

**Tehnici de Programare :**

**Tema 3**

OrderManagement

Student : Micle Bogdan

Grupa : 30221

Cluj Napoca

Aprilie , 2018

**Cuprins :**

**1. Obiectivul temei . . . . . . . . . . . . . . . 3**

**2. Analiza problemei . . . . . . . . . . . . . . 3**

**3. Proeictare . . . . . . . . . . . . . . . . . . . . . 3**

**4. Implementare si testare . . . . . . . . . . 7**

**5. Rezultate . . . . . . . . . . . . . . . . . . . . . . 13**

**6. Concluzii . . . . . . . . . . . . . . . . . . . . . . 13**

**7. Posibile dezvoltari ulterioare . . . . . 13**

**8. Bibliografie . . . . . . . . . . . . . . . . . . . . 14**

1. Obiectivul temei **:**

Cerinta: Luaţi în considerare o cerere OrderManagement pentru procesarea comenzilor clienţilor.Aplicatia foloseste (minim ) , următoarele clase: Ordine ,  OPDept ( Departament de procesare a comenzii), client, produs, şi depozit. Clasele de OPDept şi Warehouse folosi un BinarySearchTree pentru comenzi de depozitare .

Descriere : Aplicatia va avea 2 moduri de functionare . O metoda adresata unui client si o metoda adresata unui administrator . Clientul are urmatoarele optiuni : sa vizualizeze produsele existente pe stoc si sa cumpere unul pe care il doreste la alegere . Administratorul poate vedea comenzile impuse de catre client si poate sa ii confirme comanda . De asemenea administratorul mai poate sa adauge produse si sa vizualizeze produsele existente . De asemenea in cazul introducerii aceluiasi produs la acesta i se va incrementa numarul de bucati din stoc .

1. Analiza Problemei **:**

Am ales situatia clara in care avem un administrator al unei baze de date care poate face orice operatie asupra tabelelor din baza de date(Select,Insert,Delete,Update) asupra celor 4 tabele ale bazei de date Client, Stoc, Produs, Comanda . Acesta are posibilitatea de a introduce id-ul produsului. In cazul in care numele produsului nu exista , se va afisa eroare . De asemenea daca numarul de bucati comandate este mai mare decat numarul de bucati existente pe stoc se va afisa un mesaj de avertizare in care se specifica acest lucru.

Interfata grafica este usor de utilizat, numele componentelor fiind deductive.

1. Proiectare **:**

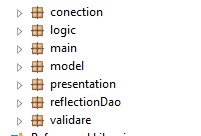
**Aplicatia este structurata in 7 pachete:**

Aceasta aplicatie este facuta dupa modelul View-Controller .

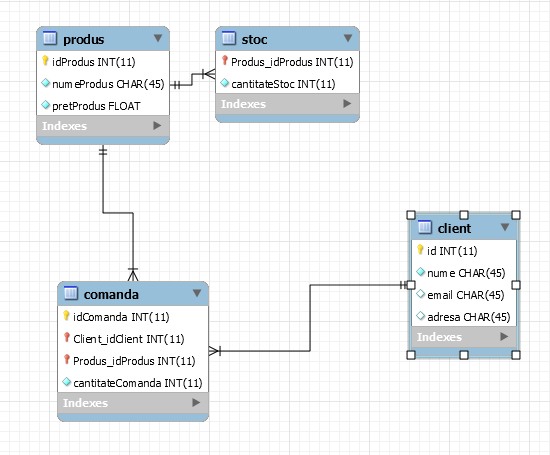
Cum construim cel mai usor o aplicatie mare? O impartim pe sectiuni, desigur. Avem astfel avantaje multiple: se poate lucra in echipa mult mai usor, putem gasi repede partea de aplicatie de care avem nevoie si ii putem oferi o anume individualitate.   
Aplicatia noastra va fi impartita in mai multe sectiuni, carora le vor corespunde cate doua clase. MC vine de la Model-Controller, reprezentand cele doua componente principale ale unei aplicatii.  
  
View-ul se ocupa de afisarea datelor. O data ce functiile sunt executate, viewului ii sunt oferite rezultatele de catre celelalte clase, iar acesta le va trimite catre browser.

Controller-ul este legatura intre model si view, intre logica aplicatiei si actiunile userului. In functie de cerintele userului, controllerul va executa o anumita functie definita special pentru sectiunea de site in care se afla userul. Functia va folosi modelul pentru a prelucra datele si va trimite rezultatele catre view, care isi va afisa apoi templateurile.

Aplicatia este impartita in mai multe pachete.

In pachetul presentation avem urmatoarele clase : o clasa View care reprezinta interfata grafica principala si extinde Jframe, o clasa Table care preia liniile din baza de date si le transfoorma intr-un tabel, si clasa Controller care tine Listener cu rol de ascultator pentru interfata grafica .In pachetul model avem clasele Client, Produs, Comanda si Soc in care se retin informatiilepreluate din baza de date avand aceleasi campuri de acelasi tip. In Produs se tin informatiile despre produse: id(unic), nume, pret, in Client informatiile despre clienti: id(unic), nume, adresa si email.In pachetul conection se realizeaza conectarea la baza de date.Pachetul logic pe baza claselor din pachetul validare si reflectionDao, opereaza cu date valide asupra bazei de date.Pachetul main contine o singura clasa, cea de test.

# Strunctura Bazei de date



**Explicatie:Fiecare Produs are un id care identifica unic tipul produsului, numele produsului si pretul acestuia.Fiecare produs are un Stoc determinat unic de id-ul prodului si o cantitate care reprezinta cantitatea din stoc a respectivului produs.Un client este determinat tot printr-un id unic, un nume, email si adresa.Fiecare client poate plasa o comanda identificata printr-un id unic, id-ul respectivului client, id-ul produsului si cantitatea comandata.**

# Implementare si testare :

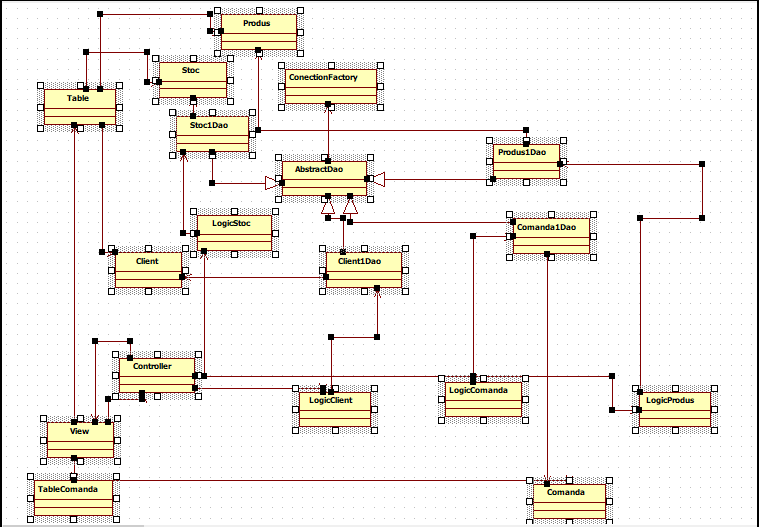
In cele ce urmeaza am sa prezint o diagrama a cazurilor de utilizare (Use Case Diagram) .Pentru inceput cateva cuvinte despre acest concept .

O diagrama a cazurilor de utilizare prezinta o colectie de cazuri de utilizare si actori si este folosita in general pentru a indica sau caracteriza functionalitatile si comportamentul intregii aplicatiei sistemului interactionand cu unul sau mai multi actori. Utilizatorii si orice sistem ce poate interactiona cu sistemul sunt actori.

In diagram de mai jos putem observa ca avem un actor numit Utilizator care poate sa : introduca timpul de simulare , numarul de cozi , interval minime si maxime de sosire client , interval minime si maxime de stat a clientilor , precum sis a dea Start acestei aplicatii.

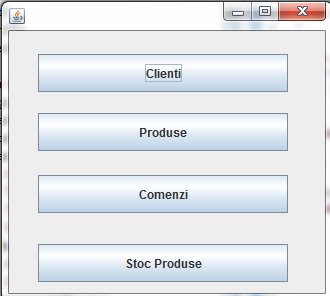
*Class diagram* este un tip de diagramă utilizată pentru descrierea structurii statice, adică a *entităţilor*sau *claselor* existente într-un sistem. Acest tip de diagramă este utilizat cel mai adesea de către dezvoltatori pentru specificarea claselor dar poate fi foarte util şi pentru specificarea structurii unor sisteme sau subsistem dintr-un business real.

Mai jos putem observa dependenta claselor .



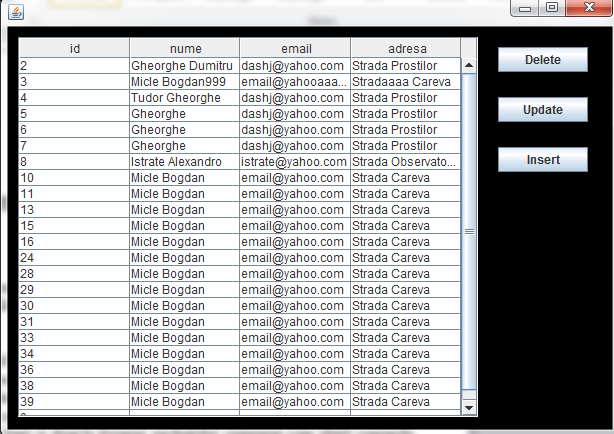
In urma testari programului se poate observa urmatoarele lucruri :

Prima fereastra reprezinta fereastr meniu in care se pot alege operatiile dorite de catre utilizatorul acestei aplicatii : Operatii precum : operatii pe Client , operatii pe Produse,operatii pe Comenzi si operatii pe Stoc Produse.



O data cu apasarea unui buton, o noua fereastra va aparea cu un tabel reprezentand datele stocate in baza de date si 3 butoane corespunzand operatiilor posibile pe acea baza de date: Insert, Update, Delete.

De exemplu daca se doreste operatii corespunzatoare unui client se da click pe butonul Clienti, in urma caruia se va deschide o noua fereastra in care se pot introduce ce clienti dorim sa introducem si in ce cantitate , banii viind nelimitati . Am ales acest punct de vedere inc are banii sunt nelimitati pentru a putea face aplicatia cat mai reala. DE exemplu ca sa updatam o anumita coloana, schimbam datele din tabel cu datele valide pe care dorim sa le introducem. Pentru stergerea unui client din baza de date se selecteaza linia din tabel pe care apare clientul si se apasa butonul Delete.Pentru operatia de insert, la sfarsitul tabelului avem o coloana goala unde se pot introduce datele.

. 

Utilizatorul aplicatiei are posibilitati multiple: de adaugare produs in stoc , de vizualizare produse stoc , de vizualizare comenzi active si tot o data de confirmare comenzi .Cand se confirma comenzile acestea vor fi atutomat toate confirmate slabiciunea constand in faptul ca nu se poate alege care comanda sa fie confirmata si care nu .

De asemenea aplicatia distinge daca produsele introduse sunt de acelasi fel iau in acest caz se incrementeaza doar numarul de bucati de tipul produsului respectiv .

Se recomanda mare atentie la introducea produselor pentru ca pretul e cel mai nesemnificativ si nu se tine cont de valoarea acestuia .

O data cu vizualizarea produselor se poate observa ca se afiseaza numele produsului , cantitatea produsului si pretul acestuia.

O data cu vizualizarea comenzilor se pot vedea doar numele produsului si cantitatea de produse comandate

. . . . .. . . .. .. . . . .. . . . . . ... . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

1. **Rezultate :**

In urma lansarii aplicatiei am putut observa ca sunt multiple cai de utilizare a acesteia. Utilizatorul isi poate asuma doua roluri : de cumparator si de vanzator . Fiecare dintre aceste roluri fiind puternic influentat de numarul de produse de pe stoc .

1. **Concluzii :**

- reprezinta un punct de plecare pentru aplicatii ulterioare care sa incerce sa particularizeze acest tip de comportament

- reprezinta un bun exemplu pentru constructia unui Observer intre clase ;

-se poate observa ca desi pare un proiect simplu defapt constructia si finalizarea lui sunt destul de greu de facut .

1. **Posibile dezvoltari ulterioare :**

Ce am reusit sa invat din aceasta aplicatie :

- un sistem de siguranta la plata , aplicatia incepand sa tina cont de pretul produselor cumparate ;

-adaugarea unei noi ferestre in care administratorul isi poate vizualiza sia dauga clientii favoriti la magazin ;

- adaugarea de reduceri la produse si in sepcial de promotii la produse : 2+1 gratis etc.

- conceptul de treeSet in java ;

- “repetitia, e mama invatarii”, experienta;

- s-au recapitulat concepte de baza legate de programarea OO;

- s-a realizat familiarizarea cu tipul de date Vector

- s-au aprofundat notiuni legate de fire de executie

- interfata grafica in Java

- concepte legate de cozi ( inserare, stergere, actualizare )