A picture containing logo

Description automatically generated**FACULTATEA DE AUTOMATICĂ ŞI CALCULATOARE**

**CATEDRA CALCULATOARE**

**LANT DE MAGAZINE DE PARFUMURI**

**Documentatie**

**Proiectul numarul 3**

**Viman Andrei-Liviu**

**Grupa: 30236**

# Enuntul problemei

Scopul acestui proiect este de a dezvolta o aplicatie desktop care poate fi utilizata intr-un lant de magazine de parfumuri. Aplicatia are doua tipuri de utilizatori: angajat al lantului de magazine dar si administratorul acestui lant. Fiecare are un set de operatii pe care le poate face in functie de rolul pe care il are in magazine. In al doilea rand scopul acestui proiect este de a utiliza un sablon arhitectural foarte folosit in ziua de azi, model – view – presenter.

# Instrumente utilizate si justificarea limbajului ales

Inainte sa ma apuc de implementarea acestui lant de magazine, a fost nevoie sa inteleg mai bine ce am de facu si cum trebuie sa functioneze aceasta aplicatie a trebuit sa folosesc StarUML pentru a construi diagrama cazurilor de utilizare iar mai apoi diagrama de clase care sa respecte arhitectura MVP. Numai dupa ce am facut aceste lucruri m-am apucat efectiv de implementarea proiectuli ales in java.

Pentru a reusi sa implementej in java am folosit mediul de dezvoltare IDE IntelliJ IDEEA 2021.3.2. Am ales sa folosesc acest ide doarece este cel mai prietenos si cel mai usor de folosit. Am ales sa folosesc limbajul java pentru implementarea proiectului deoarece este un limbaj foarte des intalnit intre programatori si este cel pe care il stiu cel mai bine. Pe de alta parte am ales java deoarece are framework-ul Java Swing cu care sunt destul de familiarizat, avand nevoie de el pentru a implementa interfata grafica. Recunos ca astfel am reusit sa invat foarte multe lucruri noi in acest limbaj.

# Descrierea diagramelor UML

Petru inceput o sa vorbesc despre diagrama cazurilor de utilizare, unde am doi actor, un angajat si un aministrator. La incept ei sunt nevoiti sa se logheze in aplicatie iar in caz de succes pentru administrator acsta trebuie sa poate prelua datele despre angajati din fiseruk XML dedicat, trebuie sa poata sa adauge un utilizator nou al aplicatiei, sa poata sa stearga un angajat din aceasta baza de date dar si sa actualizeze un angajat in aceasta baza de date. Astfel fiind nevoit sa fac operatii CRUD pe o baza de date stocata intr-un fisier XML, astfel am descoperit ca este cu mult mai usor sa lucrezi cu o baza de date stocata in mySQL decat in XML. In cazul in care are probleme la logare apar mici mesaje de notificare pentru a-i spune ce greseste.

In caz de succes pentru un angajat acesta poate sa faca mai multe operatii pe o baza de date salvata tot sub forma unui fisier XML, adica operatii de adaugare a unui nou parfum in aceasta baza de date, dar si stergera unui anumit parfum din aceasta baza de date da nu in ultimul rand si o actualizare a anumitor parfumuri. El mai poate sa vizualizeze toate parfumurile dintr-un magazin selectat, poate sa filtreze parfumurile dupa diverse criterii : producator, disponibilitate dar si pret, poate sa caute un anumit parfum dupa denumire dar poate si sa salveze liste de parfumuri in mai multe formate de exemplu csv sau json. Dupa cum se poate vedea in poza de mai jos.

Diagram

Description automatically generated

Mai departe o sa descriu diagrama UML realizata realizata dupa conceptul arhitectural MVP un model care faciliteaza testarea automata si separarea preocuparilor in logica de prezentare.

Astfel modelul defineste datele care trebuie utilizate in interfata cu utilizatorul. Prima clasa care apare aici este Parfume care contine numele parfumului si un obiect de tipul ParfumeInfo astfel reusind sa surprind toate proprietatile pe care le are un parfum. ParfumeCsvWriter este o interfata pe care o utilizez la salvarea rapoartelor in format .csv. Clasa ParfumeInfo care contine urmatoarele informatii despre un parfum: numele fabricantului, numarul de copii ramase pe stoc, codul de bare, pretul, cantitatea de parfum dintr-o sticluta dar si numarul de copii vandute. ParfumeJsonWriter este interfata cu care salvez rapoartele in formatul .json. ParfumeService este clasa care se ocupa de persistenta unui parfum adica apeleaza metode de salvare creare sau updatare ale unui parfum. Clasa abstracta ParfumeServiceInterface face legatura intre un magazin si baza de date, cu ajutorul ei citesc baza de date, adaug elemente, le sterg sau le actualizez. Clasa ParfumeStore contine ca si atribute numele unui magazin si o lista cu mai multe parfumuri care se afla in acel magazin. Acum urmeaza clasela pentru utilizatori, acestea sunt: User care are numele de utilizator, parola lui si rolul acestuia in lantul de magazine. Clasa UserService realizeaza persistenta unui utilizator si apeleaza metode din clasa abstracta pe care o extinde. Clasa abstracta UserServiceInterface este clasa care realizeaza legatura la baza de date si cu ajutorul ei pot sa salvez utilizatori noi sa sterg din ei dar si sa ii actualizez. Acestea sunt toate clasele din model.

Presentre-ul preia datele din model si le proceseza pentru a fi disponibile interfetei grafice. Acesta contine trei clase prima pe care o prezint este UserPresenter care realizeaza conexiunea dintre interfata grafica pentru angajat dar si interfata pentru administrator. Clasa EmployeePresenter realizeaza interacțiunea dintre modelul corespunzator pentru lantul de magazine de parfumuri și dintre interfața pentru angajat, iar AdministratorPresenter realizeaza interacțiunea dintre modelul corespunzator pentru utilizator și dintre interfața pentru administrator.

View-ul este interfata grafica a acestei aplicatii care afiseaza datele existente in model prin intermediul presenter-ului. Acest pachet contine trei interfete si 3 clase. Interfetele sunt IAdministratorView, IUserView, IemployeeView care fiegare din ele contin doar metode de get si set pentru clasele pe care le implementeaza. In clasa UserView se afiseaza interfața pt pagina de login prin numeroasele atribute prezente, și se gasesc metodele pentru butoanele care realizează comunicarea cu componenta din presenter corespunzatoare pagini de login. AdministratorView este clasa unde se afiseaza interfata pentru administrator unde se gasesc elementele de care are el nevoie si se implementeaza si butoanele de care are nevoie. Nu in ultimul rand clasa EmployeeView reprezinta interfața pt angajat prin numeroasele atribute prezente, și se gasesc metodele pentru butoanele care realizează comunicarea cu componenta din presenter corespunzatoare angajatului.

Diagram

Description automatically generated

# Descrierea aplicatiei

Graphical user interface

Description automatically generated with medium confidence

Aceasta ete fereastra de login unde utilizatorul se poate loga fie ca si administrator fie ca si angajat. Contine un camp unde se poate introduce un nume de utilizator cat si un camp pentru parola dar si butonul de login. Asfel utilizatorul dupa ce introduce aceste date la apasarea butonului de login se cauta dupa credentialele lui in aceasta baza de date.

Codul care stă în spatele acestei aplicații este unul destul de simplu și basic,am încercat să scriu cât mai clar și simplu toate operațiile menționate mai sus,am împărțit aplicația în clase specifice, am respectat modelul arhitectural MVP , respectiv modelul SOLID,pentru a fi mai simplu și eficient , a modifica sau a adăuga alte functionalități ,fără să se producă modificări majore liniilor de cod existente , ci doar să adăugăm alte linii.

Urmeaza mai jos sa pun celelate doau interfete grafice care fac referire la utilizatorul de tip administrator dar si utilizatorul de tip angajat. Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, email

Description automatically generated

Pot spune ca a fost a doua oara in viata mea cand am lucrat cu expresii landa in java pentru a realiza filtrarea dupa diverse criterii si chiar a fost destul de interesant. Am intelezi cum trebuie sa se modeleze viata reala datele din mai multe domenii . Principalul obiectiv al acestei teme este de a ne invata sa lucram cu expresii landa si cu MVP dar si respectarea metodologiei CRUD deoarece acest lucru ne poate usura foarte mult viata deoarece nu mai suntem nevoiti sa scriem asa de mult cod; sa recunoastem ca un programator bun va sti intodeauna sa isi refoloseasca codul pentru a creea o mai buna stabilitate peste timp a programului.