Universitatea Technica Cluj-Napoca

Facultatea de Automatica si Calculatoare

Departamentul de Calculatoare

Semestrul 2

**Tehnici de Programare**

**Tema3-OrderManagement -**

Nume:Danciu Maria Alexandra

Grupa:30222

**Cuprins**

1. **Obiectivul temei**
2. **Analiza problemei,modelare,scenarii,cazuri de utilizare**
3. **Proiectare**
4. **Implementare si testare**
5. **Rezultate**
6. **Concluzii , dezvoltari ulterioare**
7. **Bibbliografie**

**1.Obiectivul Temei**

## Cerinta:

Consider an application OrderManagement for processing customer orders for a warehouse. Relational databases are used to store the products, the clients and the orders. Furthermore, the application uses (minimally) the following classes:

Model classes - represent the data models of the application

Business Logic classes - contain the application logic

Presentation classes – classes that contain the graphical user interface

Data access classes - classes that contain the access to the database Other classes and packages can be added to implement the full functionality of the application.

Obiectivul acestei teme este de a dezvolta o aplicatie care se ocupa cu managementul unui magazine . Aceasta aplicatie interactioneaza cu baza de date si utilizatorul care doreste sa vada prima data ,dupa care sa cumpere produsele existente in magazine . Utilizatorul poate atat sa vada produsele, dar si istoricul comenzilor care s-au facut pana in acel timp. Acesta poate sa vada , sa adauge, sa stearga si sa actualizele produse. La finalizarea unei comenzi se vor afisa datele despre comanda respectiva pe o factura .

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

**Analiza Problemei**

Din analiza cerintei rezulta ca avem de creat o anumita aplicatie , ce prezinta un depozit de produse , un registru de comenzi si un tabel de clienti.

Inca din specificatia problemei apare o prima problema si anume : cazul in care sunt mai multe produse in *Warehouse* decat pot incapea , *over stock* . O alta problema este comandarea a mai multor produse decat cele disponibile pe stoc , *under stock* . In plus am mai adaugat o exceptie si anuma daca produsul nu este disponibil , nu exista in stoc -*NotAvailable.*

Pentru rezolvarea acestui proiect am avut nevoie de cunostinte in domeniul bazelor de date mysql . Comenzile executate in acest proiect au fost folosite cu aceste cunostinte . Pentru fiecare operatie a fost implementata o metoda ce foloseste o interogare sql .

**3.Proiectare**

Pentru a realiza acest proiect , am creat o baza de date “bd” folosind mySQL Workbench 8.0. Am folosit aceasta baza de date pentru a crea tabele ce vor retine datele pe care le voi folosi ulterior in proiect . Pe baza acestei baze de date am creat interogari pentru diferite operatii precum : afisare , adaugare , stergere si actualizare .

Ca structuri de date folosite , am ales folosirea ArrayList<>-urilor pentru a retine datele despre fiecare element dintr-o tabela din baza de date .

In baza de date au fost create trei tabele :

Tabela Client: Aceasta tabela retine clientii care vin la magazine si fac o comada de produse . S-a retinut pentru fiecare client un identificator de tip INT , numele de tip VARCHAR ,prenumele de tip VARCHAR si adresa sa de tip VARCHAR .Cheia primara a acestei tabele este id-ul clientului.

create table if not exists client(

idClient int primary key ,

firstName varchar(50),

lastName varchar(40),

address varchar(40)

);

Tabela Product : Aceasta retine produsele existente in magazin . Pentru fiecare produs am retinut urmatoarele date de identificare : id-ul produsului de tip INT,numele produsului de tip VARCHAR, pretul unui singur produs -DOUBLE de acest tip si stocul in magazin ,de tip INT. Cheia primara a acestei tabele este idProduct.

create table if not exists product(

idProduct int primary key ,

nameProduct varchar(40),

priceProduct double ,

stock int

);

Tabela Orders : Aceasta tabela retine comenzile efectuate de catre clienti . Pentru fiecare comanda se inregistreaza : id-ul comenzii , id-ul clientului care face comanda respectiva , id-ul produsului cumparat si de asemenea totalul de plata al comenzii care va fi calculat ulterior din inmultirea pretului produsului cu cantitatea cumparata . Cheia primara a acestei tabele este

create table if not exists orders(

idOrder int primary key auto\_increment,

idClient int ,

idProduct int,

quantity int ,

FOREIGN KEY (idClient) REFERENCES client(idClient),

foreign key (idProduct) references product(idProduct)

);

Proiectul este structurat pe o serie de pachete , fiecare avand un rol bine determinat .

Pachetele sunt :

* MODEL : Acest pachet contine clasele corespunzatoare tabelelor din baza de date .
* DAO : Data Access Layer : Acest pachet contine clasele ce realizeaza interogarile ce comunica cu baza de date . Aici sunt metodele ce implementeaza query-ul cu care se acceseaza datele din baza de date .
* CONNECTION : Acest pachet contine clasa ConnectionFactory ce face legatura aplicatiei cu baza de date .
* START:Acest pachet contine clasa BILL:generatoare de facturi in urma realizarii comenzii-in format pdf si Clasa Main de unde se va rula aplicatia .Totodata,clasa Main mai contine exceptii analizate in urma realizarii citirii din text file-ul de comenzi,interpretand correct comenzile ce vor fi efectuate asupra bazei de date.

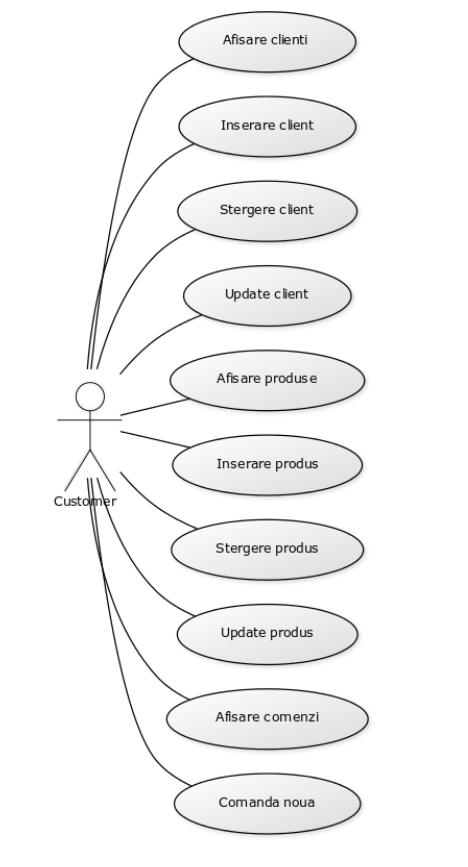
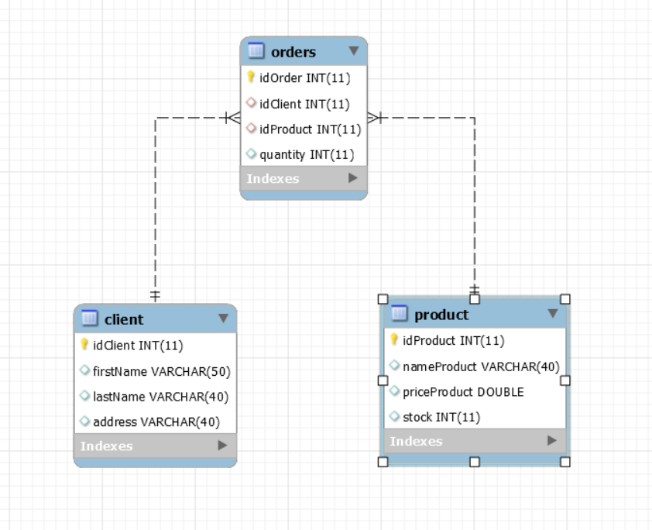
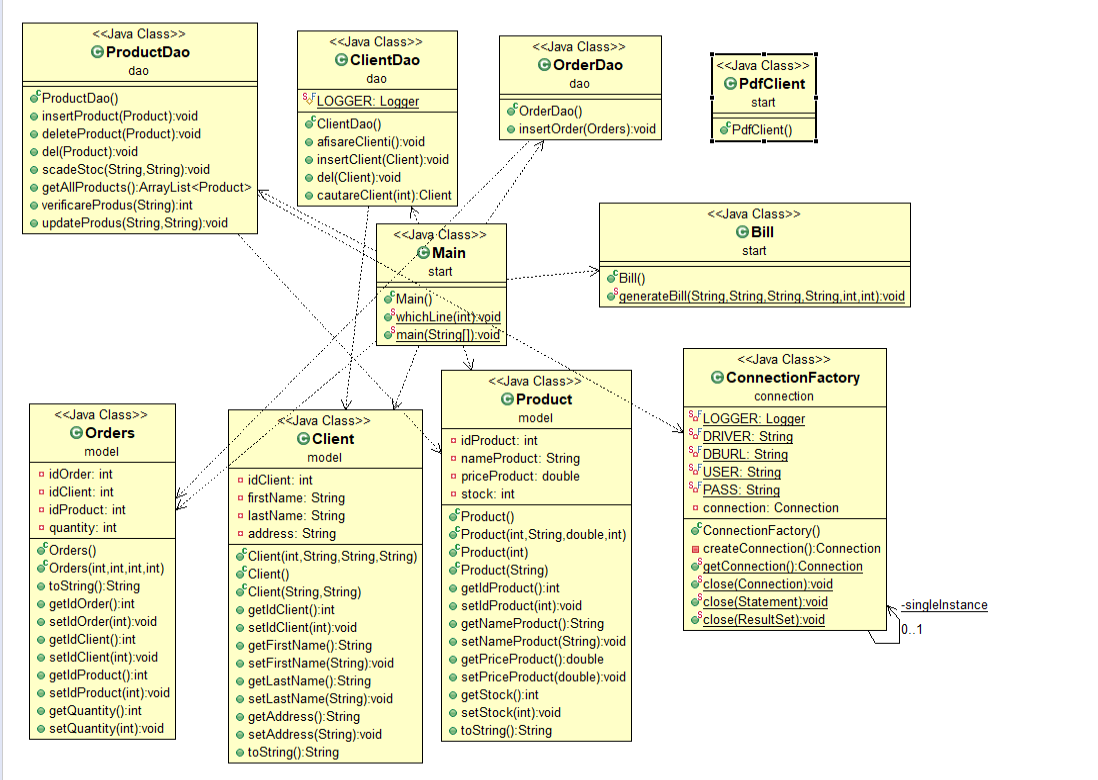
**USER CASE :** 

Diagrama bazei de data



**DIAGRAMA UML DE CLASE : **

Cu ajutorul acestei diagrame UML a avut loc proiectarea intregului proiect, fiind evidentiate pachetele , clasele , metodele dar si legaturile dintre ele , respectand formatul OOP.

**Proiectare Clase**

4.Implementare

Pachetul **MODEL** :

* Clasa **Customer** , clasa **Product** si clasa **Order** :Fiecare defineste pe rand atributele unui client, sau produs sau a unei comenzi . Toate aceste trei clase corespund tabelelor Customer , Product si Order din baza de date a magazinului . Fiecare dintre ele implementeaza interfata Comparable<Object> pentru a sorta ulterior clientii , produsele si comenzile dupa id-ul lor . S-a implementat acest lucru pentru o vizualizare cat mai ordonata a datelor .

Pachetul **DAO** : Acest pachet contine clasele care se ocupa de legatura cu baza de date .

* Clasa **CustomerDAO** : in aceasta clasa sunt implementate metodele ce folosesc interogari
* pentru a accesa date din baza de date .

Sunt implementate cinci metode , fiecare pentru afisarea , inserarea , actualizarea, stergerea si cautarea clientilor . In plus mai sunt implementate si doua metode , una de afisare a datelor in consola si una de verificare daca sa nu existe doi client cu acelasi id . Aceasta ultima metoda a fost implementata pentru a fi folosita de exemplu la adaugarea unui nou client , sa nu adaugi unul care are deja acel id , sau la stergere , sa verifice inainte de a sterge daca exista clientul pe care se doreste sa fie sters , la fel si la actualizare .

Se apeleaza Collections.sort in metoda de getCustomers ( metoda ce returneaza un ArrayList de customeri ) pentru a sorta clientii dupa id .

* Clasa **ProductDAO** : este similara cu clasa CustomerDAO ,doar ca adaptata pentru produsele din magazin . La fiecare metoda implementata se apeleaza ConnectionFactory.getConnection() pentru a realiza initial conexiunea cu baza de date .
* Clasa **OrderDAO** : Aceasta clasa este putin diferite de celalalte doua deoarece nu implementeaza si metodele de actualizare si stergere comanda, ci doar de afisare si inserare .

Pachetul **CONNECTION** : In acest pachet este create clasa **ConnectionFactory** care realizeaza conexiunea bazei de date cu aplicatia .

Pachetul **START** : in acest pachet este clasa main din care se porneste aplicatia .

Clasa Client

Clasa *Customer* este clasa care se refera la datele clientului , si anume prezinta urmatoarele atribute : *int id* – fiecare client prezinta un ID unic ; *String nume* – desigur fiecare client prezinta un nume ; *String prenume* – stocarea prenumelui fiecarui client ; *String email* – stocarea unui email pentru clienti.

Ca si metode avem *Getter-ele* si *Setter-ele* specifice fiecarui atribut si metoda *toString*.

Clasa Produs

Clasa *Product* este clasa care se refera la produsele prezente in depozit.Astfel atributele sunt urmatoarele: *int id* – fieare produs prezinta un ID unic ; *String name* – stocarea numelui produsului ; *int stoc* – stabilirea stocului initial ce este in depozit ; *double price* – stocarea pretului unui produs ; *String categorie* – am ales sa atasez fiecarui produs o categorie pentru a putea filtra produsele.

Ca si metode avem *Getter-ele si Setter-ele* specifice fiecarui atribut.

Clasa Comanda

Clasa *Order* este clasa care se stocheaza datele necesare pentru lansarea comenzilor.Ca si atribute avem: *int id* – fiecare comanda prezinta un ID unic ; *Customer customer* – se stocheaza clientul care face comanda ; *Product produs* – se refera la produsul pe care il comandam ; *int cantitate* – cantitatea care se comanda pentru un anumit produs ; *TimeStamp timeStamp* – se retine momentul in care se face lansarea comenzii si *double total* – folosit pentru stocarea totalului unei comenzi (pret\*cantitate).

Ca si metode avem *Getter-ele si Setter-ele* specifice fiecarui atribut.

* Clasa *ClientDAO*

In aceasta clasa vom realiza operatii precum adaugarea unui nou client in baza de date, editarea unui anumit client din baza de date si chiar stergerea acestuia.

Se considera in cazul editarii si stergerii ca se va alege un client existent din baza de date, in cazul in care nu se intampla acest lucru, nu se va edita / sterge nimic din baza de date.

Totodata in aceasta clasa se gaseste si metoda formClient() ce extrage informatiile referitoare la tabela clients din baza de date ordermanagement.

* Clasa *ProductDAO*

In aceasta clasa vom realiza operatii precum adaugarea unui nou produs in baza de date, editarea unui anumit produs din baza de date si chiar stergerea acestuia.

Se considera in cazul editarii si stergerii ca se va alege un produs existent din baza de date, in cazul in care nu se intampla acest lucru, nu se va edita / sterge nimic din baza de date.

Totodata in aceasta clasa se gaseste si metoda formProduct() ce extrage informatiile referitoare la tabela product din baza de date ordermanagement.

Pachetul START:

Clasa Main:

In aceast clasa am utilizat metoda whichLine:

**public** **static** **void** whichLine(**int** j)**throws** SQLException, IOException, DocumentException {

Aceasta metoda primeste ca si parametru nrLinie-1 din fisierul filename.txt pentru a executa comenzile.

String line=Files.*readAllLines*(Paths.*get*("filename.txt")).get(j);

///folosesc separator daca nu este cuvant

//stochez intr-un vector de string

//verific cu ce am in fisier

Se identifica existenta cuvintelor cheie din fiecare rand citit si stocat intr-un vector de String.

Aceste cuvinte cheie sunt :Insert,Delete,Order,iar pentru fiecare dintre acestea se apeleaza clasele corespunzatoare cu metodele lor.

Exemplu:Delete in functie de nume prenume

**for**(**int** i=0;i<words.length;i++) {

**if**(words[i].equals("Delete")&&words[i+1].contentEquals("client")) {

ClientDao client =**new** ClientDao();

client.del(**new** Client(words[i+2],words[i+3]));

**break**;

}

Tot in main se specifica whichLine(j)

Clasa Bill

Aceasta este responsabila cu crearea chitantei in urma citirii din fisier a cuvantului Order

Document document = **new** Document();

PdfWriter.*getInstance*(document, **new** FileOutputStream("chitanta\_"+firstName+r+".pdf"));

document.open();

Font redFont = FontFactory.*getFont*(FontFactory.***COURIER***, 12, Font.***BOLD***, **new** CMYKColor(0, 255, 0, 0));

Paragraph paragraphOne = **new** Paragraph("Factura comanda", redFont);

document.add(paragraphOne);

Se deschide pdf-ul ,se insereaza paragrafe,se seteaza font-ul ,iar in final se inchide pdf-ul.

**Packages**

Un pachet este o colectie de clase si interfete inrudite.Sunt folosite pentru gasirea si utilizarea mai usoara a claselor , pentru a evita conflictele de nume si pentru a controla accesul la anumite clase.In alte limbaje de programare pachetele se numesc librarii.

Desigur cum avem 4 pachete separate folosirea unei clase dintr-un alt pachet se face prin importarea pachetului respectiv : import model,dao,connection, etc.

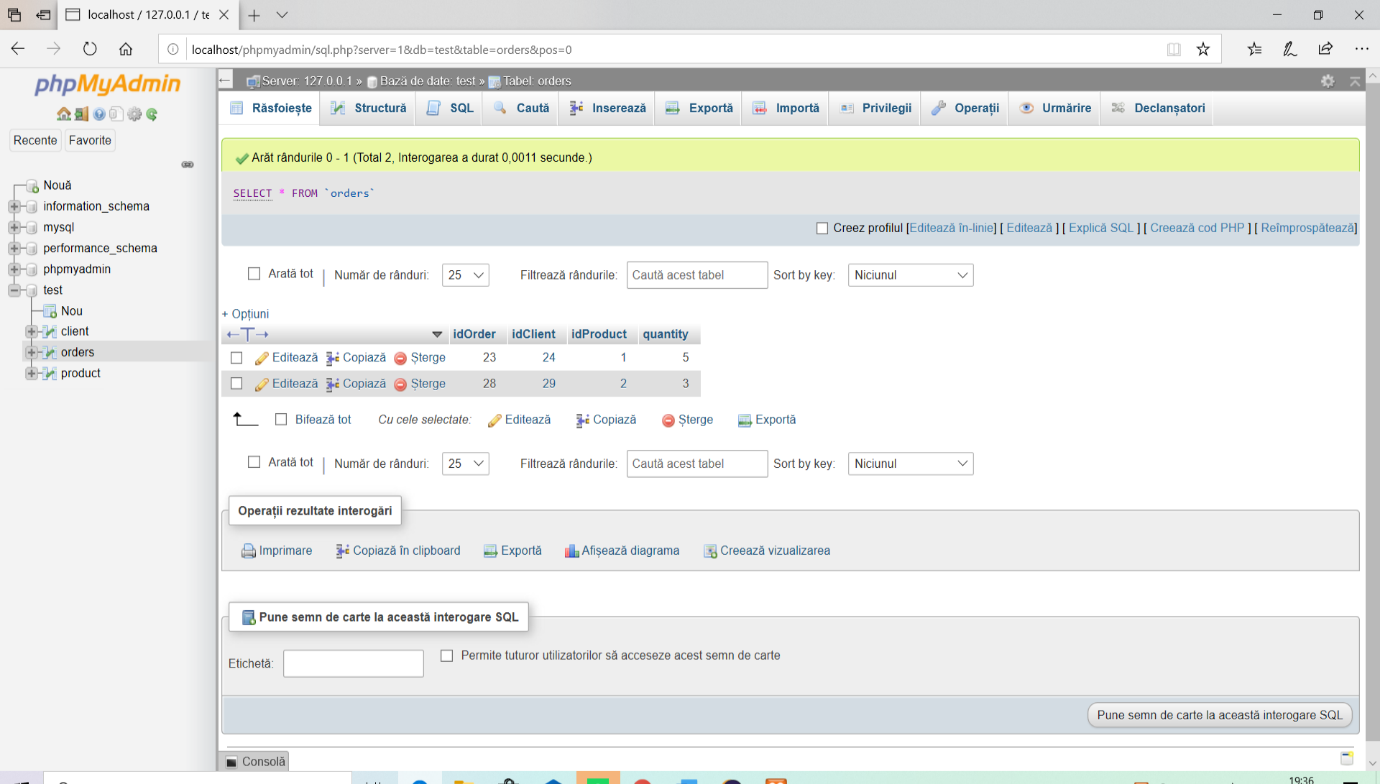
**4. Testare**

**Testare**

Pentru testare am ales sa utilizez o clasa aditionala Main in care am expus cateva exemple si pe care le-am verificat.Ulterior,am ales sa vizualizez informatiile din baza de date,cu ajutorul php my admin.In urma efectuarii fiecare operatii citite din fisier,s-au realizat update-uri ,insert-uri sau delete-uri in functie de caz.

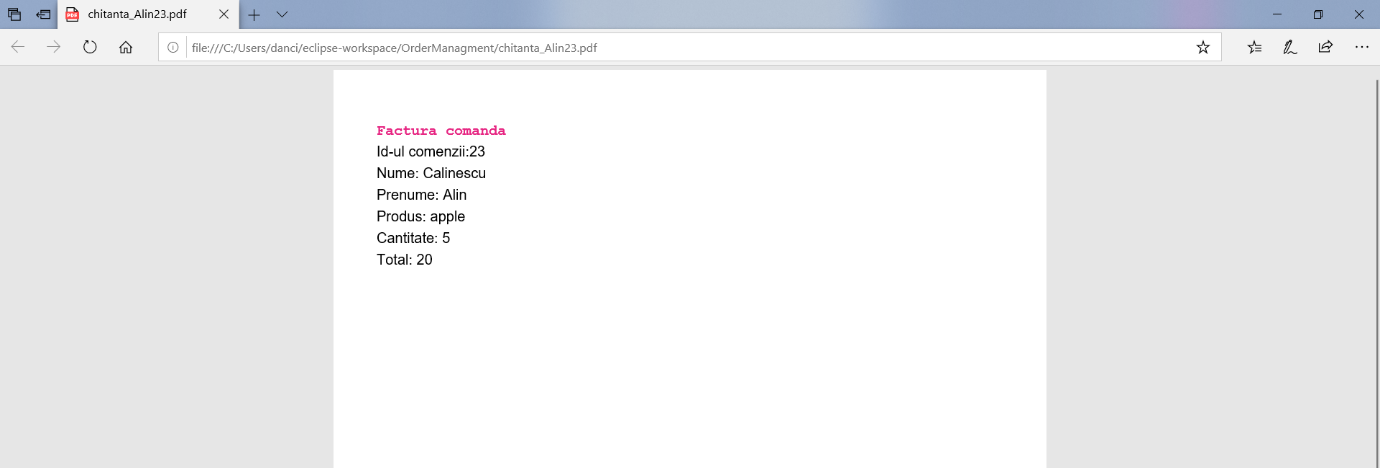
Exemplu:

Vizualizare comenzi in tabela Orders



**5.Rezultate**

Exemplu:generare chitanta comanda



**6.Concluzii**

In concluzie , au fost indeplinite obiectivele cerute in realizarea proiectului , chiar mai mult , prin prezenta de exemplu a metodei de a sterge o comanda , etc.

Din aceasta tema am imbunatatit cunostintele legate de gestionarea bazelor de date si folosirea acestora in Java .

Dezvoltari ulterioare :

Aplicatia reuseste cu succes sa creeze comenzi bazate pe datele din baza de date , dar , pe o comanda poate fi adaugat un singur produs . Ca dezolvare ulterioara se va implementa un algoritm prin care sa fie posibila adaugarea mai multor produse intr-o comanda , ca in viata de zi cu zi . Pentru aceasta implementare trebuie creat un nou tabel ProductOrdered in care sa avem ca si atribute : id-ul produsului , id-ul comenzii din care face parte , cantitatea cumparata pe comanda respectiva si eventual pretul total al produsului (cantitate \* pret pentru bucata )

In acest fel va fi posibila efectuarea mai multor comenzi .

**7.Bibliografie**

-Documentatia Java:

<https://docs.oracle.com/javase/7/docs/api/>

->mai exact

<https://docs.oracle.com/javase/7/docs/api/java/util/TreeMap.html>

-Pentru probleme de care m-am lovit

[*http://stackoverflow.com/*](http://stackoverflow.com/)

-Pentru exportul tabelului cu order in .txt ca si factura

<https://www.youtube.com/watch?v=tg7M9YvYDqo>

-Pentru a sterge continutul unui tabel

<http://stackoverflow.com/questions/3879610/how-to-clear-contents-of-a-jtable>

Access denied for password,unele erori intalnite la conectarea bazei de date

<https://www.youtube.com/watch?v=lBW77YDyMyY&list=PLr-LnFzODWU2IAGh-pIkLBdvZiHo8yJHo&index=35&t=2s>

Conectare baza de date:

<https://www.youtube.com/watch?v=B3SInSBuE_4&list=PLr-LnFzODWU2IAGh-pIkLBdvZiHo8yJHo&index=32>