MINISTERUL EDUCAȚIEI NAȚIONALE

UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Facultatea de Automatică și Calculatoare

Departamentul de Calculatoare

Manager comenzi

Profesor îndrumător:

Dan Mitrea

Student:

Andriescu Antonio

Data: 19.04.2022

Grupa 30238

Cuprins

1. Obiectivul temei

Această temă are ca obiectiv principal crearea unui program de gestionare de comenzi. Programul afișează tabelul cu clienții, tabelul cu produsele si tabelul cu comenzile. Utilizatorul poate vizualiza, adaugă, șterge sau modifica clienți si produse. Pe lângă acestea utilizatorul poate adaugă comenzi asociate unui utilizator. Implementarea acestui program presupune analiza problemei, modelarea structurilor de date inc are sunt stocate informațiile, implementarea propriu-zisa a programului si a interfeței grafice.

2. Analiza problemei, modelare, scenarii, cazuri de utilizare

Pentru a asigura persistenta datelor am utilizat baza de date MySQL. MySQL este un sistem de gestiune a bazelor de date relaționale produs de compania suedeza MYSQLAB care se bazează pe limbajul de interogare structurat SQL. Cu MySQL se pot construi aplicații in orice limbaj major. Pentru vizualizarea tabelelor si a bazei de date am utilizat aplicația MySQL Workbench. MySQL Workbench este un instrument vizual de proiectare a bazelor de date care integrează dezvoltarea, proiectarea, crearea si întreținerea bazei de date SQL intr-un singur mediu de dezvoltare integrat pentru sistemul de baze de date MYSQL.

Pentru instantierea unei singure conexiuni la baza de date am utilizat șablonul de proiectare corelațional Singleton. Singleton îmi permite sa mă asigur ca o clasa are o singura instanță. Oferind in același timp un punct de acces global la aceasta instanță.

Utilizatorul poate folosi acest program pentru gestionarea unei baze de date cu clienți, produse si comenzi. Se pot face comenzi pentru clienții existenți cu un anumit produs si o anumita cantitate, in limita stocului disponibil. Prețul final este calculat in funcție de prețul produsului si cantitatea selectata. Ca funcționalități, programul trebuie sa:

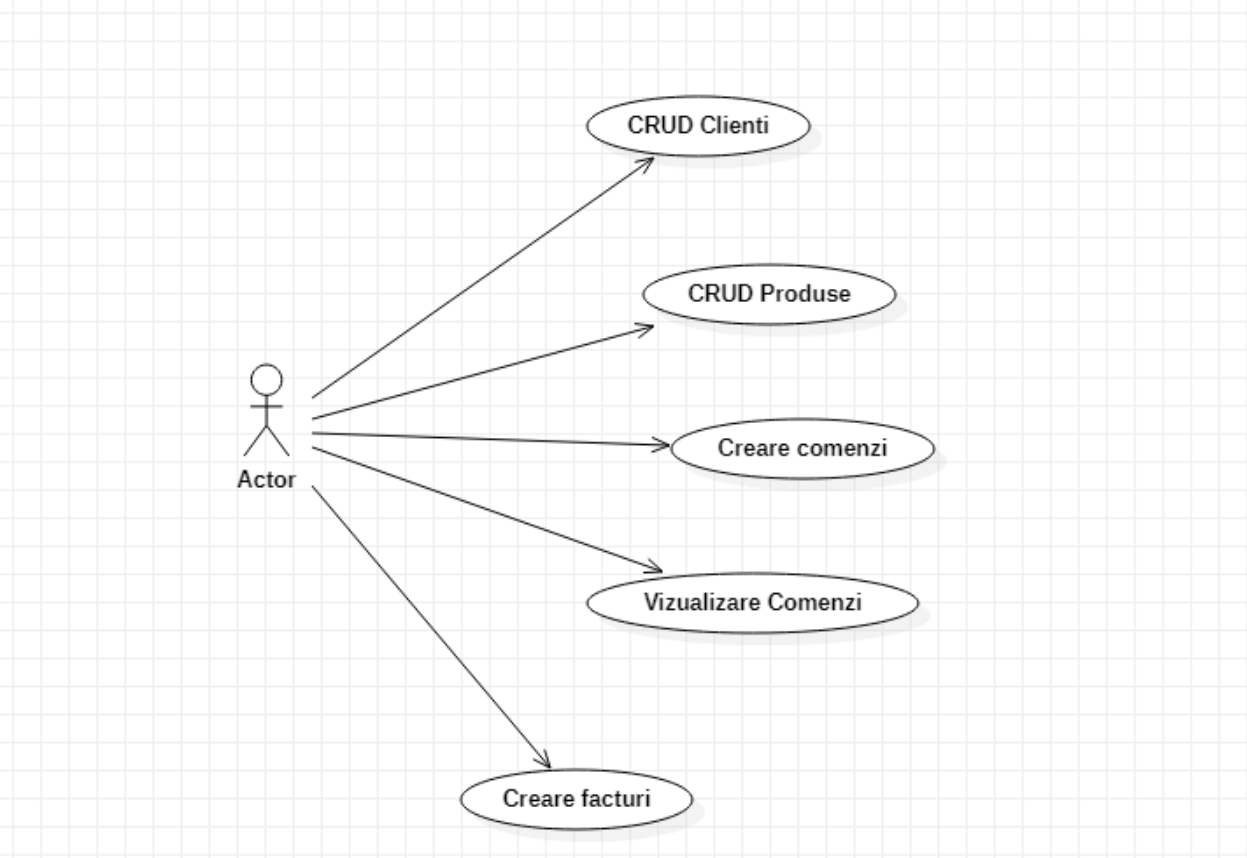
- Ii permită utilizatorului sa poată insera clienți noi
- Ii permită utilizatorului sa poată șterge clienți existenți
- Ii permită utilizatorului sa poată modifica date despre clienți existenți
- Ii permită utilizatorului sa poată vizualiza clienții existenți
- Ii permită utilizatorului sa poată insera produse noi
- Ii permită utilizatorului sa poată șterge produse existente
- Ii permită utilizatorului sa poată modifica date despre produse existente
- Ii permită utilizatorului sa poată vizualiza produsele existente
- Ii permită utilizatorului sa poată crea comenzi noi
- Ii permită utilizatorului sa poată vizualiza comenzile existente
- Ii permită utilizatorului sa poată modifica stocul
- Ii permită utilizatorului sa poată genera facturi pentru fiecare comanda

Un scenariu de utilizare ar fi:

- Utilizatorul deschide fereastra cu clienți, fereastra cu comenzi si fereastra cu clienți
- Utilizatorul introduce id-ul unui client in fereastra cu comenzi
- Utilizatorul introduce id-ul unui produs in fereastra cu comenzi
- Utilizatorul introduce cantitatea dorita in fereastra cu comenzi
- Utilizatorul apasă pe butonul „Adăugare Comanda” pentru a crea comanda
- Utilizatorul selectează comanda si apasă pe butonul „Creare raport” pentru a genera factura

Programul implementat ar putea fi utilizat intr-un magazin pentru creare de comenzi pentru clienți. Angajații sunt cei care vor selecta produsele si clienții pentru a genera comanda, si vor putea genera o factura pe care i-o pot da

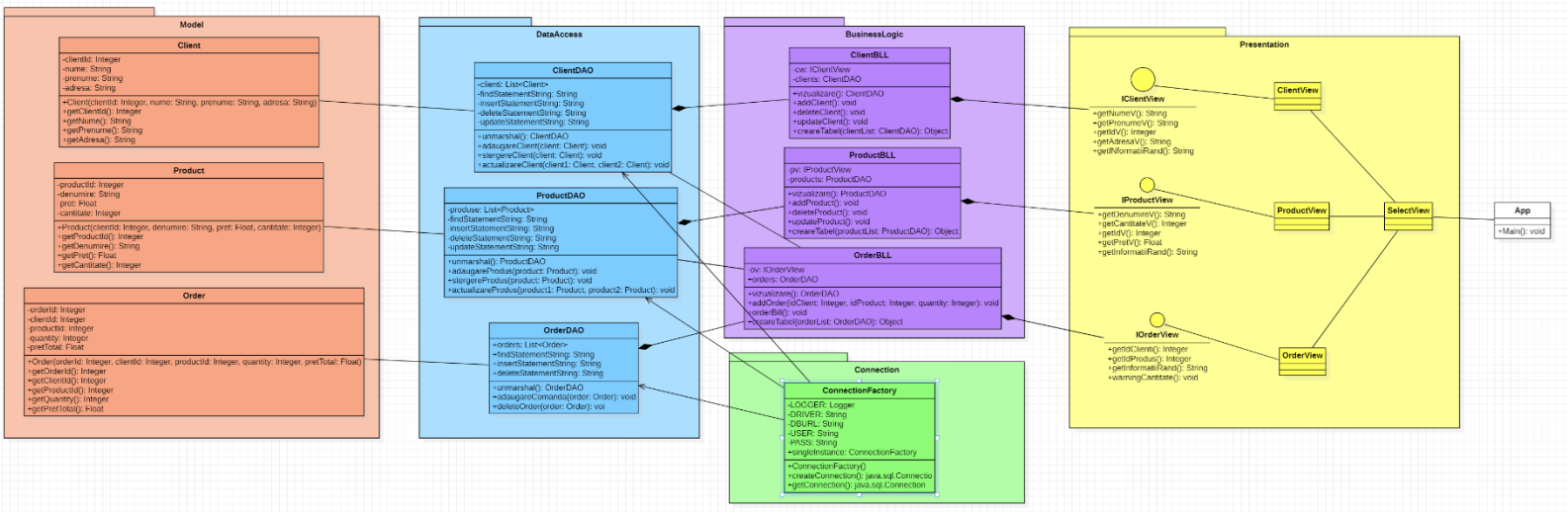
clientului care a făcut achiziția. Aceasta aplicația poate face mai ușoară gestionarea stocului unui magazin, cat si gestionarea comenzilor.



Din diagrama cazurilor de utilizare a aplicatiei se pot obsera cazurile de utilizare ale programului:

- Operatii CRUD pe clienti
- Operatii CRUD pe produse
- Crearea de comenzi
- Vizualizarea comenzilor existente
- Crearea de facturi pentru fiecare comanda

3. Proiectare



Proiectul este împărțit in cinci pachete, respectând șablonul arhitectural „Layered Architecture”:

Pachetul Model conține clasele „Client”, „Order”, „Product”. Acestea modelează structurile de date folosite in realizarea programului si corespund cu tabelele cu acelasi nume din baza de date.

Pachetul Connection conține clasa „ConnectionFactory”, clasa care este responsabila pentru realizarea conexiunii dintre aplicație si baza de date. Tot in aceasta clasa am implementat șablonul de proiectare corelațional Singleton.

Pachetul DataAccess conține clasele „ClientDAO”, „ProductDAO” si „OrderDAO” care sunt responsabile cu operațiile de vizualizare, adăugare, ștergere si actualizare pe tabelele client, product respectiv orders din baza de date aferenta programului.

Pachetul BusinessLogic conține clasele „ClientBLL”, „ProductBLL” si „OrderBLL” care fac legătura intre operațiile pe tabele/liste si interfața grafica a aplicației.

Pachetul Presentation conține clasele „ClientView”, „ProductView”, „OrderView”, „IClientView”, „IProductView” si „IOrderView”. In primele trei am implementat interfața grafica a utilizatorului, iar cele din urma le folosesc pentru a putea apela metodele din primele trei in alte contexte.

4. Implementare

4.1. Client

In clasa Client am modelat obiectul client. Acesta are patru atribute: clientID (id-ul clientului), nume (numele clientului), prenume (prenumele clientului) si adresa (adresa clientului). In aceasta clasa am cinci metode. Prima metoda este constructorul clasei prin intermediul căreia putem instanția un obiect instanță al clasei care primește ca parametri id-ul clientului, numele clientului, prenumele clientului si adresa clientului. Celelalte patru metode sunt getter-e pentru fiecare atribut al obiectului.

4.2. Product

In clasa Product am modelat obiectul product. Acesta are patru atribute: productId (id-ul produsului), denumire (denumirea produsului), preț (prețul produsului) si cantitate (cantitatea de produs disponibila in stoc). In aceasta clasa am cinci metode. Prima metoda este constructorul clasei prin intermediul caruia putem instantia un obiect instanță al clasei care primește ca parametri id-ul produsului, denumirea produsului, prețul produsului si cantitatea disponibila in stoc a produsului. Celelalte patru metode sunt getter-e pentru fiecare atribut al obiectului.

4.3. Order

In clasa Order am modelat obiectul order. Aceasta are cinci atribute: orderId (id-ul comenzii), clientId (id-ul clientului care a făcut comanda), productId (id-ul produsului care a fost comandat), quantity (cantitatea de produs comandata) si preTotal (prețul total al comenzii). In aceasta clasa avem 6 metode. Prima metoda este constructorul clasei prin intermediul căreia putem instanția un obiect instanță al clasei care primește ca parametri id-ul comenzii, id-ul clientului, id-ul produsului, cantitatea de produs si prețul total al comenzii. Celelalte 4 metode sunt getter-e pentru fiecare atribut al obiectului.

4.4. ClientDAO

Clasa ClientDAO este responsabila pentru efectuarea operațiilor asupra tabelului „client” al bazei de date. Aceasta clasa are ca atribut o lista de obiecte de tip Client. Metodele „getClienti()” si „setClienti” sunt getter-ele si setter-ele listei de obiecte. Operațiile CRUD asupra tabelului sunt implementate in următoarele metode:

- Unmarshal() – in aceasta metoda se populează lista de obiecte de tip Client luând toate datele din tabelul client din baza de date
- adaugareClient() – aceasta metoda este responsabila pentru inserarea unui client nou in tabel. Parametrul functiei este clientul pe care dorim sa îl adăugam si acesta trebuie sa fie un obiect de tipul Client
- stergereClient() – aceasta metoda este responsabila de ștergerea unui client din tabel. Parametrul funcției este clientul pe care dorim sa îl ștergem si acesta trebuie sa fie un obiect de tipul Client
- actualizareClient() – aceasta metoda este responsabila pentru modificarea datelor despre un client din tabel. Parametrii funcției sunt doua obiecte de tipul Client, primul reprezentând clientul ale cărui date dorim sa le modificam, iar al doilea este obiectul cu datele cu care dorim sa modificam clientul.

4.5. ProductDAO

Clasa ProductDAO este responsabila pentru efectuarea operațiilor asupra tabelului „product” al bazei de date. Aceasta clasa are ca atribut o lista de obiecte de tip Product. Metodele „getProducts()” si „setProducts” sunt getter-ele si setter-ele liste de obiecte. Operațiile CRUD asupra tabelului sunt implementate in următoarele metode:

- unmarshal() – in aceasta metoda se populează lista de obiecte de tip Product luând toate datele din tabelul product din baza de date
- adaugareProdus() – aceasta metoda este responsabila pentru inserarea unui produs nou in tabel. Parametrul functiei este produsul pe care dorim sa îl adăugam si acesta trebuie sa fie un obiect de tipul Product
- stergereProdus () – aceasta metoda este responsabila de ștergerea unui produs din tabel. Parametrul funcției este produsul pe care dorim sa îl ștergem si acesta trebuie sa fie un obiect de tipul Product
- actualizareProdus () – aceasta metoda este responsabila pentru modificarea datelor despre un produs din tabel. Parametrii funcției sunt doua obiecte de tipul Product, primul reprezentând produsul ale cărui date dorim sa le modificam, iar al doilea este obiectul cu datele cu care dorim sa modificam produsul.

In aceasta clasa am mai implementat metoda acutalizareCantitate() cu ajutorul căreia pot actualiza cantitatea din stoc a unui produs. Parametrii metodei sunt produsul căruia dorim sa ii modificam stocul si cantitatea cu care dorim sa decrementam stocul.

4.6. OrderDAO

Clasa OrderDAO este responsabila pentru efectuarea operațiilor asupra tabelului „orders" al bazei de date. Aceasta clasa are ca atribut o lista de obiecte de tip Order. Metodele „getOrders()" si „setOrders" sunt getter-ele si setter-ele liste de obiecte. Operațiile asupra tabelului sunt implementate in următoarele metode:

- unmarshal() – in aceasta metoda se populează lista de obiecte de tip Order luând toate datele din tabelul order din baza de date
- adaugareComanda() – aceasta metoda este responsabila pentru inserarea unei comenzi noi in tabel. Parametrul functiei este comanda pe care dorim sa o adăugăm si aceasta trebuie sa fie un obiect de tipul Order
- deleteOrder () – aceasta metoda este responsabila de ștergerea unei comenzi din tabel. Parametrul funcției este comanda pe care dorim sa o ștergem si aceasta trebuie sa fie un obiect de tipul Order

4.7. ClientBLL

Clasa ClientBLL este responsabila cu legarea clasei ClientDAO cu interfața grafica a aplicației. Prin aceasta clasa facem operații CRUD pe tabelul „client" utilizând datele din interfața grafica. Aceasta clasa are ca atribute un obiect de tip IClientView care ne permite sa utilizam metodele din clasa ClientView si un obiect de tipul ClientDAO (clients). Prima metoda implementata este constructorul clasei ClientBLL (ClientBLL()) .Operațiile sunt implementate in următoarele metode:

- vizualizare() – in aceasta metoda se populează lista de obiecte clients cu toți clienții prezenți in tabelul „client"
- addClient() – aceasta metoda este responsabila cu preluarea datelor introduse de utilizator in interfata garfica, crearea unui obiect de tipul Client folosind aceste date si de adăugarea clientului in tabel apelând metoda corespunzătoare din ClientDAO.
- deleteClient() – aceasta metoda este responsabila cu preluarea datelor despre un client selectat de către utilizator din interfața grafica, crearea obiectului de tip Client utilizând acele date si ștergerea clientului din tabel apelând metoda corespunzătoare din ClientDAO
- updateClient() – aceasta metoda este responsabila cu preluarea datelor despre un client selectat de către utilizator din interfața grafica si modificarea acestora cu datele introduse de către utilizator.
- creareTabel() – aceasta metoda este responsabila cu transformarea listei de obiecte de tip Client într-o matrice de tip Object. Aceasta este utilizata pentru crearea structurii de date ce trebuie introdusa in tabelul din interfața grafica.

4.8. ProductBLL

Clasa ProductBLL este responsabila cu legarea clasei ProductDAO cu interfața grafica a aplicației. Prin aceasta clasa facem operații CRUD pe tabelul „product" utilizând datele din interfața grafica. Aceasta clasa are ca atribute un obiect de tip IProductView care ne permite sa utilizam metodele din clasa ProductView si un obiect de tipul ProductDAO (products). Prima metoda implementata este constructorul clasei ProductBLL .Operațiile sunt implementate in următoarele metode:

- vizualizare() – in aceasta metoda se populează lista de obiecte products cu toate produsele prezente in tabelul „product"

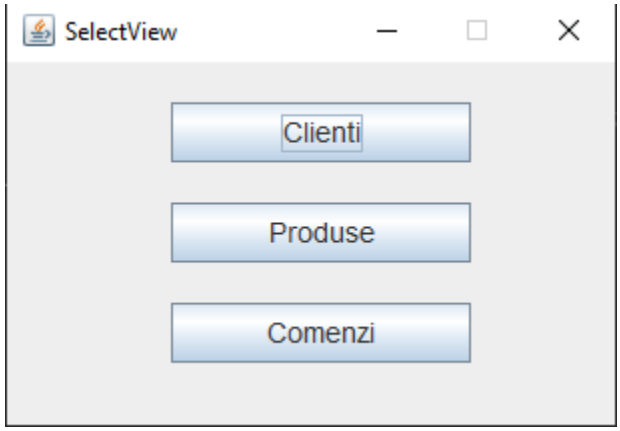
- addProduct() – aceasta metoda este responsabila cu preluarea datelor introduse de utilizator in interfața grafică, crearea unui obiect de tipul Product folosind aceste date si de adăugarea produsului in tabel apelând metoda corespunzătoare din ProductDAO.
- deleteProduct() – aceasta metoda este responsabila cu preluarea datelor despre un produs selectat de către utilizator din interfața grafica, crearea obiectului de tip Product utilizând acele date si ștergerea produsului din tabel apelând metoda corespunzătoare din ProductDAO
- updateProduct() – aceasta metoda este responsabila cu preluarea datelor despre un produs selectat de către utilizator din interfața grafica si modificarea acestora cu datele introduse de către utilizator.
- creareTabel() – aceasta metoda este responsabila cu transformarea listei de obiecte de tip Product într-o matrice de tip Object. Aceasta este utilizata pentru crearea structurii de date ce trebuie introdusa in tabelul din interfața grafica.

4.9. OrderBLL

Clasa OrderBLL este responsabila cu legarea clasei OrderDAO cu interfața grafica a aplicației. Prin aceasta clasa facem operații CRUD pe tabelul „orders" utilizând datele din interfața grafica. Aceasta clasa are ca atribute un obiect de tip IOrderView care ne permite sa utilizam metodele din clasa OrderView si un obiect de tipul OrderDAO (orders). Prima metoda implementata este constructorul clasei OrderBLL .Operațiile sunt implementate in următoarele metode:

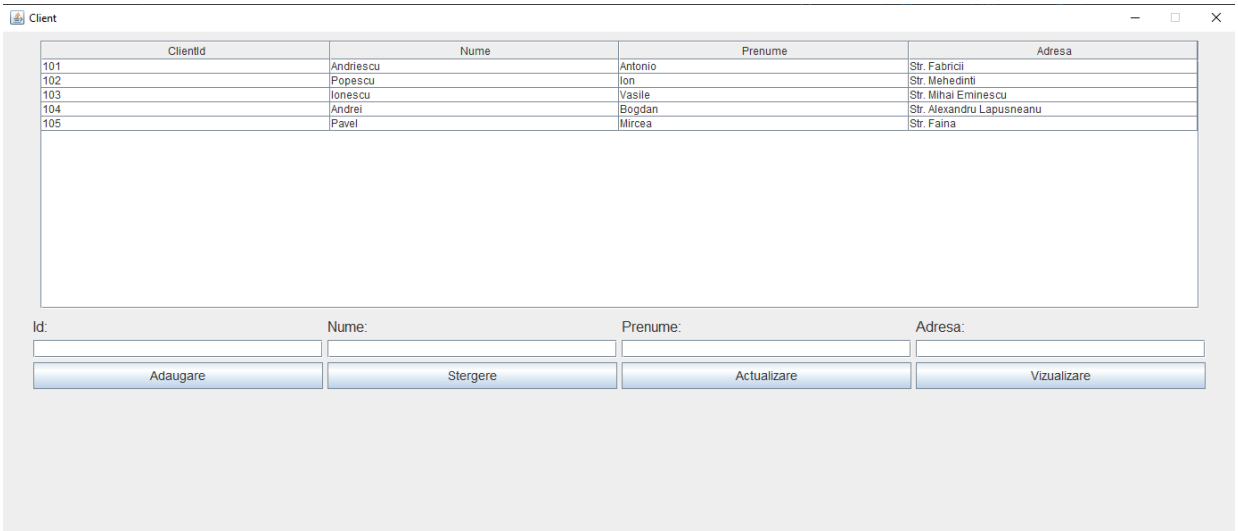
- vizualizare() – in aceasta metoda se populează lista de obiecte orders cu toate produsele prezente in tabelul „orders"
- addOrder () – aceasta metoda este responsabila crearea si inserarea unei comenzi in baza de date.
- orderBill() – in aceasta metoda se creează factura pentru o anumita comanda. Se preiau datele din OrderView de la comanda selectata de către utilizator si se generează un fișier pdf.
- creareTabel() – aceasta metoda este responsabila cu transformarea listei de obiecte de tip Product într-o matrice de tip Object. Aceasta este utilizata pentru crearea structurii de date ce trebuie introdusa in tabelul din interfața grafica.

4.10. SelectView



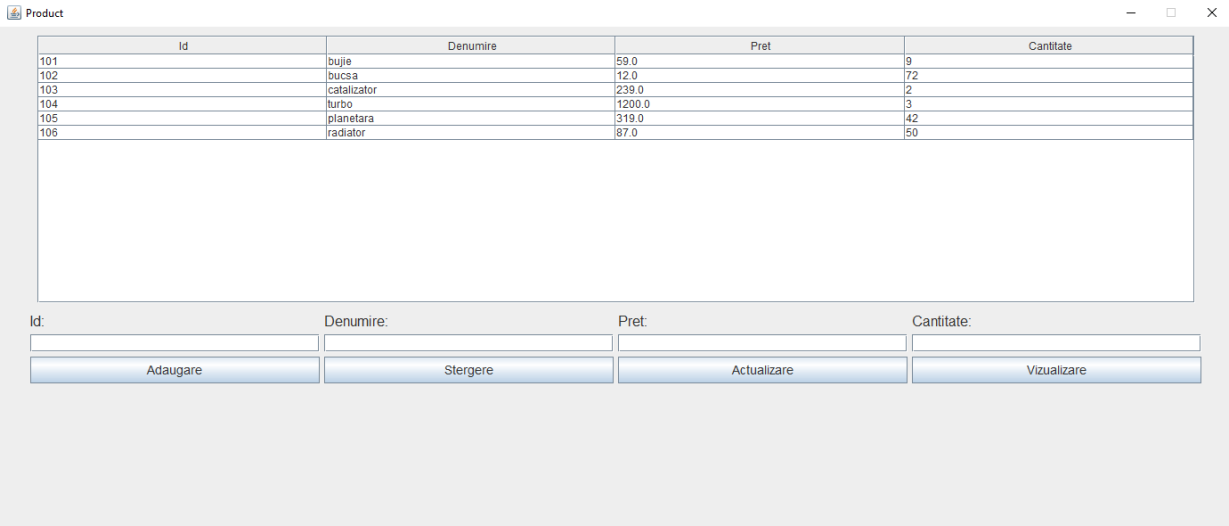
In clasa SelectVIew am implementat interfața grafica prin care utilizatorul selectează ce ferestre sa se deschidă. La apăsarea unuia dintre cele 3 butoane se va deschide fereastra corespunzătoare.

4.11. ClientView



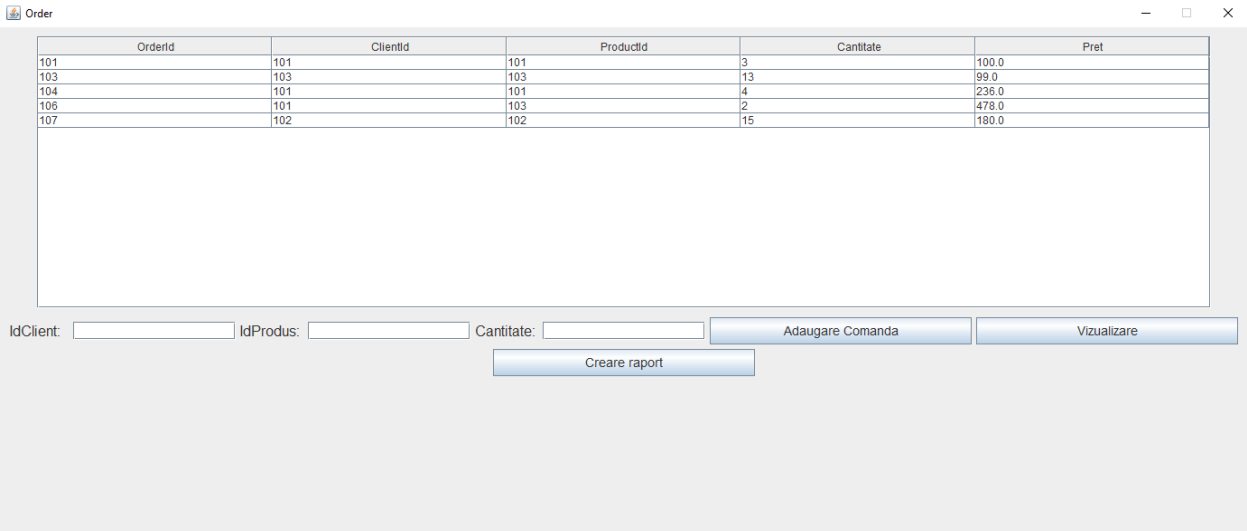
In clasa ClientView am implementat interfața grafica pentru operațiile CRUD pe clienți. In aceasta pentru a se putea vedea tabelul cu clienți utilizatorul trebuie sa apese pe butonul „Vizualizare”. Pentru a adăuga un client, utilizatorul trebuie sa completeze căsuțele corespunzătoare cu datele clientului apoi sa apeleze pe butonul „Adaugare”. Pentru ștergerea unui client, utilizatorul trebuie sa selecteze clientul pe care dorește sa îl șteargă, dând click pe rândul respectiv, apoi sa apese pe butonul „Stergere”. Pentru actualizarea unui client, utilizatorul trebuie sa selecteze clientul pe care dorește sa îl modifice si sa completeze câmpurile cu datele pe care dorește sa le modifice. Pentru a vedea modificările aduse asupra tabelului trebuie apăsat butonul „Vizualizare”

4.12. ProductView



In clasa ProductView am implementat interfața grafica pentru operațiile CRUD pe produse. Aceasta interfața funcționează exact ca cea din ClientView.

4.13. OrderView



In clasa OrderView am implementat interfața grafica pentru operațiile asupra comenzilor. In aceasta interfața putem apăsa pe butonul „Vizualizare” pentru a putea vedea tabelul cu comenzile. Pentru a adăuga o comanda completam câmpurile cu datele necesare comenzii si apăsam butonul „Adăugare Comanda”. Pentru a crea o factură pentru o comanda, utilizatorul trebuie sa selecteze comanda pentru care dorește sa creeze factura si sa apese pe butonul „Crearea Raport”. Pentru a vedea modificările aduse asupra tabelului trebuie apăsat butonul „Vizualizare”.

5. Concluzii

Realizarea acestei teme m-a familiarizat cu șablonul arhitectural "Layered Architecture" și cu folosirea bazei de date MySQL, cât și a Workbench-ului. Am folosit pentru prima dată metoda de generare a unui fișier pdf din Java și am reușit să implementez toate funcționalitățile necesare aplicației.

6. Bibliografie

[1] https://users.utcluj.ro/~igiosan/teaching_poo.html

[2] https://dsrl.eu/courses/pt/

[3] https://gitlab.com/utcn_dsrl/pt-layered-architecture

[4] https://gitlab.com/utcn_dsrl/pt-reflection-example

[5] https://www.baeldung.com/java-pdf-creation