează ”frumos” conținutul fișierelor sursă de la versiunea selectată anterior. În această pagina nu sunt alte acțiuni decât revenirea la pagina anterioară.



Figura 5.11: Conținutul versiuni

Capitolul 6

Concluzii și direcții viitoare

După acum am prezentat la începutul acestei lucrări aplicația orefă funcționalități de user & group management, project version, compiling și plagiere fiind dedicată studenților și profesorilor de la materiile de programare.

În capitolul 2 "Specificații funcționale" am prezentat detaliat funcționalitățile pe care aplicația le oferă atât desktop cât și web sau server.

Pe viitor doresc să imi concentrez atenția asupra integrării aplicației cu un API al unui sistem de versionare modern cum ar fi GitHub, BitBucket, SourceForge sau un serviciu de cloud cum ar fi AWS sau Google cloud pentru o stocare fizică a fișierelor și pentru o mai bună performanță. Aceste API-uri expun rute HTTPS pentru a putea utiliza funcționalitățile acestora.

O atenție deosebită voi acorda pe viitor dezvoltării unui algoritm de plagiere mult mai puternic. La acest punct voi utiliza un ofuscator pentru cod java acesta eliminând diferențele de denumire al variabilelor, metode, etc... atenția algoritmului fiind îndreptată către cod și nu către denumiri.

Bibliography

- [1] URL: <https://techterms.com/definition/java>.
- [2] URL: [knuthwebhttps://en.wikipedia.org/wiki/Java_\(programming_language\)](https://en.wikipedia.org/wiki/Java_(programming_language))site13.
- [3] URL: https://www.tutorialspoint.com/postgresql/postgresql_java.html.
- [4] URL: <https://dzone.com/articles/java-thread-tutorial-creating-threads-and-multithr>.
- [5] URL: <https://www.primefaces.org/showcase/ui/input/oneRadio.xhtml>.
- [6] URL: <https://www.rgagnon.com/javadetails/java-0039.html>.
- [7] URL: <https://docs.spring.io/spring/docs/4.3.14.RELEASE/spring-framework-reference/html/aop.html>.
- [8] URL: <https://profs.info.uaic.ro/~acf/java/>.
- [9] URL: [https://ro.wikipedia.org/wiki/Java_\(limbaj_de_programare\)](https://ro.wikipedia.org/wiki/Java_(limbaj_de_programare)).
- [10] URL: https://java.com/en/download/faq/whatis_java.xml.
- [11] URL: <https://www.journaldev.com/24601/java-11-features>.
- [12] URL: <https://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>.
- [13] URL: <https://docs.oracle.com/en/java/javase/11/docs/api/index.html>.
- [14] URL: <http://tutorials.jenkov.com/java/what-is-java.html>.
- [15] URL: <https://docs.oracle.com/javafx/2/overview/jfxpub-overview.html>.
- [16] URL: https://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition.
- [17] URL: <http://tutorials.jenkov.com/javafx/index.html>.
- [18] URL: <https://www.journaldev.com/2552/spring-rest-example-tutorial-spring-restful-web-services>.
- [19] URL: <https://openjfx.io/javadoc/11/>.
- [20] URL: <https://spring.io/projects/spring-boot>.
- [21] URL: <https://www.quickprogrammingtips.com/spring-boot/history-of-spring-framework-and-spring-boot.html>.
- [22] URL: <https://docs.spring.io/spring-boot/docs/2.3.1.RELEASE/reference/pdf/spring-boot-reference.pdf>.


- [23] URL: <https://stackoverflow.com/questions/36687314/how-to-style-java-fx-tableview-column-header-using-css>.
- [24] URL: <https://gist.github.com/Zookey/2759663>.
- [25] URL: <https://github.com/Marcel1123/E-Catalog/blob/master/Client/src/view/AdaugareAbsenta.java>.
- [26] URL: <https://github.com/jfoenixadmin/JFoenix/issues/411>.
- [27] URL: <http://tutorials.jenkov.com/javafx/listview.html>.
- [28] URL: <http://www.javased.com/?api=javax.tools.JavaCompiler>.
- [29] URL: <http://www.java-server-faces.org/>.
- [30] URL: http://www.javased.com/index.php?source_dir=platform_external_jsilver/src/com/google/clearsilver/jsilver/compiler/CompilingClassLoader.java.
- [31] URL: <https://www.postgresql.org/about/>.
- [32] URL: <https://www.primefaces.org/showcase/ui/data/datatable/filter.xhtml>.
- [33] URL: https://en.wikipedia.org/wiki/JavaServer_Faces.
- [34] URL: <https://www.tutorialspoint.com/jsf/>.
- [35] URL: https://www.tutorialspoint.com/spring_boot/spring_boot_introduction.html.

[illegible]

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele _FRĂSINARU CRISTIAN_

Data _22.06.2020_ Semnătura 

DECLARAȚIE privind originalitatea conținutului lucrării de licență

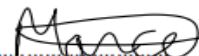
Subsemnatul(a)PÎRLOG MARCEL IONUȚ.....
domiciliul înCOM. UNIREA BRĂILA STR. ȘCOLII NR 70.....
născut(ă) la data de ...22.08.1997....., identificat prin CNP ...1970822090075.
....., absolvent(a) al(a) Universității „Alexandru Ioan Cuza” din Iași, Facultatea de
.....INFORMATICĂ..... specializarea ...INFORMATICĂ, promoția...2020., declar pe propria
răspundere, cunoscând consecințele falsului în declarații în sensul art. 326 din Noul Cod
Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la
plagiat, că lucrarea de licență cu titlul:SOFT
EDUCAȚIONAL.....elaborată sub îndrumarea dl. ____FRĂSINARU
CRISTIAN____, pe care urmează să o susțin în fața comisiei este originală,
îmi aparține și îmi asum conținutul său în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată
prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la
introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări
științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei
lucrări de licență, de diplomă sau de disertație și în acest sens, declar pe proprie
răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am
întreprins-o.

Data azi,22.06.2020.....

Semnătură student



DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul „.....*SOFT EDUCATIONAL*.....”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea „Alexandru Ioan Cuza” din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de licență.

Iași, *data*
22.06.2020

Absolvent *Prenume Nume (în clar)*

PÎRLOG MARCEL IONUȚ

(semnătura în original)

