DOCUMENTATIE

TEMA *3*

NUME STUDENT: Tîrcă Mihnea

GRUPA: 30224

Cuprins

[1. Obiectivul temei 3](#_Toc104455415)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc104455416)

[3. Proiectare 5](#_Toc104455417)

[4. Implementare 7](#_Toc104455418)

[5. Rezultate 10](#_Toc104455419)

[6. Concluzii *.* 10](#_Toc104455420)

[7. Bibliografie 10](#_Toc104455421)

# Obiectivul temei

Obiectivul acestei teme este de a implementa o aplicatie cu interfata grafica pentru gestionarea comenzilor pentru un depozit. Datele trebuie stocate folosind baza de date MySQL. Trebuie folosit de asemenea o arhitectura „Layered”.

Utilizatorul trebuie sa poata, in fiecare fereastra, sa faca anumite operatii pe date:

In fereastra Client, trebuie sa vada toti clientii, sa poata adauga, modifica si sterge clienti.

In fereastra Product, trebuie sa vada toate produsele, sa poata adauga, modifica si sterge produse.

In fereastra Order, trebuie sa vada toate comenzile si sa poata adauga comenzi.

Au trebuit scrise comentarii javadoc si a trebuit generat fisierul dump SQL, care sa contina codul de creare a bazei de date si de populare a acesteia.

# Analiza problemei, modelare, scenarii, cazuri de utilizare

**Analiza:**

Fiecarui set de operatii i se atribuie o fereastra individuala. Pentru asta, au trebuit implementate 3 ferestre diferite de operatii pe date (Client, Product, Order). Au trebuit implementate operatiile pe date de adaugare, modificare si stergere. Datele sunt stocate intr-o baza de date MySQL, la care programul se conecteaza cu numele si parola.

**Modelare:**

Structura de date „Client” reprezinta un client

Structura de date „Product” reprezinta un produs

Structura de date „Order” care reprezinta o comanda

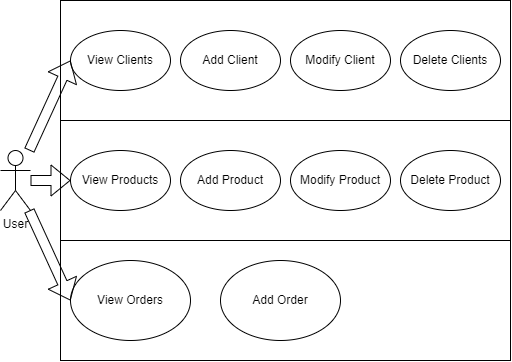
**Scenarii:**

Functia de modificare a unei comenzi nu are sens si nu a fost implementata.

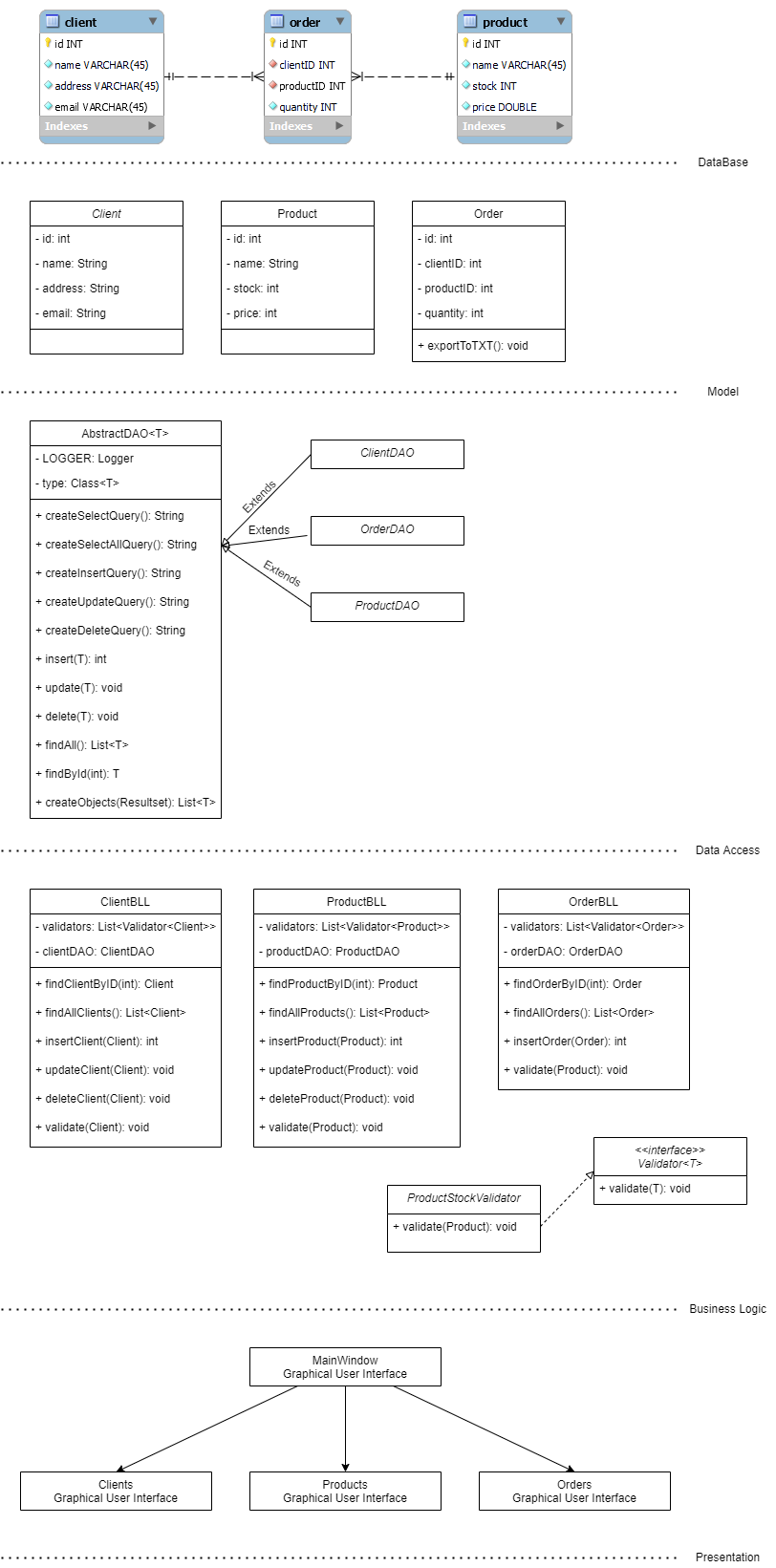
In cazul incercarii de a adauga o comanda, daca aceasta ar duce stocul produsului sub 0, o fereastra de eroare va aparea, iar comanda nu va fi efectuata.

De asemenea, nu se poate sterge un produs sau client daca cel putin unul dintre ei participa intr-o comanda. Asta se intampla din cauza cheilor straine din baza de date.

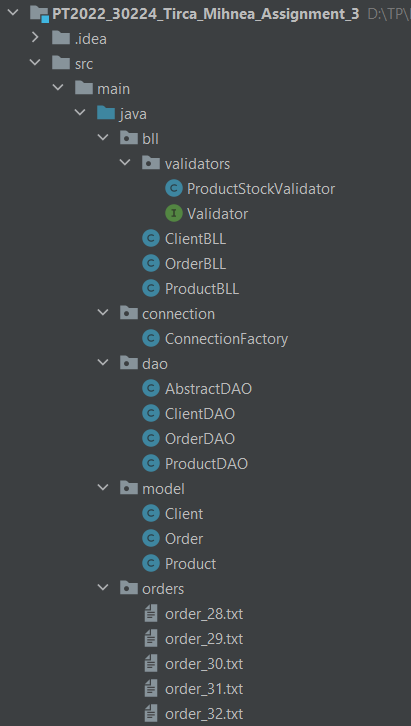
**Cazuri de utilizare**

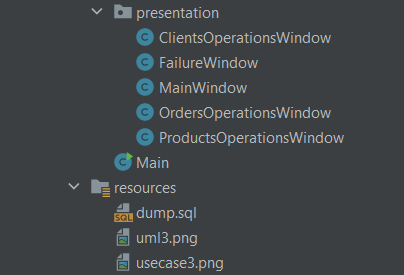


# Proiectare



Proiectul este ierarhizat pe baza arhitecturii „Layered”:





# Implementare

**Clasa MainWindow:**

Reprezinta fereastra deschisa la rularea programului. Aici utilizatorul poate alege ce fel de operatii vrea sa faca: pe produse, clienti, sau comenzi.

**Clasa ClientsOperationsWindow:**

Fereastra unde se pot face operatii pe clienti: adaugare clienti, modificare clienti, stergere clienti.

**Clasa ProductsOperationsWindow:**

Fereastra unde se pot face operatii pe produse: adaugare produse, modificare produse, stergere produse.

**Clasa OrdersOperationsWindow:**

Fereastra unde se pot face operatii pe comenzi: adaugare comenzi, stergere comenzi.

**Clasa FailureWindow:**

Fereastra de care se deschide daca o operatie a esuat din varii motive. Este implementata sa se deschida doar atunci cand se incearca adaugarea unei comenzi care ar pune un produs sub stocul 0.

**Clasa Client:**

Clasa care reprezinta clientul. Ea contine urmatoarele campuri:

private int id;  
private String name;  
private String address;  
private String email;

id: id-ul unic fiecarui client

name: numele clientului

address: adresa de domiciliu a clientului

email: adresa de email a clientului

**Clasa Product:**

Clasa care reprezinta produsul. Ea contine urmatoarele campuri:

private int id;  
private String name;  
private int stock;  
private double price;

id: id-ul unic fiecarui produs

name: numele produsului

stock: stocul produsului

price: pretul produsului

**Clasa Order:**

Clasa care reprezinta comanda. Ea contine urmatoarele campuri:

private int id;  
private int clientID;  
private int productID;  
private int quantity;

id: id-ul unic fiecarei comenzi

clitentID: id-ul clientului care a plasat comanda

productID: id-ul produsului care a fost comandat

quantity: cantitatea produsului

**Clasa AbstractDAO<T>:**

Clasa cu metode de acces de date. Este mostenita de clasele ClientDAO, OrderDAO, ProductDAO.

String createSelectQuery();

Returneaza un String reprezentand un template de query SELECT.

String createSelectAllQuery();

Returneaza un String reprezentand un template de query SELECT \*.

String createInsertQuery();

Returneaza un String reprezentand un template de query INSERT.

String createUpdateQuery();

Returneaza un String reprezentand un template de query UPDATE.

String createDeleteQuery();

Returneaza un String reprezentand un template de query DELETE.

int insert(T t);

Insereaza un element t in tabela.

void update(T t);

Updateaza un element t din tabela.

void delete(T t);

Sterge un element t din tabela.

List<T> findAll();

Returneaza toate elementele T din tabela.

T findByID(int id);

Returneaza elementul de tip T din tabela cu id-ul id.

List<T> createObjects(ResultSet resultSet);

Creeaza si returneaza o lista de obiecte T din ResultSet-ul trimis ca parametru.

**Clasa ClientDAO:**

Clasa responsabila cu accesul de date al clientilor. Mosteneste AbstractDAO<Client>.

**Clasa ProductDAO:**

Clasa responsabila cu accesul de date al produselor. Mosteneste AbstractDAO<Product>.

**Clasa OrderDAO:**

Clasa responsabila cu accesul de date al comenzilor. Mosteneste AbstractDAO<Order>.

**Clasa ClientBLL:**

Clasa de business logic pentru accesul de date al clientilor.

**Clasa ProductBLL:**

Clasa de business logic pentru accesul de date al produselor.

**Clasa OrderBLL:**

Clasa de business logic pentru accesul de date al comenzilor.

**Clasa ConnectionFactory:**

Clasa repsonsabila cu stabilirea conexiunii programului java la baza de date.

**Interfata Validator<T>:**

Este o interfata functionala care contine metoda:

void validate(T t);

Clasele de business logic o implementeaza pentru a valida datele.

**Clasa ProductStockValidator:**

Implementeaza interfata Validator<Product>. Contine doar metoda implementata. Aceasta verifica daca un produs are stocul pozitiv.

# Rezultate

Programul functioneaza corect si nu au fost gasite exceptii in functionare care nu sunt firesti.

# Concluzii *.*

Cu acest proiect mi-am aprofundat cunostintele de a lucra cu JDBC. De asemenea am invatat ce este reflexia.

# Bibliografie

<https://docs.oracle.com/javase/7/docs/>

<https://geeksforgeeks.org/>

https://stackoverflow.com/