**DOCUMENTATIE**

**TEMA 3**

NUME STUDENT : Pojar Andrei-Gabriel

GRUPA : 30223

**CUPRINS**

1. Obiectivul temei
2. Analiza problemei
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie
8. **Obiectivul temei**

Obiectivul temei este proiectarea si implementarea unei aplicatii de gestionare a comenzilor pentru procesarea comenzilor clientilor pentru un depozit. Bazele de date relaționale ar trebui să fie utilizate pentru a stoca produsele, clienții și comenzile

1. **Analiza problemei**

Avem de implementat o aplicatie de gestionare a comenzilor pentru un depozit. Aceasta aplicatie trebuie sa fie capabila sa indeplineasca toate cerintele pentru a afisa, modifica si ține evidenta comenzilor, clientilor si produselor. Acestea sunt stocate intr-o baza de date relationala, impreuna cu informatii despre utilizatorii care au acces la sistem. In ceea ce priveste datele de intrare in aplicatie, utilizatorul poate aleaga sa gestioneze trei tabele, Clients, Products și Orders. Toate tabelele au optiunile definite ca operatii CRUD, Create (inserare), Read (afisare), Update (modificare) și Delete (stergere), iar utilizatorul poate introduce valori în câmpurile specifice pentru oricare dintre cele trei tabele diferite.

De exemplu, pentru tabelul Client, utilizatorul poate introduce un ID pentru client, numele si prenumele clientului si numarul de telefon al acestuia. Pentru toata aceste campuri am implementat o clasa Validator in care validez datele pe care utilizatorul le introduce(numele si prenumele sa contina doar litere si spatii,iar numarul de telefon sa contina 10 cifre).

Pentru a utiliza aplicatia, utilizatorul are la dispozitie o interfata grafica dedicata(GUI). La rularea aplicatiei utilizatorului i se va deschide o fereastra in care vor aparea patru butoane,unul pentru a ajunge in pagina dedicata clientilor, unul pentru a ajunge in pagina dedicate produselor, unul pentru a ajunge in pagina dedicate comenzilor si un buton de exit. Cand apasam pe butonul de clients ni se va deschide o alta fereastra care este pagina pentru client,cand apasam pe butonul de orders ni se va deschide o alta fereastra care este pagina pentru comenzi,cand apasam pe butonul de products ni se va deschide o alta fereastra care este pagina pentru produse, iar cand apasam pe butonul exit se va iesi din aplicatie. In pagina de client,vom gasi patru textfield-uri in care putem introduce id-ul unui client,numele,prenumele si numarul de telefon. De asemenea gasim sase butoane: unul pentru back,unul pentru vizualizarea tuturor clientilor,unul pentru a insera un client dupa ce ai introdus date,unul pentru a sterge un client,unul pentru update si unul de exit din aplicatie. La fel si pentru comenzi si produse,doar ca la produse poti introduce in textfield-uri, id-ul,numele,pretul si cate produse sunt in stoc. La comenzi poti introduce in cele patru textfield-uri id-ul comenzii,id-ul produsului,id-ul clientului si cantitatea dorita. La comenzi apare o diferenta la butoane fata de client si produse,nu mai avem butonul de update.

1. **Proiectare**

Pentru a proiecta aceasta aplicatie, am ales sa utilizez sase pachete in care am impartit clasele:

* Pachetul BusinessLogic care cuprinde clasele: ClientBLL,OrderBLL si ProductBLL
* Pachetul Connection care cuprinde clasa: ConnectionF
* Pachetul DataAccess care cuprinde clasele: AbstractDAO, ClientDAO, OrderDAO, ProductDAO
* Pachetul Model care contine clasele: Clients,Products si Orders
* Pachetul Presentation care continue clasele: ClientController, InterfataClient, InterfataOrder,InterfataPrincipala,InterfataPrincipalaController,InterfataProduct, COrderController si ProductController
* Pachetul Validator care cuprinde clasele: ClientValidator,OrderValidator si ProductValidator

Pachetul BusinessLogic contine o clasa pentru efectuarea de operatii asupra unui client, a unui produs sau a unei comenzi. In aceasta parte a programului, datele din baza de date sunt procesate si pregatite pentru a fi afisate. Acest layer este conectat la layer-ul si pachetul DataAccess, care contine o clasa pentru realizarea conexiunii la baza de date. Un alt pachet important este pachetul Model, care contine modelele entitatilor utilizate in aceasta aplicatie.

Diagrama UML a claselor:

A picture containing text, drawing, sketch, schematic

Description automatically generated

1. **Implementare**

In pachetul BusinessLogic am implementat trei clase : ClientBLL,OrderBLL si ProductBLL. Clasa ClientBLL gestionează interacțiunea intre interfata utilizator si nivelul de acces la date (Data Access) pentru a efectua operatii CRUD (create, read, update, delete) asupra obiectelor de tip Clients. La fel si pentru clasele OrderBLL si ProductBLL care se comporta exact la fel si fac acelasi lucru.

In pachetul Connection am implementat clasa ConnectionF care este clasa responsabila cu conexiunea cu baza de date. De asemenea, are metode de inchidere a conexiunii. Aceasta stocheaza acreditarile de identificare, cum ar fi adresa serverului bazei de date, numele de utilizator, parola si driverele necesare.

In pachetul DataAccess am implementat patru clase: AbstractDAO,ClientDAO,OrderDAO si ProductDAO. In clasa AbstractDAO implementez o metoda in care formez interogarea, metoda createObjects primește un rezultat de interogare si creeaza o lista de obiecte de tipul specificat în clasa AbstractDAO. Aceasta utilizează reflectia pentru a crea si initializa obiectele folosind constructorul și metodele setter ale obiectului.

Metoda findById primeste un ID si returneaza obiectul corespunzator din baza de date. Aceasta construieste o interogare SELECT utilizand campul ID si apoi apeleaza metoda createObjects pentru a crea obiectul din rezultatul interogarii. Metoda getTable primeste un tip si returneaza numele corect al tabelei corespunzatoare acestui tip. Metoda inserare este utilizata pentru a insera un obiect in baza de date. Metoda update este utilizata pentru a actualiza un obiect existent in baza de date. Metoda createTable primeste o lista de obiecte si returneaza o tabela de tip JTable care poate fi utilizata in interfata grafica pentru afisarea datelor. Metoda findAll returneaza o lista cu toate obiectele din baza de date. Metoda stergere este utilizata pentru a sterge un obiect din baza de date in functie de ID-ul specificat. Cele trei clase ClientDAO,OrderDAO si ProductDAO extind clasa AbstractDAO si in fiecare avem un constructor.

In pachetul Model se gasesc modelele entitatilor utilizate in aceasta aplicatie: Clients,Orders,Products.

In pachetul Presentation am implementat opt clase dar care sunt legate doua cate doua: InterfataClient si ClientController,InterfataOrder si OrderController,InterfataProduct si ProductController si InterfataPrincipala si InterfataPrincipalaController. In clasele de interfata efectiv realizez ferestrele pentru home screen,pentru clienti,pentru comenzi si pentru produse. Clasele Controller fac legatura intre interfata utilizator si clasele din pachetul BusinessLogic.

In pachetul Validator am implementat trei clase: ClientValidator care are metode de validare a numelui,prenumelui,numarului de telefon si apoi o metoda care le verifica pe toate si returneaza cuvantul ,,Corect” daca datele introduse sunt corecte sau ,,Incorect” daca datele sunt invalide. La fel si pentru OrderValidator unde verificam ca cantitatea introdusa sa fie mai mare ca 0 apoi facem o metoda in care returnam ,,Corect” daca totul e in regula. La fel si pentru ProductValidator unde verificam validarea numelui introdus ,a pretului introdus(>0) si a produselor in stoc introduse(>0). O alta metoda returneaza ,,Corect” daca toate au fost indeplinite.

1. **Rezultate**

Rezultatele unei testari care e facuta de utilizator se poate vedea in interfata. El poate vizualiza sub forma de tabel clientii,comenzile si produsele si de asemenea poate face operatii de inserare,update,stergere. De asemenea, tabelele rezultate in urma testarii se pot vizualiza si in tabelele din IntelliJ,respectiv MySQL Workbench.

1. **Concluzii**

Din aceasta tema, eu am invatat sa proiectez si sa structurez un sistem de gestionare a comenzilor pentru un depozit. Acest lucru implica identificarea entitatilor si a relatiilor dintre ele. De asemenea am mai invatat cum sa interactionez cu o baza de date pentru a stoca datele legate de comenzile din depozit. Acest lucru poate implica crearea si executarea de interogari SQL, gestionarea conexiunii la baza de date si utilizarea obiectelor si metodelor specifice pentru manipularea datelor. O posibila dezvoltare ar fi adaugarea de functionalitati suplimentare pentru a extinde aplicatia. Acestea pot include gestionarea inventarului, urmarirea livrarilor si a retururilor, generarea automata a rapoartelor si a statisticiilor, integrarea cu sisteme externe (de exemplu, furnizori sau sisteme de plata).

1. **Bibliografie**
2. Connect to MySQL - <https://www.baeldung.com/java-jdbc>
3. Java Reflection - <https://www.baeldung.com/java-reflection>
4. DAO - <https://www.baeldung.com/simplifying-the-data-access-layer-with-spring-and-java-generics>
5. SQL dump file - <https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html>
6. PT - <https://dsrl.eu/courses/pt/materials/PT2023_A3_S2.pdf>