

## FOOD DELIVERY MANAGEMENT SYSTEM

### [1]Obiectivul temei

Scopul acestei teme este de a proiecta și implementa un sistem de management al livrării alimentelor pentru o companie de catering. Clientul poate să comande produse din meniul companiei. Sistemul are trei tipuri de utilizatori care se conectează folosind un nume de utilizator și o parolă: administrator, client.

Administratorul poate:

- Importa setul inițial de produse care vor completa meniul dintr-un fișier “.csv”
- Gestiona produsele din meniu: poate adăuga / șterge / modifica produsele și crea produse noi

compuse din mai multe produse (un exemplu de produs compus ar putea fi denumit „meniul zilei” compus dintr-o supă, o friptură, o garnitură și un desert).

- Genera rapoarte despre comenzile efectuate luând în considerare următoarele criterii:
  - ❖ intervalul de timp al comenzilor – se generează un raport cu comenzile efectuate

între o oră de început și o oră de sfârșit, indiferent de dată.

- ❖ produsele comandate de mai multe ori decât un număr specificat.
- ❖ clienții care au comandat de mai multe ori decât un număr specificat și valoarea comenzii fiind mai mare decât o sumă specificată.

- ❖ produsele comandate într-o zi specificată cu numărul lor de comenzi.

Clientul poate:

- Să se înregistreze utilizând numele de utilizator și parola să pentru a fi reținut în sistem.
- Vizualiza lista produselor din meniu.
- Căuta produse pe baza unui sau mai multor criterii, cum ar fi un cuvânt cheie (de exemplu, „supă”) pe baza numărului de calorii / proteine / grăsimi / sodiu / preț.
- Crea o comandă formată din mai multe produse - pentru fiecare comandă data și ora vor fi

reținute și se va genera o factură care va lista produsele comandate, prețul total și data comenzii.

### Sub-obiectivele temei sunt:

- Analizarea problemei și identificarea cerințelor: se iau în considerare scenariile de utilizare și cerințe funcționale[2]
- Proiectarea aplicației pentru procesarea comenzilor: divizarea structurală (pachete), expresii Lambda[3]
- Implementarea aplicației de comenzi: prezentarea în detaliu a claselor componente[4]

## **[2]Analiza problemei**

**Caz 1 de utilizare:**Utilizatorul este de tipul client

### Scenariu principal reusit:

1. Utilizatorul introduce username si parola si apasa LOG IN(daca s-a inregistrat deja,altfel apasa REGISTER)
2. Se deschide fereastra cu meniul
3. Poate cauta produsul dupa titlu sau alte criterii vizibile in combo box
4. Din meniu selecteaza cate un produs si apasa butonul “Add” pentru adaugarea lui in lista comenzii
- 5.Dupa alegerea tuturor produselor, utilizatorul apasa butonul “Place the order”, iar comanda lui este finalizata

### Scenariu alternativ:

- a) Utilizatorul introduce username si parola si apasa LOG IN
- b) Datele introduse nu sunt gasite in lista clientilor si apare fereastra de avertizare
- c) Utilizatorul introduce datele corecte si apasa REGISTER
- d) Utilizatorul se logheaza din nou cu aceleasi date si apare fereastra cu meniul

**Caz 2 de utilizare:**Utilizatorul este de tipul administrator

### Scenariu principal reusit:

1. Utilizatorul introduce username si parola si apasa LOG IN(nu e necesara inregistrarea lui, deoarece se afla deja in lista de conturi)
2. Se deschide fereastra pentru modificarea meniului(adaugare/stergere/update produse)
3. Administratorul doreste adaugarea unui nou produs
4. Introduce toate caracteristicile produsului si apasa butonul “Add new product”
- 5.In meniul afisat se poate observa noul produs

### Scenariu alternativ:

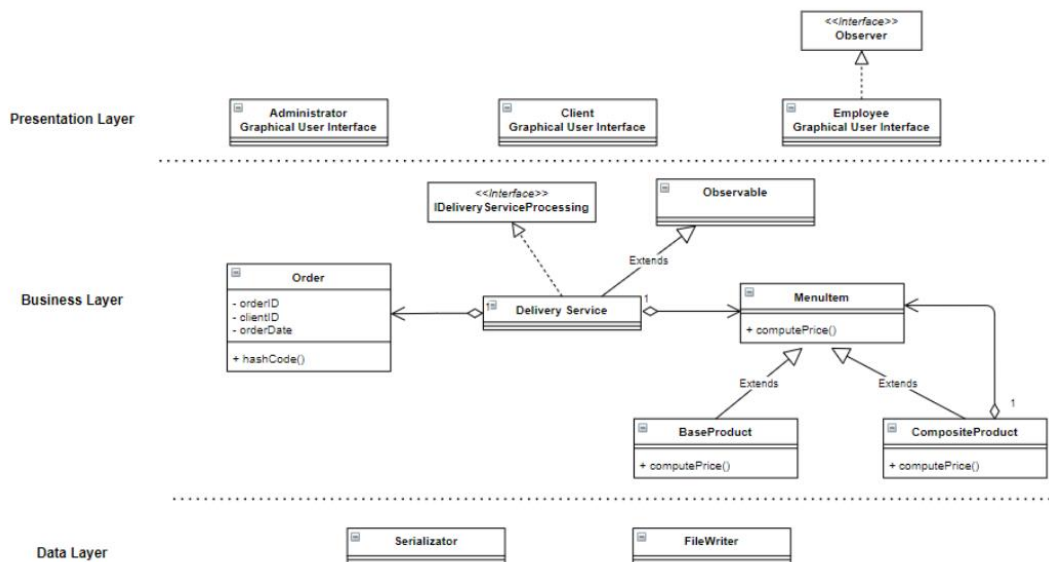
- a) Utilizatorul introduce username si parola si apasa LOG IN
- b) Datele introduse nu sunt apartin administratorului care are datele stabilite in momentul scrierii codului si apare fereastra de avertizare
- c) Utilizatorul introduce datele corecte si apasa LOG IN

## Cerinte functionale

Administratorul: poate modifica produsele din meniu(adaugare,stergere sau update),poate adauga noi produse compuse din alte produse, poate genera raporturi(cautare anumitor produse,anumitor client sau comenzi).

Clientul:poate plasa comenzi si sa caute produse dupa diverse criterii.

## [3]Proiectarea aplicatiei pentru comenzi



Aplicatia este divizata in urmatoarele pachete pentru structura organizata si eventuale modificari ulterioare:

- Presentation : contine clasele al caror scop este interfata grafica
- Business : contine clasele care compun logica din spatele aplicatiei
- Accounts : contine clasele corespunzatoare conturilor de tip client si adminstrator

## Expresii Lambda

Pentru ca expresiile lamda sunt similare metodelor, cu exceptia faptului ca nu au nevoie de nume,codul putand fi scris in interiorul parantezelor, acestea usureaza problemele de filtrare dupa criterii.In exemplul de mai jos, portiunea de cod descrie filtrarea dupa nume a unui produs din meniu.

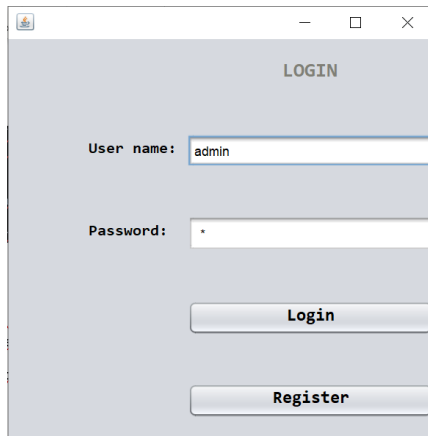
```
List<MenuItem> filteredList= deliveryService.getAllProducts().stream().filter(
p->p.getTitle().contains(keyword)).collect(Collectors.toList() );
HashSet<MenuItem> hashSet = new HashSet<>(); //produsele filtrate sa fie tot un HashSet asa cum erau retinute
initial
filteredList.forEach(value -> { hashSet.add(value); });
```

## Interfata

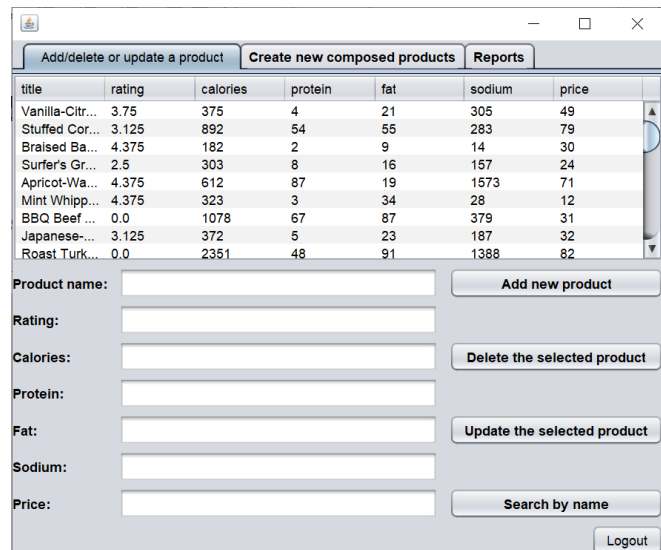
Este conceputa pentru 2 tipuri de utilizatori: administrator si clienti. Administratorul face managementul produselor existente in meniu, iar clientul poate doar sa plaseze comanda.

Prima fereastră este cea de logare

Introducem un cont de administrator



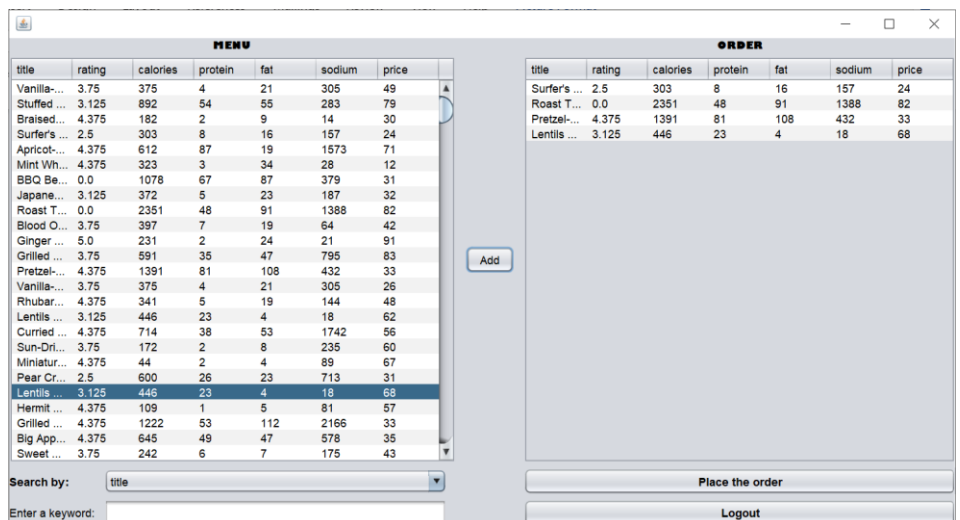
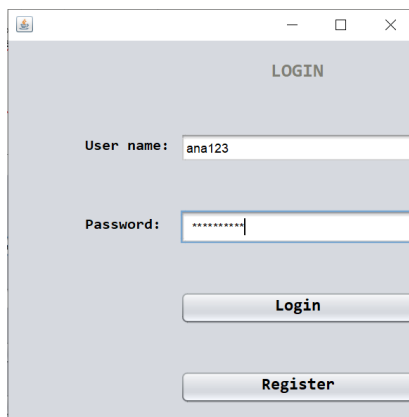
Urmatoarea fereastră contine 3 tab-uri in care poate lucra administratorul



title	rating	calories	protein	fat	sodium	price
Vanilla-Citr...	3.75	375	4	21	305	49
Stuffed Cor...	3.125	892	54	55	283	79
Braised Ba...	4.375	182	2	9	14	30
Surfer's Gr...	2.5	303	8	16	157	24
Apricot-Wa...	4.375	612	87	19	1573	71
Mint Whipp...	4.375	323	3	34	28	12
BBQ Beef ...	0.0	1078	67	87	379	31
Japanese-...	3.125	372	5	23	187	32
Roast Turk...	0.0	2351	48	91	1388	82

Introducem un cont de client, intai se inregistreaza si apoi se logheaza

Fereastră in care clientul isi poate plasa comanda



title	rating	calories	protein	fat	sodium	price
Vanilla-...	3.75	375	4	21	305	49
Stuffed ...	3.125	892	54	55	283	79
Braised ...	4.375	182	2	9	14	30
Surfer's ...	2.5	303	8	16	157	24
Apricot-...	4.375	612	87	19	1573	71
Mint Wh...	4.375	323	3	34	28	12
BBQ Be...	0.0	1078	67	87	379	31
Japane...	3.125	372	5	23	187	32
Roast T...	0.0	2351	48	91	1388	82
Blood O...	3.75	397	7	19	64	42
Ginger ...	5.0	231	2	24	21	91
Grilled ...	3.75	591	35	47	795	83
Pretzel-...	4.375	1391	81	108	432	33
Vanilla-...	3.75	375	4	21	305	26
Rhubar...	4.375	341	5	19	144	48
Lentils ...	3.125	446	23	4	18	62
Curried ...	4.375	714	38	53	1742	56
Sun-Dr...	3.75	172	2	8	235	60
Miniatur...	4.375	44	2	4	89	67
Pear Cr...	2.5	600	26	23	713	31
Lentils ...	3.125	446	23	4	18	68
Hermit ...	4.375	109	1	5	81	57
Grilled ...	4.375	1222	53	112	2166	33
Big App...	4.375	645	49	47	578	35
Sweet ...	3.75	242	6	7	175	43

Dupa finalizarea unei comenzi se genereaza factura cu totalul de plata si produsele cumparate:

```
1 04-06-2021 13:20:37
2 Clientul/clienta cu id-ul 0 a plasat comanda cu numarul 1
3 A achizitionat 3 produse in valoare TOTALA de 152
4 Produsele comandate:
5 [
6 Title='Vanilla-Citrus Tea Ring ',
7 rating=3.75,
8 calories=375,
9 protein=4,
10 fat=21,
11 sodium=305,
12 price=49
13 ,
14 Title='Stuffed Cornish Hens with Port Sauce ',
15 rating=3.125,
16 calories=892,
17 protein=54,
18 fat=55,
19 sodium=283,
20 price=79
21 ,
22 Title='Surfer's Granola ',
23 rating=2.5,
24 calories=303,
25 protein=8,
26 fat=16,
27 sodium=157,
28 price=24
29 ]
```

#### [4]Implementarea aplicatiei pentru comenzi

Clase:

- Pachet Accounts:

- a) Account: Clasa retine datele despre un cont: username si password
- b) AccountBank: retine lista de conturi de client sau administrator cu un id autoincrement pt fiecare cont nou de client
- c) AdminAccount: reprezinta un cont de admin cu username="admin" si password="1"
- d) ClientAccount: reprezinta un cont de client care primeste ca id, id-ul ultimului cont de client din lista+1

- Pachet Business:

- a) BaseProduct(mosteneste clasa MenuItem): reprezinta un produs simplu cu atributele title, rating, calorii, proteine, grasimi, sodiu si pret
- b) CompositeProduct(mosteneste clasa MenuItem): reprezinta un produs compus din mai multe produse simple-contine numele sau si o lista de produse simple
- c) MenuItem: clasa abstracta care are metode abstracte ce calculeaza ratingul, kaloriile, proteinele, grasimile, pretul si sodiul pentru produse simple sau compuse.
- d) Order: reprezinta o comanda cu id-ul propriu(incrementat la fiecare comanda), id-ul clientului ce a comandat si data comenzii
- e) IDeliveryServiceProcessing: interfata care contine headere de metode specifice modificarii produselor din meniu(adaugare, stergere, update)

f)DeliveryService: implementeaza metodele din interfata si realizeaza citirea datelor din fisierul .csv si operatiile de baza(adaugare,stergere,update produs,creare produs nou compus,creare comanda).Are ca si atribut lista de produse citita din .csv de tip HashSet<MenuItem> pentru a evita duplicatele si retine obiecte de tip BaseProduct sau ComposedProduct.

Citirea din .csv se face astfel:

```
Reader reader = Files.newBufferedReader(Paths.get(CSV_FILE_PATH));

ColumnPositionMappingStrategy strategy = new ColumnPositionMappingStrategy();
strategy.setType(BaseProduct.class);
String[] memberFieldsToBindTo = { "title", "rating", "calories",
    "protein", "fat", "sodium", "price" };
strategy.setColumnMapping(memberFieldsToBindTo);
//Se mapeaza coloanele cu pozitiile date in sirul memberFieldsToBindTo

//Mai jos vom avea o lista cu obiecte BaseProduct citite din fisier,evitand primul rand care este header-ul tabelului
CsvToBean<BaseProduct> csvToBean = new CsvToBeanBuilder(reader)
    .withMappingStrategy(strategy)
    .withSkipLines(1)
    .withIgnoreLeadingWhiteSpace(true)
    .build();
```

- Pachet Presentation:

- a) AdminController: controleaza interactiunea dintre utilizatorul admin si interfata corespunzatoare administratorului,deoarece clientul si adminul au tipuri diferite de interfețe grafice,unul doar comanda produse si celalalt le gestioneaza.Sunt 3 tab-uri disponibile,cel de modificare produse din meniu,cel de creare produs compus nou si cel in care se face generearea raportului.

Daca intra in primul tab,se face refresh la meniu pentru a fi mereu actualizate produsele din meniu.Prin refresh doar se genereaza tabelul cu produsele din lista data ca parametru si se seteaza ca model tabelului din interfata.La fel se actualizeaza si tabelul din al 2-lea tab din care se pot selecta produse pentru crearea de noi produse compuse.Butonul de logout va face invizibila fereastra pentru admin si vizibila fereastra de logare.Generearea raporturilor cu intervalul de timp al comenzilor, produsele comandate de mai multe ori decat un numar specificat, clienții care au comandat de mai multe ori decat un număr specificat și valoarea comenzii fiind mai mare decat o sumă specificată si produsele comandate într-o zi specificată cu numărul lor de comenzi se regasesc tot in aceasta clasa, construite prin metoda expresiilor lambda.

Exemplu expresie lambda pentru raportul -produse comandate într-o zi specificată cu numărul lor de comenzi:

```
String date=guiAdmin.getDate().getText();
HashMap<MenuItem,Integer> o=new HashMap<>(); //are cheia produs si valoarea numarul de comenzi corespunzator

DateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
DateFormat sd = new SimpleDateFormat("dd-MM-yyyy HH:mm:ss");

deliveryService.getOrders().keySet().stream().filter(order->
```

```

{
    try { return sdf.format(sd.parse(order.getOrderDate())).equals(date); }
    catch (ParseException exception) { exception.printStackTrace(); }
    return false; //pana aici comenzile sunt filtrate dupa data
})

    .map(order -> deliveryService.getProductList(order))
    .forEach(
//se parcurge lista filtrata si se incrementeaza acolo unde un produs apare de mai multe ori
        list -> list.forEach(menuItem -> {
            if(o.containsKey(menuItem))
                o.replace(menuItem,o.get(menuItem)+1);
            else
                o.put(menuItem,1);
        }
    ));
StringBuilder sb=new StringBuilder();
sb.append("Produsele comandate din ziua specificata :\n");

o.keySet().forEach( menuItem ->
    sb.append(menuItem.getTitle()).append(" cu numar comenzi= ").append(o.get(menuItem)).append("\n\n") );

```

- b) ClientController: controleaza interactiunea dintre utilizatorul client si interfata corespunzatoare clientului. Automat la aparitia ferestrei se actualizeaza tabelul cu meniul din care isi poate alege clientul produse. Combo box-ul are listenere care, la selectarea unui criteriu (titlu, rating, calorii, proteine, grasimi, sodiu sau pret), va genera automat cautarea dupa acel criteriu si cuvantul cheie introdus in text-box.

Butonul de logout va face invizibila fereastra pentru client si vizibila fereastra de logare.

Selectarea si adaugarea produsului in lista comenzii si plasarea comenzii este implementata in aceasta clasa.

Exemplu pentru filtrarea dupa titlu:

```

String keyword=guiClient.getKeyword().getText();
JComboBox comboBox =(JComboBox) e.getSource();
if(comboBox.getSelectedIndex()==0) //search dupa title
{
    List<MenuItem> filteredList=
deliveryService.getAllProducts().stream().filter(p->
    p.getTitle().contains(keyword)).
    collect(Collectors.toList());

//daca vreun obiect din lista are titlul cautat, atunci se colecteaza in
filteredList

//La final se transforma filteredList in HashSet, deoarece metoda de generare tabel cere ca parametru o lista de tip
HashSet

HashSet<MenuItem> hashSet = new HashSet<>();

filteredList.forEach(value -> { hashSet.add(value); });
tableProducts.generateTable(hashSet);
guiClient.setProductsTableModel(tableProducts.getTableModel());

```

- c) LoginController: controleaza interactiunea dintre utilizator si interfata pentru logare,fiind prima fereasta care apare la rularea aplicatiei.Retine o lista de conturi,de aceea daca se apasa LOGIN si clientul nu apare in lista,trebuie intai inregistrat si apoi logat.Contul de admin nu trebuie creat, el este inregistrat automat.Daca username si parola apartin adminului, atunci se face vizibila fereasta pentru admin si invizibila fereasta pentru logare.Altfel, rezulta ca utilizatorul este un client si daca datele introduse sunt corecte,apare fereasta pentru client si dispare cea de logare.

Metoda de LOGIN doar cauta contul si verifica daca datele sunt corecte,iar daca nu exista apare un mesaj de avertizare.

Metoda de REGISTER adauga contul in lista de conturi.

- d) MainController: gestioneaza vizibilitatea ferestrelor. LoginController-ul comunica cu aceasta clasa si ii transmite care ferestre trebuie sa fie vizibile si care invizibile, iar aceasta clasa seteaza vizibilitatea ferestrei transmise.MainController are ca attribute toate cele 3 ferestre-GUILogin,GUIAdmin,GUIClient-si astfel seteaza vizibilitatea frame-urilor.

Exemplu:

```
public void setGuiLoginVisible(boolean visibility)
{
    guiLogin.setVisible(visibility);
}
```

- e) Table: creeaza un obiect de tip TableModel cu numele coloanelor { "titlu", "rating", "calories", "protein", "fat", "sodium", "price" } si popularea tabelului cu elemente din lista de produse data ca parametru.
- f) GUILogin,GUIAdmin,GUIClient: clasele interfetei fiecare avand ActionListener pentru toate butoanele si combo box

## Concluzii

Tema sistemului de livrare de mancare a adus cu sine noutati in ceea ce priveste interactiunea dintre un meniu cu produse de dimensiuni mari si mai multi utilizatori,2 tipuri de utilizatori fiecare avand capabilitatile specifice.Cautarea produselor dupa diverse criterii sau generarea raporturilor dupa anumite specificatii a fost imbunatatita cu ajutorul expresiilor lambda care au transformat o implementare normala in cateva linii de cod.

Dezvoltarile ulterioare ar putea include implementarea serializarii,pentru ca la oprirea aplicatiei sa nu se piarda toate conturile inregistrate.De asemenea,o fereasta noua pentru angajat ar putea fi creata,iar la fiecare comanda noua sa-i apara angajatului comenzile.Dupa ce angajatul finalizeaza comanda,el ar putea apasa un buton „Ridica comanda”, afisand clientului respectiv acest mesaj.



## **Bibliografie**

<https://www.callicoder.com/java-read-write-csv-file-opencsv/>

Support Presentation

StackOverFlow