DOCUMENTATIE

TEMA 3

NUME STUDENT: Vlase Rafaella

GRUPA: 30225

# CUPRINS

[1. Obiectivul temei 3](#_Toc95297885)

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

[3. Proiectare 6](#_Toc95297887)

[4. Implementare 8](#_Toc95297888)

[5. Rezultate 15](#_Toc95297889)

[6. Concluzii 15](#_Toc95297890)

[7. Bibliografie 16](#_Toc95297891)

# Obiectivul temei

Obiectivul a fost acela de a implementa o aplicație pentru gestionarea comenzilor la un depozit. Datele au fost stocate cu ajutorul bazei de date MySQL.

Pentru implementarea proiectului am folosit o arhitectură de tip Layered.

Utilizatorul va avea la îndemână mai multe ferestre, după cum urmează:

Fereastra de meniu pentru a selecta pe ce tabel vom efectua operații

Fereastra Client, în care se văd toți clienții, se pot adăuga și șterge clienți

Fereastra Product, în care se văd toate produsele, se pot adăuga și șterge produse

Fereastra Order, în care se văd toate comenzile, se pot plasa și șterge comenzi

Pentru documentație s-au generat fișiere JavaDoc, de asemenea a fost făcut un fișier dump SQL, care conține codul de creare și populare a bazei de date.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

Analiza: Fiecare set de operații are o fereastră proprie (Client, Product, Orders)

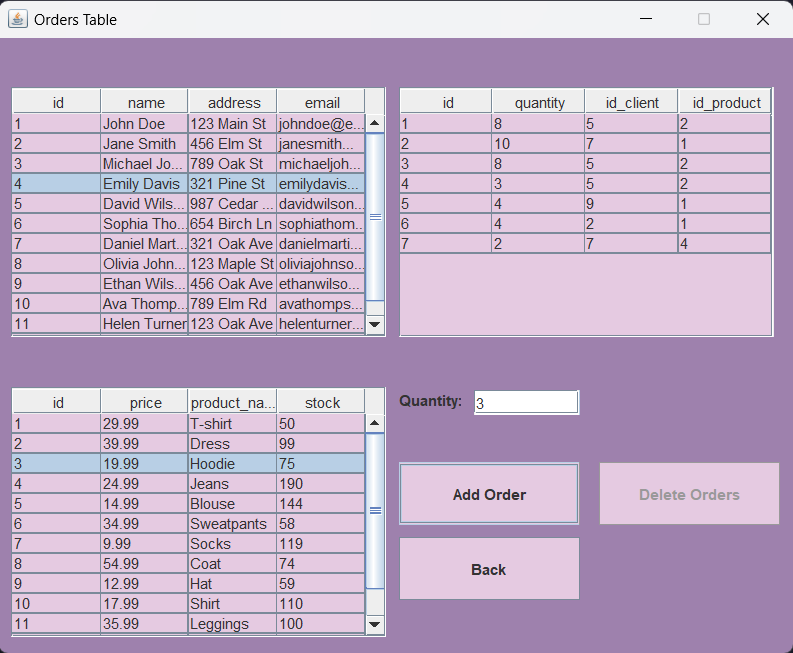
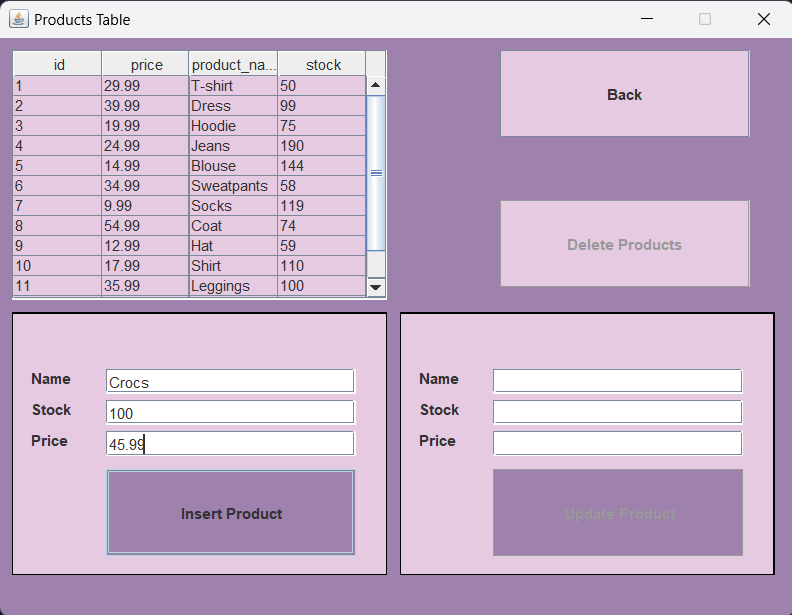
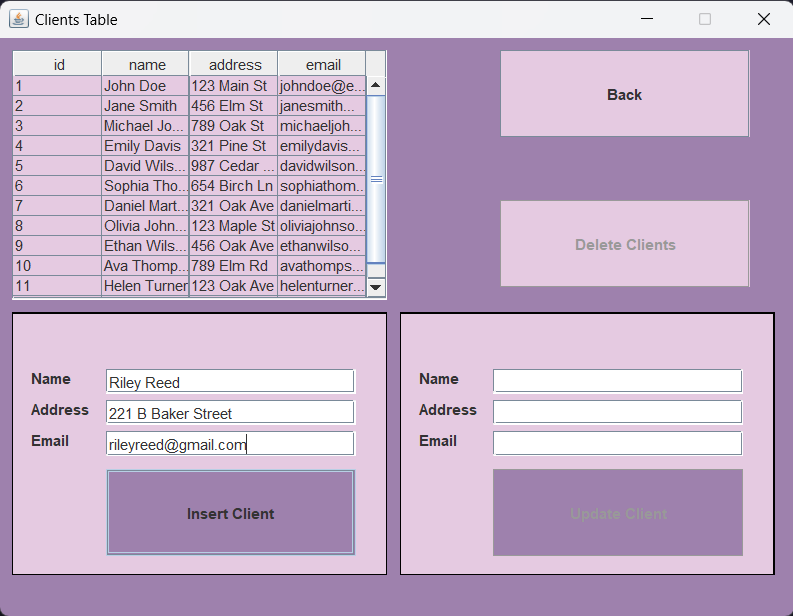
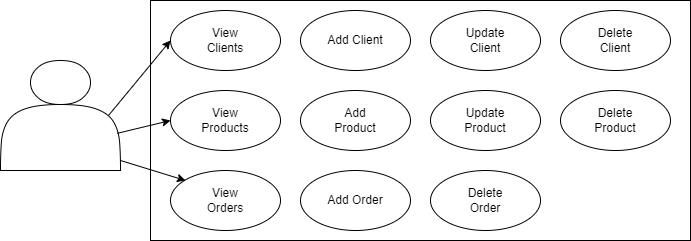
Operații implementate: adăugare, modificare, ștergere

Stocare date: bază e date MySQL, la care programul este conectat prin username și parolă

Modelare: Clasele Client, Product și Orders reprezintă câte un obiect cu același nume

Scenarii: Nu se vor putea șterge produse sau clienți implicați într-o comandă plasată din cauza cheilor străine din tabelul Orders

Use cases

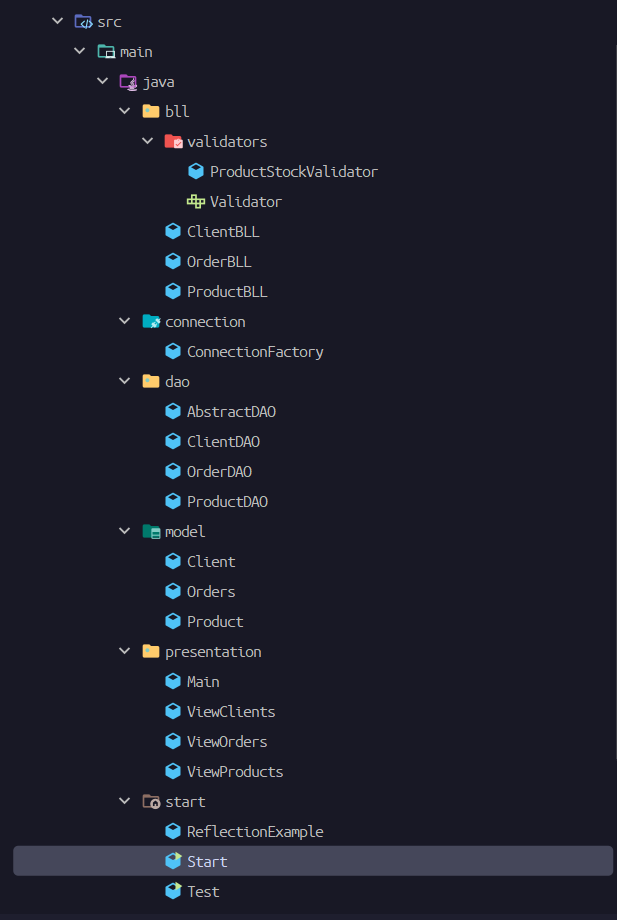


# Proiectare

Diagrama UML



Ierarhia de fișiere a proiectului (Arhitectura Layered)



# Implementare

Clasa Start: clasa în care se instanțiază interfața grafică

Clasele ClientBLL, ProductBLL, Order BLL: clasele de business logic pentru accesul de date al clienților, produselor, respectiv comenzilor

Clasa ConnectionFactory: stabilește conexiunea programului Java la baza de date MySQL

Clasa ProductStockValidator: verifică dacă un produs are stocul pozitiv, pentru a putea fi comandat

Clasa AbstractDAO este o clasă generică abstractă care implementează funcționalitatea de bază a operațiilor CRUD (Create, Read, Update, Delete) pentru entitățile din baza de date. Ea folosește Java Reflection pentru a accesa și manipula câmpurile și valorile obiectelor.

Atributele clasei:

LOGGER: un obiect de tip Logger utilizat pentru înregistrarea de mesaje și excepții în jurnal.

type: un obiect de tip Class<T> care reprezintă clasa entității cu care lucrează clasa AbstractDAO.

Metodele clasei:

AbstractDAO(): constructorul clasei care obține tipul generic al clasei concrete care o extinde.

createSelectQuery(): returnează o interogare SQL pentru a selecta un obiect din baza de date bazat pe ID.

createSelectAll(): returnează o interogare SQL pentru a selecta toate obiectele din baza de date.

findAll(): returnează o listă de obiecte care reprezintă toate înregistrările din tabela corespunzătoare entității.

findById(int id): returnează un obiect care corespunde unui anumit ID.

createObjects(ResultSet resultSet): convertește rezultatul unei interogări SQL într-o listă de obiecte de tipul entității.

createInsert(): returnează o interogare SQL pentru a insera un obiect în baza de date.

createUpdate(): returnează o interogare SQL pentru a actualiza un obiect în baza de date bazat pe ID.

createDelete(): returnează o interogare SQL pentru a șterge un obiect din baza de date bazat pe ID.

insert(T t): inserează un obiect în baza de date și returnează obiectul cu ID-ul generat.

update(T t): actualizează un obiect în baza de date bazat pe ID.

delete(T t): șterge un obiect din baza de date bazat pe ID.

makeTable(): generează un obiect DefaultTableModel care poate fi utilizat pentru a crea un tabel grafic cu datele din baza de date.

Clasele ClientDAO, ProductDAO, OrderDAO: clase responsabile cu accearea datelor. Moștenesc clasa abstractă AbstractDAO.

Clasa Client reprezintă o entitate care definește un client în cadrul sistemului. Ea conține următoarele atribute și metode:

Atribute:

id: un întreg care reprezintă ID-ul clientului.

name: un șir de caractere care reprezintă numele clientului.

address: un șir de caractere care reprezintă adresa clientului.

email: un șir de caractere care reprezintă adresa de email a clientului.

Metode:

Client(): constructorul fără parametri al clasei.

Client(int id, String name, String address, String email): constructor care primește valorile pentru toate atributele clientului și le setează.

Client(int id): constructor care primește doar ID-ul clientului și îl setează.

Client(String name, String address, String email): constructor care primește valorile pentru nume, adresă și email și le setează.

getId(): returnează ID-ul clientului.

setId(int id): setează ID-ul clientului.

getName(): returnează numele clientului.

setName(String name): setează numele clientului.

getAddress(): returnează adresa clientului.

setAddress(String address): setează adresa clientului.

getEmail(): returnează adresa de email a clientului.

setEmail(String email): setează adresa de email a clientului.

toString(): returnează o reprezentare textuală a obiectului Client, folosind valorile atributelor.

Clasa Product reprezintă o entitate care definește un produs în cadrul sistemului. Ea conține următoarele atribute și metode:

Atribute:

id: un întreg care reprezintă ID-ul produsului.

price: un număr în virgulă mobilă care reprezintă prețul produsului.

product\_name: un șir de caractere care reprezintă numele produsului.

stock: un întreg care reprezintă stocul disponibil al produsului.

Metode:

Product(): constructorul fără parametri al clasei.

Product(int id, float price, String product\_name, int stock): constructor care primește valorile pentru toate atributele produsului și le setează.

Product(float price, String name, int stock): constructor care primește valorile pentru preț, nume și stoc și le setează.

Product(int id): constructor care primește doar ID-ul produsului și îl setează.

getId(): returnează ID-ul produsului.

setId(int id): setează ID-ul produsului.

getPrice(): returnează prețul produsului.

setPrice(float price): setează prețul produsului.

getProduct\_name(): returnează numele produsului.

setProduct\_name(String product\_name): setează numele produsului.

getStock(): returnează stocul disponibil al produsului.

setStock(int stock): setează stocul disponibil al produsului.

toString(): returnează o reprezentare textuală a obiectului Product, folosind valorile atributelor.

Clasa Orders reprezintă o entitate care definește o comandă în cadrul sistemului. Ea conține următoarele atribute și metode:

Atribute:

id: un întreg care reprezintă ID-ul comenzii.

quantity: un întreg care reprezintă cantitatea de produse comandate.

id\_client: un întreg care reprezintă ID-ul clientului asociat comenzii.

id\_product: un întreg care reprezintă ID-ul produsului comandat.

Metode:

Orders(): constructorul fără parametri al clasei.

Orders(int id, int quantity, int id\_client, int id\_product): constructor care primește valorile pentru toate atributele comenzii și le setează.

Orders(int quantity, int id\_client, int id\_product): constructor care primește valorile pentru cantitate, ID-ul clientului și ID-ul produsului și le setează.

Orders(int id): constructor care primește doar ID-ul comenzii și îl setează.

getId(): returnează ID-ul comenzii.

setId(int id): setează ID-ul comenzii.

getQuantity(): returnează cantitatea de produse comandate.

setQuantity(int quantity): setează cantitatea de produse comandate.

getId\_client(): returnează ID-ul clientului asociat comenzii.

setId\_client(int id\_client): setează ID-ul clientului asociat comenzii.

getId\_product(): returnează ID-ul produsului comandat.

setId\_product(int id\_product): setează ID-ul produsului comandat.

toString(): returnează o reprezentare textuală a obiectului Orders, care include informații despre comandă, precum ID-ul comenzii, ID-ul clientului, numele clientului, ID-ul produsului, numele produsului, prețul produsului, cantitatea comandată și prețul total al comenzii.

Clasa Main este o clasă care extinde clasa JFrame și reprezintă fereastra principală a aplicației. Ea conține următoarele atribute și metode:

Atribute:

contentPanel: un obiect de tip JPanel care reprezintă panoul de conținut al ferestrei.

clientButton, orderButton, productButton: obiecte de tip JButton care reprezintă butoanele pentru operațiile legate de clienți, comenzi și produse.

Metode:

Main(): constructorul clasei, care inițializează fereastra principală și configurează panoul de conținut, butoanele și alte proprietăți ale ferestrei.

addActionListeners(): o metodă privată care adaugă ascultători de evenimente pentru butoanele clientButton, orderButton și productButton. Când unul dintre aceste butoane este apăsat, se vor executa acțiuni specifice, cum ar fi afișarea ferestrei pentru vizualizarea clienților sau a comenzilor.

Clasa Main este responsabilă de inițializarea și afișarea ferestrei principale a aplicației, precum și de gestionarea evenimentelor asociate butoanelor. Ea utilizează biblioteca Swing pentru crearea interfeței grafice. Panoul de conținut conține trei butoane, fiecare având un text specific și o culoare de fundal. Când un buton este apăsat, fereastra curentă se închide și se deschide o nouă fereastră corespunzătoare operației selectate.

Această clasă face parte din componenta de prezentare a aplicației și are rolul de a oferi interacțiunea cu utilizatorul și de a direcționa acțiunile acestuia către alte componente ale sistemului, cum ar fi vizualizările clienților, comenzilor și produselor.

Clasa ViewClients este o clasă care extinde clasa JFrame și reprezintă fereastra pentru vizualizarea și gestionarea datelor despre clienți. Ea conține următoarele atribute și metode:

Atribute:

contentPanel: un obiect de tip JPanel care reprezintă panoul de conținut al ferestrei.

clientsPanel, insertPanel, updatePanel: obiecte de tip JPanel care reprezintă panourile specifice pentru afișarea tabelului cu clienți, inserarea unui nou client și actualizarea informațiilor despre un client existent.

insertNameLabel, insertNameField, insertAddressLabel, insertAddressField, insertEmailLabel, insertEmailField: obiecte de tip JLabel și JTextField pentru introducerea datelor noului client în panoul de inserare.

insertButton, updateButton, deleteButton, backButton: obiecte de tip JButton pentru efectuarea diferitelor acțiuni precum inserarea, actualizarea, ștergerea și revenirea în fereastra principală.

tableModel, tableScrollPane, clientsTable: obiecte necesare afișării și gestionării tabelului cu clienți.

Metode:

ViewClients(): constructorul clasei, care inițializează fereastra de vizualizare a clienților și configurează panourile, butoanele, tabelul și alte proprietăți ale ferestrei.

makeTable(): o metodă privată care construiește panoul și tabelul pentru afișarea clienților.

makeInsertPanel(): o metodă privată care construiește panoul și elementele necesare pentru inserarea unui nou client.

makeUpdatePanel(): o metodă privată care construiește panoul și elementele necesare pentru actualizarea informațiilor despre un client existent.

makeButtons(): o metodă privată care construiește butoanele pentru diferite acțiuni, cum ar fi ștergerea și revenirea.

makeListeners(): o metodă privată care adaugă ascultători de evenimente pentru diferite elemente ale interfeței grafice, precum selecția în tabel, modificarea câmpurilor text și acțiunile butoanelor.

clearInsertFields(): o metodă privată care curăță câmpurile de inserare pentru introducerea unui nou client.

clearUpdateFields(): o metodă privată care curăță câmpurile de actualizare pentru informațiile despre un client existent.

Clasa ViewProducts este o clasă care extinde clasa JFrame și reprezintă o interfață grafică pentru afișarea, inserarea, actualizarea și ștergerea produselor. Aceasta are următoarele atribute și metode:

Atribute:

contentPanel: Un obiect de tip JPanel care reprezintă panoul principal al ferestrei.

productsPanel: Un obiect de tip JPanel care conține tabelul cu produse.

productsTable: Un obiect de tip JTable care afișează datele produselor.

insertPanel: Un obiect de tip JPanel care conține elementele necesare pentru inserarea unui nou produs.

updatePanel: Un obiect de tip JPanel care conține elementele necesare pentru actualizarea unui produs existent.

tableModel: Un obiect de tip DefaultTableModel care servește drept model pentru tabelul cu produse.

tableScrollPane: Un obiect de tip JScrollPane care permite derularea conținutului tabelului.

Metode:

makeTable(): Creează tabelul cu produse și îl adaugă în panoul productsPanel.

makeInsertPanel(): Creează panoul pentru inserarea unui nou produs și îl adaugă în cadrul ferestrei.

makeUpdatePanel(): Creează panoul pentru actualizarea unui produs existent și îl adaugă în cadrul ferestrei.

makeButtons(): Creează butoanele "Delete Products" și "Back" și le adaugă în cadrul ferestrei.

makeListeners(): Adaugă ascultători evenimentelor pentru selecția unei linii din tabel, cât și pentru câmpurile de inserare a datelor unui nou produs.

clearInsertFields(): Golește câmpurile de inserare pentru un nou produs.

clearUpdateFields(): Golește câmpurile de actualizare pentru un produs existent.

Constructorul ViewProducts(): Inițializează fereastra, setează dimensiunea și fundalul, adaugă elementele în fereastră, setează vizibilitatea și alte opțiuni pentru fereastră.

Clasa ViewOrders este o clasă de prezentare (interfață grafică) care extinde clasa JFrame. Atributele și metodele clasei sunt:

Atribute:

contentPanel: Un obiect de tip JPanel care reprezintă panoul principal al ferestrei.

ordersPanel, clientsPanel, productsPanel: Obiecte JPanel care reprezintă panourile pentru afișarea tabelelor de comenzi, clienți și produse.

ordersLabel, clientsLabel, productsLabel: Obiecte JLabel care reprezintă etichetele pentru tabelele de comenzi, clienți și produse.

ordersTable, clientsTable, productsTable: Obiecte JTable care reprezintă tabelele de comenzi, clienți și produse.

quantityLabel: Un obiect JLabel care reprezintă eticheta pentru câmpul de cantitate.

quantityField: Un obiect JTextField care reprezintă câmpul de introducere a cantității.

insertButton, deleteButton, backButton: Obiecte JButton care reprezintă butoanele "Add Order", "Delete Orders" și "Back".

clientsTableModel, productsTableModel, ordersTableModel: Obiecte DefaultTableModel care reprezintă modelele de date pentru tabelele de clienți, produse și comenzi.

clientsTableScrollPane, productsTableScrollPane, ordersTableScrollPane: Obiecte JScrollPane care permit derularea tabelelor de clienți, produse și comenzi.

Metode:

ViewOrders(): Constructorul clasei. Inițializează fereastra, setează dimensiunile și panoul principal, apelează metoda makeTables() pentru a crea tabelele și metoda makeOrders() pentru a crea elementele pentru manipularea comenzilor. De asemenea, adaugă ascultătorii de evenimente și setează fereastra vizibilă.

makeTables(): Creează și configurează tabelele pentru afișarea clienților, produselor și comenzilor.

makeOrders(): Creează și configurează elementele pentru manipularea comenzilor (etichete, câmpuri de introducere și butoane).

makeListeners(): Creează și adaugă ascultătorii de evenimente pentru tabelele și elementele interfeței grafice. Acești ascultători gestionează selecția de rânduri în tabele, validarea cantității introduse și acțiunile de adăugare și ștergere a comenzilor.

# Rezultate

Programul funcționează conform cerințelor, nu există excepții nefirești.

# Concluzii

Cu acest proiect mi-am aprofundat cunostințele de a lucra cu JDBC.

Ca dezvoltare ulterioară se poate genera o factură pentru fiecare comandă plasată.

# Bibliografie

<https://www.baeldung.com/java-jdbc>

<http://www.mkyong.com/jdbc/how-to-connect-to-mysql-with-jdbc-driver-java/>

<https://dzone.com/articles/layers-standard-enterprise>

<http://tutorials.jenkov.com/java-reflection/index.html>

<https://www.baeldung.com/java-pdf-creation>

<https://www.baeldung.com/javadoc>

<https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html>