**Documentație**

**Procesarea datelor dintr-un depozit**

Student: Costea Ovidiu-Bogdan

Grupa:30229

# 

**Contents**

[**1.Obiectivul temei:** 3](#_Toc512479675)

[**2.Analiza problemei ,modelare,cazuri de utilizare** 3](#_Toc512479676)

[**2.1Analiza problemei** 3](#_Toc512479677)

[**2.2. Modelarea** 3](#_Toc512479678)

[**2.3.Cazuri de utilizare** 3](#_Toc512479679)

[**3.Proiectare** 4](#_Toc512479680)

[**3.1. Structuri de date** 4](#_Toc512479681)

[**3.2. Clase** 5](#_Toc512479682)

[**3.3Algoritmi folosiți** 5](#_Toc512479683)

[**3.4. Pachete** 5](#_Toc512479684)

[**3.5. Interfețe** 6](#_Toc512479685)

[**3.6.Interfața cu utilizatorul** 6](#_Toc512479686)

[**4. Implementare și Testare** 10](#_Toc512479687)

[**5.Rezultat** 11](#_Toc512479688)

[**6.Concluzie** 12](#_Toc512479689)

[**7.Bibliografie** 12](#_Toc512479691)

# **1.Obiectivul temei:**

Principalul scop al acestei teme este realizarea unei aplicati care proceseaza clienti unui depozit.Implementarea se bazeaza pe utilizarea bazelor de date, dar mai mult decat atat am folosit si design pathernul Layout.Proiectul a fost realizat in limbajul de programare Java ,iar ca mediu de programare am folosit IDE-ul Eclipse.

Proiectul pe care deja l-am implementat ofera posibilitatea utilizari atat de catre amanageri cat si de clienti, activitatile pe care acestia pe pot executa fiind foarte bine conturate

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

# **2.1Analiza problemei**

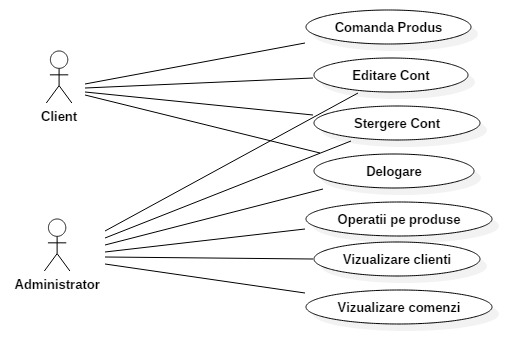
Dupa o scurta analiza a cerintei obiectivele care trebuie indeplinite sunt foarte bine conturate. Astfel aplicatie trebuie sa permita lucrul cu o baza de date in care vom avea mai multe tabele pentru: clienti, produse, comenzi si administrator. In functie de tipul contului de utilizare actvitatile noastre ar putea fi restrictionate sau largite, facandu-se diferenta intre client si administrator.

# **2.2. Modelarea**

Pentru implementarea acestui sistem de gestiune a datelor dintr-un depozit am avut nevoie de mai multe clase.In primul rand pentru fiecare tabel aveam nevoie de cateo clasa care sa contina variabile instanta identice ca tip si denumire pentru fiecare tabel din baza de date.Pe langa acesta clase am mai avut folosit clase care se ocupa cu operatiile de baza care pot fi realizate ,interogari. De o clasa care va in care se va implementa interfata grafica si alte clase folosite pentru implementare si realizarea design paternului Layout.

# **2.3.Cazuri de utilizare**

Inainte de a incepe sa utilizam aceasta aplicatie avem nevoie de creearea unui nou client de utilizator lucru care se poate face in doi pasi foarte simpli. In cazul in care dorim sa creeam un cont cu drepturi de administrator suntem nevoiti ca in campul cod administrator sa introducem stringul ADM. Dupa creearea unui nou cont clientul va avea acces la catevo operati respectiv: editare contului cu care s-a logat, stergerea acestui cont, delogarea din applicatie ,dar ofera si posibilitate vizualizari produselor din depozit si trimiterea unei noi comenzi( scrierea in tabel si creearea unui fisier care va avea un nume sugestiv si asociat cu id comenzi care tocmai a fost create).

**Diagrama USE-CASE: **

# **3.Proiectare**

Aplicatia a fost dezvoltată printr-o abordare top-down , sistemul fiind proiectat ca un tot, apoi acesta a fost divizat in subsisteme. Clasele au fost realizate intr-un mod abstract, acestea asteptandu-se la un rezultat specific fara a furniza instructiuni de returnare a acelui rezultat. Dupa ce un model acceptabil a fost atins metodele au putut fi definite.

Motivul utilizarii acestei abordari a fost faptul că oferă un foarte bun nivel de abstractizare, acest lucru imbunatățind viteza de dezvoltare a aplicației, un cod mai ușor de întreținut și mai pușin predispus la erori.

# **3.1. Structuri de date**

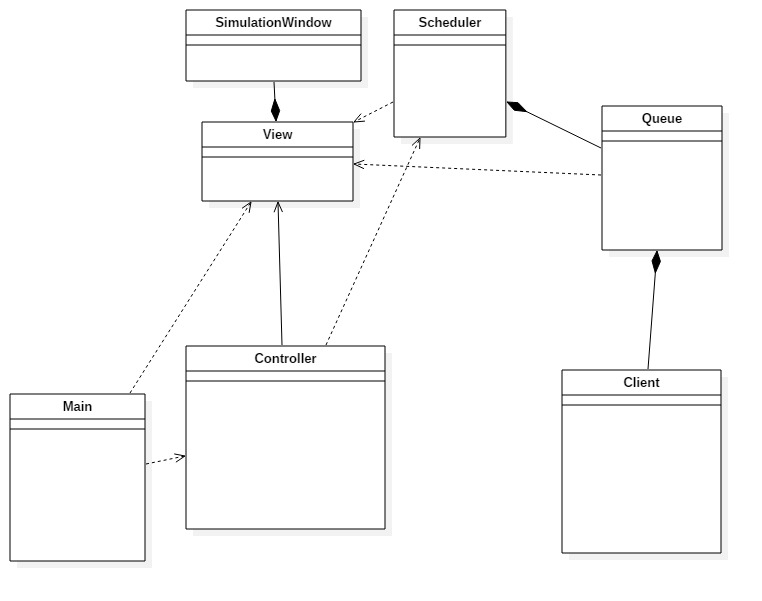
Din cauza faptului ca numarul de clienti, administratori, comenzi si produse continute de efiecare tabela este variabil am optat pentru utilizarea unei structuri de date de tip List,in cazul in care rezultatul returnat de interogare de inregistrari este mai mare de 1, care se va comporta exact ca o coada (FIFO- Firts In First Out).

# **3.2. Clase**

Funcționalitatea programului conține următoarele șapte clase. Clasa ”app” reprezintă clasa de unde porneste executia programului.

In clasa View se gaseste implementare interfetei grafice pecum si ascultatori pentru fiecare buton in parte.Clasele al caror nume se termina cu BLL(Business Logic) incapsuleaza logica acestei aplicatii , iar clasele cu terminatie DAO ( Data Access Layer ) contin interogarile la tabele, operatiile de baza care se pot realiza pe acestea ca de exemplu : inserare, stergere , update.

Diagrama UML de clase :

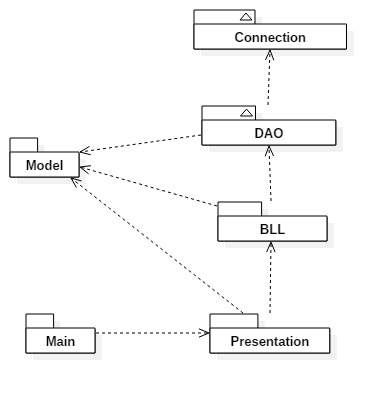


# **3.3Algoritmi folosiți**

In dezvoltarea acestei aplicatii nu am avut nevoie de algoritmi care sa se fie eficienti din punct de vedere al timpului si al memorie ,datorita lucrului cu baze de date. Singurele implementari care au fost facute au fost verificare usernameului si respectiv a parolei pentru un utilizator , asigurarea nedepasiri numarului de cantitati in cazul unei noi comenzi , dar si verificarea id pentru care se doreste editarea unui client/administrator/produs.O imbunatatire ar putea fi reprezentarea informatiei intr-o alta strutura de date care sa confere regasirea mult mai usoara a infromatiei din tabele dupa cheia primara.

# **3.4. Pachete**

Folosirea design paternului Layer impune o folosire foarte stricta a pachetelor astfel am folosit un numar de sase pachete. Care contin clasele specifice pentru fiecar parte a implementarii acestui design. Pachetul Model contine doar clasele care mapeaza baza de date,pachetul Connection contine clasa ConnnectionFactory care implementeaza metodele necesare conexiuni la baza de date, dar si metode care implementeaza interogari. Aceste pachete comunica intre ele implentand astfel ideea principala a acestui patern ( impartierea in diferite layere , in care giecare layer contine functii si are cate un scop special).



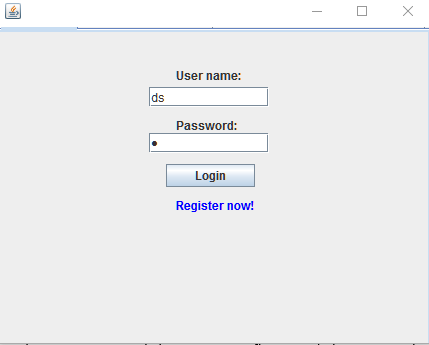
# **3.5. Interfețe**

Aplicatie nu implementeaza nicio interfata predefinita sau definita ,in dezvoltarea acestei aplicati. Pentru a nu declara tot timpul anumite operatii legate de interogarea tabelului am optat pentru dezvoltarea unei clase generica care sa poate fi instatiata cu diferitele clase unde se mapeaza tabelele din baza de date, astfel am decis ca anumite operatii specifice fiecarui tabel sa fie rezolvate in clasa Dao specifica tabelului care la randul ei va extinde clasa Abstracta generica.

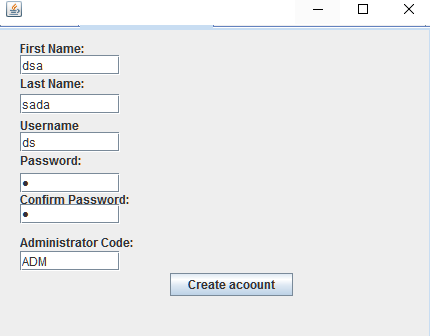
# **3.6.Interfața cu utilizatorul**

Interfata grafica este destul de simpla și intuitivă pentru orice utilizator, caruia ii ofera posibilitatea de a creea un nou cont prin simpla apasare asupra textului care sugereaza acest lucru. Interfata conține un JtabbedPane unde au fost adaugate mai multe containere Jpanel, utilizarea acestei componente faciliteaza lucrul cu ferestre ofrind astfel sansa de a reduce numarul de ferestre utilizate.

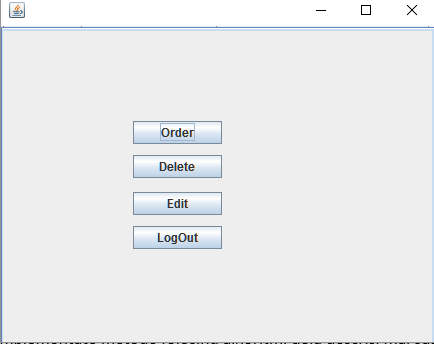
Fiecare Jpanel a fost populat la randul lui cu componentele grafice de care aveam nevoie. Taburile pentru creearea unui nou cont sunt populate cu mai mult JtextFielduri, Jlabel si Jbutoon care ii ofera utilizatorul ajutorul necesar pentru adaugarea unui nou cont in baza de date. Jtabel reprezinta componenta pe care pana acum un am avut oprtunitatea sa o folosec. Acesta piesa mi-a permis prin simplele sale metode sa afisez continul unei tabele. Odata ce o noua componenta a fost creata avem posibilitatea sa-i asignam un tablou bidimesional de tipul Object in care sunt stocate informatiile din tabel ,dar si numele fiecarei coloane in parte printr-un vector de tipul Object unde in fiecare camp a fost asignat cu cate un nume ce reprezinta coloana din tabel.



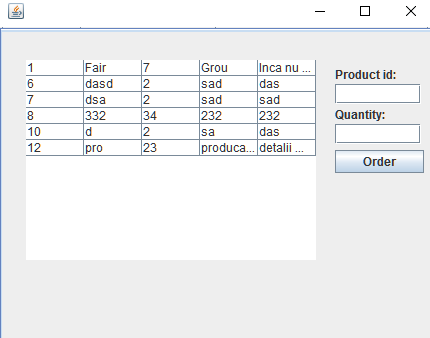
In imagina de mai sus este prezentata pagina de logare care prin completarea campului de username si password permite utilizatorului/ adminitratorului accesul in aplicatie. In cazul in care nu avem un cont vom avea sansa de a creea unul prin simpla apasare a Jlabelului intitulat ”Register now!” caruia i-a fost assignat un acultator de mouse, iar atunci cand vom da click pe el ne va duce la un alt tab.



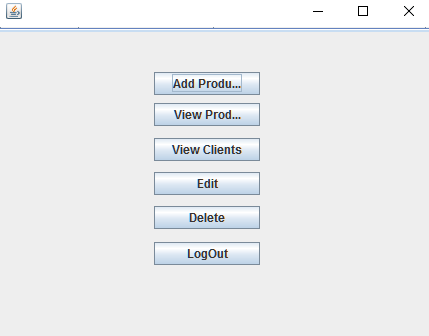
Asa cum am precizat si mai sus pe langa faptul ca putem creea noi conturi de utilizator am optat si pentru optiunea de inserari si noi conturi de administrator. Acest lucru este posibil in cazul in care campul Administrator Code va fi completat cu caracterele ”ADM”. Daca acest camp nu este completat utilizator va fi dus la o noua pagina unde va trebui sa completeze anumite detalii legate de datele personale ( date de care nu este neaparata nevoie in activitatile unui simplu administrator ).



In cazul in care logarea unui client a fost cu succes acestuia ii va aparea un meniu in care sunt prezentate anumite optiuni pe care le poate activa. De exemplu prin simpla apasare a butonului delete contul cu care s-a logat va fi sters definitiv din baza de date. Obtiunea Order ii ofera posibilitatea de a vizualica produsele care sunt stocate in depozit la momentul actual. In partea dreapta a tabului vor fi pozitionate de textfielduri sugestiv denumite pentru cantitatea si id unui produs anume.



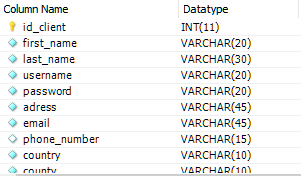
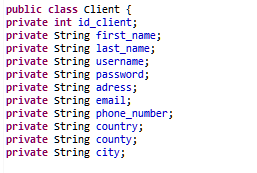
Pentru cazul logari cu un cont de administrator aplicatia va seta tabul care permite operatiile premise.



# **4. Implementare și Testare**

Programul a fost scris prin implementarea claselor deja menționate. În fiecare clasă au fost implementate metode folosind algoritmi deja descriși mai sus, acestea au fost declarate public ,iar campurile private fiind acesibile doar cu ajutorul getterelor si setterelor pentru a aigura permisiunile la aceste date.

**Clasa Client:**



Clasa Client apartine pachetului Model impreuna cu clasele Administrator, Product, Order. Principala utilizare acestora a fost maparea la tabelele din baza de date ,prin urmare fiecare variabila instanta a clasei client va avea atat denumirea cat si tipul identic cu cel al coloanei corespunzatoare din tabel. Cu ajutorul tehnicii de reflexie fiecarei variabile instanta i-a fost asignata valoarea corespunzatoare din tabel. O metoda mult mai simpla ar fi folosirea frameworkului Hibernate care mapeaza direct un tabel folosind annotatiile facute de noi intr-un xml sau prin simpla anotare inainte declararii.

**Clasa AbstractDAO:**

Clasa abstract DAO este clasa generic care implementeaza operatiile de interogare executate asupra tabelelor. Ulterior aceasta a fost extinsa de alte clase DAO care au fost ulterior implementate, favorizand astfel utilizarea acestor metode fara a mai fi nevoie sa le declaram inca o data. In anumite metode implementate in aceasta clasa a fost folosita techinica de reflexie. Pentru a avea acces la baza de date am folosit JDBC. Clasa AbstractDAO are ca varibile instanta un camp care v-a retine tipul clasei parametrizabile tipul clasei parametrizabile este determinat chiar in constructorul acestei clase

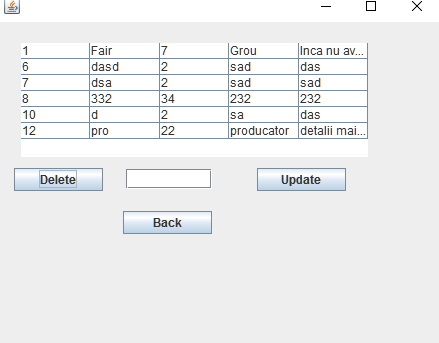


Metoda getList returneaza o lista de obiecte obtinute in urma exectiei unui query, folosid reflexia am reusit sa aflam campurile clasei T ,iar prin intermediul setterelor am reusit sa setam fiecare camp astfel incat toate datele de inregistrare sunt associate unei variabile instante. In cazul inserari nu va mai fi nevoie sa precizam campul id deaorece acesta a fost setat in baza de date ca auto complete si va fi generat automat.

Pachetul connection care contine clasa ConnectionFactory contine metodele necesare acestei pentru conexiunea la baza de date prin intermediul unui API.JDBC este o interfata (API) pentru limbajul de programare Java care defineste cum client acceseaza baza de date. Este o tehnologie de acces folosita pentru conectarea la baza de date.

# **5.Rezultat**

Produsele disponobile in acest moment la deposit vor fi vizibile prin simpla apasare a butonului Order care va schimba tabul si va initializa JTableul cu informatiile preluate din baza de date.Acest lucru este folosit si in cazul tabelelor pentru clienti ( tabela vizibila de administatori).



# **6.Concluzie**

# Acest proiect reprezinta daca as putea sa-l numesc machete unei aplicati de care permite realizarea unor cumparaturi de catre client.Astfel vom avea sansa ca tea casa sa ne comanda produsele dorite. Exista posibilitatea extiderii acesteia prin adaugarea unor poze fiecarui produs in parte.O alta imbanatatire care ar putea fi adusa este extinderea aplicatiei cu un chat in care client sa poata contacta direct pe cei ce se ocupa de vanzari , avand astfel sansa de-a le adresa intrebari primind raspunsuri in timp real. Am putea oferi utilizatorilor sansa de asi exprima parerea asupra unui anumit produs.

# **7.Bibliografie**

<https://docs.oracle.com/javase/tutorial/uiswing/components/table.html>

<https://stackoverflow.com/questions/5260462/can-i-add-an-action-listener-to-a-jlabel>

<https://www.youtube.com/watch?v=0Zlh56mTS6c>

<https://docs.jelastic.com/jelastic-db-hibernate>

<https://docs.oracle.com/javase/tutorial/jdbc/basics/connecting.html>