

DOCUMENTATIE

TEMA *3*

Titlu:

--Order Management--

NUME STUDENT: Pop Andreea Gabriela

GRUPA: 30226

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc95297886)

[3. Proiectare 3](#_Toc95297887)

[4. Implementare 3](#_Toc95297888)

[5. Rezultate 3](#_Toc95297889)

[6. Concluzii 3](#_Toc95297890)

[7. Bibliografie 3](#_Toc95297891)

# Obiectivul temei

Obiectivul temei constă în proiectarea și implementarea unui sistem de gestionare a comenzilor, care va utiliza o bază de date cu 3 tabele și va permite efectuarea diferitelor operații asupra acestora. Se vor construi soluții corecte și valide, demonstrându-se corectitudinea lor. Utilizatorul va interacționa cu aplicația prin intermediul unei interfețe grafice intuitive, care va permite efectuarea operațiilor de gestionare a comenzilor.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Pentru rezolvarea cerinței, trebuie să înțelegem conceptele necesare. O bază de date reprezintă o colecție organizată de informații sau date structurate, stocate electronic într-un computer. Baza de date este, în general, controlată de un sistem de management al bazelor de date (DBMS). Datele, DBMS-ul și aplicațiile asociate formează un sistem de baze de date. În cele mai obișnuite tipuri de baze de date, datele sunt distribuite în linii și coloane, în diferite tabele, pentru a eficientiza procesarea și interogarea datelor. Pentru accesarea, gestionarea, modificarea și organizarea datelor se folosește un limbaj structurat de interogare (SQL).

Sintaxa operațiilor asupra bazelor de date este următoarea:

Inserare: INSERT INTO nume\_tabel (coloană1, coloană2, coloană3, ...) VALUES (valoare1, valoare2, valoare3, ...);

Actualizare: UPDATE nume\_tabel SET coloană1 = valoare1, coloană2 = valoare2, ... WHERE condiție;

Ștergere: DELETE FROM nume\_tabel WHERE condiție;

Vizualizare: SELECT \* FROM nume\_tabel;

Aceste operații vor fi utilizate pentru a accesa baza de date. Aplicația va avea o interfață prietenoasă pentru utilizator, astfel încât să fie ușor de utilizat de către oricine dorește să efectueze operații de gestionare a comenzilor.

Caz de utilizare:

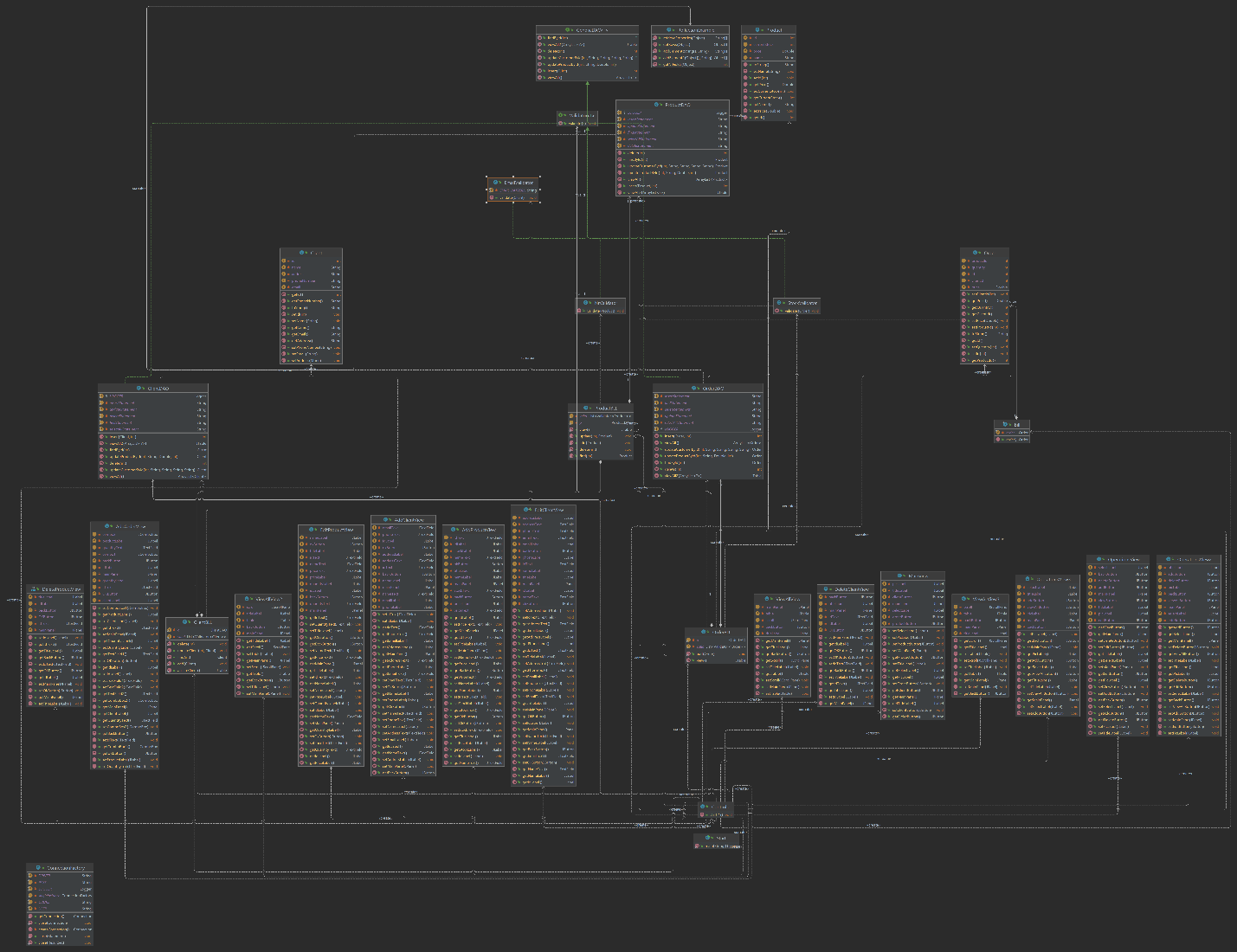
Utilizatorul selectează inițial tipul de tabel asupra căruia dorește să efectueze o operație.

Ulterior, utilizatorul selectează tipul de operație dorită.

Datele furnizate sunt verificate pentru a se asigura corectitudinea lor.

Programul efectuează operația selectată, iar rezultatul poate fi observat accesând operația de vizualizare.

Mai jos este atașată diagrama UML.



# Implementare

Deoarece proiectul este complex, am structurat totul în 6 pachete: BLL, DAO, CONNECTION, MODEL, PRESENTATION și START.

Pachetul MODEL conține următoarele clase:

Client, cu atributele: id, nume, email, număr de telefon și adresă. Această clasă are doar metode de acces (getter) și modificare (setter).

Product, cu atributele: id, nume, preț și stoc. Această clasă are doar metode de acces (getter) și modificare (setter).

Order, cu atributele: id, clientId, productId, cantitate și preț. Această clasă are doar metode de acces (getter) și modificare (setter).

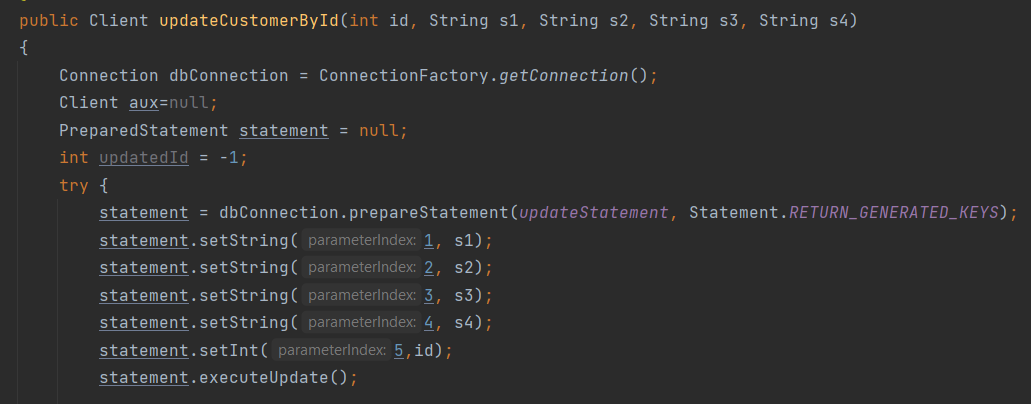
Bill, care este o clasă imutabilă din pachetul "java records" și generează bonul comenzii sub forma unui fișier text.

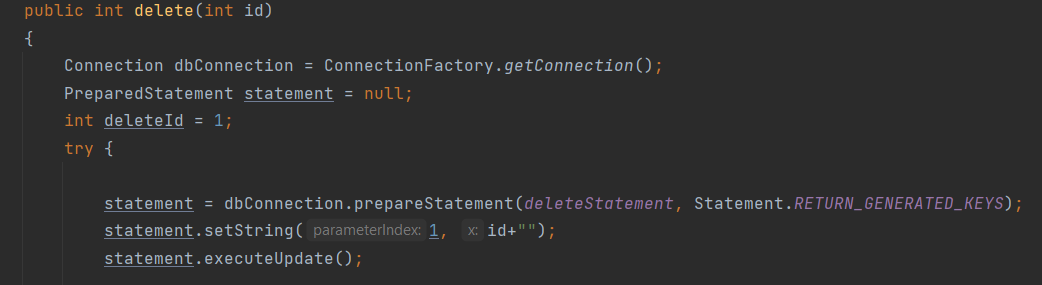
Pachetul DAO conține următoarele clase:

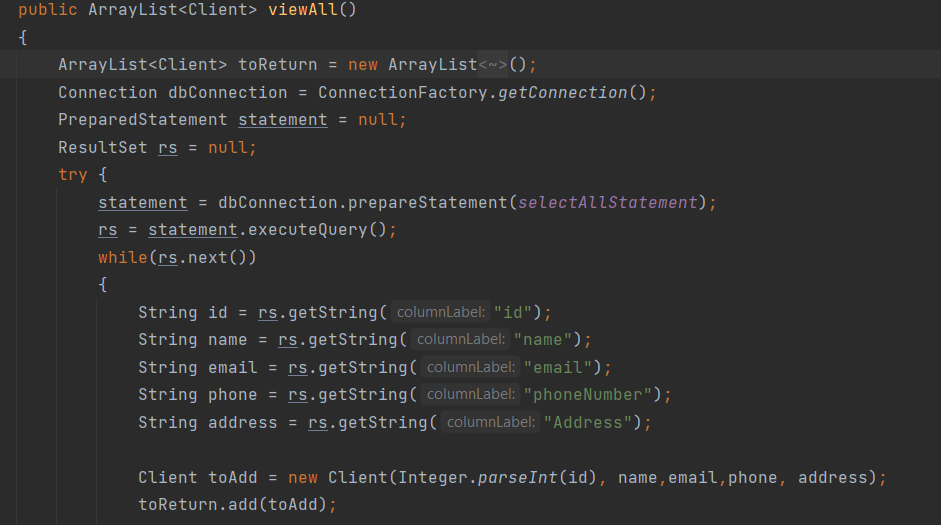
Interfața GenericDao, care conține declarațiile funcțiilor pentru operațiile asupra bazei de date.

Clasa ClientDao, care conține implementarea operațiilor asupra tabelului de clienți. Se utilizează sintaxa menționată anterior.

Clasa OrderDao și ProductDao funcționează în mod similar, diferind în atributele obiectului asupra căruia se execută instrucțiunea.



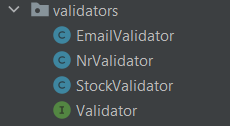


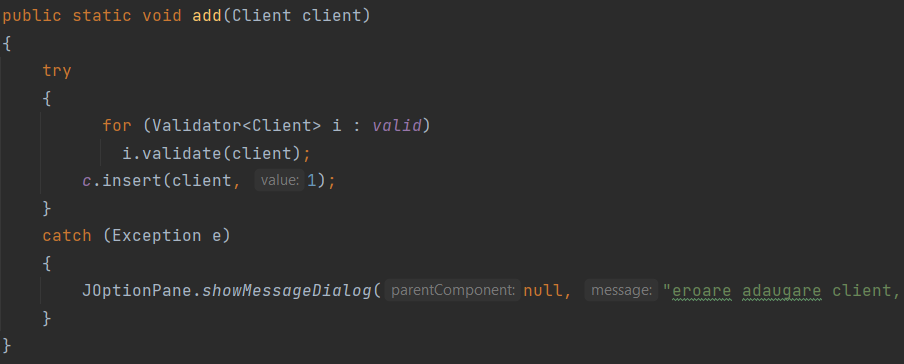


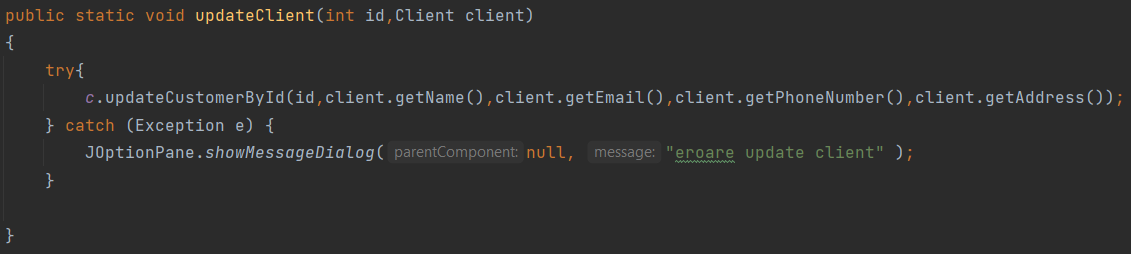
Pachetul BLL conține următoarele:

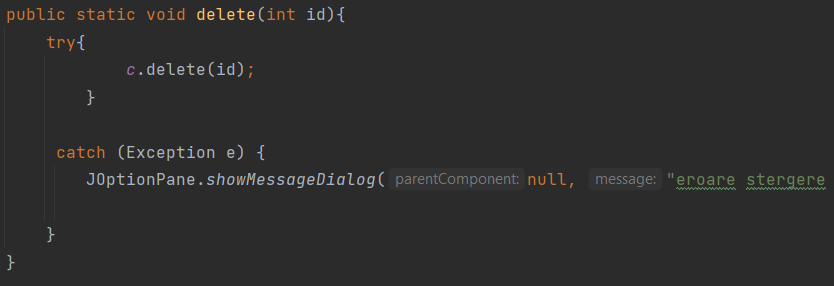
Validatori pentru a verifica corectitudinea datelor preluate din interfață.

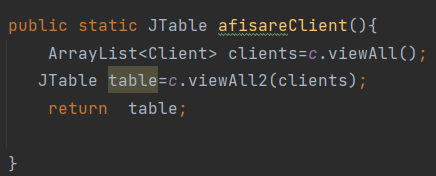
Clasele ClientBLL, ProductBLL și OrderBLL, care interpretează datele introduse și apelează metodele de acces la baza de date.





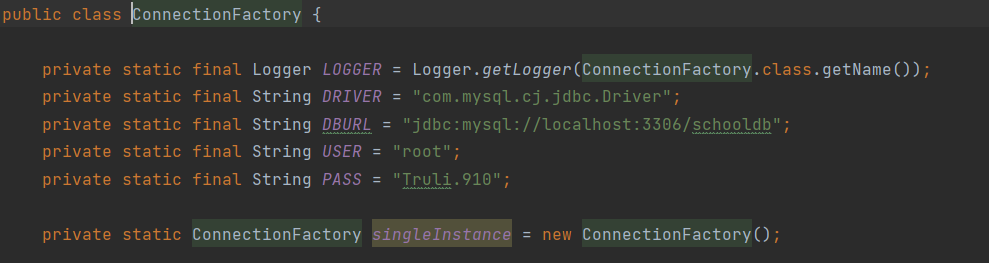


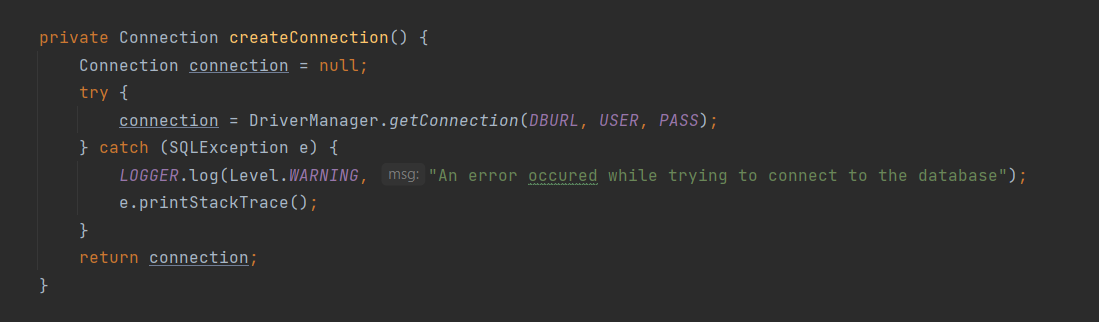




Pachetul CONNECTION conține clasa ConnectionFactory, care stabilește conexiunea cu baza de date MySQL.

Pachetul PRESENTATION conține interfețele grafice pentru interacțiunea cu utilizatorul.



****

1. **Rezultate**

Rezultatele pot fi observate atât în interfața grafică, cât și în baza de date propriu-zisă.

Pentru demonstrație, avem tabelul de clienți și dorim să efectuăm operația de inserare a unui nou client. În cazul în care datele introduse sunt valide, se poate observa că operația a avut loc cu succes. În caz contrar, se afișează un mesaj de eroare.

# Concluzii

Tema abordată necesită aplicarea cunoștințelor de programare și de baze de date SQL, care sunt utile pentru stocarea și gestionarea datelor de dimensiuni mari. Aplicația creată poate fi utilizată cu ușurință de către cei care doresc să gestioneze comenzile unui magazin online. În viitoarele versiuni, se poate îmbunătăți interfața pentru a fi și mai intuitivă, se pot trata anumite excepții care pot fi omise și se poate lucra la simplificarea codului prin crearea de metode abstracte care utilizează tehnicile de reflexie.

*.*

# Bibliografie

<https://www.javatpoint.com/steps-to-connect-to-the-database-in-java>

<https://www.guru99.com/java-swing-gui.html>

<https://www3.ntu.edu.sg/home/ehchua/programming/java/jdbc_basic.html>

<https://www.marcobehler.com/guides/java-databases>

<https://www.developer.com/java/java-databases/>