Aplicatie pentru facilitarea livrarii de alimente

Aplicatie realizata de

Bursuc Andrei

Grupa 30234

Contents

[1. Prezentare generala 3](#_Toc509484356)

[2. Decizii de proiectare 3](#_Toc509484357)

[3. Proiectarea bazei de date 4](#_Toc509484358)

[Diagrama baza de date 5](#_Toc509484359)

[4. Diagrama UML 6](#_Toc509484360)

[Proiectarea claselor si relatiile dintre ele 7](#_Toc509484361)

[5. Concluzii și dezvoltări ulterioare 10](#_Toc509484362)

# Prezentare generala

Acesta tema urmareste realizarea unei aplicatii ce ajuta la comandarea si livrarea de produse alimentare. Aplicatia are doua tipuri de utilizatori: administratorul, ce poate vedea si modifica datele clientilor, dar si modifica si adauga produse in lista de produse, si utilizatorul obisnuit ce isi poate crea un cont nou ale carui date le poate edita cand doreste, de asemenea acesta in meniu isi creaza un cos nou in care adauga produse, putand efectua plata la finalizarea comenzii fie cu cardul sau cash.

# Decizii de proiectare

Pentru realizarea aplicatiei am optat pentru utilizarea arhitecturii stratificate care imparte aplicatia in mai multe straturi:

„Presentation Layer” contine clasele folosite la definirea interfetei utilizatorului. Rolul său este de a evidenția informația obținută prin intermediul „Data Acces Layer”. Interfața a fost realizată cu ajutorul bibliotecii Javax.Swing utilizând plugin-ul special din Intelij, ce permite realizarea mai ușoară a unei interfețe atractive pentru utilizator și ușor de folosit.

„Business Layer” contine clasele care incapsuleaza logica aplicatiei

„Data Acces Layer” contine clasele ce contin interogarile pentru baza de date si practic in cazul de fata definesc functiile prin care programul interactioneaza cu baza de date;

In plus in cadrul acestei architecturi mai avem „Modelul” ce contine clase care modeleaza fiecare tabel in parte din baza de date cu care programul interactioneaza.

O alta tehnologie utilizata in cadrul proiectului este o baza de date, aceasta reprezentand o modalitate de stocare a unor [informatii](http://ro.wikipedia.org/wiki/Informa%C8%9Bie) si [date](http://ro.wikipedia.org/wiki/Dat%C4%83) pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii usoare si a regasirii rapide a acestora.

**SQL** (de la numele englez Structured Query Language - limbaj de interogare structurat - care se pronunță [*es kiu el*)este un limbaj de programare specific pentru manipularea datelor în sistemele de manipulare a bazelor de date relaționale (RDBMS),  iar la origine este un limbaj bazat pe algebra relațională. Acesta are ca scop inserarea datelor, interogații, actualizare și ștergere, modificarea și crearea schemelor, precum și controlul accesului la date. A devenit un standard în domeniu (standardizat ANSI-ISO), fiind cel mai popular limbaj utilizat pentru creearea, modificarea, regăsirea și manipularea datelor de către SGBD-urile (Sistemele de Gestiune a Bazelor de Date) relaționale. Pe lângă versiunile standardizate ale limbajului, există o mulțime de dialecte și variante, unele proprietare, fiind specifice anumitor SGBD-uri și de asemenea conținând extensii pentru a suporta SBD-urile (Sistemele de Baze de Date) obiectuale (obiectual-relaționale).

SQL a fost inițial dezvoltat la compania IBM de către Donal D.Chamberlin și Raymond F.Boyce  în anul 1974. Inițial numele era **SEQUEL** (tot de la *Structured Query Language*) și avea ca scop manipularea și regăsirea datelor stocate în bazele de date relaționale IBM

Operaţiile de bază pe o bază de date sunt:

* crearea unui tabel

CREATE TABEL tbl\_name ([data\_type] [col\_name], ...)

* afişarea datelor unui table

SELECT select\_expr [, select\_expr ...]

[FROM table\_references [WHERE where\_condition]

* inserarea în baza de date

INSERT [INTO] tbl\_name [(col\_name,...)] VALUE ({expr},...),(...),...)

* modificarea datelor

UPDATE table\_reference

SET col\_name1={expr1|DEFAULT} [, col\_name2={expr2|DEFAULT}] ...

[WHERE where\_condition]

* ştergerea datelor

DELETE FROM tbl\_name

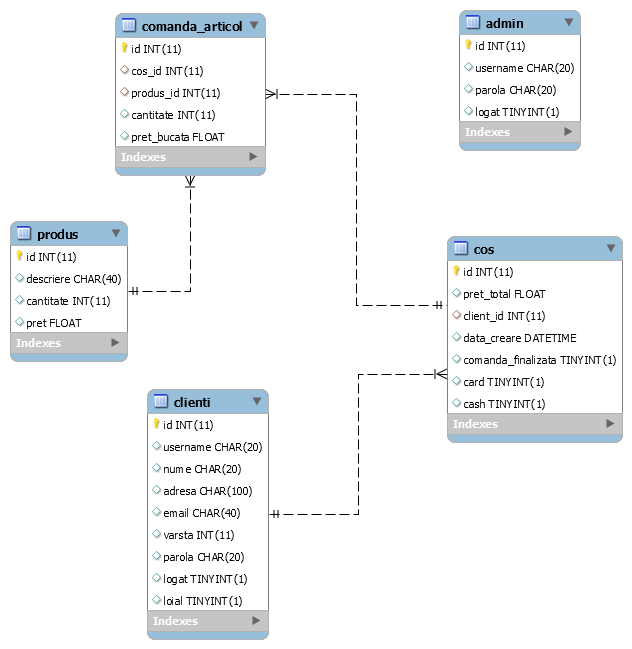
[WHERE where\_condition]

Codul a fost scris si compilat utilizând IDE-ul IntelIJ IDEA, mediu de dezvoltare software dezvoltat de catre compania JetBrains ( cunoscută anterior sub numele de IntelIJ ). El vine in doua variante ”Community Edition” și ”Ultimate Edition”, aceasta din urmă fiind disponibilă gratis studenților. Prima versiune a IntelIJ a fost lansată in Ianuarie 2001 și a fost unul dintre primele IDE-uri pentru java care dispunea de funcții avansate de ” Code Navigation” și ”Code Refactoring”. Ultima versiune 2017.3 dispune de suport pentru Java 8, posibilitatea de dezvoltare de interfață grafică pentru Android, Play 2.0 și Scala.

# Proiectarea bazei de date

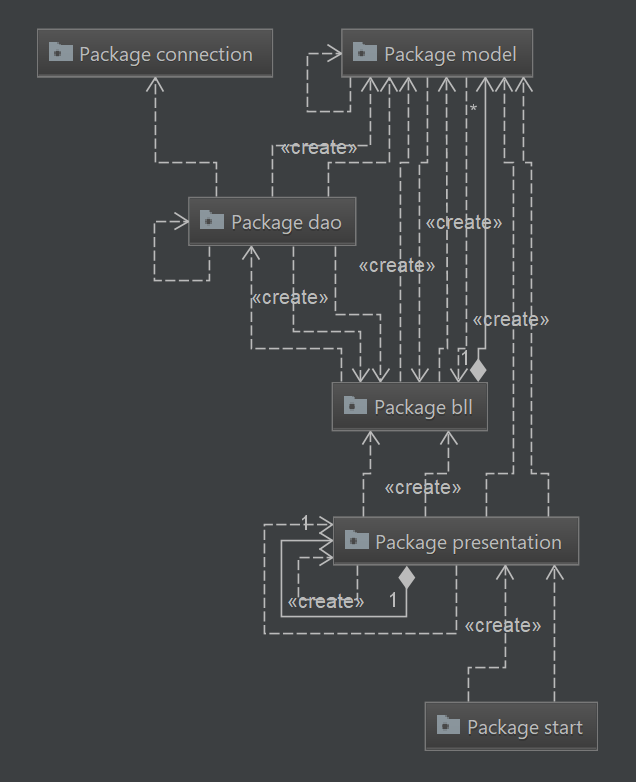
Baza de date este alcatuita din 5 tabele: Admin, Clienti, Produs, Comanda\_Articol si Cos. Tabelul admin este definit de campurile: id, username, parola, logat (camp boolean setat pe true cand unul din administratori este logat) Tabelul clienti este definit de campurile: id, username, nume, adresa, email, varsta, parola, logat (camp de tip boolean care confirma daca este sau nu logat clientul) si loial (camp boolean ce poate fi activat de admin daca el considera dupa vizualizarea comenzilor clientului ca el este loial, daca acest camp este activat clientul beneficiaza de o reducere de 5%), acest tabel ne ajuta sa stocam datele despre client si ne spune daca un client este logat sau nu. Tabelul Produs cu campurile: id, descriere, cantitate, pret stocheaza datele despre produse; Tabelul cos definit de: id, pret\_total, client\_id, data\_creare, comanda\_finalizata, cash, card ne ajuta sa facem legatura intre comenzi si fiecare client in parte, un client putand sa aiba mai multe comenzi ce au acelasi cos\_id, de asemenea campurile boleene cash si card ne ajuta sa identificam modul in care clientul a efectuat plata (cash sau cu cardul), tabelul comanda\_articol este definit de: id, cos\_id, produs\_id, cantitate, pret\_bucata.

## Diagrama baza de date



# Diagrama UML

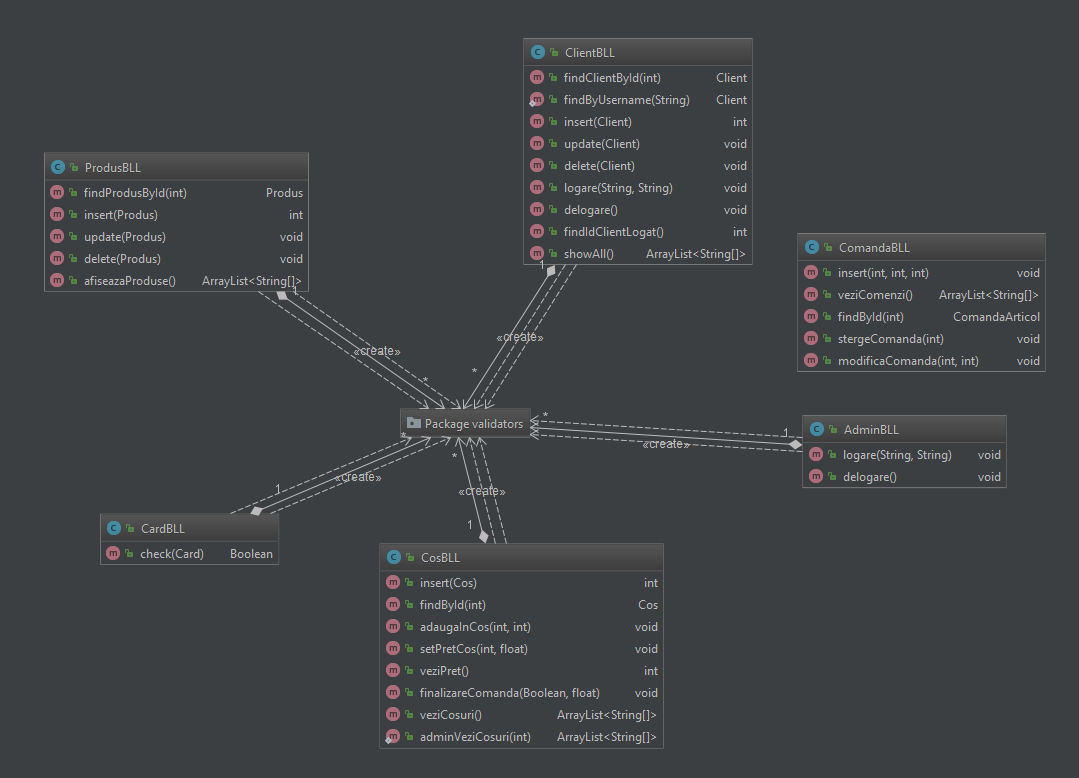
In continuare este prezentata diagrama UML a pachetelor folosite in program iar mai apoi diagrama UML a fiecarui pachet in parte:



## Proiectarea claselor si relatiile dintre ele

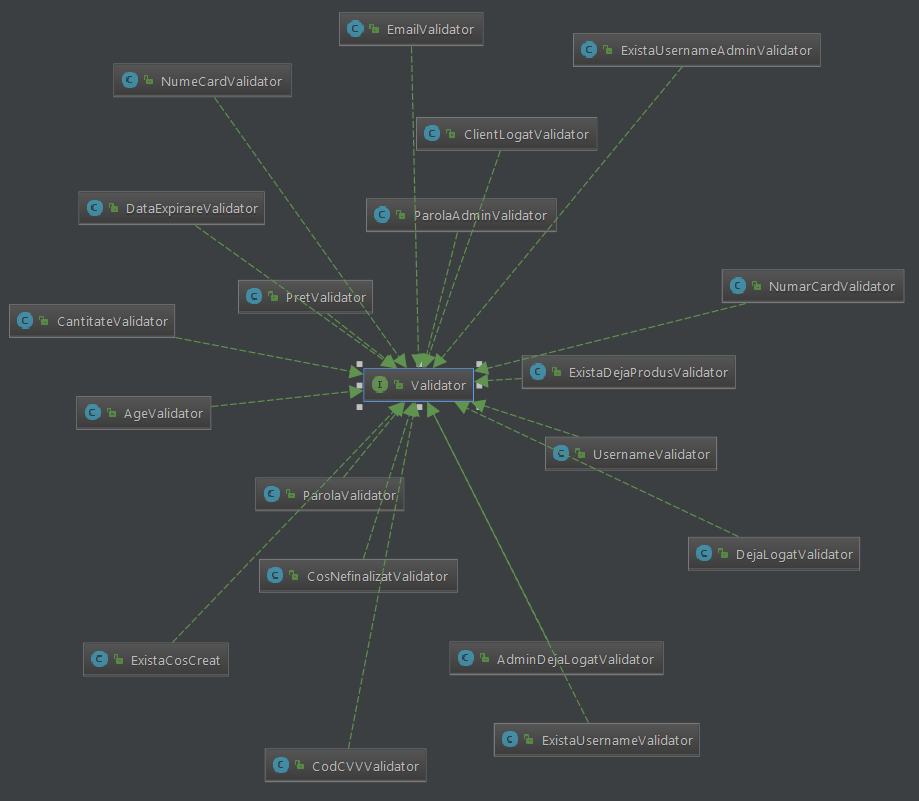
Proiectul fiind alcatuit din mai multe pachete vom prezenta unele dintre pachetele cele mai importante cu diagrama UML corespunzatoare.

**Package BLL ( Business Logic )**



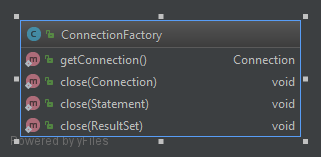
Acest pachet contine clasele ce verifica functiile din pachetul „Data Acces” aplicand pe ele validatorii ce spun daca interogarea pe baza de date se poate sau nu executa. Un exemplu ar fi clasa ClientBLL.

**Package bll.validators**



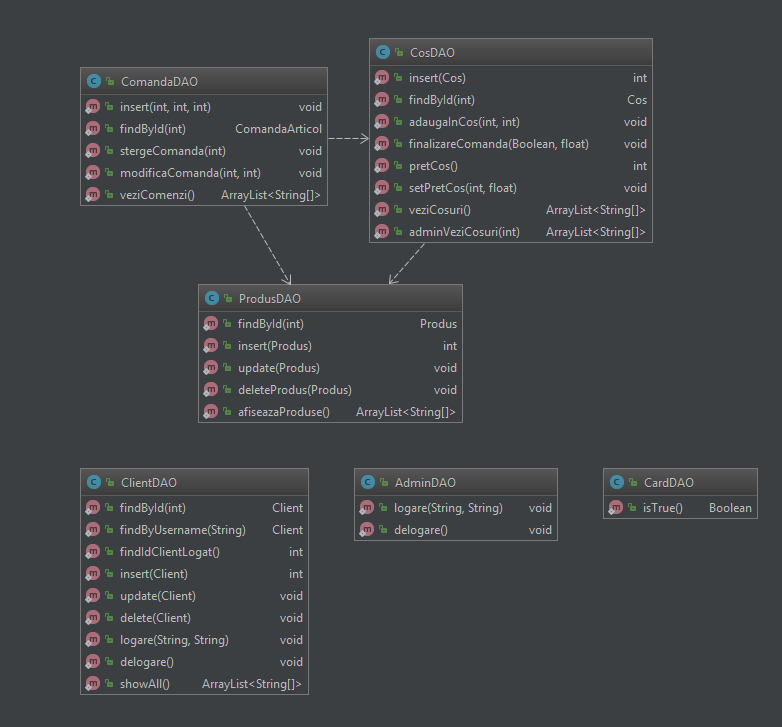
Scopul claselor din acest pachet este sa verifice daca datele introduse de utilizator sunt corecte, iar in cazul in care acestea sunt gresite metoda validate din clasa va arunca o exceptie.

**Package Connection**



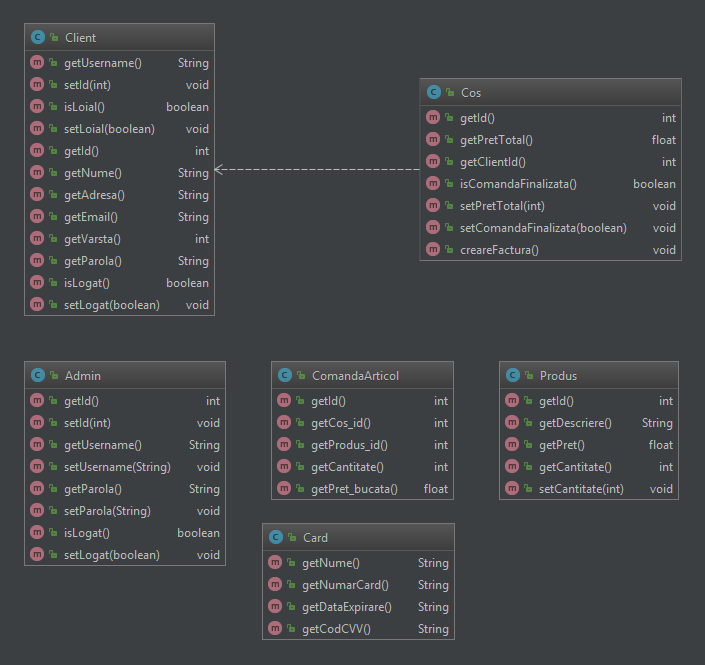
Acest pachet este alcatuit dintr-o singura clasa in care se foloseste pattern-ul singleton ce permite crearea doar a unei singure instante a clasei ConnectionFactory, acest lucru usurandu-ne modul in care realizam conexiunea cu baza de date inainte de apelarea unei interogari.

**Package dao**



Aici sunt definite clasele care contin metodele prin care interactionam cu baza de date si putem vedea sau modifica continutul ei.

**Package model**



Clasele din acest pachet reprezinta corespondentul tabelelor din baza de date (cu exceptia clasei Card creata pentru a facilita verificarea formularului pentru card).

# Concluzii și dezvoltări ulterioare

Dezvoltarea acestei aplicații m-a ajutat sa invat mai mult despre arhitectura stratificata si despre cum putem organiza mai bine un program prin impartirea acestuia in mai multe pachete cu funcționalitati diferite.

Posibile dezvoltari ulterioare ale programului ar fi prezenta unei descreri mai detaliate a produselor ( poze, descriere), posibilitatea creerii unei liste de dorinte si aparitia unei notificari cand produsul apare pe stoc. Introducerea unei modalitati de recuperare a parolei in cazul in care utilizatorul o uita, dar si posibilitatea de a modifica datele produsului direct pe tabel fara a mai fi nevoie de casutele aditionale.