



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

PROIECT

Proiectare Software

Student: Mihaila Alexia

Grupa: 30235



Cuprins

1. Obiectiv.....	3
2. Cerințe.....	3
3. Enuntul problemei:	4
4. Diagrama cazurilor de utilizare	5
5. Instrumente utilizate	5
6. Justificarea limbajului ales	6
7. Diagrama de clase	7
8. Diagrama entitate relatie	11
9. Functionalitatea aplicatiei	13



1. Obiectiv:

Obiectivul acestui proiect este familiarizarea cu șablonul architectural Client/Server, cu șabloanele arhitecturale orientate pe servicii (SOA) și cu șabloanele de proiectare. Pentru persistența informației se va utiliza o bază de date relațională (SQL Server, MySQL, etc.).

2. Cerințe:

Transformați aplicația implementată la tema 3 într-o aplicație client/server astfel încât să utilizați minim 5 șabloane de proiectare (minim un șablon de proiectare creațional, un șablon de proiectare comportamental și un șablon de proiectare structural) și o arhitectură orientată pe servicii (SOA).

❖ În faza de analiză se va realiza diagrama cazurilor de utilizare și diagramele de activități pentru fiecare caz de utilizare.

❖ În faza de proiectare se vor realiza:

➤ 2 diagrame de clase corespunzătoare aplicației server și aplicației client respectând principiile GRASP și DDD și folosind o arhitectură orientată pe servicii (SOA) și minim 5 șabloane de proiectare;

➤ diagrama entitate-relație corespunzătoare bazei de date;

➤ diagrame de secvență corespunzătoare tuturor cazurilor de utilizare.

❖ În faza de implementare se va scrie cod pentru îndeplinirea tuturor funcționalităților precizate de diagrama cazurilor de utilizare utilizând:

➤ proiectarea dată de diagramele de clase și diagramele de secvență;

➤ unul dintre următoarele limbaje de programare: C#, C++, Java, Python.

❖ Finalizarea temei va consta în predarea unui director ce va cuprinde:

➤ Un fișier cu diagramele UML realizate;

➤ Baza de date;



- Aplicația soft;
- Documentația (minim 20 pagini)-un fișier care cuprinde:
 - numele studentului, grupa;
 - enunțul problemei;
 - instrumente utilizate;
 - justificarea limbajului de programare ales;
 - descrierea diagramelor UML;
 - descrierea aplicației.

3. Enunțul problemei:

Dezvoltați o aplicație care poate fi utilizată într-un lanț de magazine de parfumuri. Aplicația va avea 3 tipuri de utilizatori: angajat al unui magazin de parfumuri, manager al lanțului de magazine de parfumuri și administrator.

Utilizatorii de tip angajat al unui magazin de parfumuri pot efectua următoarele operații după autentificare:

- ❖ Vizualizarea listei tuturor parfumurilor din parfumeria la care este angajat sortată după următoarele criterii: denumire și preț;
- ❖ Filtrarea parfumurilor după următoarele criterii: producător, disponibilitate, preț;
- ❖ Căutarea unui parfum după denumire;
- ❖ Operații CRUD în ceea ce privește persistența parfumurilor din magazinul la care lucrează acel angajat;
- ❖ Salvare liste cuparfumuridin parfumeria la care este angajat în mai multe formate: csv, json, xml, txt.

Utilizatorii de tip manager al lanțului de magazine de parfumuri pot efectua următoarele operații după autentificare:

- ❖ Vizualizarea listei tuturor parfumurilor dintr-un magazin selectat sortată după următoarele criterii: denumire și preț;



❖ Filtrarea parfumurilor după următoarele criterii: parfumerie, producător, disponibilitate, preț;

❖ Căutarea unui parfum după denumire;

❖ Salvare liste cu situația parfumurilor din lanțul de parfumerii în mai multe formate: csv, json, xml, txt;

❖ Vizualizarea unor statistici legate de produsele din lanțul de parfumerii utilizând grafice (structură radială, structură inelară, de tip coloană, etc.).

Utilizatorii de tip administrator pot efectua următoarele operații după autentificare:

❖ Operații CRUD pentru informațiile legate de utilizatori;

❖ Vizualizarea listei tuturor utilizatorilor și filtrarea listei utilizatorilor după tipul utilizatorilor.

Interfața grafică a aplicației va fi disponibilă în cel puțin 3 limbi de circulație internațională.

❖ Notificarea fiecărui utilizator prin cel puțin 2 variante (email, SMS, WhatsApp, Skype, etc.) la orice modificare a informațiilor de autentificare aferente aceluși utilizator.

4. Diagrama cazurilor de utilizare

Toți utilizatorii pot să se logheze, iar după logare fiecare are funcționalități diferite, în funcție de rolul pe care îl prezintă. Fata de tema anterioară, tema 2, s-au mai adăugat câteva funcționalități, cum ar fi, manager-ul poate să creeze niste grafice de tip structură radială, inelară și de tip coloană. Această diagramă se poate observa în figura 1.

5. Instrumente utilizate.

Pentru realizarea diagramei cazurilor de utilizare am folosit aplicația draw.io iar pentru generarea diagramei de clase, am folosit aplicația Star UML și am realizat manual diagrama de clase. Mediul de dezvoltare ales este IntelliJ, în care am utilizat biblioteca Java Swing pentru crearea unei interfețe grafice pentru utilizator. Pentru crearea bazei de date și pentru conexiunea dintre program și baza de date am folosit MySQL workbench, de unde am generat și o diagramă a



bazei de date. Suplimentar, pentru realizarea unor diagrame de activitati, am folosit Visual Paradigm online.

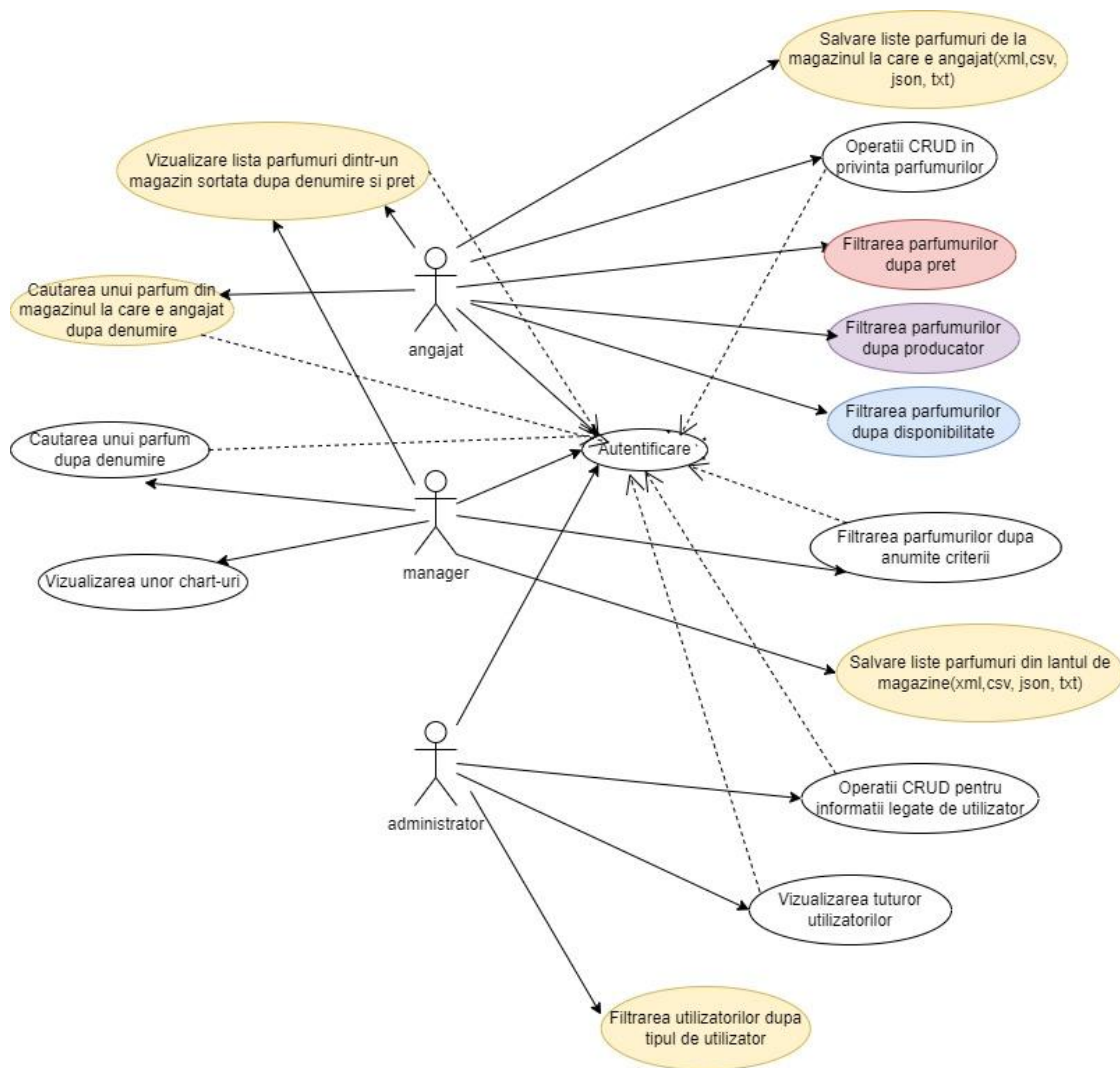


Fig. 1

6. Justificarea limbajului ales

Pentru implementarea acestui proiect am ales sa folosesc limbajul Java, deoarece este limbajul pe care il stapanesc cel mai bine, si care se potriveste pentru acest tip de aplicatie, deoarece are la dispozitie biblioteci cu ajutorul carora interfata grafica se creaza foarte usor.



Totodata, acest limbaj este folosit pentru programarea orientate pe obiecte, pe care o trebuie sa o aplicam si in proiectul nostru. Acest limbaj permite crearea de aplicatii desktop, care permit comunicarea dintre cod si interfata, pentru a crea un program functional. Este un limbaj actual, care se foloseste des in diverse aplicatii.

7. Diagrama de clase

In figura 2 puteti observa diagrama de clase a aplicatiei. In primul rand, pentru acest proiect am ales sa implementam arhitectura Client Server, asadar, vom avea 2 diagrame de clase.

Pentru inceput, vom prezenta structura acestei arhitecturi. Arhitectura client-server este un model de arhitectură software care implică două componente principale: clientul și serverul. Aceste componente comunică între ele pentru a realiza funcționalități specifice.

Arhitectura client-server este utilizată într-o gamă largă de aplicații și servicii, cum ar fi aplicații web, aplicații de rețele sociale, jocuri online, sisteme de mesagerie și multe altele. Ea oferă o modalitate flexibilă și scalabilă de a împărți sarcinile și resursele între client și server, permițând dezvoltarea de aplicații robuste și distribuite.

În Java, arhitectura client-server poate fi implementată utilizând diferite tehnologii și biblioteci, cum ar fi sockets, biblioteca si tehnica pe care am folosit-o si eu in proiectul meu. Totodata am mai folosit si `objectInputStream` si `objectOutputStream`. Sockets și `ObjectInputStream` / `ObjectOutputStream` sunt instrumente puternice pentru comunicarea între client și server în Java. Ele oferă posibilitatea de a trimite și primi date complexe și structurate, cum ar fi obiecte, între două componente software. Acestea sunt adesea utilizate în implementarea aplicațiilor client-server pentru transferul de date între aplicații și servere prin rețea.

Astfel, am impartit tema anterioara in 2 proiecte diferite, unul de client si unul de server.

Un proiect de tip **SERVER** în Java este o aplicație care rulează pe o mașină și așteaptă conexiuni de la clienți. Acesta ascultă pe un anumit port pentru cereri de conectare. Odată ce primește o cerere de conexiune de la un client, serverul acceptă conexiunea și stabilește o legătură cu acel client.

După stabilirea conexiunii, serverul poate primi date sau cereri de la client și poate răspunde în consecință. Aceasta implică utilizarea fluxurilor de intrare și ieșire pentru a trimite și primi date între server și client prin intermediul socket-urilor.

In proiectul de **SERVER** avem urmatoarele pachete:



Model: reprezintă componenta care se ocupă de stocarea și gestionarea datelor aplicației. Modelul poate include date de intrare (input), procesare și stocare.

Business Logic: reprezintă componenta care se ocupă de logica de afaceri a aplicației. Aceasta componenta primește date de la client prin intermediul Stream-urilor de input și efectuează operațiile necesare pe baza informațiilor primite. Apoi returnează înapoi către client rezultatele obținute

Am folosit aceste pachete pentru a separa logic componentele unei aplicații într-un mod ușor de gestionat și de extins. Prin separarea componentelor aplicației, dezvoltatorii pot lucra independent asupra fiecărei componente fără a afecta celelalte componente. Acest lucru face posibilă dezvoltarea și întreținerea aplicațiilor mai mari și mai complexe.

În primul rând, putem observa că avem un pachet "Model", în care am inclus mai multe clase. Clasele Store, Perfume, Stock, User și clasa de tip enumeratie UserType sunt clase care conțin atributele corespundente și metode de get și set. Aceste clase corespund cu tabelele din baza de date și cu coloanele lor și ele reprezintă entitățile de care vom avea nevoie. Tot în acest pachet se regăsește și o interfață Observer care va fi implementată de către user, deoarece vrem ca acesta să fie notificat când limba se schimbă în interfață. Mai avem și interfetele observer și prototype care ne ajută în implementarea unor design pattern-uri.

În continuare, tot în acest pachet se poate observa un alt pachet numit "Persistency", care conține mai multe clase de tip persistentă, în care se regăsesc funcții pentru query-urile efectuate pentru a face anumite operații cu baza de date. În clasa UserPersistency am creat o funcție pentru a insera utilizatori în baza de date. În clasa PerfumePersistency am creat o funcție pentru a insera parfumuri în baza de date. În clasa StorePersistency am creat o funcție pentru a insera magazine în baza de date. În clasa Stocks persistency am creat o funcție pentru a insera utilizatori în baza de date. În clasa PerfumeFromStorePersistency am creat funcțiile necesare pentru ca Angajatul să poată să efectueze operațiile pe care trebuie să le facă. În clasa ManagerPersistency am creat funcțiile necesare pentru ca Managerul magazinului să poată opera baza de date și să extragă sau să introducă datele necesare în baza de date. Același lucru l-am făcut și pentru AdminPersistency. Astfel, pachetul Persistency conține funcții care creează și efectuează Queryuri pe baza de date, pentru a putea manipula datele din ea. Suplimentar, față de tema 1, în acest pachet am mai adăugat 4 clase: GenerateCSV, GenerateXML, GenerateJSON, GenerateTXT. care ne ajută să generăm fișiere de un anumit tip, care să conțină date despre parfumurile din baza de date

În Business Logic avem clase de tip Controller care gestionează logica aplicației. În aceste clase sunt primite de la client datele asupra cărora se dorește efectuarea unor operații care sunt în legătură cu baza de date. Cu ajutorul acestor date primite, apelăm funcții din persistentă și returnăm un rezultat care va fi trimis înapoi la client pt ca acesta să fie expus în interfață.

Mai avem o singura clasa care este clasa MainClass, in care avem o singura metoda statica main. Aceasta clasa ne porneste functionalitatea aplicatiei server. Diagrama de clase pentru SERVER se poate observa in urmatoarea figura (Fig2):

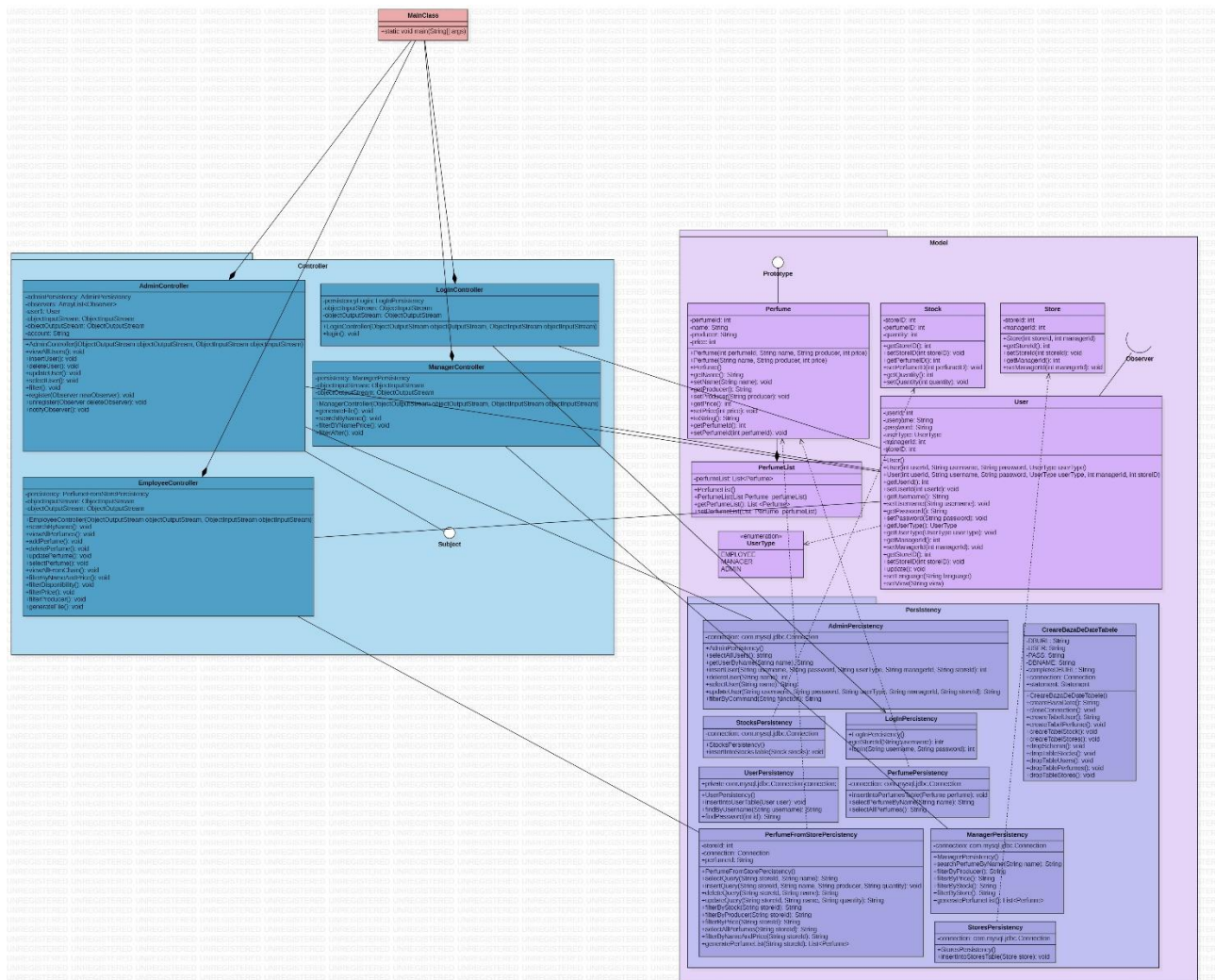


Fig 2

In continuare avem si un proiect de tip **CLIENT**. Într-un proiect de tip client, aplicația este responsabilă de inițierea conexiunii cu serverul și de trimiterea cererilor către acesta. De obicei, clientul se conectează la server utilizând adresa IP și portul serverului specificat. După stabilirea conexiunii, clientul poate trimite cereri către server și poate aștepta răspunsurile corespunzătoare.

Clientul poate solicita diverse servicii sau resurse de la server, cum ar fi obținerea de date, procesarea informațiilor, efectuarea de operații sau interacțiunea cu alte componente ale sistemului. Un proiect de client implică, de obicei, gestionarea interacțiunii cu utilizatorul,



precum și procesarea și afișarea răspunsurilor primite de la server. Acesta poate implica implementarea unei interfețe grafice pentru a permite utilizatorului să interacționeze cu aplicația

În proiectul de **CLIENT** avem următoarele pachete:

View: reprezintă componenta care afișează datele modelului și permite interacțiunea cu utilizatorul. Interfața utilizator este formată din obiecte grafice și elemente precum butoane, casete de text, liste etc.

Controller: Acest pachet primește date de la utilizator prin intermediul interfeței grafice pe care mai apoi le trimite la server. Datele returnate de server prin intermediul stream-urilor de input se afișează înapoi în interfața.

Helper: Continuu clase pt trimiterea SMS-urilor si a Email-urilor.

Connection: Se ocupa de conexiunea dintre client si server

În programul nostru, pachetul View conține clase în care se creează interfața grafică. Aici se creează design-ul interfeței cu ajutorul bibliotecii Java Swing. Am creat 4 clase, una pentru pagina de log in, una pentru pagina angajatului, una pt pagina managerului și una pt pagina administratorului. Aceste clase conțin butoane, text-field-uri din care se preiau informațiile și diverse etichete și text-areas în care să se dispună rezultatele obținute în urma apăsării butoanelor. Pe lângă acestea, regăsim și metode getter și setter pentru extragerea și setarea atributelor din clase. O implementare suplimentară adăugată proiectului este faptul că interfețele sunt dispuse în 3 limbi de circulație internațională, anume: română, engleză, franceză. Utilizatorul are 3 butoane la dispoziție prin care poate alege limba dorită. Totodată, dacă din pagina de log-in se alege o anumită limbă, atunci ferestrele pentru manager, admin și angajat se vor deschide cu limba aleasă deja în pagina de login. Bineînțeles, aceasta se va putea schimba ulterior prin apăsarea butoanelor. În același timp, în aceste clase găsim și o metodă în care prin ajutorul unui resource bundle setăm limba corespunzătoare.

În pachetul Controller avem 4 clase care se ocupă de gestionarea interacțiunii dintre client și server. Aceste clase sunt: LoginController, AdminController, EmployeeController și Manager Controller.

Această clasă este responsabilă de gestionarea evenimentelor generate de utilizator în cadrul ferestrei de autentificare (interfața LoginView).

Mai jos sunt explicații pe scurt pentru diferitele părți ale codului: Se importă clasele necesare din alte pachete pentru conexiunea la server și interfața utilizator (View). Clasa LoginController are o referință la o instanță a LoginView, un șir de caractere pentru limba selectată și o instanță a clasei ServerConnection denumită proxyServer. Constructorul clasei



primește o instanță a LoginView și setează evenimente de ascultare pentru butoanele din interfață, cum ar fi butonul de logare și butoanele pentru a schimba limba.

Metodele engleza(), franceza() și romana() sunt apelate atunci când utilizatorul dă clic pe butoanele de limbă corespunzătoare. Aceste metode actualizează limba interfeței și apelează metoda onLocaleChange() a LoginView pentru a actualiza elementele interfeței în funcție de noua limbă selectată.

Metoda login() este apelată atunci când utilizatorul dă clic pe butonul de autentificare. Aceasta preia informațiile de la utilizator (username și password) și le trimite către server utilizând proxyServer. Apoi, primește un răspuns de la server și acționează în consecință, afișând un mesaj de eroare sau deschizând o nouă fereastră (view) în funcție de rolul utilizatorului.

În funcție de rolul utilizatorului, se utilizează clasa FactoryView pentru a obține o instanță a unei ferestre specifice, precum EMPLOYEE, MANAGER sau ADMIN.

Acest cod demonstrează un exemplu simplu de interacțiune între o interfață de autentificare și un server prin intermediul unei conexiuni de server. El gestionează evenimentele utilizatorului și realizează comunicarea cu serverul pentru a efectua operațiile necesare în funcție de cerințele aplicației.

Mai jos se poate vedea diagrama de clasa pentru proiectul CLIENT (Fig 4).

8. Diagrama entitate relatie

În aceasta diagrama(Figura 5), se pot observa tabelele existente în baza de date. Am create 4 tabele. Tabelul users contine utilizatorii aplicatiei si are coloanele username, password, userType (aceasta coloana continut tipul utilizatorului si dupa aceasta coloana ne vom ghida atunci cand alegem ce pagina sa deschidem in aplicatie), apoi mai avem managerId si storeId, care pot sa fie si nule, in functie de rolul utilizatorului.

În continuare avem un tabel perfumes, în care se regasesc parfumurile care exista în stocul lanțului de magazine, și care pot sau nu se regaseasca și în câte un magazin care aparține lanțului. Acest tabel contine id-ul parfumului, producatorul, numele și prețul parfumului.

Mai departe avem și tabelul stocks, care ne spune ce parfumuri se regasesc în fiecare magazin din lanțul nostru. Avem coloanele perfumeId, storeId, și quantity, acestea indicându-ne relația dintre tabelul stocks și perfumes.

Un ultim tabel este tabelul stores, care ne indica ce magazine avem în lanț și ce manager au acestea.

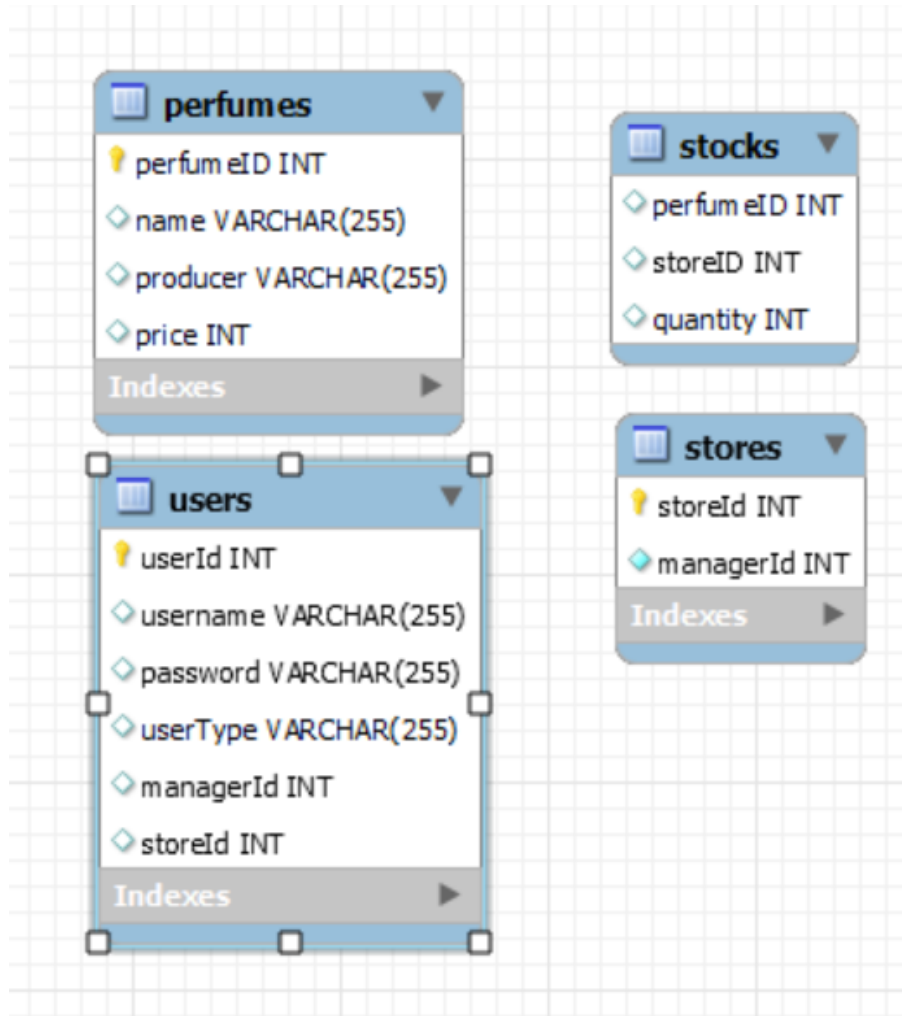


Fig. 5

9. Functionalitatea aplicatiei

La pornirea aplicatiei, se va deschide o fereastră de log-in. În această fereastră, utilizatorul își va introduce numele și parola, iar dacă câmpurile vor fi lăsate goale, pe ecran va apărea un mesaj care să indice utilizatorului că trebuie completate câmpurile. Apoi, la apăsarea butonului de log-in, username-ul și parola vor fi extrase din interfața, iar apoi pe baza acestora se caută în baza de date și se găsește care este rolul utilizatorului. Apoi, în funcție de asta, se va deschide o fereastră fie pentru angajat, fie pentru admin fie pentru manager.

În imaginile de mai jos putem vedea cum funcționează fereastră de login. În primul rând, trebuie să știm că în baza de date sunt deja create 3 tipuri de utilizator. Aceștia sunt:

angajat – username: mariapopa, password: abcd;

admin – username: andreeaB, password: abcd;



manager: username – alexiam, password abcd;

Se poate observa cazul in care in campuri nu s-au introdus date. In aceasta situatie se afiseaza un mesaj in care utilizatorul este atentionat in vede cu acest fapt(fig 6)

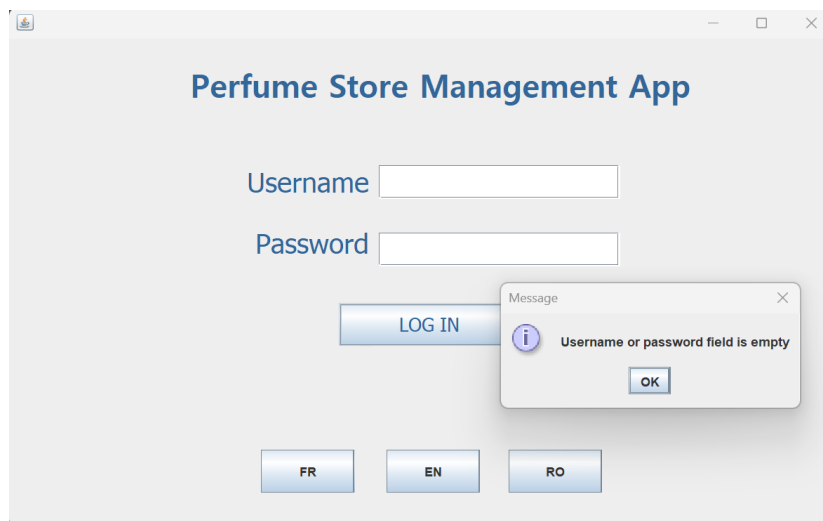


Fig 6.

In cazul in care s-a introdus un nume sau o parola gresita, acest aspect va aparea intr-o caseta pe ecran, continand un mesaj care sa indice asta(Fig 7):

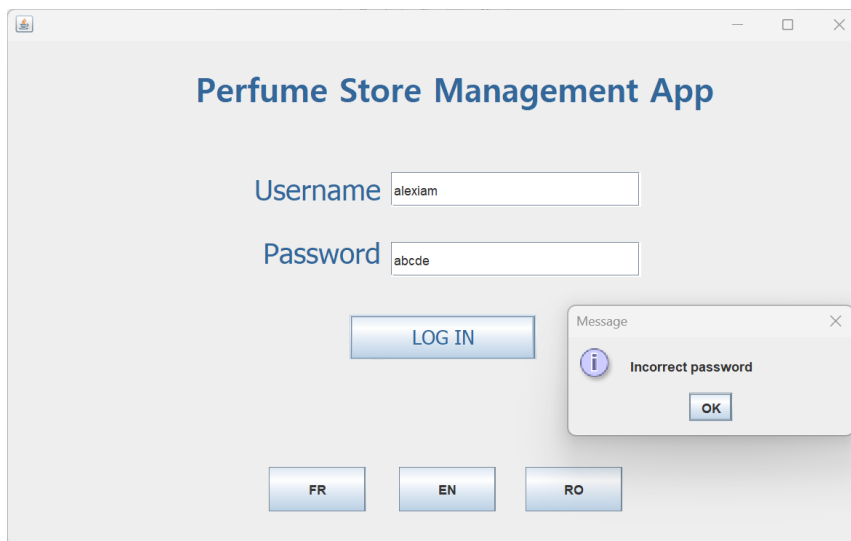


Fig 7.



Mai avem un caz in care utilizatorul introduce corect datele, iar in functie de rolul acestuia o sa se deschida o fereastră care sa corespunda.

Daca utilizatorul este angajat, acesta o sa fie angajat intr-un magazin, a carui id o sa apara intr-un camp din interfata. In interfata avem un textArea in care se vor vedea toate informatiile extrase din baza de date. Utilizatorul poate sa vada toate parfumurile existente din magazinul la care lucreaza, care vor fi afisate in textArea la apasarea butonului. Totodata poate sa vada si toate parfumurile existente in baza de date a lantului de magazine, ca sa stie ce poate sa adauge in magazinul la care lucreaza. Mai apoi, acesta poate sa filtreze parfumurile din magazine in functie de 3 criterii. Disponibilitate, pret si numele producatorului. Poate sa filtreze parfumurile si in functie de nume si de pret. Aceste operatii pot fi efectuate apasand cate un buton care corespunde fiecaruia dintre ele. Dupa apasarea butonului, se pot vizualiza rezultatele in textArea-ul din partea dreapta. Ce mai poate sa faca angajatul, e sa selecteze un parfum in functie de denumirea lui, iar in textArea o sa se afiseze attributele parfumului cautat. Acesta mai poate sa insereze un parfum in magazinul la care lucreaza, daca acest parfum exista deja in tabelul de parfumuri, adica in stocul lantului de parfumuri. Totodata, poate sa stearga in parfum sau sa-l actualizeze stocul din magazine. De asemenea, angajatul poate sa genereze fisiere in format XML, CSV, JSON si TXT care contin date despre parfmurile din parfumeria la care lucreaza. Daca utilizatorul este manager, acesta poate sa cauta un parfum din tabelul de parfumuri, dupa denumire.

Fereastră de EMPLOYEE arata asa(Fig 8):

Fig 8.



Managerul poate cauta in lista de parfumuri existente in lantul de magazine, nu intr-un magazine particular. Parfumul cautat va fi afisat in text-Area-ul din partea dreapta la apasarea butonului “Search perfume by name”. Acesta poate filtra toate parfmurile din lantul de magazine, in functie de producator si pret,. Poate filtra si dupa disponibilitatea in magazine si s-a mai adaugat un criteriu de filtrare fata de tema1. Totodata, pentru un magazine dorit, mai poate sa filtreze in functie de pret si de nume. Rezultatele vor aparea in textArea. Mai poate sa genereze fisiere in format XML, CSV, JSON si TXT care contin date despre parfmurile din lantul de parfumerii.

Fereastra de MANAGER arata asa(Fig. 9):

Fig 9.

Daca utilizatorul e admin, acesta poate efectua calule asupra tabelului de utilizatori. Acesta poate vizualiza toti utilizatorii din aplicatie si poate sa insereze utilizatori, poate sa ii stearga, poate sa ii actualizeze si poate sa selecteze cate un utilizator. Administratorul poate sa filtreze utilizatorii in functie de tipul utilizatorului.

Fereastra de admin arata asa(Fig 10):



Perfume Store Management App

ADMIN

Username

Password

Function

Manager id

Store id

FILTER BY

Fig. 10

Totodata, pentru toate cele 4 interfete (Login, Admin, Employee, Manager), interfata se poate dispune in 3 limbi diferite: Romana, Engleza, Franceza. Aplicatia se va deschide in limba engleza, iar daca in fereastra de log-in se schimba limba, fereastra care se va deschide in functie de rolul utilizatorului va fi afisata in limba care a fost selectata in login. Schimbarea limbilor se poate face prin intermediul a 3 butoane care sunt afisate in fiecare interfata. Butonul FR se foloseste pentru a schimba limba in franceza, butonul EN se foloseste pentru a schimba limba in engleza iar RO se foloseste pentru a schimba limba in Romana.

Vom exemplifica cele 3 limbi pe interfata Angajatului. Engleza (Fig 11), Romana(Fig 12), Franceza(Fig 13).



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Perfume Store Management App

EMPLOYEE

Store ID

Name

Producer

Price

Stock

INSERT DELETE

UPDATE SELECT

Search perfume by name

Filter After

Producer Disponibility

Price

Filter By Name and Price

View All Perfumes From The Selected Store

View Perfumes from the Chain

Generate File in Format Generate

FR EN RO

Fig 11.

Aplicatie de gestiune a parfumurilor

ANGAJAT

ID Magazin

Nume

Producator

Pret

Stoc

INSEREAZA STERGE

ACTUALIZEAZA SELECTEAZA

Cauta parfum dupa nume

Filtreaza Dupa

Producator Disponibilitate

Pret

Filtreaza dupa nume si pret

Vezi toate parfumurile din magazinul selectat

Vezi parfumurile din lantul de magazine

Genereaza Fisier in Format Genereaza

FR EN RO

Fig 12.

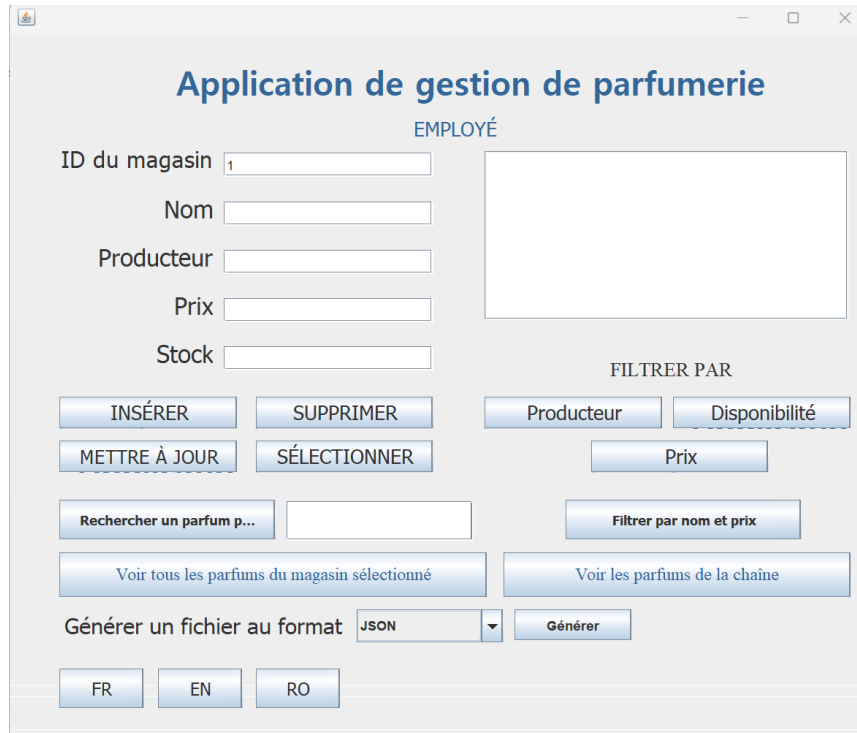


Fig 13.

10. Diagrame de activitate

In continuare puteti observa si cateva diagrame de activitate.

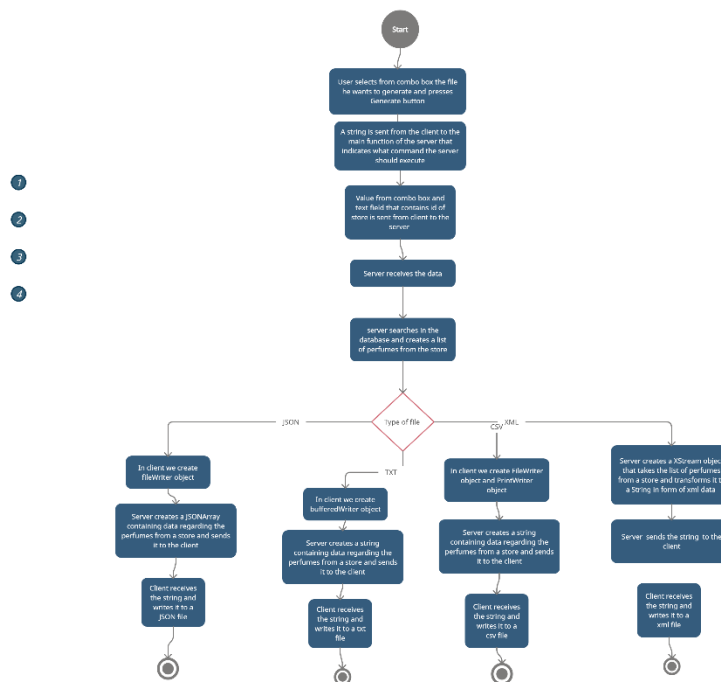


Fig 14

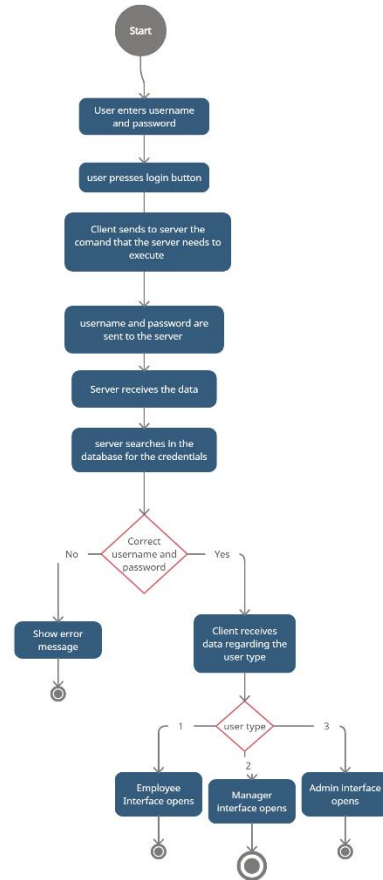


Fig. 15

11.Bibliografie

<https://www.w3schools.com/sql/>

<https://www.geeksforgeeks.org/junit-testing-for-mysql-project-injava>