Food Delivery Management

Assigment 4



**Profesor Coordonator:**  **Student:**

Dr.Pop Cristina Bianca Vaduva Calin Liviu

**Grupa**: 30228 **An:** 2

Cuprins

**1. Obiectivul Temei** . ..............................................................................................

**2. Analiza Problemei** ........... ..................................................................................

2.1 Diagrama Use-Case ............ ...........................................................................

2.2 Cazuri de utilizare .................................. .......................................................

**3. Proiectare** ........................................................................ ..................................

3.1 Diagrama UML ...............................................................................................

3.2 Cazuri de utilizare ................................................................ ..........................

**4. Implementare** .................................................................................. ..................

4.1 Clase .................................................................................................... ..........

4.2 Interfata Grafica ..................................................................................... .......

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

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

**7 Documentatie** .................................................................................................... ..

1.Obiectivul temei

Obiectivul principal al temei este de a proiecta si implementa un sistem de management al livrarilor unor produse pentru o companie de catering. Aceasta aplicatie ar trebuii sa poata avea trei tipuri de urilizatori care se pot conecta simultan utilizand nume de urilizator si parola, acestia au 3 roluri diferite si anume administrator , angajat si client .

Obiectivele secundare a acestei teme sunt :

* Analiza problemei ( va fi prezentata in capitolul 2 ) -- care cere prezentarea cerintelor

functionale si cerintelor de utilizare.

* Proiectare ( va fi prezentata in capitolul 3 ) – contine proiectarea orientata pe obiect a aplicatiei, diagramele UML, de clase, de obiecte, interfetele si algoritmi folositi .
* Implementare ( va fi prezentata in capitolul 4 ) – contine descrierea claselor , metodelor folosite, si descrierea implementarii interfetei grafice.
* Rezultate ( prezentata in capitolul 5 )

2. Analiza Problemei

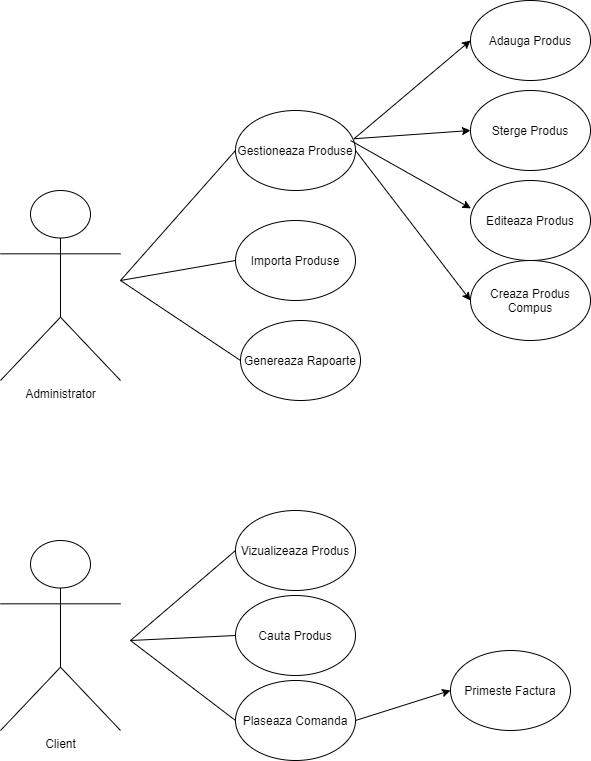
Aplicatia ar trebuii sa ofere posibilitatea de a creea utilizatori pe care ii numim Clienti si prin intermediul acestora aplicatia trebuie sa ofere posibilitatea de a vedea o serie de meniuri de care dispune firma de catering si sa poata cauta in acea lista in functie de selectia unui criteriu precum a cauta in funtie de un cuvant inserat, a cauta toate produsele cu un anumit numar de calori sau produsele care au un anumit pret, totodata aplicatia trebuie sa permita clientului sa selecteze un produs si sa il comande primind o factura cu pretul final. Tot in intermediu aplicatiei trebuie sa mai existe un user pentru administrarea lucrurilor, astfel user-ul de tip administrator se ocupa de aceasta parte, el are accesul la lista de meniuri, le poate accesa, trebuie sa poata sa creeze noi produse , chiar meniuri compuse din produse deja existente , ar trebuii sa poata sa stearga si sa modifice un produs dupa bunul sau plac si sa creeze ulterior rapoarte in funtie de comenziile clientilor sai .

Pe scurt aplicatia trebuie sa permita : - Utilizarea unui meniu initial dintr-un fisier de tip .csv

- O fereastra pentru Client care sa se ocupe de crearea comenziilor

-O fereastra pentru Administator ca sa se ocupe de gestiunea comenziilor si de fabricarea meniului .

**Diagrama Use-Case a aplicatiei:**



**Actorii**: Administrator si Client . . . . . . . . . . . . . . . . .. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

**Cazuri de utilizare a aplicatiei** :

1. Administrator: - Vizualizeaza interfata si importa produsele in meniu

-Realizeaza operatii asupra produselor din meniu : operatii de adaugare a unui nou produs ,adaugarea unui produs compus din mai multe, operatii de stergere a unui produs sau modificere a oricarui produs din meniu .

-Genereaza rapoarte in legatura cu comenzile date de clienti : rapoarte in functie de numarul de comenzi cu acelasi produs , ora dintrun interval a comenzii etc.

2. Client : - Vizualizeaza Meniul adaugat de administrator

- Realizeaza operatii de cautare in meniu in functie de criterii selectate precum titlu, rateing, numarul de calorii, pretul comenzii

- Creaza o comanda si genereaza o factura cu pretul aferent produselor selectate .

**Descrierea cazurilor de utilizare :**

**Scenariul de succes :**

1) Se porneste aplicatia , si se inregistreaza un client prin introducerea unui id, username si password , apoi se logheaza cu acelasi nume si parola pentru a intra in fereastra clientului

2) Se introduc tot in fereastra de login datele : UserName = Admin , Password= Admin si se apasa butonul “Login” , astfel se va intra in fereastra adminului .

3) In fereastra de admin se apasa butonul Import pentru a se importa meniul din fisierul .csv

4) Adminul poate sa introduca sa modifice sau sa stearga un produs prin introducerea in casutelor text a unui nou produs si apasarea butonului ADD , MODIFY sau DELETE . Pentru creearea unui produs compus , administratorul trebuie sa introduca in cele 3 casute text din mijlocul ferestrei de admin urmatoarele informatii : - In casuta de id = id-ul meniului care va fi creeat ,

- In casuta CompositeID se va insera o serie de id-ul despartite printr-un spatiu ex:CompositeID = 1 2 , astfel se va creea un nou produs compus din produsul cu id-ul 1 si id-ul 2

- In casuta text Title va fi noul nume dat produsului compus .

5) Tot Administratorul poate sa genereze rapoarte in funtie de comenziile date de client, daca nu sunt comenzi nu se va genera nici un raport . Un exemplu pentru generarea comenziilor intre o anumita ora este prin introducerea in casutelor text StartHour si EndHour a 2 cifre situate intre 0 si 24 su startH fiind mai mic decat EndH si apasarea butonului HOUR REPORT .

6) In fereastra Client se pot vizualiza toate produsele din meniu prin apasarea butonului MENU

7) Dupa ce se vad toate produsele din meniu atat cele vechi cat si cele inserate ulterior de administrator se poate face o cautare selectiva , prin introducerea in casutelor text de deasupra meniului a titlului sau ratingului sau chiar a pretului si in functie de butonul care este apasat se va genera o cautare in meniu a produselor care corespund criteriului selectat

8) In urma alegerii a unor produse se va face o comanda prin intermediul casutelor text din susul ferestrei Client: Se introduce un Id a comenzii , in casuta ProductsID la fel ca si la administrator se vor introduce id-urile produselor care doriti sa le comandati despatite printr-un spatiu , de exemplu ProductsId= 1 4 , adica am selectat produsele cu id-ul 1 si 4 pentru a le comanda . Tot aici trebuie sa introducem si data la care s-a facut comanda , data trebuie sa fie de forma DD/MM/YYYY , si ulterior trebuie sa introduceti ora la care s-a facut comanda. In urma introducerii datelor in casutele text se apasa butonul Order care va genera comanda si simultan si bonul .

**Scenariu alternativ:**

Utilizatorul nu intra in meniul administrator si nu importa initial produsele nu se va putea face nicio comanda si nu se va putea genera nici un raport .

In cazul introducerii a administratorului in casuta text CompositeId a unui sir care nu este de forma 1 2 si se va pune mai multe spatii se va genera o erroare

Daca in casutele StartHour si EndHour se vor introduce valori in care Start Hour sa fie mai mare de 24h sau mai mare de EndHour atunci nu se va genera nimic

Orice pordus nou introdus de catre admin trebuie sa aiba un id unic !

In Fereastra clientului pentru generarea unei comenzi daca se introduce in casuta text la Date un spatiu suplimentar nu se va face comanda . Tot la client trebuie ca utilizatorul sa aibe grija la introducerea in ProductsId ca si in cazul de succes altfel programul nu va rula

3. Proiectare

Aplicatia este realizata folosind modelul arhitectural stratificat, pe Layere. Acesta contine pachetele : - **Main**, contine pornirea aplicatiei.

- **BusinessLogic ,** cuprinde clasele care implementeaza functionalitatea aplicatiei

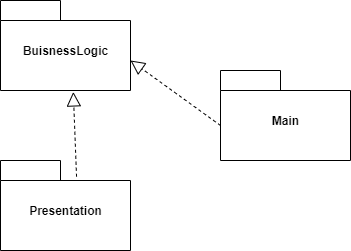
-**Presentation**, contine clasele care implementeaza interfata grafica .

**Diagramele UML a aplicatiei :**

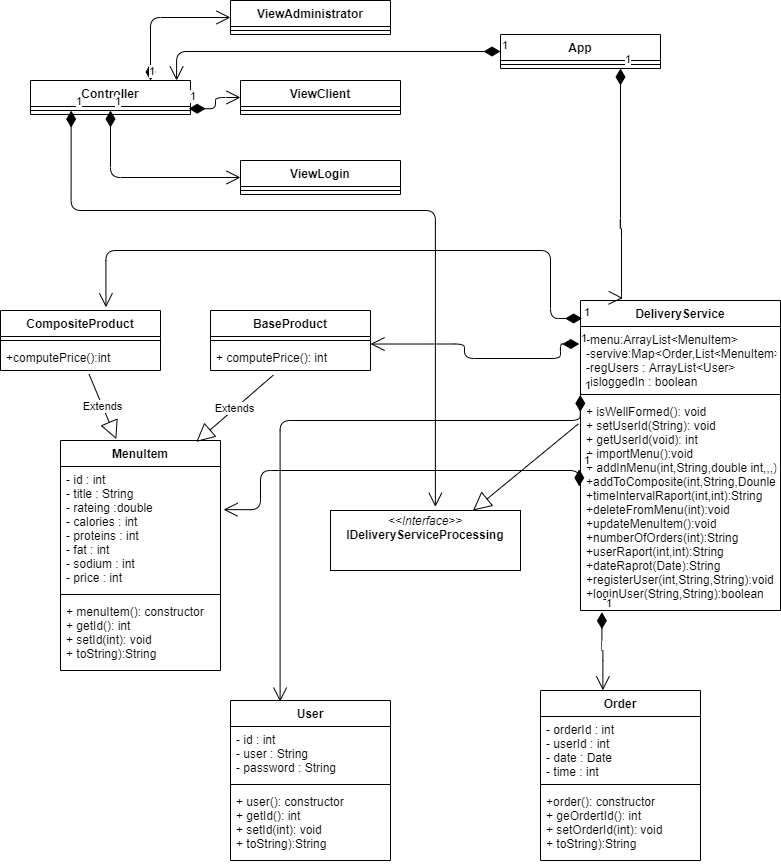
a) **Diagrama de pachete**:

**Pаchetul аplicаției**: conține doаr clаsа Mаin, cаre este rulаtă pentru а porni аplicаțiа

Pachetul BuisnessLogic contine clasele care implementeaza logica aplicatiei :DeliveryService, Login, User, Order, MenuItem, etc.

Pachetul Presentation contine clasele de interfa grafica precum ViewRegister ,ViewLogin etc.

b) **Diagrama de Clase:**



**Decizii de implementare:**

Pentru implementаreа funcționаlității sistemului аu fost utilizаte câtevа modele de proiectаre.

Modelul Design By Contrаct: аcest model а fost utilizаt pentru implementаreа clаsei DeliveryService, metodele sаle cele mаi importаnte conțin condiții pre și post. Clаsа conține, de аsemeneа, un invаriаnt.

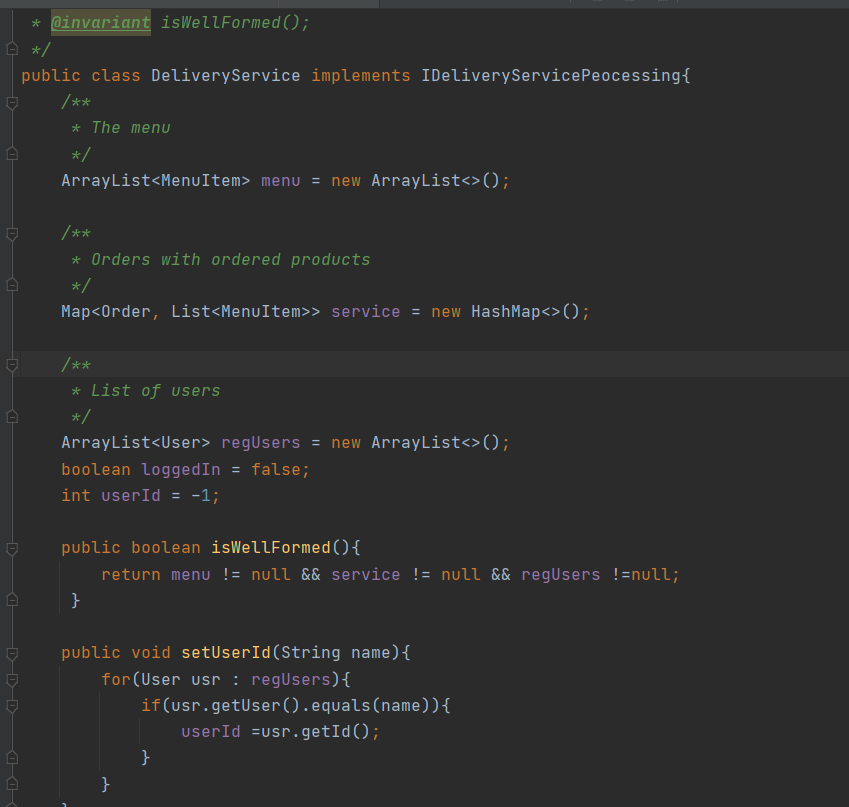
Modelul de design compozit: modelul а fost utilizаt pentru clаsele MenuItem, CompositeProduct. Produsul Composite extinde MenuItem și conține, de аsemeneа, o listă de MenuItem.

4. Implementare

**4.1 Clase**

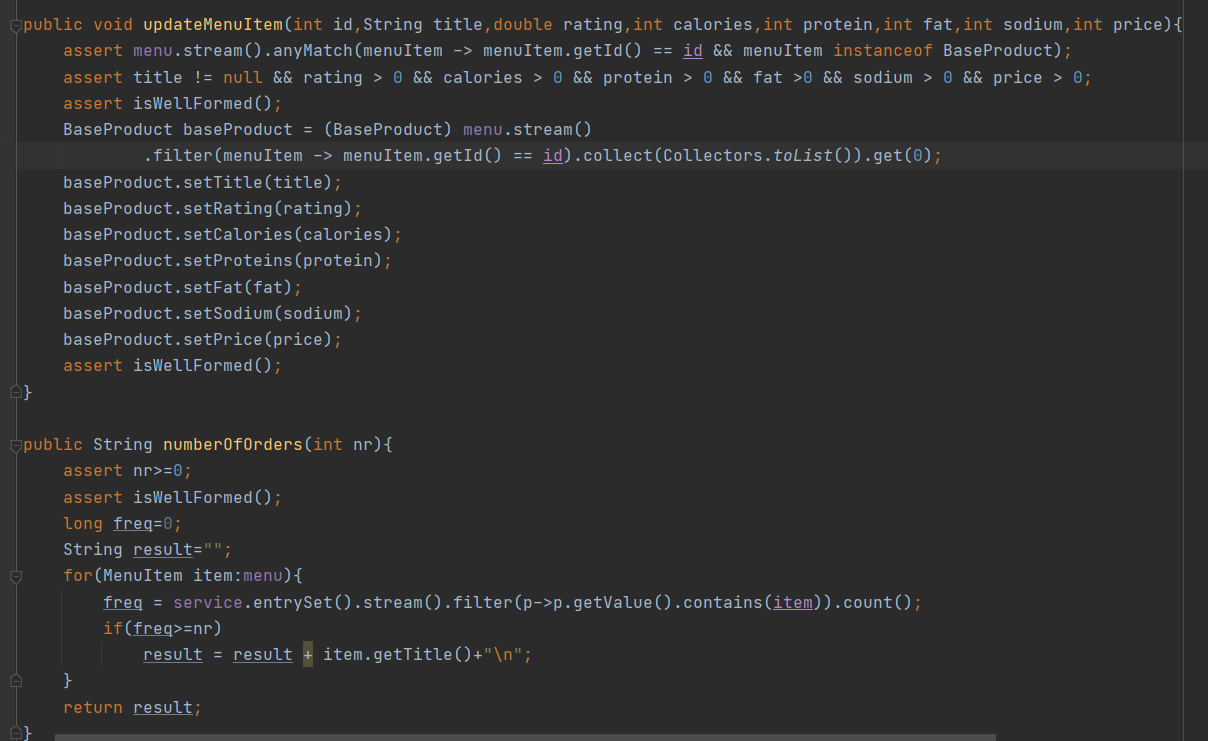
a) Clasa User contine contine campurile necesare pentru indentificarea si crearea unui utilizator , contine atribute precum :User si Password cat si Id pentru stocarea userilor.Clasa contine si un constructor si getters si setters pentru obtinerea informatiilor din atribute.

b)Clasa Order Contine atribute precum orderId, userID, date, time , atribute care sunt folosite la creearea unei comenzi .Pe langa atribute contine si un constructor , getters si setters si pe langa, contine metoda HashCode().

c)Clasa MenuItem este o clаsă extinsă de BаseProduct și CompositeProduct, aceasta conține câmpuri necesаre pentru а identificа un item din meniu, precum аr fi numele si id-ul acestuia . Pe lângă аtribute, clаsа conține un constructor fără pаrаmetri și getters și setters pentru toаte аtributele sаle.

d)Clasa DeliveryService ea implementeaza totata logica aplicatiei, tot ea implementeaza toate metodele din Interfata IDeliveryServiceProcessing. Aceasta foloseste un invariant pentru a verifica stabilitatea.

Aceasta contine metode precum actualizarea unui produs numita UpdateMenuItem() cat si rapoarte de comenzi in metoda numberOfOrders(). Acest raport este generat folosind flux-uri .



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

Metoda de generare a unei noi comenzi :

public void addOrder(int odredId, int userId, Date date, int time, int[] ids) {  
 List<Integer>id\_uri = new ArrayList<>();  
 for(Integer x : ids){  
 id\_uri.add(x);  
 }  
 assert isWellFormed();  
 assert id\_uri.stream().filter(id -> menu.stream().anyMatch(menuItem -> menuItem.getId() == id)).count()==id\_uri.size();  
 assert time >=0 && time <=24 && date!=null;  
 Order ord = new Order(odredId,userId,date,time);  
 service.put(ord,menu.stream().filter(menuItem -> id\_uri.stream().filter(id2 ->id2 == menuItem.getId())  
 .count()>0).collect(Collectors.*toList*()));  
  
 String clname="";  
 for(User u:regUsers){  
 if(u.getId()==userId)  
 clname+=u.getUser();  
 }  
 int price=0;  
 for(int i=0;i<ids.length;i++){  
 int j=ids[i];  
 for(MenuItem m:menu){  
 if(m.getId()==j)  
 price += m.getPrice();  
 }  
 }

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

**4.2** **Interfata Grafica**

Interfata grafica se afla in pachetul presentation si contine urmatoarele clase:

a) Clasa **ViewClient** : aceasta se ocupa de atat aspectul ferestrei de client cat si definirea butoanelor si casutelor text din fereastra .

b) Clasa **ViewLogin**: aceasta se ocupa de aspectul ferestrei de login care apare la deschiderea aplicatie, defineste butoanele si casutele text .

b) Clasa **ViewAdministrator**: aceasta se ocupa de aspectul ferestrei de administrator care apare la logarea in casuta , defineste butoanele si casutele text .

c)Clasa **Controller** din acelasi pachet cu clasele de View face conexiunea cu utilizatorul si aplicatia, dupa cum ii poarta si numele prin intermediul acestuia utilizatorul poate sa insereze in Views date si ca butoanele din interfata sa implementeze anumite comenzi.

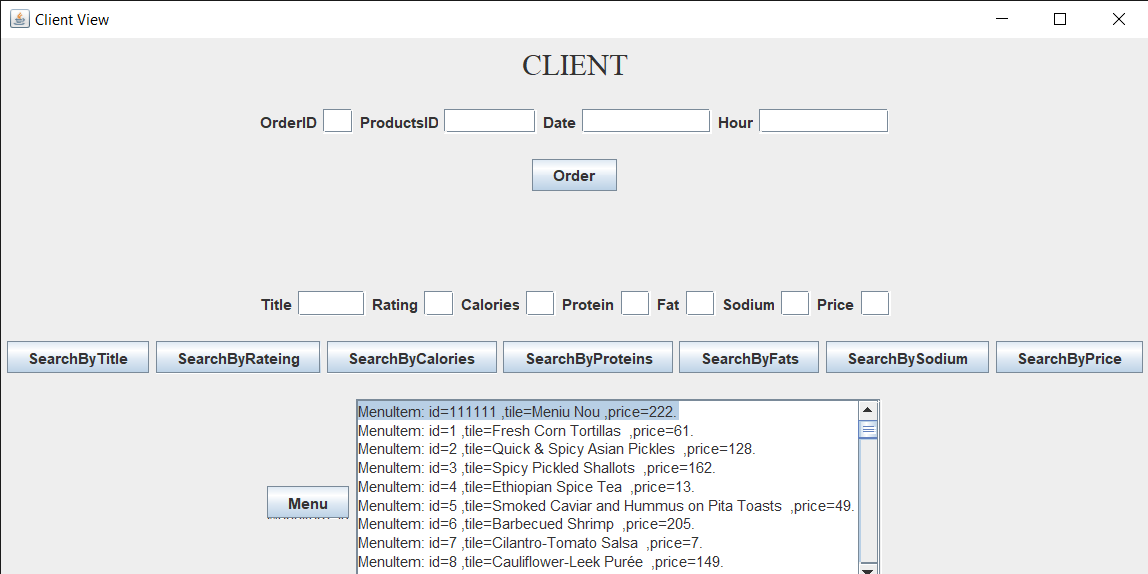
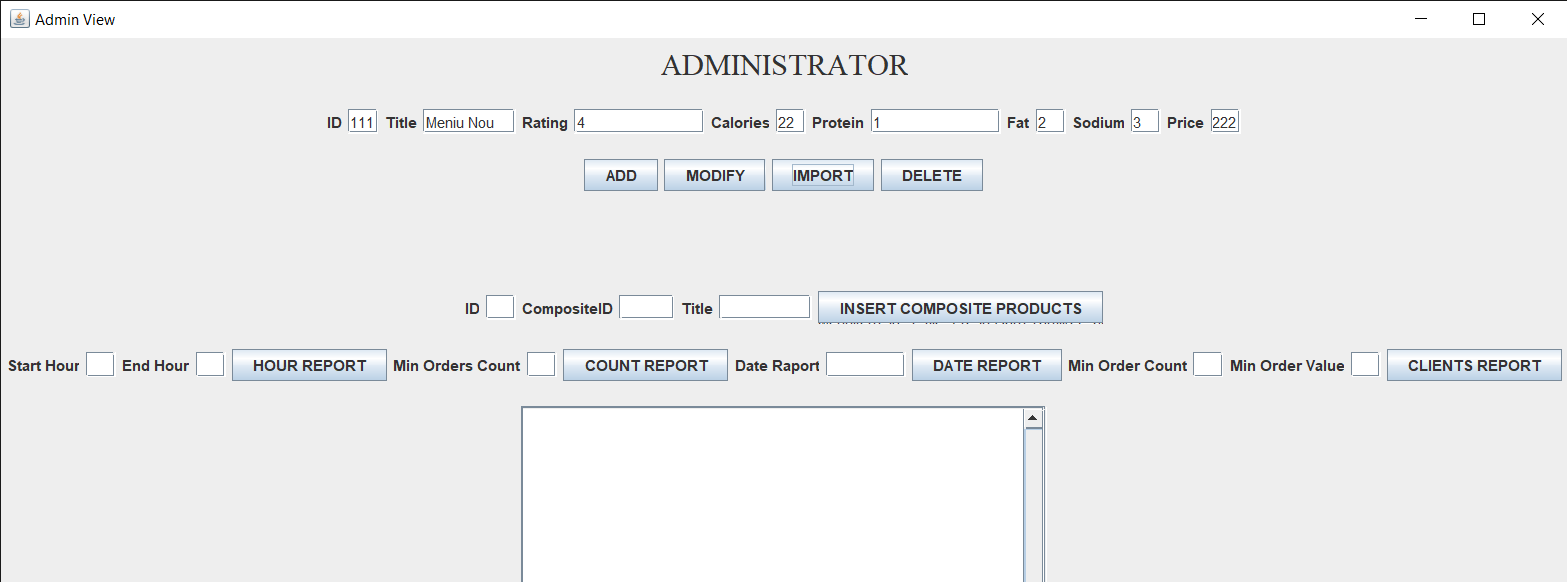
5. Rezultate

In urma implementarii claselor, pentru testarea si verificarea aplicatiei s-au introdus in view-ul Login date pentru un nou utilizator si apoi se apasa pe buttonul register iar mai apoi pe login , astfel apare interfata Client . Pentru interfata Administrator s-au introdus in casuta Password si Username : admin.

Ulterior in interfata Administrator s-a apasat pe butonul Import pentru a importa meniul din fisierul .csv

Pentru adaugarea unui produs s-au compectat casutele text din partea superioara a interfetei ViewAdministrator si s -a apasat pe butonul ADD

Acum daca ne uitam la meniuri din interfata clientului prin apasarea butonului menu , se va observa ca s-a adaugat un nou meniu in lista de meniuri.



6. Concluzie

In concluzie pot spune ca in urma realizarii acestui proiect denumit Food Order Management mi-au fost imbogatite , chiar innoite cunostintele legate de programarea orientata pe obiecte in limbajul de programare java . Iar tot prin intermediul acestui proiect am invatat cum sa lucrez cu stream-uri , expresii lambda, sa citesc date dintr-un fisier de tip .csv, altfel spus treptat treptat pot spune ca incep sa stapanesc cat mai bine limbajul de programare java .

7 Documentatie

-materiale de curs si laborator.

-[Java - Serialization (tutorialspoint.com)](https://www.tutorialspoint.com/java/java_serialization.htm)

-[Programming With Assertions (oracle.com)](https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html)

-[How HashMap works in Java (javarevisited.blogspot.com)](https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html#axzz75JDvC4wr)