**DOCUMENTATIE**

**TEMA 3**

**Barbu Andrada Corina**

**Grupa 30224**

Contents

[1. Cerinte Functionale 3](#_Toc135601019)

[2. Obiective 3](#_Toc135601020)

[2.1. Obiectiv Principal: 3](#_Toc135601021)

[2.2. Obective Secundare: 3](#_Toc135601022)

[3. Analiza Problemei 3](#_Toc135601023)

[4. Proiectare 4](#_Toc135601024)

[4.1. Structuri de date 4](#_Toc135601025)

[4.2. Diagrama de clase 5](#_Toc135601026)

[4.3. Algoritmi 5](#_Toc135601027)

[5. Implementare 6](#_Toc135601028)

[6. Concluzii si Dezvoltari Ulterioare 17](#_Toc135601029)

[7. Bibliografie 18](#_Toc135601030)

# Cerinte Functionale

Trebuie implementata o aplicatie de gestionare a comenzilor clientilor intr-un deposit. Bazele de date relationare sunt utilizate paentru a stoca produsele, clientii si comenzile.

# Obiective

## Obiectiv Principal:

Obiectivul principal al acestui proiect consta in dezvoltarea unei aplicatii care are drept scop preluarea si prelucarea comenzilor primite de la clienti de catre un depozit de produse. Astfel, in functie de produsele aflate in stoc, aplicatia va incerca sa efectueze comenzile clientilor, in caz contrar afisand un mesaj ’Stoc indisponibil’. Pentru efectuarea acestor cerinte intr-un mod mai usor si placut, am implementat o interfata grafica(GUI) usor de folosit de catre orice utilizator. Aceasta ii permite administratorului de depozit sa adauge, sa editeze, sa stearga si sa vizualizeze produsele depozitului, iar clientilor sa plaseze/sa comande produsele dorite.

## Obective Secundare:

Pentru indeplinirea obiectivul principal a fost necesara alegerea structurilor de date,crearea unei baze de date in care sa fie stocate informatii ce contribuie la buna functionare a unui depozit de produse, impartirea pe clase si pachete, conectarea la baza de date create, dezvoltarea unor instructiuni prin intermediul carora informatiile din baza de date au fost prelucrate, implementarea solutiei alese si nu in ultimul rand testarea proiectului.

# Analiza Problemei

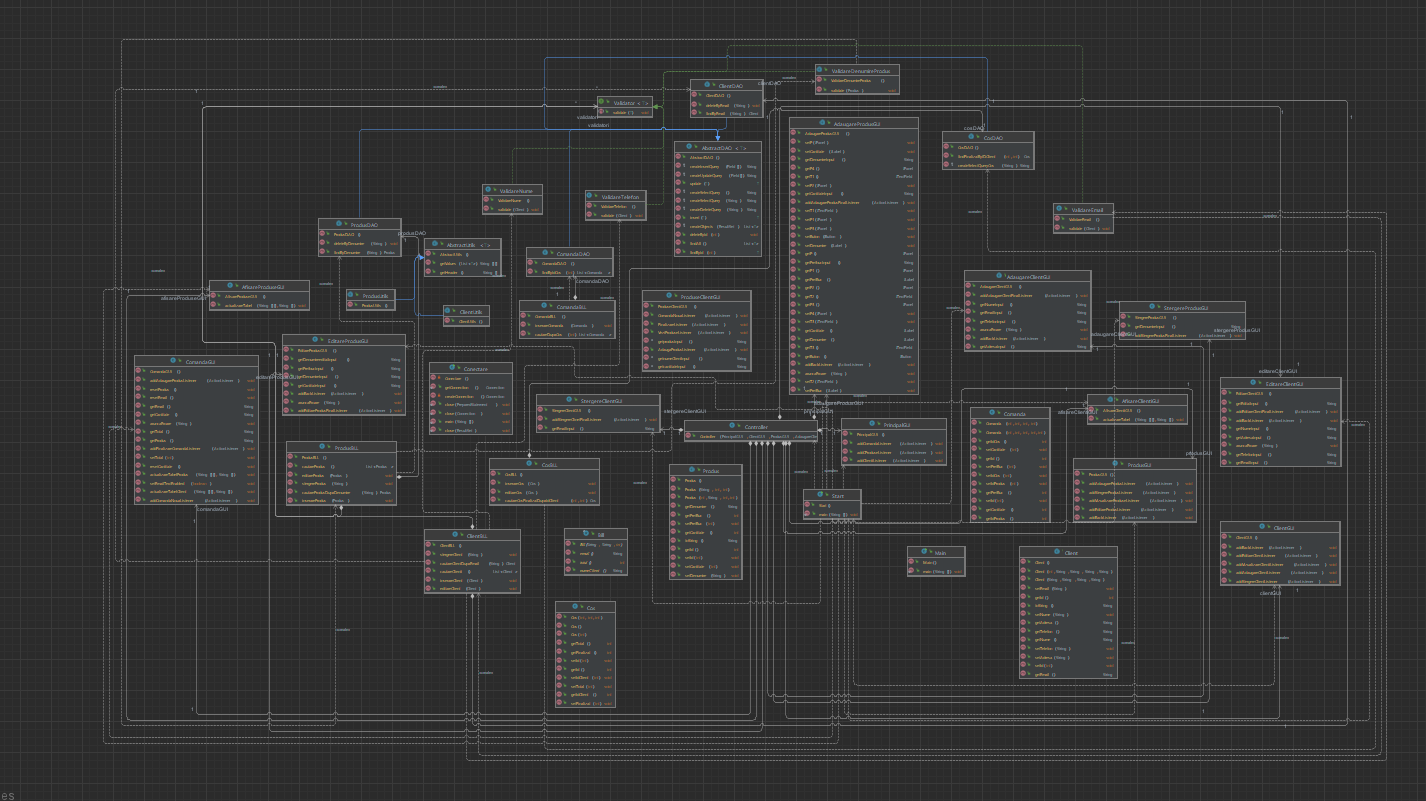
Este crucial ca proiectantul să acorde o atenție deosebită acestei etape de analiză pentru a stabili temelia clară pe care trebuie să se construiască proiectul. Prin cerința problemei, putem deduce că trebuie să ne concentrăm pe preluarea și procesarea comenzilor primite de un depozit de produse de la clienți (un scenariu obișnuit în viața de zi cu zi). Pentru a realiza acest pas, ne-am bazat în principal pe principiul de funcționare a unui depozit, unde elementul principal este stocul de produse. Etapa următoare pe care am analizat-o implică modelarea proiectului, adică colectarea de informații cu privire la datele necesare și locul și modul în care acestea sunt stocate după ce au fost introduse. După aceste analize, am ajuns la concluzia că avem nevoie de o bază de date, care, la o primă vedere, ar trebui să fie structurată astfel: tabela "Client", caracterizată de un ID, nume, adresă, telefon, adresă de email și parola clientului; tabela "Produs", caracterizată de un ID, denumire, preț per bucată și cantitatea de produse din stocul depozitului; și în final, tabela "Comanda", caracterizată de clientul care a plasat comanda, produsele comandate de acesta și prețul total al produselor. La o primă vedere, am putea spune că această repartizare pe tabele a întregului proiect este una foarte bună, dar un lucru foarte important pe care l-am omis este că în MySQL nu se pot realiza relații de tipul "mulți la mulți" (M:N). În cazul nostru, clienții nu pot comanda mai multe produse, iar pentru a remedia această problemă, a fost necesară introducerea unei tabele numită "Cos". În acest moment, o parte din caracteristicile tabelei "Comanda" au fost preluate de tabela "Cos". În urma acestor modificări, tabela "Cos" va fi caracterizată de un ID, clientul care plasează comanda, totalul comenzii și un indicator care indică dacă comanda a fost finalizată sau nu. Tabela "Comanda" este acum caracterizată de un ID, ID-ul tabelei "Cos", ID-ul produsului, prețul per bucată și cantitatea de produse cumpărate. Această bază de date reprezintă "temelia" depozitului de produse. În acest moment, realizăm că trebuie să trecem la îndeplinirea cerințelor problemei și să verificăm buna funcționare a acestora. Acest lucru conduce la introducerea unor clase precum: "Client", "Produs", "Comanda", "Cos", fiecare având o clasă corespunzătoare de tip DAO în care sunt implementate metodele de prelucrare a informațiilor din baza de date și o clasăPentru realiza conexiunea intre baza de date si aceste clase a fost necesare crearea unei noi clase ConectareBD care sa se ocupe de acest lucru prin intermediul conectorului, a adresei URL spre baza noastra de date si nu in ultimul rand a user-ului si a parolei utilizate de MySQL Server.

# Proiectare

## Structuri de date

In acest proiect nu putem spune ca folosim structure de date precise ca in cele anterioare, insa la baza proiectului nostru se afla baza de date create in MySQL Server care ne ajuta sa gestionam informatiile legate de depozitul de produse si clientii acestuia. O alta structura de date poate fi reprezentata de ’interfata Validator’ care poate fi folosita pentru fiecare entitate din baza de date fara a define metode separate pentru acestea.

## Diagrama de clase



## Algoritmi

* Algoritmul claselor DAO
  + Clasa AbstractDAO este o clasă generica care furnizează funcționalități comune pentru accesul la date și manipularea acestora într-o bază de date. Aceasta este o clasă abstractă și poate fi extinsă pentru a crea clase DAO specifice pentru fiecare entitate din sistemul de gestionare a depozitului de produse.
  + Clasa AbstractDAO utilizează genericitatea pentru a permite manipularea și accesul la date pentru orice tip de entitate.Este utilizată pentru operațiile de bază CRUD (Create, Read, Update, Delete) în baza de date.Metodele acestei clase abstracte pot fi suprascrise pentru a adăuga funcționalități specifice pentru fiecare entitate.Clasa se bazează pe conexiunea la baza de date realizată prin intermediul clasei Conectare (care este responsabilă de conectarea și deconectarea de la baza de date).
* Algoritmul clasei Conectare
  + Prin intermediul acestei clase se va realiza conexiunea la baza de date. Aceasta clasa a fost distribuita in pachetul separat ’conectare’, iar ea a fost codificata in asa fel incat sa se poata crea o singura instanta a ei.
* Algoritmul pachetului bll.validatori
  + Acest pachet contine diferite clase care se ocupa cu validarea datelor inainte de introducerea lor in baza de date.

# Implementare

La finalul implementarii, proiectul nostru contine atat o baza de date alcatuita din tabelele Client, Produs, Cos si Comanda, cat si 10 pachete

Asa cum am precizat si la inceput, baza de date este alcatuita din 4 tabele pentru a se respecta relatiile de tip (M:N) dintre clienti si produse. Totodata tabelele din baza de date au fost legate intre ele prin intermediul cheilor straine astfel:tabela Cos contine un camp in care este stocata cheia straina ’id\_client’ care face referire la caracteristica ’id’ din tabela client; tabela Comanda contine campurile ’id\_cos’ si ’id\_produs’ care fac referire la caracteristicile ’id’ din tabela Cos respective ’id’ din tabela Produs. Implementarea bazaei de date este prezentata mai jos:

CREATE DATABASE IF NOT EXISTS ordersmanagement;

use ordersmanagement ;

DROP TABLE IF EXISTS `client`;

create table if not exists `client`

(id int unique not null auto\_increment primary key,

nume varchar(30),

adresa varchar(30),

telefon varchar(10),

email varchar(30));

DROP TABLE IF EXISTS `produs`;

create table if not exists `produs`

(id int unique not null auto\_increment key,

denumire varchar(30),

pretBuc int,

cantitate int);

DROP TABLE IF EXISTS `cos`;

create table if not exists `cos`

(id int unique not null auto\_increment primary key,

idClient int,

total int,

finalizat int,

foreign key (idClient) references client(id));

DROP TABLE IF EXISTS `comanda`;

create table if not exists `comanda`

(id int unique not null auto\_increment primary key,

idCos int,

idProdus int,

pretBuc int,

cantitate int,

foreign key (idCos) references cos(id),

foreign key (idProdus) references produs(id));

Prin intermediul celor 10 pachete am reusit sa realizam un proiect care are o arhitectura stratificata(Layered Architecture).

* Pachetul bll
  + Contine 4 clase: ClientBLL, ComandaBLL, CosBLL si ProdusBLL
  + Clasa ClientBLL: contine o serie de metode prin intermediul carora sunt validate datele care urmeaza sa fie prelucrate(validate)
  + Clasa ComandaBLL: contine o serie de metode prin intermediul carora sunt apelate metodele din clasa ComandaDAO care corespund acestora
  + Clasa CosBLL: contine o serie de metode prin intermediul carora sunt apelate metodele din clasa CosDAO care corespund acestora
  + Clasa ProdusBLL: contine o serie de metode prin intermediul carora sunt validate datele care urmeaza sa fie prelucrate
* Pachetul bll.validatori
  + Contine 4 clase(de validare) si interfata Validator care scoate din calcul implementarea, in unele cazuri, a acelorasi metode de validare pentru clase diferite
  + ValidareEmail: verifica cu ajutorul regex formatul unui email, in cazul in care aceste cu corespunde cu formatul general va arunca o exceptie
  + ValidareNume: verifica cu ajutorul regex formatul numelui, iar in cazul in care acesta nu este correct, arunca o exceptie
  + ValidareTelfon: verifica cu ajutorul regex formatul telefonului, iar in cazul in care acesta nu este corect, arunca o exceptie
  + ValidareDenumireProdus: verifica daca produsul introdus exista deja sau nu, iar in caz afirmativ arunca o exceptie
* Pachetul conectare
  + Este alcatuit din clasa Conectare prin intermediul careia ne este oferit accesul la baza de date creata in MySQL. Aceasta clasa poate fi instantiata o singura data, deci ne vom conecta la baza de date o singura si nu de fiecare data cand dorim sa accesam si sa prelucram informatii din baza de date. Conexiunea la baza de date se realizeaza prin intermediul adresei URL si a user-ului si parolei utilizate de MySQL Server.
* Pachetul dao
  + Continue o clasa AbstractDAO care urmeaza sa fie mostenita de toate lecelalte clase aflate in acest pachet
  + Clasa ClientDAO: contine metodele de delete si findByEmail necesare
  + Clasa ComandaDAO: continue metoda de findByIdCos
  + Clasa CosDAO: continue metodele de createSelectQueyCos si findFinalizatByIDClient
  + Clasa ProdusDAO: continue metode de delete si find in functie de denumirea produsului
* Pachetul Presentation
* Pachetul interfata

Pachetul interfata este cel care contine toate clasele care alcatuiesc interfata grafica(GUI). Totodata contine si clasa Controller care se ocupa cu buna functionare a interfetei grafice. Deci, putem spune ca interfata cu utilizatorul este bazata pe principiul MVC.

Interactiunea cu GUI se face astfel: se alegere metoda de logare: client sau administrator.

* Pachetul Controller

Clasa Controller:

* In fereastra generata de ClientGUI se preia numele clientului atunci cand se apasa butonul “Comanda noua”, se cauta clientul in baza de date in functie de numele acestuia iar daca acesta este gasit se adauga in tabela Cos un nou cos care are initial valoarea din campurile total si finalizat ‘0’; se preiau denumirea produsului, cantitatea si numele clientului atunci cand este apasat butonul “Adauga produs” , se cauta produsul in baza de date dupa denumire, se cauta cosul in baza de date in functie de numele clientului introdus, se compara cantitatea introdusa cu stocul produsului(daca prima este mai mare se va afisa mesajul “Nu este in stoc!”) si daca este mai mica atunci se incrementeaza totalul(cantitate\*pretbuc), se decrementeaza stocul produsului respectiv prin editare, se creaza o noua comanda in tabela Comanda din baza de date, se actualizeaza totalul cosului prin editare si se seteaza in fereastra ‘Total’ pretul final actual pe care clientul trebuie sa-l plateasca; de asemenea factura se va crea la orice validare realizata de butonul ”Finalizare” si va colecta din tabela Comanda toata comenzile care au id-ul cosului egal cu cel actual si campul finalizat din tabela Cos egal cu ’1’.

O imagine care conține captură de ecran, text, software, calculator

Descriere generată automat

O imagine care conține captură de ecran, text, software, Sistem de operare

Descriere generată automat

O imagine care conține text, captură de ecran, software

Descriere generată automat

O imagine care conține text, captură de ecran, diagramă, Paralel

Descriere generată automat

* + Adaugare Client: la apasarea acestui buton se va deschide o fereastra in care trebuie sa introducem datele cerute

`O imagine care conține text, captură de ecran, afișaj, software

Descriere generată automat

* + Editare Client: la apasarea acestui buton se va deschide o fereastra in care: introducem prima data email-ul clientului din baza de date a carui date se doresc a fi editate, iar in celelalte campuri se introduc noile date

O imagine care conține text, captură de ecran, număr, software

Descriere generată automat

* + Sterge Client: la apasarea acestui buton se va deschide o fereastra in care trebuie sa introducem numele si email-ul clientului pe care dorim sa-l stergem din baza de date; informatiile introduse de utilizator in TextField-uri se vor cauta in baza de date astfel incat numele clientului introdus sa fie in stransa legatura cu adresa de email introdusa; daca cele doua caracterizeaza acelasi client, atunci se va efectua stergere, iar in caz contrar nu

O imagine care conține captură de ecran, Dreptunghi

Descriere generată automat

* + Vizualizare Clienti: la apasarea acestui buton se va deschide o fereastra in care va fi afisat un tabel cu toti clientii existenti in baza de date si caracteristicile acestora

O imagine care conține text, captură de ecran, Dreptunghi

Descriere generată automat

* + Adaugare Produs: la fel ca si la Adaugare Client se va deschide o fereastra in care trebuie sa introducem denumirea, pretul/bucata si cantitatea produsului ce urmeaza sa fie adaugat in baza de date

O imagine care conține text, captură de ecran, software, număr

Descriere generată automat

* + Editare Produs: la fel ca si la Editare Client se va deschide o fereastra in care in primul rand va trebui sa introducem denumirea produsului a carui caracteristici din baza de date se doresc a fi editate, iar apoi, in celelalte casute noile date; existent produsului ce urmeaza a fi editat in baza de date se confirma in functie de denumirea introdusa de utilizator

O imagine care conține text, captură de ecran, software, număr

Descriere generată automat

* + Stergere Produs: efectiv se va introduce denumirea produsului ce se doreste a fi sters; pentru a se efectua stergerea acel produs trebuie sa existe in baza de date

O imagine care conține captură de ecran, text, Dreptunghi

Descriere generată automat

* + Vizualizare Produs: la apasarea acestui buton se va deschide o fereastra in care va fi afisat un tabel cu toate produsele existente in baza de date si caracteristicile acestora

O imagine care conține captură de ecran

Descriere generată automat

* Pachetul model
  + Contine 5 clase: Client, Comanda, Cos si Produs si Bill. Aceste clase sunt identice ca si structura si caracteristici cu tabelele create in baza de date
  + Clasa Client: caracterizata de variabilele instanta: id, nume, adresa, telefon, email si parola.
  + Clasa Comanda: caracterizata de variabilele instanta: id, id\_cos, produs\_id, pretbuc, cantitate.
  + Clasa Cos: caracterizata de varibilele instanta: id, id\_client, total, finalizat.
  + Clasa Produs: caracterizata de varibilele instant: id, denumire, pretbuc, cantitate.
* Pachetul start
  + Contine clasa Start care face legatura intre interfata utilizator si controller, ea facand posibila interactiunea utilizatorului cu aplicatia create
* Pachetul utils
  + Contine 3 clase: AbstractUtils, ClientUtils si ProdusUtils
  + Clasa AbstractUtils utilizează genericitatea pentru a permite manipularea și accesul la date pentru orice tip de obiect.Este utilizată pentru a obține valori și anteturi dintr-o listă de obiecte, util pentru afișarea datelor într-un format tabular.
  + Clasa ClientUtils mosteneste clasa AbstractUtils
  + Clasa ProdusUtils mosteneste clasa AbstractUtils

# Concluzii si Dezvoltari Ulterioare

In final, pot spune, ca prin intermediul acestui proiect, am reusit sa imi imbunatesc si sa imi clarific mai bine conceptele programarii orientate pe obiect si folosirea limbajului JAVA datorita faptului ca acest proiect a reprezentat o problema care a necesitat transpunerea ei in subprobleme. De asemenea, am reusit sa asimilez cunostinte noi in ceeea ce priveste lucrul cu o baza de date utilizand programul MySQL Server si conectarea la aceasta.

Proiectu s-ar putea imbunatatii cu urmatoarele:

* dezvoltarea interfetei grafice in ceea ce priveste aspectul visual
* Crearea unei baze de date mai complexe
* Eliberarea unei facturi in urma achizitionarii unor produse

# Bibliografie

[www.wikipedia.com](http://www.wikipedia.com)

[http://www.w3schools.com/](http://l.facebook.com/l.php?u=http%3A%2F%2Fwww.w3schools.com%2F&h=ATPZjV1TKhgHLRV-0S4712kj7dCkIwStvwhHuRmpb3Y0t1pkr3bYB2Uda4GtfYIE_ZZRGB5Dqo3B5A5YdG99rXbUetKOEDbqDG_G5SIoz60_mSeE9EN72z-k6L3dt7IZvTS6dGeD)

<http://stackoverflow.com/questions>