DOCUMENTATIE

TEMA *3*

ORDERS MANAGEMENT

NUME STUDENT: CRISTE CASIAN
GRUPA: 30224

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei: Dezvoltarea unei aplicații de gestionare a clienților, comenzilor și produselor.

Obiective secundare:

1. Crearea unei interfețe grafice intuitive și atrăgătoare pentru utilizatori. (Detaliat în cadrul clasei MainMenu)

2. Implementarea funcționalității de vizualizare a clienților, comenzilor și produselor într-un tabel. (Detaliat în cadrul clasei OrdersMenu)

3. Implementarea funcționalității de creare a unei comenzi și actualizarea stocului de produse. (Detaliat în cadrul clasei OrdersMenu)

4. Implementarea funcționalității de modificare și ștergere a clienților, comenzilor și produselor. (Detaliat în cadrul claselor ClientsMenu, OrdersMenu și ProductsMenu)

5. Integrarea aplicației cu baza de date pentru stocarea și gestionarea datelor. (Detaliat în cadrul claselor DAO - ClientDAO, OrdersDAO, ProductsDAO)

6. Asigurarea securității datelor utilizatorilor și protejarea împotriva accesului neautorizat. (Detaliat în cadrul claselor DAO și al funcționalităților relevante)

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

Cadrul de cerințe funcționale și non-funcționale:

Cerințe funcționale:
1. Vizualizarea listei de clienți existenți.
2. Adăugarea unui nou client.
3. Modificarea informațiilor unui client existent.
4. Ștergerea unui client.
5. Vizualizarea listei de comenzi existente.
6. Adăugarea unei noi comenzi.
7. Modificarea unei comenzi existente.
8. Ștergerea unei comenzi.
9. Vizualizarea listei de produse existente.
10. Adăugarea unui nou produs.
11. Modificarea informațiilor unui produs existent.
12. Ștergerea unui produs.

Cerințe non-funcționale:
1. Interfața grafică intuitivă și atrăgătoare.

2. Performanță și timp de răspuns rapid.
3. Securitatea datelor și protecția împotriva accesului neautorizat.
4. Stabilitatea și fiabilitatea aplicației.

Cazurile de utilizare și diagramele:

1. Vizualizarea listei de clienți:
   - Utilizatorul deschide aplicația.
   - Utilizatorul accesează meniul "Clients".
   - Aplicația afișează lista de clienți existenți.

2. Adăugarea unui nou client:
   - Utilizatorul deschide aplicația.
   - Utilizatorul accesează meniul "Clients".
   - Utilizatorul apasă butonul "Create".
   - Utilizatorul completează informațiile despre noul client.
   - Utilizatorul apasă butonul "Save" pentru a adăuga clientul în sistem.
   - Aplicația actualizează lista de clienți și afișează noul client adăugat.

3. Modificarea informațiilor unui client existent:
   - Utilizatorul deschide aplicația.
   - Utilizatorul accesează meniul "Clients".
   - Utilizatorul selectează clientul dorit din lista existentă.
   - Utilizatorul apasă butonul "Modify".
   - Utilizatorul modifică informațiile dorite ale clientului.
   - Utilizatorul apasă butonul "Ok" pentru a salva modificările.
   - Aplicația actualizează lista de clienți și afișează modificările.

4. Ștergerea unui client:
   - Utilizatorul deschide aplicația.
   - Utilizatorul accesează meniul "Clients".
   - Utilizatorul selectează clientul dorit din lista existentă.
   - Utilizatorul apasă butonul "Delete".
   - Utilizatorul confirmă acțiunea de ștergere.
   - Aplicația elimină clientul din sistem și actualizează lista de clienți.

5. Vizualizarea listei de comenzi:
   - Utilizatorul deschide aplicația.
   - Utilizatorul accesează meniul "Orders".
   - Aplicația afișează lista de comenzi existente.

6. Adăugarea unei noi comenzi:
   - Utilizatorul deschide aplicația.
   - Utilizatorul accesează meniul "Orders".
   - Utilizatorul apasă butonul "Create".

- Utilizatorul selectează clientul și produsul pentru comandă.
- Utilizatorul completează cantitatea dorită.
- Utilizatorul apasă butonul "Ok" pentru a adăuga comanda în sistem.
- Aplicația actualizează lista de comenzi și afișează noua comandă adăugată.

7. Vizualizarea listei de produse:
- Utilizatorul deschide aplicația.
- Utilizatorul accesează meniul "Products".
- Aplicația afișează lista de produse existente.

8. Adăugarea unui nou produs:
- Utilizatorul deschide aplicația.
- Utilizatorul accesează meniul "Products".
- Utilizatorul apasă butonul "Create".
- Utilizatorul completează informațiile despre noul produs.
- Utilizatorul apasă butonul "Save" pentru a adăuga produsul în sistem.
- Aplicația actualizează lista de produse și afișează noul produs adăugat.

9. Modificarea informațiilor unui produs existent:
- Utilizatorul deschide aplicația.
- Utilizatorul accesează meniul "Products".
- Utilizatorul selectează produsul dorit din lista existentă.
- Utilizatorul apasă butonul "Modify".
- Utilizatorul modifică informațiile dorite ale produsului.
- Utilizatorul apasă butonul "Ok" pentru a salva modificările.
- Aplicația actualizează lista de produse și afișează modificările.

10. Ștergerea unui produs:
- Utilizatorul deschide aplicația.
- Utilizatorul accesează meniul "Products".
- Utilizatorul selectează produsul dorit din lista existentă.
- Utilizatorul apasă butonul "Delete".
- Utilizatorul confirmă acțiunea de ștergere.
- Aplicația elimină produsul din sistem și actualizează lista de produse.

3. Proiectare

În cadrul proiectării OOP a aplicației cu clienți, comenzi și produse, se pot utiliza următoarele elemente:

1. Diagrama UML de clase:
- Clasa `Client` pentru reprezentarea informațiilor despre un client.
- Clasa `Orders` pentru reprezentarea informațiilor despre o comandă.
- Clasa `Products` pentru reprezentarea informațiilor despre un produs.
- Clasa `ClientDAO` pentru gestionarea operațiilor de acces la date pentru clienți.

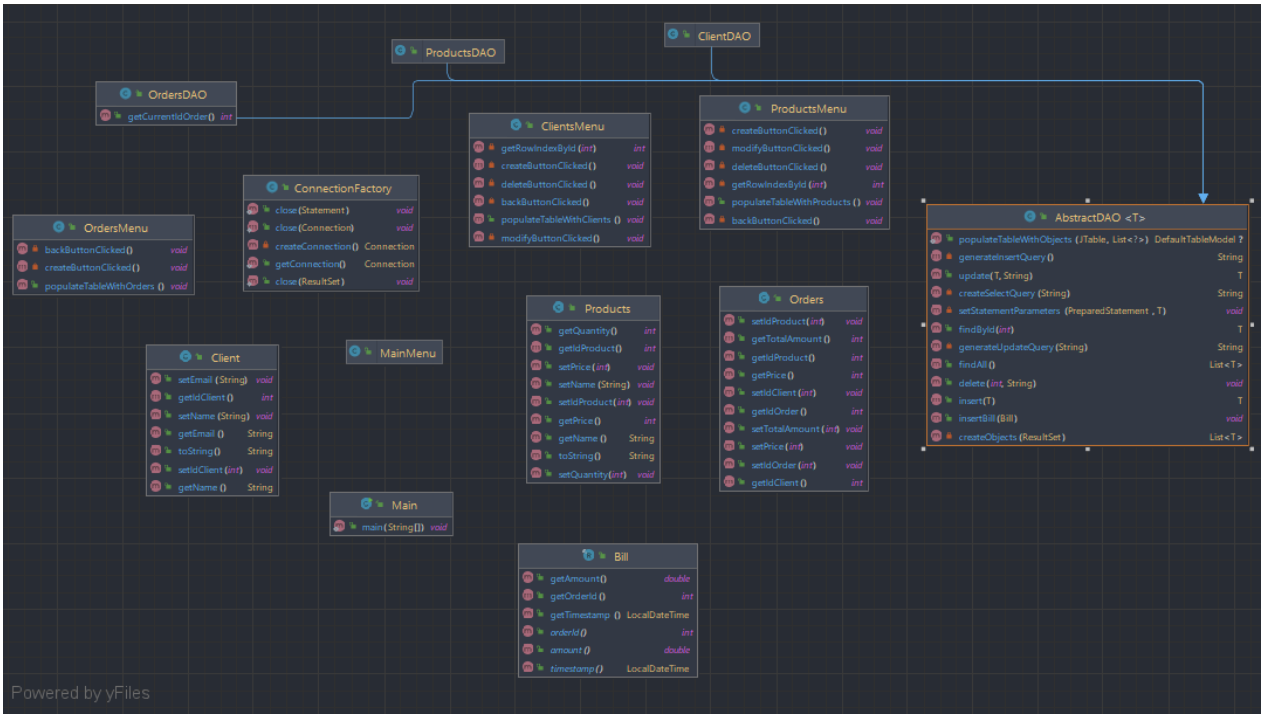
- Clasa `OrderDAO` pentru gestionarea operațiilor de acces la date pentru comenzi.
- Clasa `ProductDAO` pentru gestionarea operațiilor de acces la date pentru produse.
- Clasa `MainMenu` pentru reprezentarea ferestrei principale a aplicației.
- Clasa `ClientsMenu` pentru reprezentarea ferestrei de gestionare a clienților.
- Clasa `OrdersMenu` pentru reprezentarea ferestrei de gestionare a comenzilor.
- Clasa `ProductsMenu` pentru reprezentarea ferestrei de gestionare a produselor.

2. Diagrama UML de pachete:
- Pachetul `BusinessLogic.Model` pentru clasele `Client`, `Order` și `Product`.
- Pachetul `BusinessLogic` pentru clasele `ClientDAO`, `OrderDAO` și `ProductDAO`.
- Pachetul `DataAccess` pentru gestionarea accesului la date în baza de date.
- Pachetul `Presentation` pentru clasele care se ocupă de interfața grafică.

3. Structurile de date folosite:
- Listele pentru stocarea și manipularea obiectelor de tip `Client`, `Order` și `Product`.

4. Interfețe definite:
- Interfața `DAO` pentru definirea metodelor de acces la date comune pentru clasele `ClientDAO`, `OrderDAO` și `ProductDAO`.

5. Algoritmi folosiți:
- Algoritmi pentru adăugarea, modificarea și ștergerea obiectelor din listele de clienți, comenzi și produse.
- Algoritmi pentru accesul la date în baza de date, prin intermediul claselor `ClientDAO`, `OrderDAO` și `ProductDAO`.



4. Implementare

Clase și metode importante:

1. Clasa `Client`:
   - Campuri:
     - `idClient` (int) - identificatorul unic al clientului
     - `name` (String) - numele clientului
     - `email` (String) - adresa de email a clientului
   - Metode:
     - Constructori pentru inițializarea obiectelor de tip `Client`
     - Metode de acces (getteri și setteri) pentru fiecare câmp
     - Metode suplimentare pentru operații specifice, cum ar fi afișarea informațiilor despre client

2. Clasa `Orders`:
   - Campuri:
     - `idOrder` (int) - identificatorul unic al comenzii
     - `idClient` (int) - identificatorul clientului asociat comenzii
     - `idProduct` (int) - identificatorul produsului comandat
     - `quantity` (int) - cantitatea produsului comandat
     - `totalAmount` (double) - suma totală pentru comanda

   - Metode:
     - Constructori pentru inițializarea obiectelor de tip `Order`
     - Metode de acces (getteri și setteri) pentru fiecare câmp

3. Clasa `Products`:
   - Campuri:
     - `idProduct` (int) - identificatorul unic al produsului
     - `name` (String) - numele produsului
     - `price` (double) - prețul produsului
     - `quantity` (int) - cantitatea disponibilă a produsului

   - Metode:
     - Constructori pentru inițializarea obiectelor de tip `Product`
     - Metode de acces (getteri și setteri) pentru fiecare camp

4. Clasa `AbstractDAO`:
   - Această clasă abstractă servește drept schelet pentru clasele DAO specifice (ex. `ClientDAO`, `OrderDAO`, `ProductDAO`).
   - Conține metode generice pentru operațiile de bază (inserare, actualizare, ștergere) care sunt comune tuturor claselor DAO.
   - Metodele abstracte specifice fiecărei clase DAO sunt implementate în clasele derivate.

- Metodele generice includ:
  - `findAll()` - returnează o listă cu toate obiectele de tipul specific din baza de date
  - `findById(int id)` - returnează un obiect de tipul specific cu id-ul specificat din baza de date
  - `insert(T object)` - inserează un nou obiect în baza de date
  - `update(T object)` - actualizează informațiile unui obiect existent în baza de date
  - `delete(int id)` - șterge obiectul cu id-ul specificat din baza de date

5. Clasa `MainMenu`:
  - Această clasă reprezintă fereastra principală a aplicației.
  - Afisează un meniu cu opțiuni pentru a accesa ferestrele specifice (ex. `ClientsMenu`, `OrdersMenu`, `ProductsMenu`).
  - Metodele importante includ:
    - Metode pentru gestionarea evenimentelor de apăsare a butoanelor pentru accesarea ferestrelor specifice

6. Clasa `ClientsMenu`:
  - Această clasă reprezintă fereastra pentru gestionarea clienților.
  - Afișează o listă cu toți clienții existenți în baza de date și opțiuni pentru adăugarea, modificarea și ștergerea unui client.
  - Metodele importante includ:
    - Metode pentru gestionarea evenimentelor de apăsare a butoanelor și interacțiunea cu clasa `ClientDAO` pentru a realiza operațiile specifice (ex. inserare, actualizare, ștergere)

7. Clasa `ProductsMenu`:
  - Această clasă reprezintă fereastra pentru gestionarea produselor.
  - Afișează o listă cu toate produsele existente în baza de date și opțiuni pentru adăugarea, modificarea și ștergerea unui produs.
  - Metodele importante includ:
    - Metode pentru gestionarea evenimentelor de apăsare a butoanelor și interacțiunea cu clasa `ProductDAO` pentru a realiza operațiile specifice (ex. inserare, actualizare, ștergere)

8. Clasa `OrdersMenu`:
  - Această clasă reprezintă fereastra pentru gestionarea comenzilor
  - Afișează o listă cu toate comenzile existente în baza de date și opțiuni pentru crearea unei noi comenzi.
  - Metodele importante includ:
    - `populateTableWithOrders()` - populează tabela cu comenzile existente din baza de date
    - `createButtonClicked()` - acțiunea realizată la apăsarea butonului "Create" pentru crearea unei noi comenzi, inclusiv interacțiunea cu clasele `ClientDAO`, `ProductDAO` și `OrderDAO`
    - `backButtonClicked()` - acțiunea realizată la apăsarea butonului "Back" pentru a reveni la fereastra principală

6. Clasa `Bill`:
   - Această clasă reprezintă o factură asociată unei comenzi.
   - Conține informații despre id-ul comenzii, suma totală și data facturării.
   - Metodele importante includ:
     - Constructor pentru inițializarea obiectului
     - Metode getter și setter pentru accesarea și modificarea atributelor obiectului

7. Clasa `ConnectionFactory`:
   - Această clasă reprezintă conexiunea cu baza de date
   - Se folosește de JDBC

8. Clasele `ClientDAO`, `OrdersDAO`, `ProductsDAO`
   - Aceste clase facilitează lucrul cu Reflection, folosindu-se de parametrii generici.
   - Extinde clasa `AbstractDAO`

9. Clasa `Main`
   - Pornește programul apelând `MainMenu`.

5. Concluzii

În concluzie, realizarea acestei aplicații de gestionare a unui magazin online cu funcționalități de gestionare a clienților, produselor și comenzilor a oferit oportunitatea de a exersa și aprofunda cunoștințele în programarea orientată pe obiecte și utilizarea limbajului Java.

Prin implementarea diferitelor clase și interfețe, am învățat cum să proiectăm și să organizăm corect codul sursă, utilizând concepte precum încapsularea, moștenirea, polimorfismul și abstracția. Am înțeles importanța utilizării principiilor SOLID pentru a obține un cod flexibil, ușor de întreținut și extensibil.

De asemenea, am învățat cum să utilizăm JDBC pentru a conecta aplicația noastră la o bază de date MySQL și să realizăm operațiuni CRUD (create, read, update, delete) asupra datelor. Am înțeles importanța unei gestionări corecte a resurselor de conexiune și am implementat aceasta utilizând clasa `ConnectionFactory`.

Posibile dezvoltări ulterioare ale aplicației pot include adăugarea de funcționalități suplimentare, cum ar fi gestionarea stocului produselor, gestionarea facturilor și rapoartelor, implementarea unui sistem de autentificare și securitate, optimizarea performanței prin utilizarea unor interogări SQL mai eficiente etc.

În ansamblu, realizarea acestui proiect ne-a permis să aplicăm cunoștințele teoretice într-un context practic, dezvoltându-ne abilitățile de proiectare și implementare a unei aplicații Java.

6. Bibliografie

1. https://stackoverflow.com
2. https://www.w3schools.com