**DOCUMENTAȚIE**

**ORDERS MANAGEMENT**

**TEMA 3**

NUME STUDENT: **Tecuci Dragoș**

GRUPA:**30222**

**CUPRINS**

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare
4. Imprelementare
5. Rezultate
6. Concluzii
7. Bibliografie

**1.Obiectivul temei**

Tema abordată se concentrează asupra realizării unui proiect cu obiective clare și un nivel înalt de profesionalism. Scopul principal al acestei inițiative constă în dezvoltarea și implementarea unei aplicații sofisticate de gestionare a clienților, a produselor și a comenzilor într-un depozit diversificat. Această aplicație se bazează pe o interfață grafică interactivă, ce permite utilizatorului să realizeze operațiuni de inserare, actualizare și ștergere a clienților și produselor, precum și să plaseze comenzi prin selectarea unui client și a unui produs.

Prin intermediul acestei aplicații avansate, utilizatorul beneficiază de un sistem de notificare eficient, care îl alertează în timp real în cazul în care stocul unui produs este excedentar sau insuficient. Astfel, gestiunea și controlul depozitului devin mai eficiente și mai precise.

Interfața intuitivă a aplicației, structurată în trei module distincte, facilitează accesul rapid și ușor la funcționalitățile dorite. Utilizatorul poate efectua operații asupra produselor din depozit, poate gestiona și actualiza informațiile referitoare la clienți și poate administra comenzile într-un mod simplu și eficient. Meniul din interfața Java asigură o navigare fluidă între aceste module și permite selecția opțiunii de interes cu doar câteva clicuri.

Fiecare panel al interfeței prezintă un tabel actualizat în timp real, ce afișează informațiile despre produsele, clienții și comenzile existente în baza de date. Astfel, utilizatorul poate avea mereu acces la datele actualizate și poate lua decizii în cunoștință de cauză.

Proiectul a fost abordat cu o metodologie riguroasă, parcurgând etapele de analiză a problemei, modelare, definirea scenariilor și a cazurilor de utilizare. O atenție deosebită a fost acordată proiectării aplicației, astfel încât să ofere o funcționalitate superioară și o experiență plăcută utilizatorului. Implementarea propriu-zisă a fost realizată cu meticulozitate și profesionalism, având în vedere standardele și practicile de programare moderne.

Obiectivele secundare ale acestui proiect au fost urmărite cu atenție pentru a sprijini realizarea obiectivului principal, iar rezultatele lor contribuie la o implementare de succes a aplicației de gestionare a clienților, produselor și comenzilor unui depozit. Aceste obiective secundare includ:

1. Analiza problemei și modelarea: S-a efectuat o analiză exhaustivă a cerințelor și a contextului proiectului, identificându-se nevoile și fluxurile de lucru specifice depozitului și clienților săi. S-au realizat modele și diagrame relevante pentru a asigura o înțelegere clară a structurii și interacțiunilor sistemului.
2. Definirea scenariilor și cazurilor de utilizare: S-au identificat și documentat scenariile și cazurile de utilizare relevante pentru aplicație, luând în considerare diferitele situații și acțiuni pe care utilizatorii le pot întâlni în timpul gestionării clienților, produselor și comenzilor.
3. Proiectarea aplicației: S-a realizat o proiectare detaliată a aplicației, inclusiv a interfeței grafice, a structurii bazei de date și a componentelor funcționale necesare pentru a realiza funcționalitățile dorite. S-au aplicat principii de design și s-au urmărit cele mai bune practici de dezvoltare software.
4. Implementarea propriu-zisă a aplicației: S-a dezvoltat codul sursă al aplicației, respectând proiectarea și specificațiile definite anterior. S-au utilizat tehnologii și limbaje de programare adecvate pentru a asigura o funcționalitate solidă și o performanță optimă.
5. Testarea aplicației: S-a realizat o testare exhaustivă a aplicației pentru a verifica funcționalitatea corectă, stabilitatea și securitatea acesteia. S-au efectuat teste unitare, teste de integrare și teste de performanță pentru a asigura o calitate ridicată a aplicației.

Prin urmărirea acestor obiective secundare, proiectul a atins o implementare de succes și a furnizat o aplicație robustă și intuitivă de gestionare a clienților, produselor și comenzilor, îndeplinind astfel nevoile și cerințele depozitului într-un mod eficient și profesional.

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

Pentru a rezolva cerințele temei într-un mod eficient, primul pas esențial constă într-o înțelegere solidă a bazelor de date MySQL. Acest aspect este de o importanță crucială, deoarece toate comenzile pe care dorim să le executăm, cum ar fi vizualizarea, ștergerea, actualizarea și efectuarea comenzilor, se vor realiza direct pe baza de date.

Problema pe care ne-am propus să o abordăm constă în implementarea unei aplicații de management a comenzilor efectuate de diferiți clienți, care să fie utilă și accesibilă oricărui utilizator. Aplicația utilizează o bază de date externă formată în MySql WorkBench, unde sunt reținute toate informațiile despre clienți, produse, comenzi. Pe lângă gestionarea comenzilor, în cadrul aplicației se pot efectua și alte operații precum adăugarea, ștergerea și actualizarea clienților, produselor și furnizorilor. În cazul introducerii unei valori invalide (de exemplu, o cantitate întreagă într-un câmp de tip șir de caractere) sau a neglijării completării unor câmpuri obligatorii, aplicația va afișa o atenționare, așteptând introducerea unor date valide și corecte. De asemenea, orice modificare realizată în aplicație se va reflecta în timp real în baza de date, asigurând astfel o sincronizare corespunzătoare. Aplicația oferă, de asemenea, posibilitatea vizualizării conținutului bazei de date, cum ar fi lista clienților sau a produselor, prin intermediul unor tabele care sunt în concordanță cu structura bazei de date.

* Cerințe funcționale
* Aplicația trebuie să permită utilizatorului să aleagă operațiunea dorită: Operații pe tabela de clienți, operații pe tabela de produse și operații pe tabela de comenzi.
* Aplicația trebuie să permită utilizatorului să adauge un client nou.
* Aplicația trebuie să permită utilizatorului să editeze un client existent.
* Aplicația trebuie să permită utilizatorului să șteargă un client existent.
* Aplicația trebuie să permită utilizatorului să adauge un produs nou.
* Aplicația trebuie să permită utilizatorului să editeze un produs existent.
* Aplicația trebuie să permită utilizatorului să șteargă un produs existent.
* Aplicația trebuie să permită utilizatorului să creeze o comandă prin selectarea unui client și a unui produs dintr-un tabel de clienți și un tabel de produse și introducerea unei cantități valide, iar apoi inserarea acestei comenzi în tabela de comenzi.
* Cerințe non-funcționale
* Aplicația trebuie să fie intuitivă și ușor de utilizat.
* Aplicația trebuie să aibă o interfață grafică interactivă care să faciliteze utilizarea
* A picture containing diagram, text, line, screenshot

  Description automatically generatedCazuri de utilizare

**Cazuri de utilizare:** Operații pe tabela de clienți, Operații pe tabela de produse

1. Utilizatorul alege din fereastra principală operațiunea dorită: Operații pe tabela de clienți sau pe tabela de produse.

2.1.Pentru adăugarea unui nou client/produs, utilizatorul introduce datele, iar apoi apasă butonul “INSERT”.

2.2.Pentru actualizarea unui client/produs, utilizatorul selectează clientul/produsul din tabel și introduce noile date, iar apoi apasă butonul „UPDATE”.

2.3.Pentru ștergerea unui client/produs, utilizatorul selectează clientul/produsul ce se dorește a se șterge, iar apoi apasă butonul “DELETE”.

3.Baza de date corespunzătoare clienților/produselor se actualizează în funcție de operația realizată la punctul anterior.

**Cazuri de utilizare:** Operații pe tabela de comenzi

1.Utilizatorul alege din fereastra principală operațiunea dorită: Operații pe tabela de comenzi.

2.Utilizatorul trebuie să selecteze un client, un produs și să introducă o cantitate validă, iar apoi apasă butonul “Make order”, ceea ce va determina inserarea unei noi comenzi în tabelă.

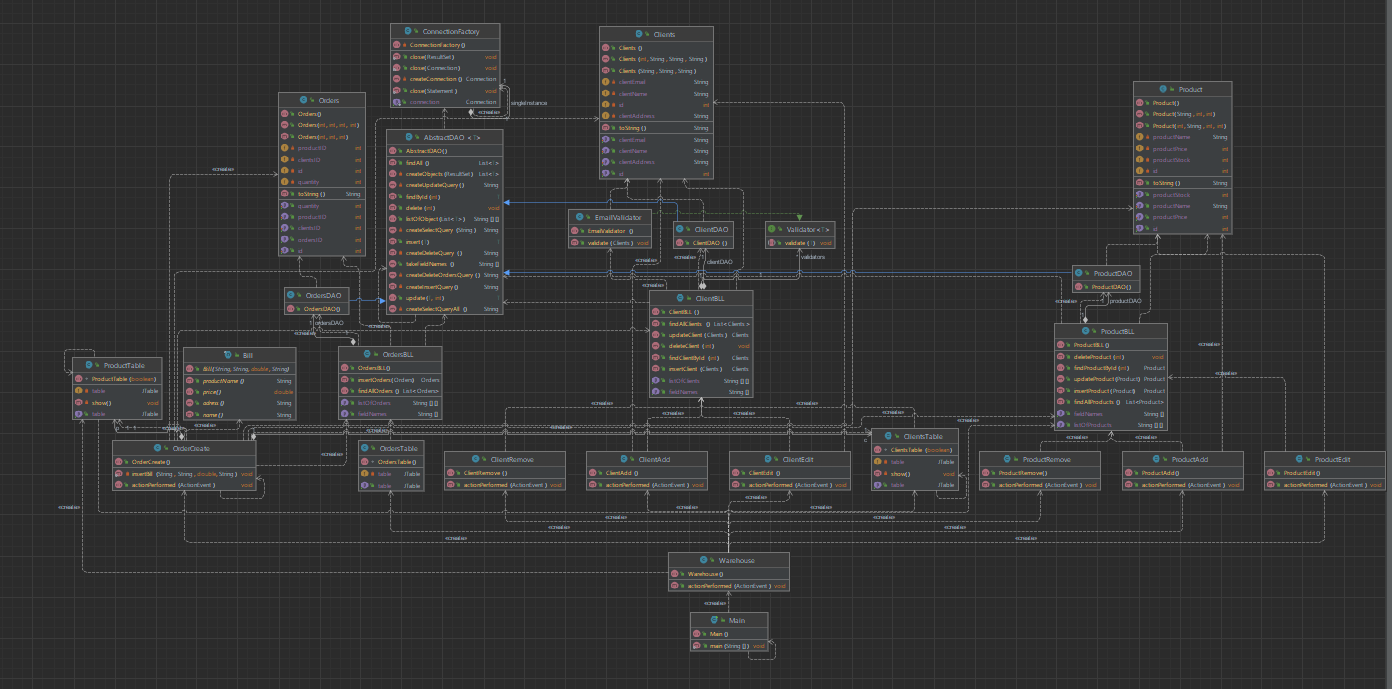
3. Baza de date corespunzătoare comenzilor se actualizează.

**Scenariu alternativ:** Email invalid, cantitate invalidă.

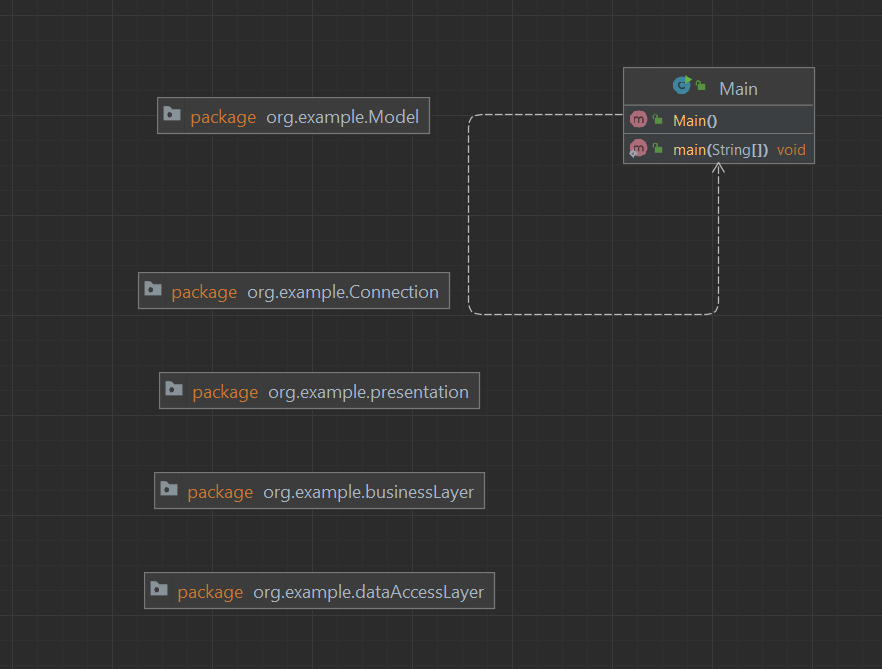
Pentru cazul când utilizatorul dorește introducerea unui nou client în baza de date, dacă email-ul nu respecta pattern-ul stabilit, se va afișa un mesaj corespunzător și utilizatorul va trebui să introducă un alt mail-ul.

Pentru cazul când utilizatorul dorește introducerea unei cantități pentru crearea unei comenzi, dacă această cantitate este negativă sau este mai mare decât stocul existent, se va afișa un mesaj corespunzător și utilizatorul va trebui să introducă o altă cantitate.

**3.Proiectare**

**Diagrama UML**

Package-uri:



**4.Implementare/Proiectare clase**

În cadrul implementării și proiectării claselor , am urmat abordarea prezentata anterior.Am început prin definirea unui pachet numit "connection" care conține clasa "ConnectionFactory". Această clasă conține metodele necesare pentru conectarea aplicației Java la o bază de date creată de noi. Am folosit codul primit în cadrul laboratorului pentru a implementa această clasă. Clasa "ConnectionFactory" are atributele necesare pentru conectarea la baza de date și conține patru metode statice care se ocupă de conectare, inițializare și închidere a conexiunii la baza de date. De asemenea, clasa conține trei metode suplimentare pentru închiderea unei conexiuni, unui statement și a unui resultSet.

Următorul pachet este "bll" (business logic layer), care conține pachetul "validators" și alte patru clase: "ClientBLL", "OrdersBLL", "ProductBLL" În pachetul "validators", am implementat trei clase care aplică condiții asupra datelor de intrare și le validează conform numelor lor. Clasele menționate anterior au următoarele roluri:

Clasa "ClientBLL" are un atribut numit "validators" de tipul "List<Validators<Client>>", ceea ce înseamnă că avem o listă de validatori pentru client. Constructorul clasei creează o instanță a claselor care validează datele de intrare. Metodele din această clasă sunt "insertClient", "findAll", "updateClient", "deleteClient" și "findClient". În aceste metode, se apelează și metoda "validate" pentru a verifica datele. Dacă datele nu respectă condițiile de validare, metoda aruncă o excepție, care este tratată în interfața grafică prin afișarea unui mesaj utilizatorului aplicației. În cazul clientilor, verificăm doar dacă vârsta acestora se încadrează între 10 și 60 de ani, așa cum a fost descris în prezentare.

Clasa "ProductBLL" are și ea un atribut numit "validators" de tipul "List<Validators<Product>>", ceea ce înseamnă că avem o listă de validatori pentru produse. În această clasă am implementat metodele "findProduct", "insertProduct", "findAll", "updateProduct" și "deleteProduct". Aceste metode creează o nouă instanță a clasei "ProductDAO" și generează interogarea.

Clasa "OrdersBLL" are, de asemenea, atributul "validators" (discutat și prezentat în clasele descrise anterior).

Următorul pachet este pachetul model, care conține clasele principale prezente în tabelele inițiale din baza de date. Astfel, în acest pachet avem următoarele clase: Clasa Client, Clasa Product și Clasa Orders. Clasa Client conține următoarele atribute: id (de tip int), name (de tip String), address (de tip String), email (de tip String) și age (de tip int). Constructorul acestor clase are parametri identici cu atributele descrise mai sus. În această clasă am generat getteri și setteri prin intermediul IntelliJ-ului, iar de asemenea, și metoda toString a fost generată tot prin aceeași metodă pentru a fi mai ușor de implementat.

Clasa Product conține următoarele atribute (specifice bazei de date generate în MySQL Workbench): id (de tip int), name (de tip String) și stock (de tip int). Constructorul acestei clase are, de asemenea, 3 parametri, aceștia fiind identici cu atributele menționate anterior. Avem și un constructor null pentru a crea o instanță a clasei. La fel ca și în clasa anterioară, am generat getteri și setteri prin intermediul IntelliJ-ului și de asemenea și metoda toString().

Clasa Orders conține următoarele atribute: idOrder (de tip int), idClient (tot de tip int), nameClient (de tip String), idProduct (de tip int), nameProduct (de tip String) și quantity (de tip int). Constructorul clasei are parametri identici cu atributele. La fel ca și în clasa anterioară, am generat getteri și setteri prin intermediul IntelliJ-ului.

Următorul pachet este pachetul DAO (Data Access Object), care este pachetul principal ce funcționează pe baza de date. Acest pachet conține următoarele clase: ClientDAO, ProductDAO și OrdersDAO, care moștenesc clasa AbstractDAO. Aceasta din urmă este cea care transmite comenzile în baza de date. În această clasă avem ca și argument de tip privat Logger-ul pentru logarea în baza de date, respectiv tipul, care stabilește tabela pe care se lucrează metodele de tip query. Acestea au ca scop crearea comenzii care trebuie trimisă în baza de date pentru a fi executată, realizându-se prin intermediul unui StringBuilder. Prin intermediul acestora se realizează funcțiile de insert, update, delete și selecție din tabele. Selectia se face atât în funcție de id, cât și de nume, și se returnează un obiect de tipul cerut. De asemenea, se poate genera și o listă care conține totalitatea obiectelor de acel tip din baza de date. În acest pachet se folosesc tipurile generice pentru realizarea lucrului pe un anumit tabel din baza de date, nefiind necesare alte metode în clasele ce moștenesc clasa principală.

**5.Rezultate**

Rezultatele obținute în urma testării aplicației de gestionare a comenzilor confirmă faptul că aceasta își îndeplinește scopul, respectiv implementarea unui mecanism de management al comenzilor. Managementul comenzilor cuprinde trei componente principale: managementul produselor, managementul clienților și managementul efectiv al comenzilor.

Managementul produselor presupune posibilitatea adăugării unui produs nou, modificarea cantității unui produs selectat pe baza identificatorului unic (ID) al produsului și ștergerea unui produs selectat tot pe baza ID-ului.

Managementul clienților presupune posibilitatea adăugării unui client nou împreună cu toate informațiile de contact necesare, precum și ștergerea unui client din baza de date. Clientul este selectat în funcție de ID-ul său unic.

Managementul comenzilor presupune adăugarea unei noi comenzi. O comandă poate conține unul sau mai multe produse. După selectarea fiecărui produs pe baza ID-ului acestuia, se apasă butonul "Adaugă produs", iar la final se apasă butonul "Finalizează comanda", care va adăuga toate produsele în lista de comenzi și va actualiza cantitatea de produse rămase în stoc ("Warehouse") după adăugarea comenzii.

Pentru a facilita selecția produselor, a clienților și a comenzilor asupra cărora se efectuează operațiile de adăugare, ștergere și modificare, am optat pentru afișarea datelor deja existente în baza de date sub forma unui tabel în interfața aplicației. Aceste informații sunt actualizate după fiecare modificare efectuată în baza de date și în obiectele Java care stochează datele din baza de date. Astfel, se asigură o experiență mai ușoară și mai intuitivă pentru utilizatori.

**6.Concluzii**

În urma acestui proiect de dezvoltare a unei aplicații de gestionare a comenzilor, am reușit să implementăm un sistem eficient și funcțional pentru managementul produselor, clienților și comenzilor. Aplicația oferă utilizatorilor posibilitatea de a adăuga, modifica și șterge produse și clienți, precum și de a plasa și finaliza comenzile.

Prin intermediul interfeței intuitive, utilizatorii pot accesa și actualiza rapid informațiile din baza de date, iar vizualizarea datelor existente într-un tabel facilitează selecția și monitorizarea comenzilor și produselor disponibile.

Rezultatele obținute în timpul testării confirmă că aplicația își îndeplinește scopul propus, oferind un mecanism eficient pentru gestionarea comenzilor într-un mediu ușor de utilizat. Aceasta contribuie la îmbunătățirea procesului de management al companiei, asigurând o mai mare eficiență și control asupra stocurilor și relației cu clienții.

În viitor, aplicația poate fi extinsă și îmbunătățită cu funcționalități suplimentare, precum generarea de rapoarte și statistici pentru analiza performanțelor și identificarea tendințelor în vânzări. De asemenea, poate fi integrată cu alte sisteme sau platforme, pentru a facilita gestionarea comenzilor și a informațiilor într-un mod mai cuprinzător.

În ansamblu, acest proiect reprezintă o soluție valoroasă pentru eficientizarea procesului de management al comenzilor, contribuind la creșterea performanței și productivității în cadrul unei organizații.

**7.Bibliografie**

<https://gitlab.com/utcn_dsrl/pt-reflection-example/-/tree/master/>

<https://www.geeksforgeeks.org/>

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

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