DOCUMENTATIE

TEMA *3*

NUME STUDENT: CHARYYEVA ALTYN
GRUPA: 30224

CUPRINS

1. Obiectivul temei

Cerintele proiectului constau in proiectarea aplicatiei de Gestionare a Comenzilor pentru procesarea comenzilor clienților pentru un depozit. Bazele de date relaționale ar trebui să fie utilizate pentru a stoca produsele, clienții și comenzile. Site-ul aplicația ar trebui să fie proiectată în conformitate cu modelul de arhitectură stratificată și ar trebui să utilizeze (minim) următoarele clase:

- Clase de model - reprezintă modelele de date ale aplicației.

- Clase de logică de afaceri - conțin logica aplicației.

- Clase de prezentare - clase legate de interfața grafică.

- Clase de acces la date - clase care conțin accesul la baza de date.

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

Conform problemei, trebuie să implementăm un sistem de gestionare a comenzilor, ceea ce înseamnă că trebuie să existe un depozit care să conțină produse și trebuie să existe clienți care să poată comanda aceste produse, ceea ce înseamnă că trebuie să avem o bază de date în care să putem stoca informații despre produse, clienți, comenzi ale clienților și comenzi deja efectuate.

Mai mult, vorbim despre clienți, produse și comenzi, ceea ce înseamnă că vom avea nevoie ca acestea să poată fi considerate ca modele în aplicația noastră.

Apoi, trebuie să implementăm o conexiune între baza de date și modele. În cele din urmă, toate acestea trebuie să se întâmple într-un mod ușor de înțeles pentru utilizator, ceea ce înseamnă că trebuie să implementăm o interfață grafică pentru interacțiune cu utilizator.

- ***Caz de utilizare:*** adăugarea unui produs (la fel si pentru client)

Actor primar: angajat

Scenariul principal de succes:

1. Angajatul selectează opțiunea de adăugare a unui nou produs
2. Aplicația va afișa un formular în care se vor afișa detaliile produsului

trebuie să fie inserate

3. Angajatul inserează numele produsului, prețul acestuia și

stocul curent

4. Angajatul face clic pe butonul "Add" (Adaugă)
5. Aplicația stochează datele despre produs în baza de date și

afișează un mesaj de confirmare

Secvență alternativă: Valori nevalabile pentru datele produsului

- Utilizatorul introduce o valoare negativă pentru stocul produsului.
- Aplicația afișează un mesaj de eroare și solicită utilizatorului să

să introducă un stoc valid

- Scenariul revine la etapa 3

- ***Caz de utilizare:*** editarea informatia despre unui produs (la fel si pentru client)

Actor primar: angajat

Scenariul principal de succes:

1. Angajatul selectează opțiunea de editarea informatii despre a unui nou produs
2. Aplicația va afișa un formular în care se vor afișa detaliile produsului
3. Angajatul inserează datele care trebuie sa fie modificate.
4. Angajatul face clic pe butonul "Edit".
5. Aplicația actualizeaza datele despre produs în baza de date și

afișează un mesaj de confirmare.

Secvență alternativă: Valori nevalabile pentru datele produsului

- Utilizatorul introduce o valoare negativă pentru stocul produsului.
- Aplicația afișează un mesaj de eroare și solicită utilizatorului să

să introducă un stoc valid

- Scenariul revine la etapa 3.

- ***Caz de utilizare:*** efectuarea comenzii

  Actor primar: angajat

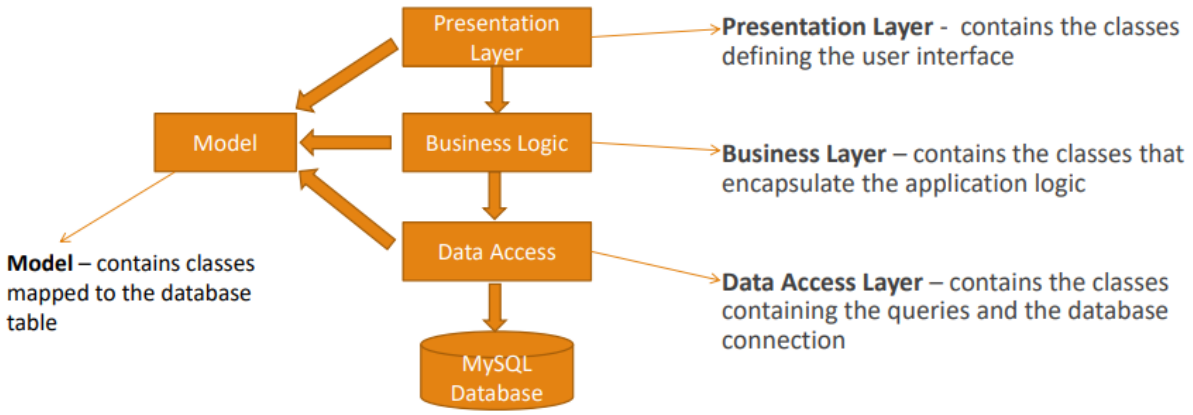
Scenariul principal de succes:

1. Angajatul selecteeaza optiunea de efectuarea comenzii.
2. Aplicatia va afisa un formular in care trebuie sa fie inserate date despre comanda.
3. Angajatul insereaza datele necesare(numele produsului, numele clientului, cantitate produsului si asa mai departe).
4. Angajatul face clic pe butonul "Make order".
5. Aplicația actualizeaza datele despre produs în baza de date și
   afișează un mesaj de confirmare.

3. Proiectare

Proiectarea generală a sistemului.

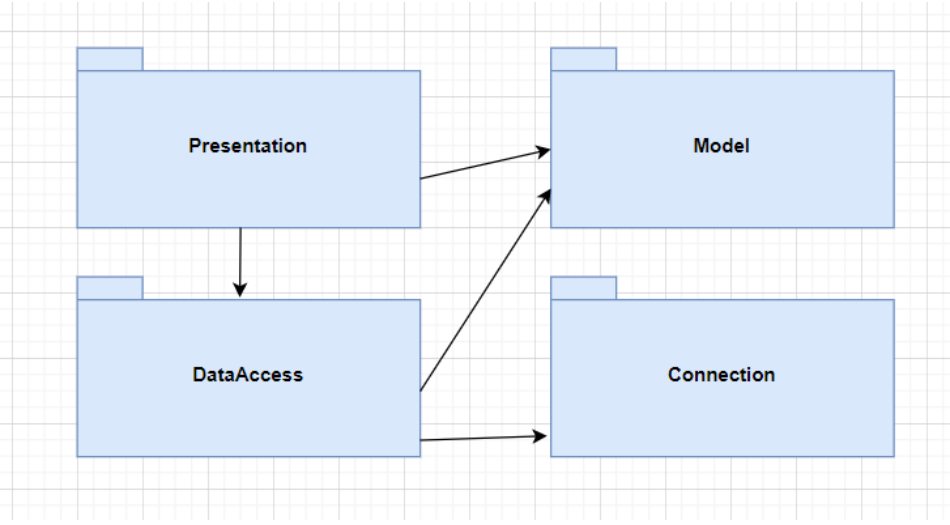
În general, aplicația este formată din mai multe component cum ar fi stratul de prezentare, modelele de date, logica de afaceri și așa mai departe. Fiecare componentă are un rol important în funcționalitatea aplicației.



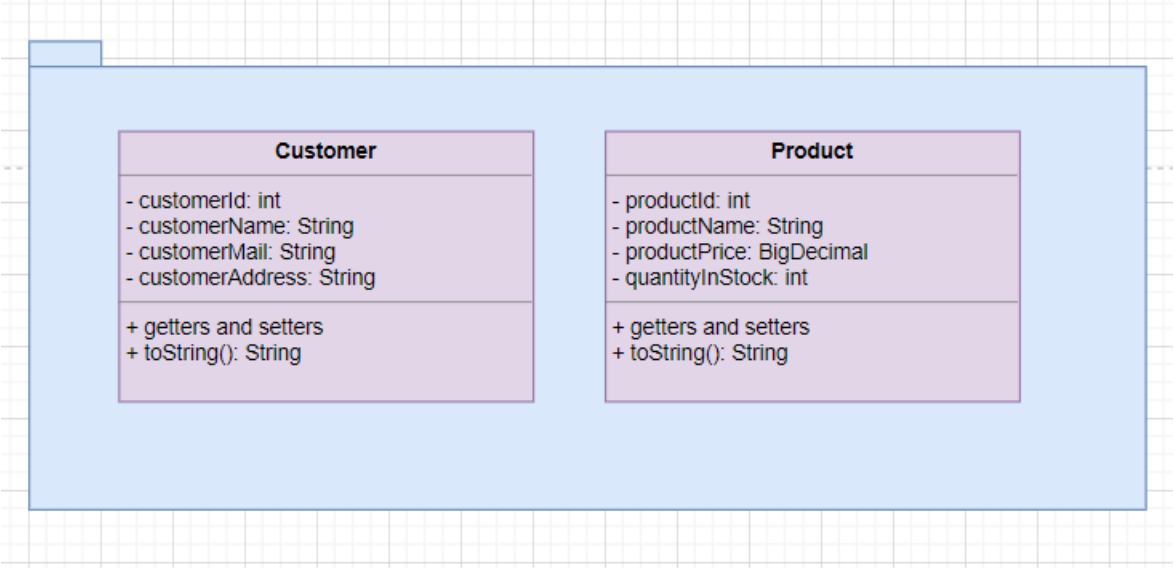
După cum vedem în imagine, la implementarea aplicației vom avea nevoie de un pachet pentru fiecare dintre aceste straturi. Modelul de arhitectură stratificată are următoarele avantaje:

- Cadrul este simplu și ușor de învățat și de implementat.
- Există o dependență redusă, deoarece funcția fiecărui strat este separată de celelalte straturi.
- Testarea este mai ușoară datorită componentelor separate, fiecare componentă poate fi testată individual.
- Costurile generale sunt destul de reduse.

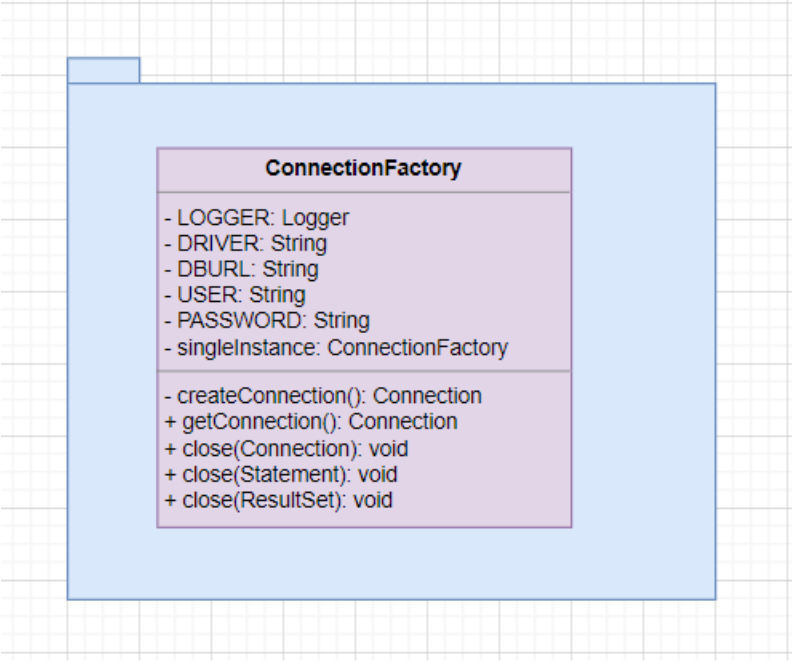
Am folosit acesti patru pachete pentru proiectarea generala a aplicatiei.



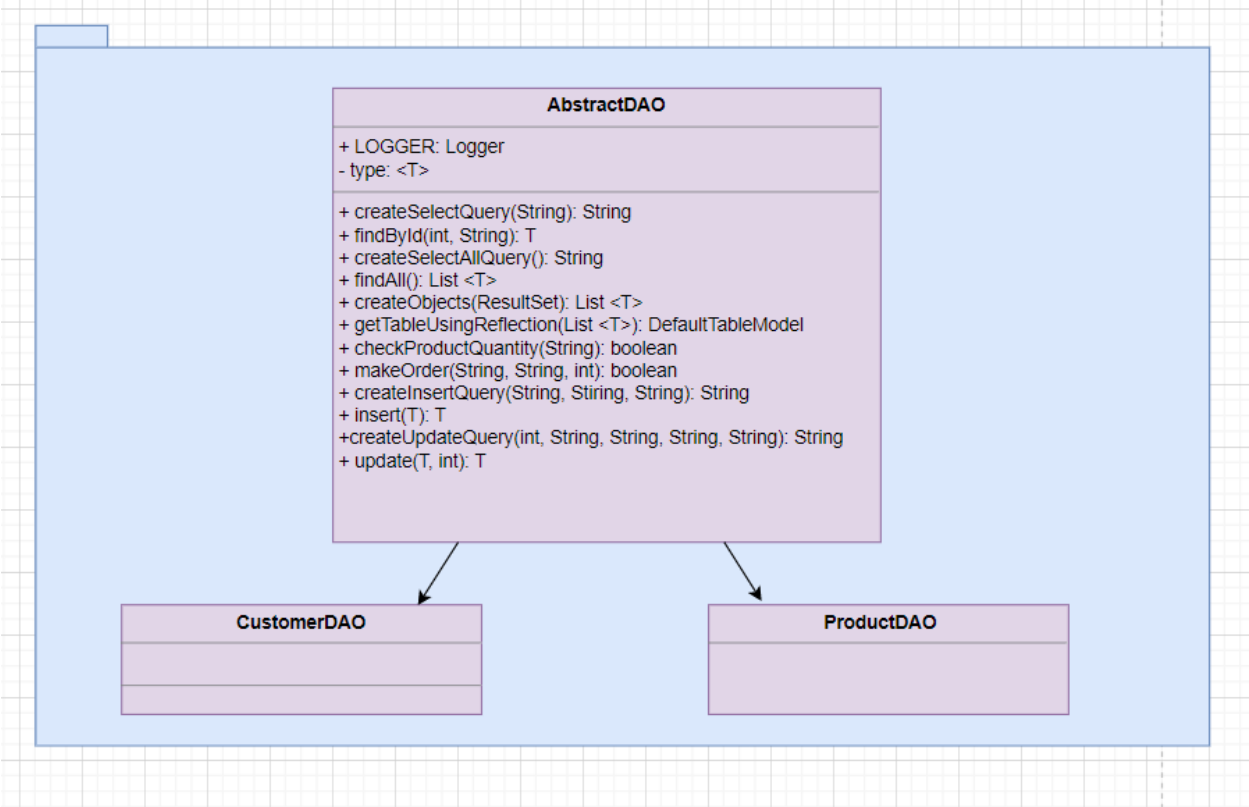
Pacheta Model si clasele sale:



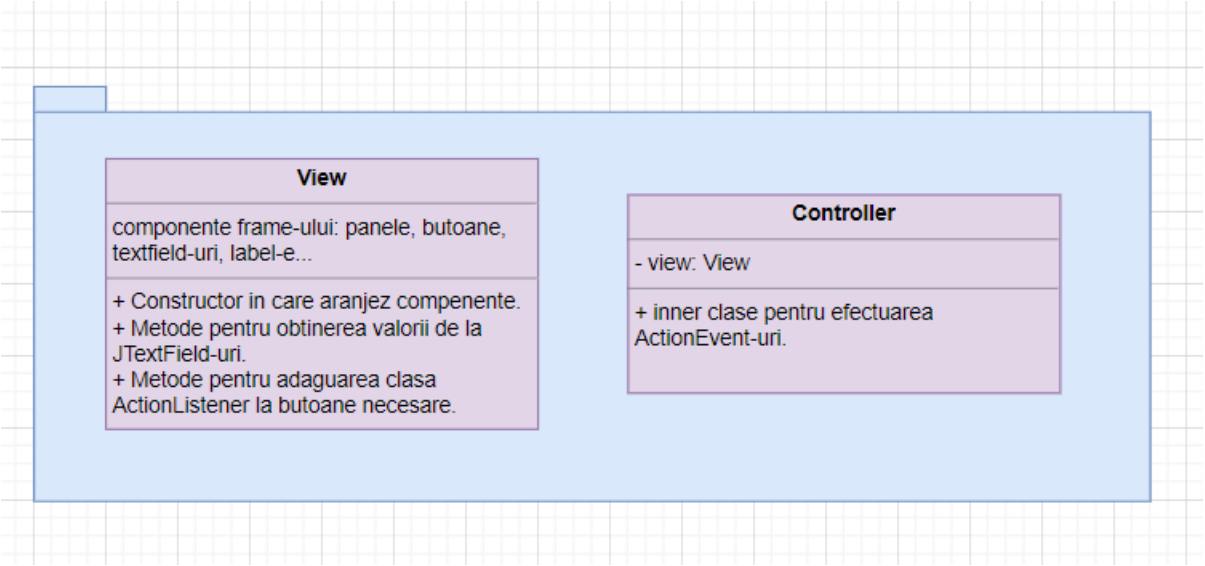
Pacheta Connection si clasa sa:



Pacheta DataAccess si clasele sale:



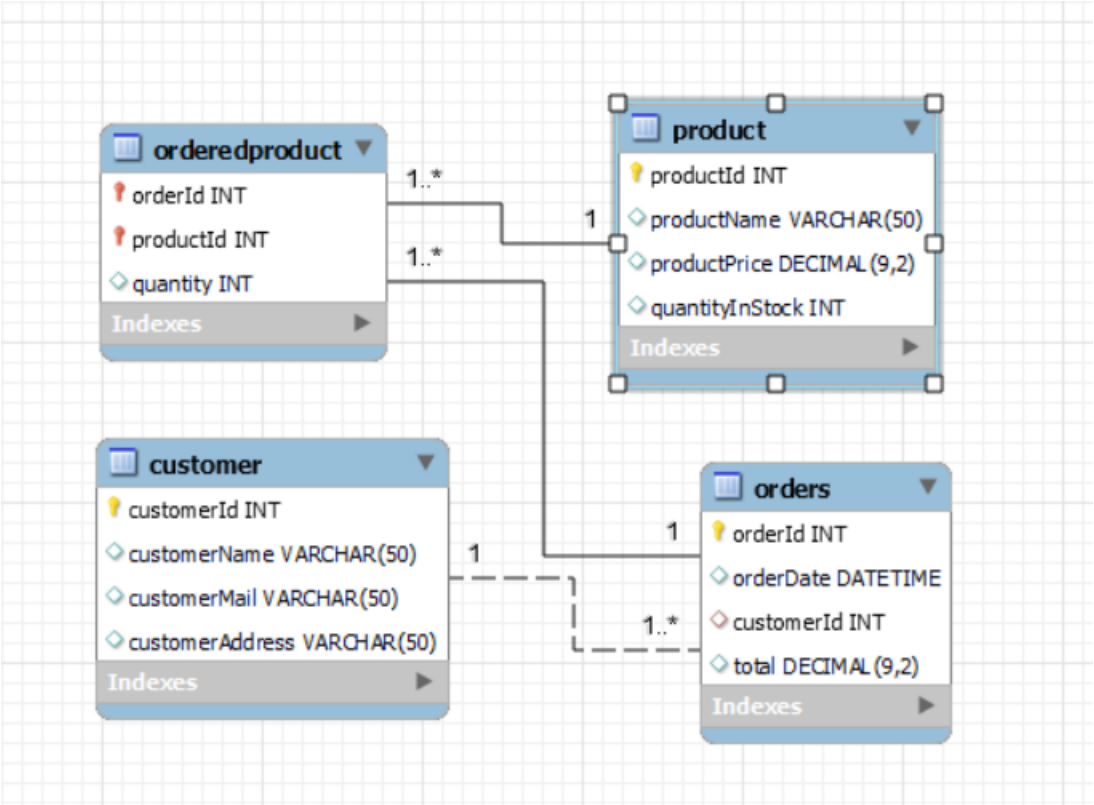
Pacheta Presentation si clasele sale:



4. Implementare

Baza de date a aplicatiei este facuta folosiind mySQLWorkbench. Baza de date se numeste **orders_management** si are urmatoare tabele:

1. Customer
2. Product
3. Orders
4. OrderedProduct



Ca sa implement aplicatia am folosit aceste clase in pachetele acele pe care am aratat in capitoulu anterior.

Pacheta Model

Clasele din pachetul Model au exact aceleași nume ca și numele tabelelor din baza de date și exact aceleași câmpuri ca și numele coloanelor din tabele. Acest mod de implementare a claselor va fi util atunci când se accesează baza de date folosind tehnici de reflecție.

- Clasa Customer

  Clasa Customer are atribute customerId(int), customerName(String), customerMail(String) si customerAddress(String). Clasa are doi constructori una primeste taote campuri celalalt primeste toate campuri in afara de campul ID. Metodele clasei sunt gettere si settere pentru fiecare camp si toString().

- Clasa Product

  Clasa Product are atribute productId(int), productName(String), productPrice(BigDecimal) si quantityInStock(int). Clasa are doi constructori una primeste taote campuri celalalt primeste toate campuri in afara de campul ID. Metodele clasei sunt gettere si settere pentru fiecare camp si toString().

Pacheta Connection

- Clasa ConnectionFactory

  Clasa ConnectionFactory are atribute LOGGER(Logger) – pentru registrarea orice eroare, DRIVER(String) – pentru conectarea folosiind JDBC, DBURL(String) – adresa bazei de date(locatia in mySQLQorkbench si numele bazei de date), USER(String) – numele utilizatorului bazei de date in MySQL, PASSWORD(String) – parola utilizatorului si o instanta de clasa insusi. Metodele sunt:

- createConnection() – creeaza conexiunea cu baza de date.
- getConnection() – returneaza conexiunea creata.
- trei metode de close() pentru inchiderea conexiuni cu cu Statement, ResultSet si Connection.

Pacheta DataAccess

- Clasa AbstractDAO

  Aceasta clasa contine implementarea generala a metodelor CRUD(Create, Read, Update, Delete). Are atribute type(T – care o sa transforma la clasa necesara in timpul de rulare). Are metode createSelectQuery(), createSelectAllQuery(), createInsertQuery() si createUpdateQuery() care toate formeaza un query general pentru basic CRUD metode. Clasa asta mai are metode:

- findById() – foloseste query-ul generat de metoda createSelectQuery, creaza conexiune folosind metode din clasa Connectionfactory si gaseste campul din baza de date potrivit id-ul dat ca parametru.
- findAll() - creaza conexiune folosind metode din clasa Connectionfactory si gaseste toate campurile a tipului dat si returneaza campurile gasite din baza de date ca si un List.
- createObjects() – metoda generala pentru obtinerea valori returnat din ResultSet. Valoriile alea se pun intr-un List si se returneaza.
- getTableUsingReflection() - creaza conexiune folosind metode din clasa Connectionfactory, ia toate campuri clasei potrivit, ia numele campurilor si valoriile campurilor folosiind

tehnnici de Reflection si pune valoriile intr-un tabel. Tabela asta se returneaza de catre metoda.

- checkProductQuantiy() = verifica cantitate a produselor in baza de data daca nu mai este un anumit produs metoda returneaza valoarea “false” un mesaj de warining se afiseaza in GUI.
- makeOrder() - creaza conexiune folosind metode din clasa Connectionfactory, executa o procedura din baza de date. Procedura asta in baza de date actualizeaza campurile necesare in tabelele necesare(creaza un nou rand in tabela Orders si OrderedProducts si decrementeaza valoarea respectiv a produsului in tabela Product).
- Insert() – face inserare in tabela Customer sau Product in baza de date.
- Update() – actualizeaza campuri furnizate de catre parametre primate de catre metoda in baza de date.

    - Clasa CustomerDAO
      Clasa CustomerDAO este mostenita de AbstractDAO poate implementa metode in clasa parinte.
    - Clasa ProductDAO
      Clasa ProductDAO este mostenita de AbstractDAO poate implementa metode in clasa parinte.

Pacheta Presentation

- Clasa View
- Clasa Controller

5. Rezultate

6. Concluzii

7. Bibliografie