MINISTERUL EDUCAȚIEI NAȚIONALE
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

Facultatea de Automatica si Calculatoare

Tema 3:
Sistem de gestiune a bazei de date a unui depozit

Disciplina: Tehnici de programare

Student:
Coman Vasile
An II
Grupa 30221

Profesor coordonator:
Antal Marcel
Profesor curs:
Ioan Salomie

Cuprins

1.Obiectivul temei

Obiectivul principal al temei este cel de a implementa si gestiona baza de date a unui depozit. Lucrul acesta il putem face prin urmatoarele etape:

-creare bazei de date;

-conexiune la baza de date;

-impartirea programului in pachete corespunzatoare: data acces layer, business layer, presentation layer, model;

-folosirea de JTable;

-utilizarea tehnicilor de reflaction.
Toate aceste lucruri vor fi detaliate in urmatoarele capitole ale documentatiei.

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

Aceasta aplicatie va putea adauga, modifica, sterge si afisa clientii si produsele. Un client va putea selecta un produs pe care va dori sa-l achizitioneze si cantitatea acestuia si va putea genera comanda in acelasi timp se va emite un bon intr-un fisier text.

Am folosit validatori pentru email, pentru cantitate si pret care nu vor putea fi negative de asemenea utilizatorului i se va afisa mesaj daca cantitatea pe care doreste sa o comanda este mai mare decat cea disponibila in stoc.

Toate celelalte date pe care utilizatorul le va introduce se presupune ca vor fi valide si nu vor putea fi nule.

Modificare, stergerea si afisarea dupa id se vor face dupa un id dat de utilizator care se presupune ca fiind valid.

3.Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator, modul de tratare a erorilor)

Baza de date este formata din 6 tabele:

-tabela client care contine date despre client(id, nume, adresa, email);

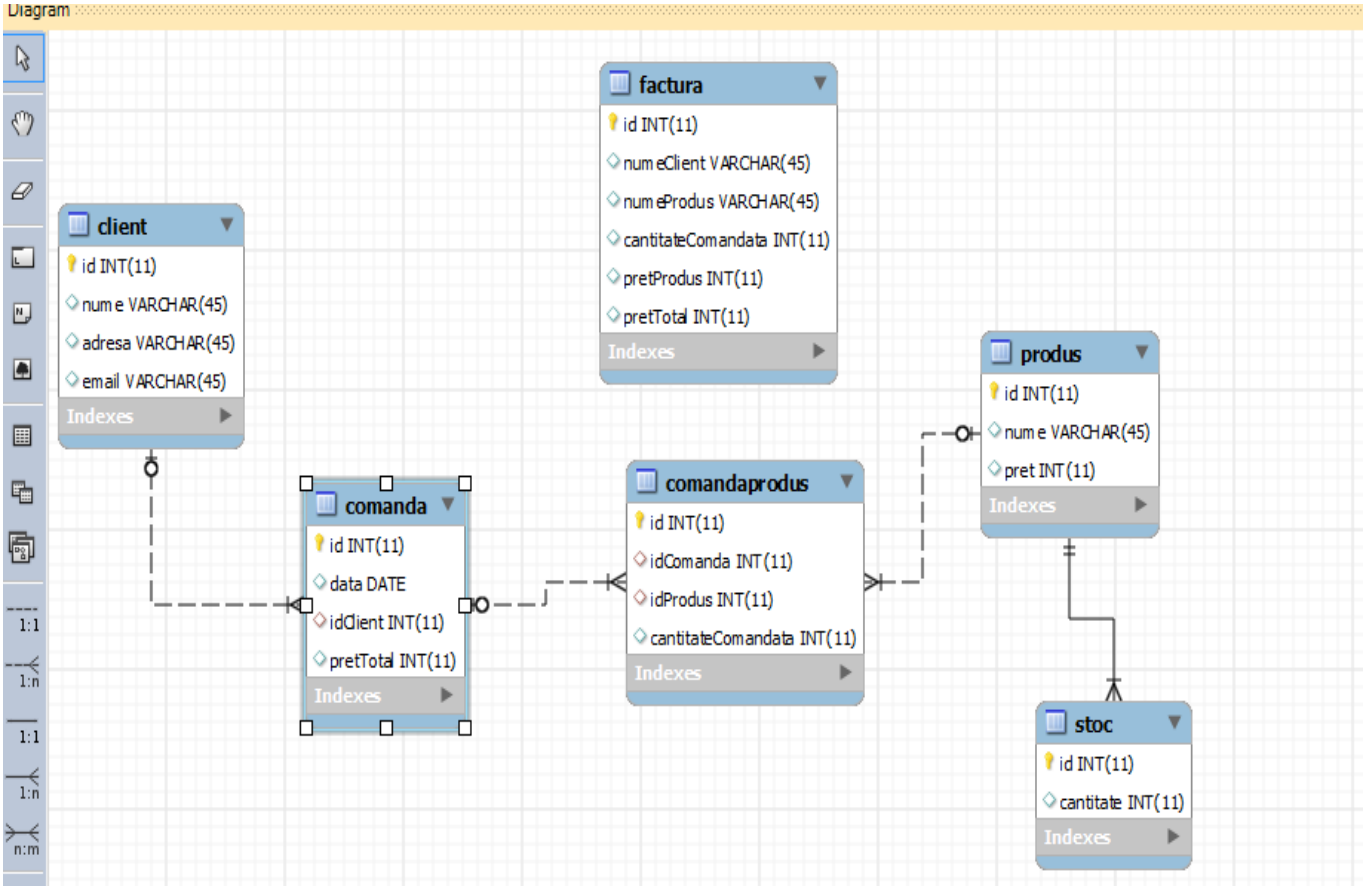
-tabela produs contine date drespre un produs(id, nume, pret);

-un produs are un stoc(id, cantitate);

-tabela comanda(id, idClient, id, pretTotal, data)

-tabela comandaprodus care face legatura dintre comanda si produs(id, idProdus, idComanda, cantiateComandata);

-tabela factura care contine date despre o comada(data comenzii, cine a comandat, ce produs si valoarea totala).



Am ales sa impart proiectul in 7 pachete.

Primult pachet: model care contine clasele Client, Comanda, Produs, ComandaProdus, Stoc si Factura. Fiecare clasa contine variabile instanta corespunzatoare cu numele campurilor din fiecare tabel din baza de data.

Al doilea pachet: connection contine clasa ConnectionFactory care realizeaca conexiunea la baza de date.

Al treilea pachet: dao face accesul la datele din baza de date, contine clasa AbstractDAO unde ce creeaza queryurile de inseare, adaugare, stergere si selectare si se executa aceste queryuri returnand rezultatul acestor queryuri. Clasele ClientDAO, ComandaDAO, ComandaProdusDAO, StocDAO, ProdusDAO si FacturaDAO care extind clasa

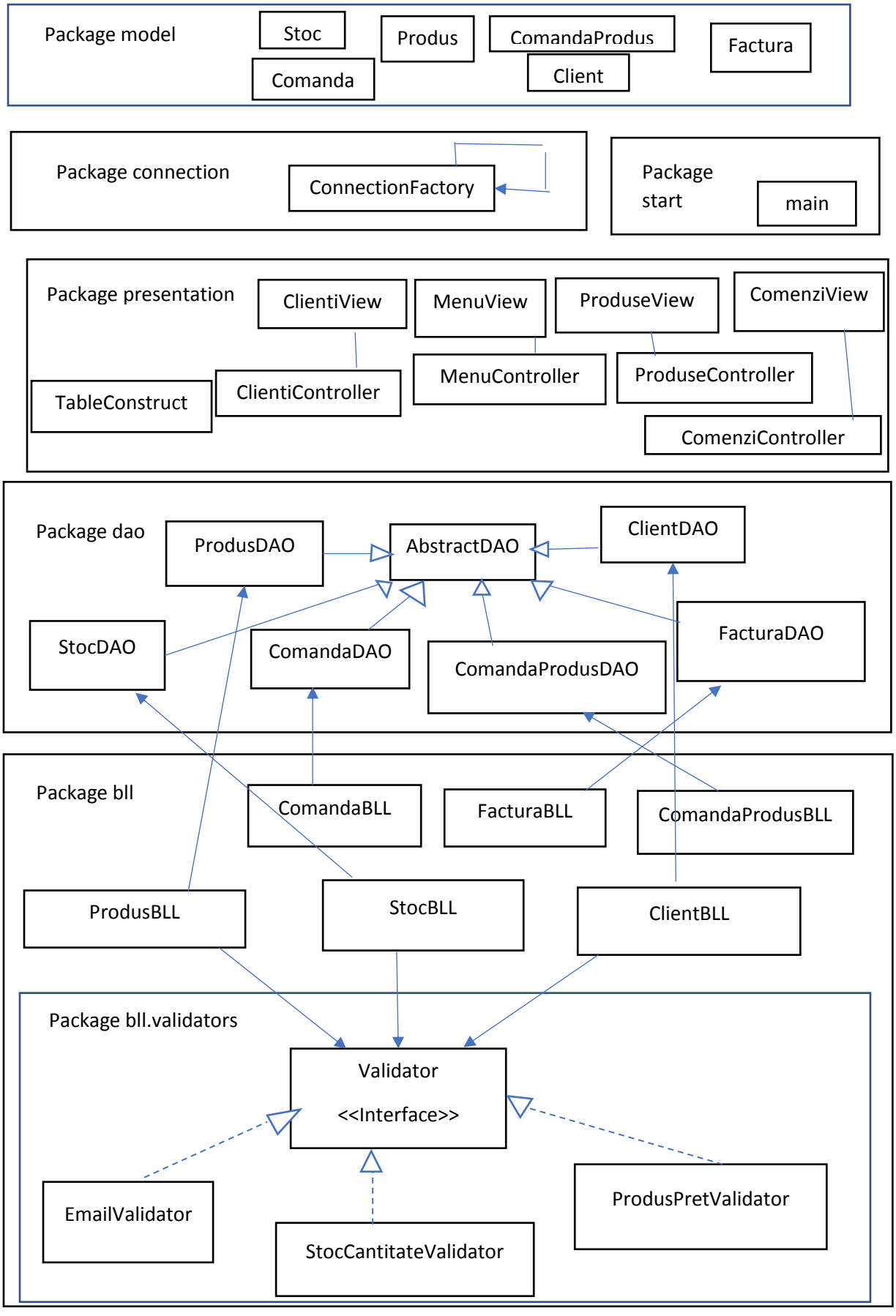
AbstractDAO. In aceste clase vom putea adauga queryuri specifice fiecarui tabel, cele generale sunt implementate in AbstractDAO.

Al patrulea pachet: bll contine clasele ClientBLL, ComandaBLL, ComandaProdusBLL, FacturaBLL, ProdusBLL, StocBLL care contine variabile instante obiecte de tip DAO si validatori care pot apela metodele de insert, update, delete, select.

Al cincilea pachet: bll.validators contine clase ce implementeaza interfata Validator care contine medota validate care arunca exceptie in caz ca dorim sa inseram un produs care nu corespunde cerintelor validatorilor.

Al saselea pachet: presentation contine clasele de interfata grafica dupa modelul controller-view pentru clienti, produse si comenzi si meniu si clasa ce creeaza un obiect de tip JTable folosind o lista de obiecte.

Al saptelea pachet: start contine clasa main unde apelam frame-ul de meniu.


Package model
Stoc
Produs
ComandaProdus
Factura
Comanda
Client
Package connection
ConnectionFactory
Package start
main
Package presentation
ClientiView
MenuView
ProduseView
ComenziView
TableConstruct
ClientiController
MenuController
ProduseController
ComenziController
Package dao
ProdusDAO
AbstractDAO
ClientDAO
StocDAO
ComandaDAO
ComandaProdusDAO
FacturaDAO
Package bll
ComandaBLL
FacturaBLL
ComandaProdusBLL
ProdusBLL
StocBLL
ClientBLL
Package bll.validators
Validator
<<Interface>>
EmailValidator
StocCantitateValidator
ProdusPretValidator

4.Implementare

Clasele din pachetul model Client, Factura, Produs, ComandaProdus, Comanda si Stoc au campurile corespunzatoare campurilor din tabel. Aceste clase contin doar metode de get si set si constructori.

Clasa din pachetul ConnectionFactory creaza conexiune la baze de date astfel face legatura din aplicatie si baza de date din MySQL.

Clasele din pachetul dao- data access care permit accesul la datele din baza de date. Clasa AbstractDAO contine metode de construire a queryurilor de insert, update, delete, select.

De asemenea contine metode care executa queryurile si o metoda de creare a rezultatului queryului prin crearea unei liste de obiecte. Restul claselor din pachetul dao permit scrierea de queryuri specifice fiecarui tabel, cele generale fiind descrie in clasa AbstractDAO.

Pachetul bll contine logica aplicatiei formata din clasele: ClientBLL, ProdusBLL, ComandaBLL, ComandaProdusBLL, StocBLL, FacturaBLL.

Fiecare clasa din acest pachet are urmatoarele metode:

-o metoda care gaseste un obiect dupa un id dat unde un obiect de tip dao apeleaza metoda findById din AbstractDAO;

-o metoda care returneaza toate obiectele unde un obiect de tip dao apeleaza medota findALL din AbstractDAO;

-o metodd care insereaza un obiect apeland metoda de insert din AbstractDAO;

-o metoda care sterge un obiect apeland delete din AbstractDAO;

-o metoda care modifica un obiect apeland update din AbstractDAO.

De asemenea in unele clase se gasesc validatori pentru a verifica inainte de inserare daca datele introduse corespund cerintelor.

Pachetul bll.validators contine interfata Validator care contine metoda neimplementata validate. Clasa EmailValidator verifica daca emailul introdus este valid in caz contrar va arunca o exceptie. Clasa ProdusPretValidator nu permite utilizatorului sa introduca un pret negativ unui produs. Si clasa StocCantitateValidator nu permite utilizatorului sa introduca o cantitate negativa unui produs.

In pachetul de presentation se aflasa clasele pentru generarea interfetei grafice. Fiecare frame a fost implementat dupa modelul de constructie controller-view unde clasa view construieste frame-ul iar clasa controller implementeaza interfata ActionListener si suprascrie metoda actionPerformed.

Clasa MenuView extinde clasa JFrame si MenuController implementeaza interfata ActionListener. Aceste clase construiesc primul frame in care utilizatorul poate sa aleaga ce frame sa deschida dintre Comenzi, Produse si Clienti.

Clasa ClientiView si clasa ClientiController creeaza cel de-al doilea frame si permit utilizatorului sa introduca un client, sa afiseze toti clientii intr-un JTable, sa afiseze clientii dupa un anumit id dat intr-un textfield, sa modifice un client si sa stearga un client.

Aici utilizatorului i se va verifica daca a introdus un email valid altfel se va afisa mesaj.

Clasa ProduseView si clasa ProduseController creeaza cel de-al treilea frame si utilizatorul poate de asemenea sa introduca un produs, sa modifice un produs, sa afiseze toate produsele, sa afiseze un produs dupa un id dat din interfata si sa stearga un produs. Tot aici se va verifica daca pretul introdus este pozitiv in caz contrar se va afisa mesaj si de ase daca cantitate introdusa in stoc este pozitiva.

Clasa ComenziView si clasa ComenziController creeaza ultimul frame iar utilizatorul poate sa introduca o comanda, sa vada toate comenzile si sa vada o anumita comanda dupa id.

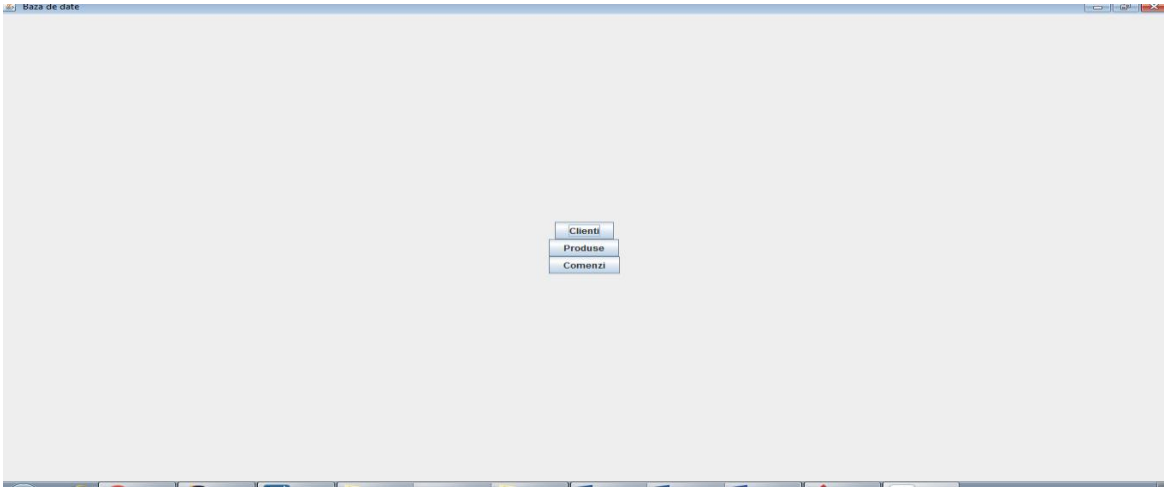
Se verifica daca cantitatea dorita care urmeaza a fi comandata este mai mica decat cantitatea din stocul produsului. Si tot aici avem o metoda write care scrie intr-un fisier un bon care arata date despre comanda.

O clasa mai speciala este cea TableConstruct care extinde DefaultTableModel. Aceasta clasa are metoda createTable care primeste o lista de obiecte. Aceasta metoda parcurge fiecare camp al listei de obiecte si adauga capurile de tabel, adica numele campurilor din tabelul din baza de date trimis ca si lista de obiecte. Apoi parcurgem lista de obiecte si salvam intr-un vector iar la final adaugam acel vector ca si linie la tabel si in final modelul este creat.

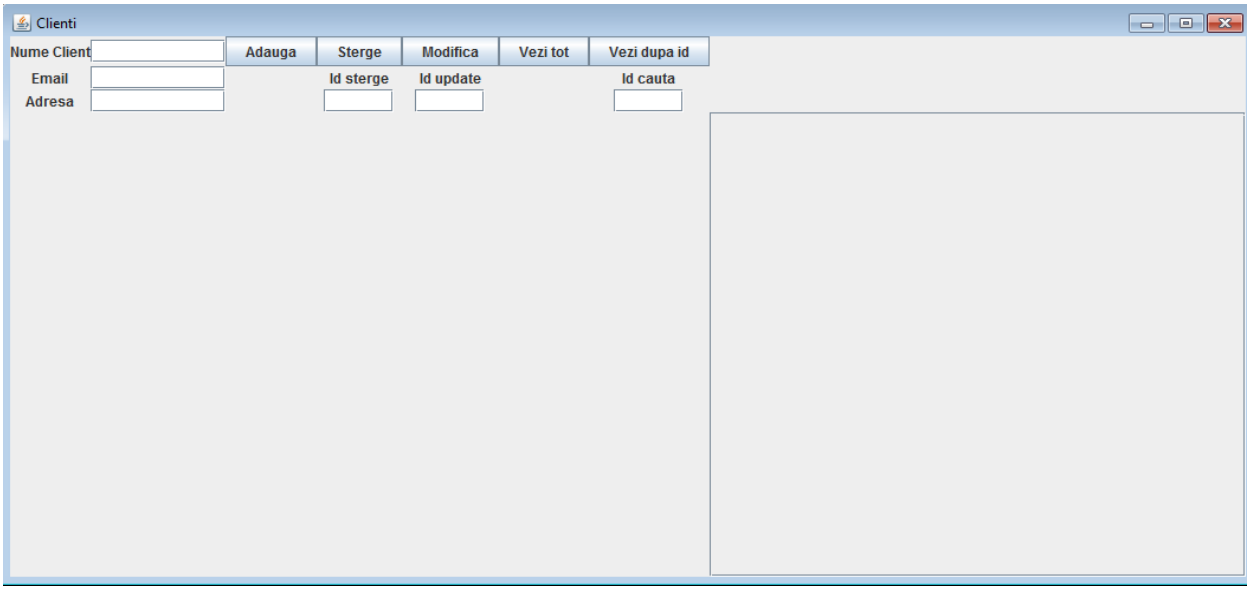
In pachetul start avem clasa Main care apeleaza fereastra principala adica cea de meniu din care se vor deschide urmatoarele frame-uri.

5.Rezultate

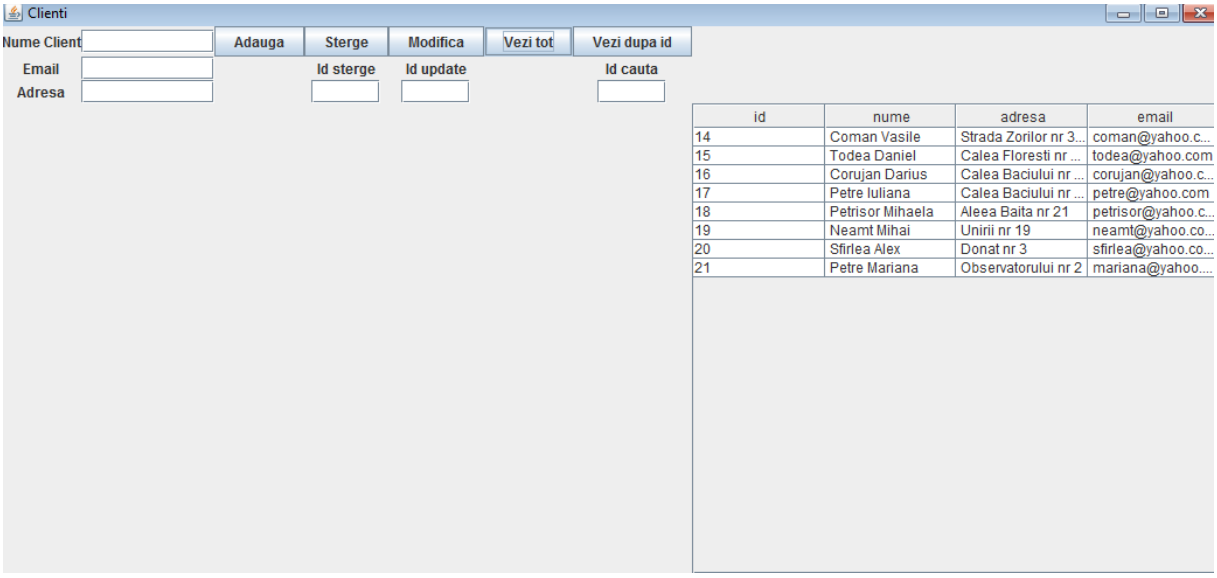
La rularea aplicatiei utilizatorului ii va aparea un meniu pentru a alege ce frame sa se deschida in continuare.



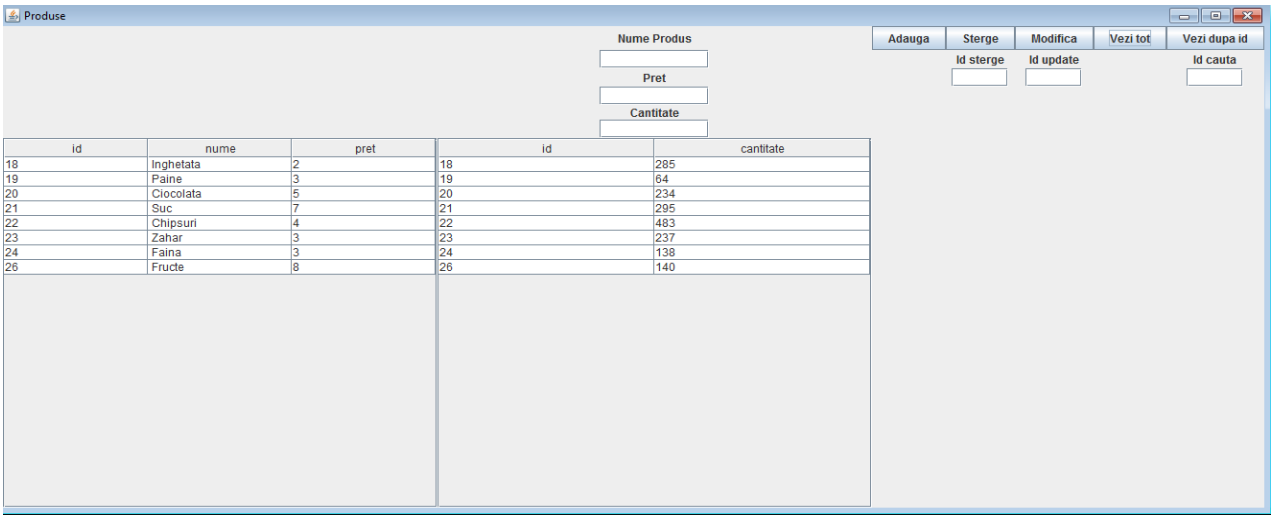
Daca selecteaza butonul clienti ii va aparea urmatoarea fereastra:



El va putea introduce un client adaugand datele in cele 3 textfield-uri, va putea sterge modifica sau cauta dupa id intr-o ducand un id valid in acele casute. Daca selecteaza butonul vezi tot ii va aparea un JTable cu toti clientii.

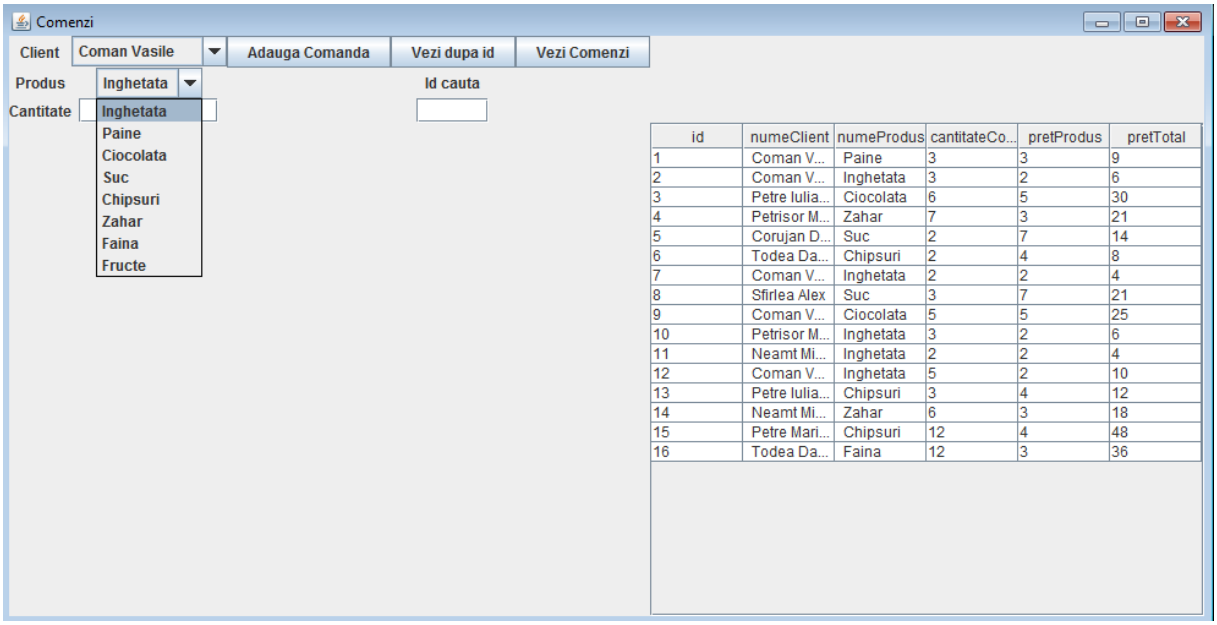


Daca selecteaza butonul Produse din meniu ii va aparea urmatoarea fereastra:



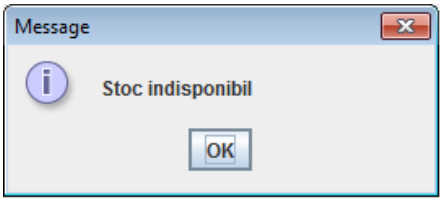
De asemenea ca si la cealalta fereastra va putea adauga, insera, sterge, modifica si vizualiza toate produsele sau un produs dupa un anumit id.

Frameul Comenzi este urmatorul:



Aici utilizatorul va putea sa faca comenzi selectand un client, un produs si o cantitate dorita.

In cazul in care utilizatorul introduce date nevalide ii vor apararea ferestre de dialog de tipul urmator:



6.Concluzii

In concluzie aceasta aplicatie implementeaza in mod corect si eficient gestiunea unui depozit oferind utilizatorului o foarte usoara interactiune cu managementul depozitului si crearea de comenzi prin interfata grafica usor de utilizat.

Posibilitati de dezvoltari ulterioare:

-posibilitatea de adaugare a mai multor tabele de legatura pentru a putea adauga intr-o comanda mai mult de un produs;

-posibilitatea de impartire a produselor pe magazine introducand un tabel magazin;

-posibilitatea de a adauga unele produse la favorite pentru a urmari evolutia modificarii pretului lor in timp;

-diferite calcule de profit si numar de vanzari pe o luna;

-adaugarea de angajati si administratori;

-posibilitatea de a te inregistra in aplicatie pe baza unui username si a unei parole si in functie de rol sa fie vizibile sau nu unele operatii din aplicatie.

Implementarea acestei aplicatii m-a invatat sa introduc si sa preiau informatii dintr-o baza de date, sa folosesc tehnica de reflection, sa utilizez JTable pentru a afisa datele dintr-un tabel trimitand o lista de obiecte, sa-mi structurez aplicatia pe straturi si sa scriu intr-un fisier text.

7.Bibliografie

https://stackoverflow.com/
http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html
http://www.coned.utcluj.ro/~marcel99/PT/
http://www.coned.utcluj.ro/~salomie/PT_Lic/
http://ww1.javahash.com/
https://www.w3schools.com/sql
http://www.java67.com
http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/
http://tutorials.jenkov.com/java-reflection/index.html
https://docs.oracle.com/javase/tutorial/uiswing/components/table.html