DOCUMENTATIE

TEMA 3

NUME STUDENT: Dragotoniu Corina Mădălina

GRUPA: 30224

# CUPRINS

[1. Obiectivul temei 3](file:///C:\Users\40770\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297885)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 4](file:///C:\Users\40770\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297886)

[3. Proiectare 5](file:///C:\Users\40770\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297887)

[4. Implementare 7](file:///C:\Users\40770\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297888)

[5. Rezultate 8](file:///C:\Users\40770\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297889)

[6. Concluzii 8](file:///C:\Users\40770\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297890)

[7. Bibliografie 8](file:///C:\Users\40770\Downloads\PT2021-2022_Documentation_Template%20(1).doc#_Toc95297891)

# Obiectivul temei

Obiectivul acestei teme este de a propune și implementa o aplicație Java ce are scopul de a procesa comenzile clienților unui depozit. Se va folosi o bază de date relațională pentru a stoca date despre produse, clienți si comenzi. În plus, aplicația va fi structurată în pachete, conform unei arhitecturi stratificate.

Pentru a realiza acest obiectiv principal, definim următoarele obiective secundare:

|  |  |  |
| --- | --- | --- |
| **Obiectiv** | **Descriere** | **Capitol** |
| Analiza problemei | Se analizează enunțul problemei și se dezvoltă un design inițial | **2** |
| Alegerea structurilor de date potrivite | Se aleg structuri de date care să faciliteze accesarea datelor din baza de date | **3** |
| Proiectare | Se vor crea clase și pachete în concordanță cu design-ul dorit și se vor scrie algoritmi pentru realizarea prin metode reflexive a operațiilor asupra datelor din baza de date | **3** |
| Implementare | Se va realiza conexiunea la baza de date și implementarea operațiilor asupra datelor din baza de date | **4** |
| Realizarea interfeței grafice | Se va realiza interfața utilizator | **4** |
| Testarea | Se vor efectua teste pentru diferite input-uri | **5** |

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

O bază de date relațională este un tip de bază de date care stochează și oferă acces la punctele de date care sunt legate între ele. Bazele de date relaționale se bazează pe modelul relațional, o modalitate intuitivă și simplă de reprezentare a datelor în tabele. Într-o bază de date relațională, fiecare rând din tabel este o înregistrare cu un cod unic numit cheie. Coloanele din tabel au atribute ale datelor și fiecare înregistrare are, de obicei, o valoare pentru fiecare atribut, facilitând stabilirea relațiilor dintre punctele de date. Așadar, folosind o bază de date relațională putem facilita stocarea datelor referitoare la comenzile din cadrul unui depozit, în cazul de față.

Analizând enunțul temei propuse, observăm că trebuie să începem prin a crea o bază de date relațională (management), care să conțină cel puțin trei tabele – product, client și clientorder. Apoi, în cadrul proiectului Java, creăm pachetele model (unde se vor afla clasele care vor fi mapate la tabelele din baza de date), view si controller (care conține clasele unde se dezvoltă interfața utilizator), bll (adică Business Logic Layer – unde se vor afla clasele care încapsulează logica aplicației), dao (adică Data Access Layer – unde se află clasele care conține interogări asupra bazei de date și conexiunea cu aceasta) și connection (care va conține clasa în care se face conexiunea inițială cu baza de date).

Trecând la partea de modelare, după tabelele din baza de date ne dăm seama care vor fi clasele din pachetul model, și anume Client, Product și Order. Clasele din pachetul dao – AbstractDAO (pentru a implementa prin metode reflexive accesul datelor din baza de date) și ClientDAO, ProductDAO, ClientOrderDAO – clase care vor extinde clasa AbstractDAO, preluând toate metodele ei. Clasele din bll vor fi ClientBLL, ProductBLL și OrderBLL. În pachetele view si controller se vor afla clasele legate de interfața, respectiv de control , iar în pachetul connection vom avea clasa ConnectionFactory.

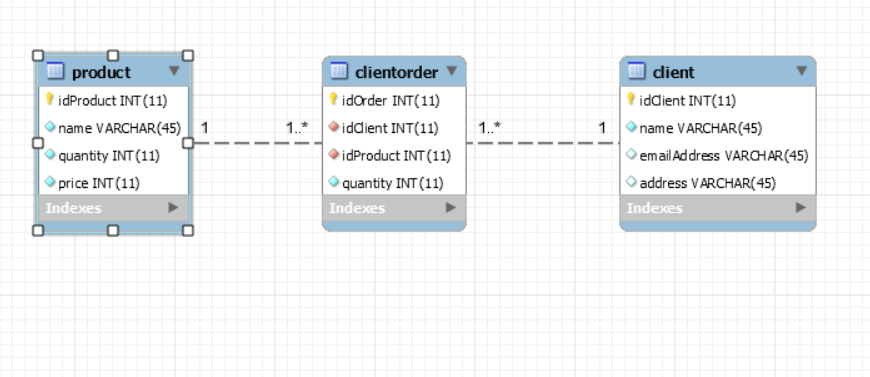
În continuare, voi prezenta cazul de utilizare ideal:

1. Utilizatorul alege tabelul pe care vrea să îl vadă/modifice apăsând unul dintre butoanele Clienti, Produse sau Comenzi.
2. Utilizatorul alege acțiunea pe care vrea să o realizeze asupra tabelului respectiv – View (pentru toate cele trei tabele – îi arată toate datele din tabel), Adauga (pentru toate cele trei tabele – adaugă o nouă linie in tabel, după ce se completează câmpurile care se doresc a fi introduse), Sterge (pentru toate cele trei tabele) și Edit
   1. Dacă apasă View, va vedea toate datele din tabelul ales.
   2. Pentru a adăuga o nouă linie în oricare dintre tabele, va trebuie să completeze câmpurile cu datele cu care vrea să populeze tabelul, apoi să apese butonul Adauga pentru a adăuga în baza de date și View pentru a vedea tabelul modificat.
   3. Pentru a șterge o linie din tabel, va trebuie să furnizeze id-ul liniei care să fie ștearsă, într-un text field din inferfața grafică și să apese butonul Sterge, apoi View pentru a vedea tabelul modificat.
   4. Pentru a edita o linie din tabel, va trebuie sa furnizeze id-ul liniei care să fie editată și datele care să se modifice, să apese butonul Edit și apoi View pentru a vedea tabelul modificat.
3. După ce a terminat operațiile asupra unui tabel, utilizatorul poate alege alt tabel sau să închidă interfața grafică.

# 3. Proiectare

O imagine care conține text, perete, interior, negru

Descriere generată automat



Pentru proiectarea acestei aplicații voi folosi Reflection Techniques și voi respecta modelul unei arhitecturi de tip Layered Architecture. Reflection Techniques presupune realizarea unei clase ce va conține metode generice pentru accesarea bazei de date. Arhitectura de tip Layered presupune existența a cel puțin 4 pachete:

* model: acest pachet va cuprinde clasele ce modelează obiectele de interes pentru aplicația de față, și anume: Produs, Client, Comanda (Product, Client, ClientOrder).
* dao: acest pachet conține atât metodele generice pentru efectuarea operațiilor asupra bazei de date, cât și metode pentru fiecare tip de obiect în parte. Metodele generice vor fi cele de INSERT, DELETE și UPDATE, iar metodele ce vor fi implementate în fiecare clasă sunt pentru găsirea unui anume obiect în baza de date, respectiv găsirea tuturor obiectelor de un anumit tip din baza de date.
* bll: acest pachet face legătura între ceea ce dorește utilizatorul și ceea ce oferă baza de date. Este implementată o variantă mai accesibilă a metodelor de INSERT, UPDATE, DELETE, FindById, care folosește de fapt metodele din pachetul dao.
* View si controller: fiind o aplicație destinată utilizatorilor fără cunoștințe în programarea și utilizarea unei aplicații din linia de comandă, trebuie să avem o interfață accesibilă și user-friendly, pentru a facilita accesul la aceasta pentru orice utilizator, indiferent de nivelul său de cunoștințe în utilizarea unui sistem de calcul.

Aplicația mai conține pachetul connection, în care este realizată conexiunea la baza de date, dar despre acesta voi vorbi în detaliu în capitolul 4.

Pentru aceasta aplicație, se va folosi o bază de date relațională, care va conține 3 tabele: product, client și clientorder.

Product va conține atributele idProduct, name, price respectiv quantity. ID-ul va fi cheie primară cu auto-increment,name va fi de tip varchar(45), iar price si quantity de tip int.

Client va conține atributele idClient, name, emailAddress și address. ID-ul va fi cheie primara cu auto-increment, iar restul atributelor vor fi de tip varchar(45).

Clinetorder va conține atributele idOrder – care va fi cheie primară, idProduct și idClient care vor fi chei străine ce fac referința atributelor cu același nume din tabelele Product respectiv Client și quantity, care va fi de tip int și va memora cantitatea de produs cu ID idProduct, cumpărată de clientul cu ID idClient în cadrul comenzii cu ID-ul idOrder.

# 4. Implementare

* Clasa **Client** simbolizează un client ce ar putea plasa o comandă. Acesta este caracterizat de o serie de atribute, idClient de tip Integer, name, emailAddress și address de tip String. Clasa are 2 constructori, unul cu toate atributele și unul cu doar ultimele 3 atribute, deoarece la inserarea în baza de date, id-ul clientului nu contează, acesta fiind ales în baza de date și fiind de tip auto-increment.
* Clasa **Product** conturează un produs care va fi stocat în depozit. Acesta este caracterizat de un nume, de preț respectiv cantitatea din produsul respectiv care se află în depozit. Asemeni clientului, vine cu 2 constructori respectiv settere și gettere pentru fiecare atribut.
* Clasa **ClientOrder** care simbolizează comanda plasată de un anumit client. Această comandă este caracterizată de un ID, de ID-ul produsului care este comandat, de ID-ul clientului care efectueaza comanda respectiv de cantitatea din produs pe care acesta o dorește. Clasa are 2 constructori și settere și gettere pentru fiecare atribut.
* Clasa **ConnectionFactory** este cea care prelucrează conexiunea cu baza de date, având metode pentru crearea conexiunii, pentru conectare și pentru închiderea obiectelor specifice conexiunii cu baza de date: conexiunea, statementurile respectiv seturile de date rezultate. În această clasă avem parametrii care ne definesc conexiunea, precum URL-ul bazei de date, utilizatorul respectiv parola acestuia.
* Clasa **AbstractDAO** face parte din pachetul destinat realizării operațiilor specifice pe baza de date. În această clasă sunt implementate 4 metode de creare a unei interogări MySQL, pentru operațiile de insert, update, delete. Pentru operația de delete, avem o interogare suplimentară, și anume în momentul ștergerii unui produs sau client, sunt șterse și comenzile care erau afiliate obiectului șters. Pentru crearea interogărilor se folosește un StringBuilder. Mai departe, avem 2 metode, findAll() respectiv findById() care sunt suprascrise respectiv implementate de fiecare clasă în parte. Metodele generice pentru operații, sunt cele de insert, delete și update. Spre exemplu, în cadrul metodei de insert, se creează conexiunea la baze de date, după care se construiește un statement folosindu-se metoda de createInsertQuery. Fiind o metodă generică, luăm într-un atribut de tip Class clasa obiectului pentru care se apelează metoda.
* Clasa **ClientDAO** extinde clasa AbstractDAO, având acces la toate metodele
* Clasele **ProductDAO** și **OrdersDAO** extind de asemenea clasa AbstractDAO
* Clasa **ProductBLL** face parte din pachetul bll și instanțiază un obiect de tip ProductDAO pentru a avea acces la operațiile pe baza de date
* Clasa **ClientBLL** este similară clasei ProductBLL și conține metode ce apelează folosind un obiect de tip ClientDAO, operațiile pe baza de date: findClientById, insertClient, deleteClient, getAllClients respectiv updateClient.
* Clasa **OrderBLL** conține un obiect de tip OrdersDAO care ne dă acces la operațiile din OrdersDAO și AbstractDAO.
* De asemenea, mai avem si clasele care se ocupa de interfata, cele care fac parte din pachetele view, respectiv controller.

# 5. Rezultate

# Au fost efectuate mai multe testari, al caror rezultat pozitiv a fost vizibil in interfata grafica, dat si in fisierele txt, care au reprezentat comenzile emise.

# 6. Concluzii

Concluzionând, prin rezolvarea acestei teme mi-am dezvoltat cunoștințele cu privire la arhitectura de tip Layered și la Reflection Techniques. Am reușit prin abstractizare să scriu metode generice pentru orice tip de obiect, care micșorează semnificativ cantitatea de cod din cadrul aplicației. Pe partea de aplicație, aceasta ar putea fi devoltată pentru mai multe tipuri de utilizatori, la momentul de față fiind destinată angajatului care prelucrează fluxul de produse din depozit. Utilizatorii aplicației ar fi adminul, care poate adăuga angajați, angajații, pentru care aplicația ar fi cea de față, și clienți, care ar putea plasa singuri comenzi.

# 7. Bibliografie

<https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_3.pdf>

<https://dsrl.eu/courses/pt/materials/A3_Support_Presentation.pdf>

<https://www.javatpoint.com/java-jtable>

<https://www.jetbrains.com/help/idea/working-with-code-documentation.html>

<https://www.w3schools.com/php/php_mysql_intro.asp>