

# DOCUMENTATIE

## TEMA 4

### *FOOD DELIVERY MANAGEMENT SYSTEM*

*Nume student: SZAKACS EMMA-EVELIN*

*Grupa: 30221*

# CUPRINS

1. Obiectivul temei.....	3
2. Analiza problemei, modelare, scenarii, cazuri de utilizare .....	3
3. Proiectare .....	6
3.1. Diagrama UML .....	6
3.2. Structuri de date folosite .....	6
4. Implementare .....	7
4.1. Pachete, clase si metode.....	7
4.2. Graphical User Interface .....	10
5. Rezultate.....	13
6. Concluzii.....	14
7. Bibliografie .....	15

## 1. Obiectivul temei

**Obiectivul principal** acestei teme a fost sa implementam un sistem de management pentru un sistem de livrarea a mancarii. Utilizatorii trebuie sa se poata loga in aplicatie avand unul din rolurile: client, administrator, angajat. Clientul poate sa caute produse dupa anumite criterii si sa dea comenzi. Angajatul va primi notificari legate de comenzile puse. Administratorul are cel mai dificil rol. El poate adauga, sterge sau modifica produse in meniu. Produsele vor fi luate din dintr-un fisier CSV. De asemenea, admininstratorul poate genera rapoarte legat de comenzi sau sa adauge in meniu un produs compus din mai multe produse de baza.

**Obiectivele secundare** care contribuie la realizarea obiectivului principal sunt:

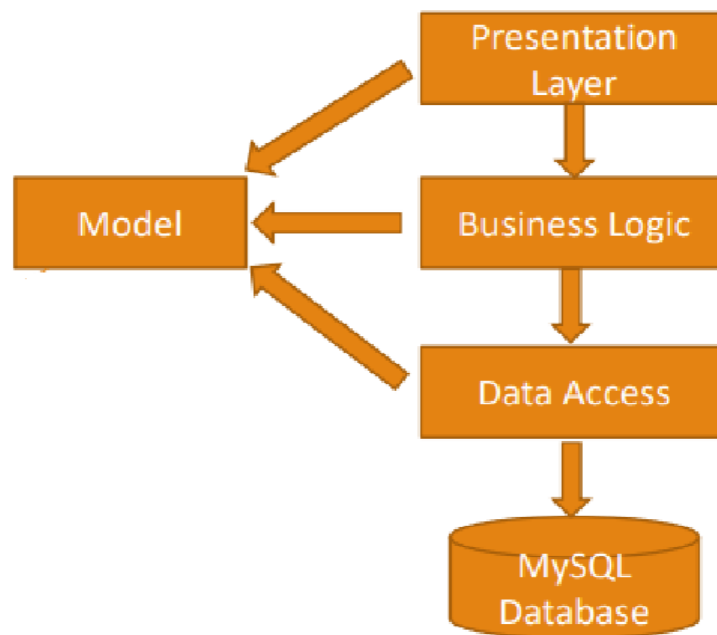
1. Analiza si identificarea cerintelor.	Capitolul 1
2. Modelarea problemei	Capitolul 3
3 Implementarea propriu zisa a aplicatiei	Capitolul 2
4. Folosirea unei interfete grafice "User Friendly" – folosind Java Swing	Capitolul 4
5. Testarea aplicatiei	Capitolul 5 - Rezultate

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

Cel mai comun model de arhitectură este modelul de arhitectură stratificat, altfel cunoscut sub numele de model de arhitectură n-tier. Acest model este standardul de facto pentru majoritatea aplicațiilor Java EE și, prin urmare, este cunoscut pe scară largă de majoritatea arhitecților, designerilor și dezvoltatorilor. Modelul de arhitectură stratificat se potrivește îndeaproape cu structurile tradiționale de comunicare și organizare IT întâlnite în majoritatea companiilor, ceea ce îl face o alegere naturală pentru majoritatea incercărilor de dezvoltare a aplicațiilor de afaceri.

Componentele din modelul de arhitectură stratificată sunt organizate în straturi orizontale, fiecare strat îndeplinind un rol specific în cadrul aplicației (de exemplu, logica de

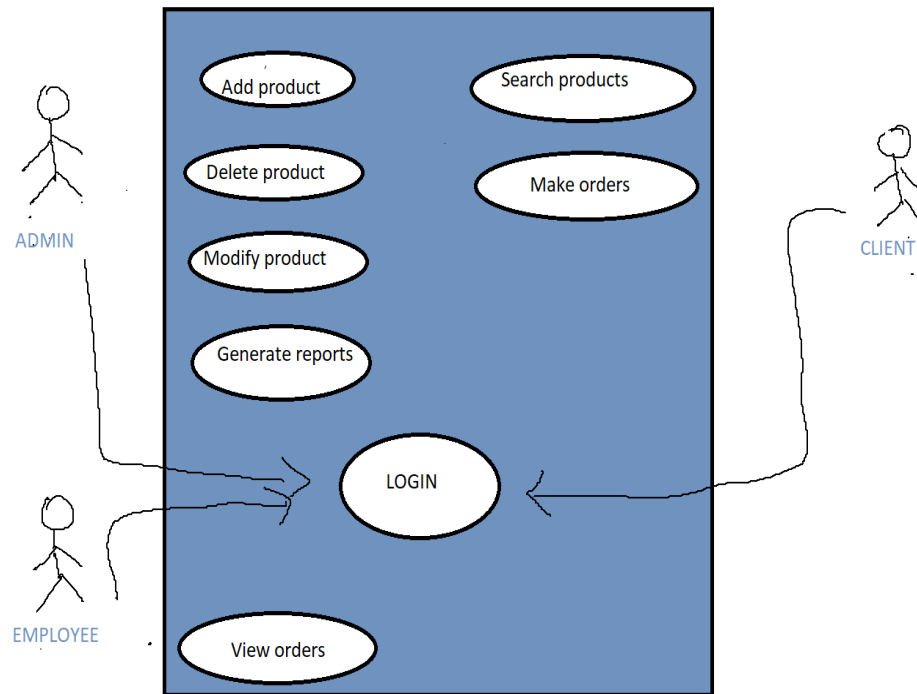
prezentare - BLL sau logica de afaceri - DAO). Deși modelul de arhitectură stratificată nu specifică numărul și tipurile de straturi care trebuie să existe în model, majoritatea arhitecturilor stratificate constau din patru straturi standard: prezentare, afaceri, persistență și bază de date. În unele cazuri, stratul de afaceri și stratul de persistență sunt combinate într-un singur strat de afaceri, în special atunci când logica de persistență (de exemplu, SQL sau HSQL) este încorporată în componentele stratului de afaceri. Astfel, aplicațiile mai mici pot avea doar trei straturi, în timp ce aplicațiile de afaceri mai mari și mai complexe pot conține cinci sau mai multe straturi.



### ***Cazuri de utilizare***

Actorul principal în acest scenariu este angajatul care se va ocupa de gestionarea unor comezi. Angajatorul poate selecta opțiunea de client, produs sau order. Dacă selectează produs, are opțiunile de inserare, stergere și editare a unui produs. Va avea la vedere un tabel cu toate produsele existente. Dacă vrea să proceseze un produs, va apăsa pe randul din tabel specific produsului. La fel va proceda și în cazul în care vrea să introducă, să steargă sau să editeze un

client. In cazul in care angajatul vrea sa realizeze o comanda, va alege un client si un produs, va seta cantitatea si va apasa pe un buton care v-a realiza comanda.



## ***Modelarea proiectului***

Am impartit clasele proiectului in mai multe pachete.

### **BLL**

Clase continue: BaseProduct, MenuItem, CompositeProduct, DeliveryService, iDeliveryService, Order, Usee, Bill

### **DAL**

Clase continue: FileSplit, Serializator

### **Presentation**

Clase continue: -Controlerele: Controler, AdminControler, RaportControler, EmployeeControler, ClientControler

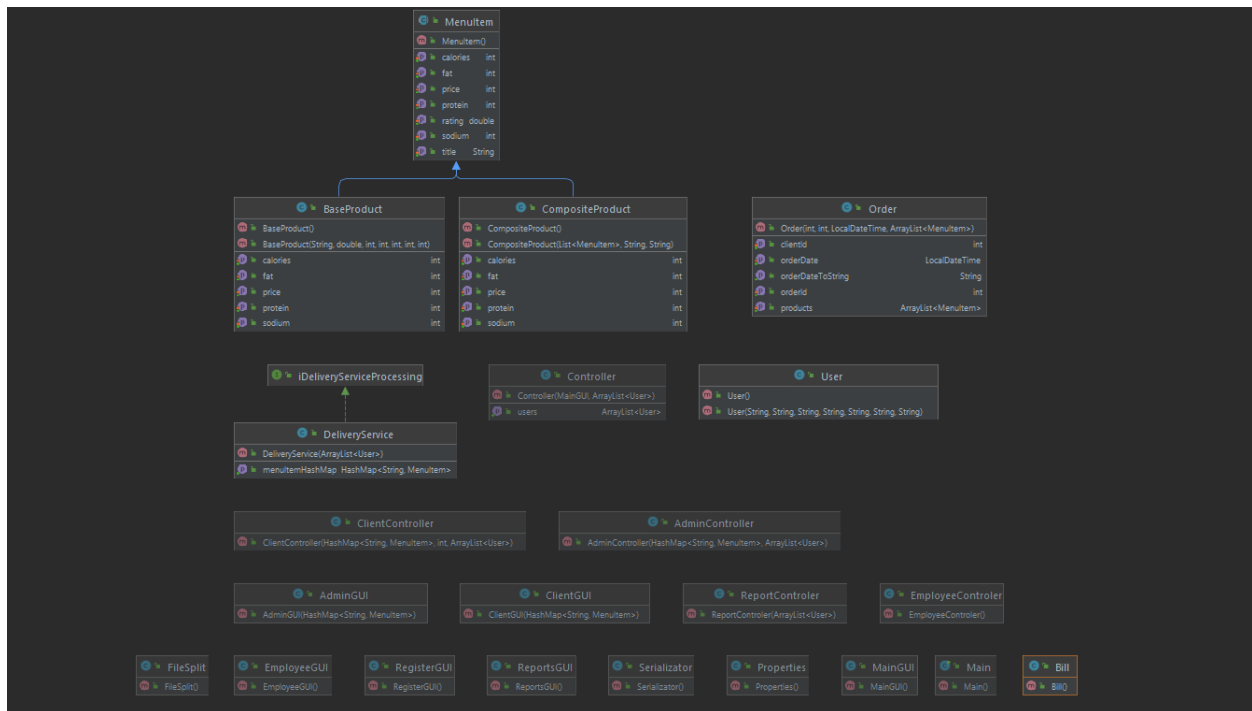
-GUI: MainGUI, RegisterGUI, ReportsGUI, ClientGUI, AdminGUI, EmployeeGUI

### 3. Proiectare

#### 3.1. Diagrama UML

Unified Modeling Language sau UML pe scurt este un limbaj standard pentru descrierea de modele si specificatii pentru software. UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora ( numite și obiecte ). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT.

Diagrama UML generata pentru proiectul acesta este urmatoarea:



#### 3.2. Structuri de date folosite

Listele in Java ofera mentinerea colectiei ordonate. Listele contin metoda de inserare, de actualizare, de stergere si de cautare a elementelor, fiind mult mai eficiente decat array-urile. De asemenea listele permit elemente duplicate si pot retine elemente nule. Am folosit liste pentru retinerea diferitelor obiecte.

HashMap în Java este o colecție bazată pe hartă și constă din perechi cheie-valoare. Un HashMap este notat cu sau. Un element HashMap poate fi accesat folosind o cheie, adică trebuie să știm cheia pentru a accesa elementul HashMap. Am folosit HashMap pentru

salvarea produselor din CSV si pentru salvarea comenzilor, ca mai apoi cand folosesc serializarea sa gestionez totul mai usor.

## 4. Implementare

### 4.1. Pachete, clase si metode

#### PACHETUL BLL

##### Clasa MenuItem

Este o clasa abstracta care implementeaza Serializable astfel incat clasele care o extend sa poata folosi serializarea. Variabilele instantate sunt de tipul protected astfel incat clasele care extend clasa abstracta sa aiba acces la variabile ca si cum ar fi proprii. Metodele de get sunt abstracte, in afara de titlu si rating.

##### Clasa BaseProduct

Extinde clasa abstracta MenuItem si implementeaza metodele abstracte.

##### Clasa CompositeProduct

Extinde clasa abstracta MenuItem si ii implementeaza metodele. Ca variabila instantata avea o lista de MenuItems, iar pentru metodele de get se va face suma pentru fiecare variabila.

##### Clasa User

Contine variabile instantate pentru attributele unui user si constructorul.

##### Clasa Order

Are ca variabile instantate orderID, ClientID, data locala si lista de produse a comenzii.

##### Interfata IDeliveryServiceProcessing

Este o interfata implementata ulterior de catre clasa DeliveryService. Metodele care vor fi implementate sunt:

```
public void createOrder(Order order); //implies computing the price
public void createBill(int id);
public void searchProduct(int id);
public void addProduct(BaseProduct menuItem);
public void deleteProduct(String key);
public void modifyProduct(BaseProduct menuItem);
public void addCompositeProduct(ArrayList<String> titles, String name, String rating);
```

### Clasa DeliveryService

\_Gestioneaza toata aplicatia, legand intre ele toate componentele. In aceasta clasa se afla HashMap-ul pentru produsele din meniu si HshMap-ul pentru comenzi.

#### Metode:

```
public void createOrder(Order order)
```

Primește ca parametru o comandă, pe care o introduce în HashMap-ul cu comenzi având ca și cheie comanda respectivă și ca și valoare lista de produse din comandă. După aceea s-a introdus comanda în HashMap, se serializează comenzile. De asemenea, se apelează metoda de createBill din clasa Bill pentru a se crea un fișier cu comanda respectivă. Pentru notificarea angajatului asupra comenzii efectuate, se apelează metoda notifyObservers având ca parametru comanda respectivă.

```
public void addCompositeProduct(ArrayList<String> titles, String name, String rating)
```

Această metodă primește ca parametru o listă de stringuri cu titlurile produselor care compun produsul, numele noului produs și ratingul lui. Se caută produsele din listă în HashMap-ul produsului, iar apoi se creează noul produs compus. După creare se serializează.

```
public ArrayList<MenuItem> filterProducts(String title, int minPrice, int maxPrice, int minRating, int maxRating, int minCalories, int maxCalories, int minFat, int maxFat, int minSodium, int maxSodium, int minProtein, int maxProtein){
```

Această metodă primește ca parametrii valorile din interfața clientului pentru filtrarea produselor. Pentru implementarea filtrării am folosit stream-uri și lambda expressions. Mai întâi am luat perechea (cheie-valoare) din HashMap-ul produselor, apoi am creat un stream pe care l-am filtrat în funcție de parametrii dați. La final am folosit map pentru extragerea doar a listei de menuitems.

```
public void reportOne(int startHour, int endHour)
```

Reprezintă metoda pentru primul raport, care generează un raport despre comenzi, luând în considerare următoarea criterie: comenzile date în intervalul de timp startHour și endHour. Pentru implementare am folosit stream-uri și lambda expressions astfel încât să salvez într-o listă de comenzi comenzile filtrate după timpul dorit. După ce am obținut rezultatul dorit, voi scrie într-un nou fișier raportul respectiv.

```
public void reportTwo(int times)
```

Reprezintă metoda pentru al doilea raport. Acest raport generează raportul despre comenzile care au fost efectuate de mai multe ori decât un număr dat ca parametru. Pentru această metodă am creat un nou HashMap care are ca și cheie titlul unui produs și ca și valoare un



Integer care reprezinta numarul de aparitii in comenzi. Cu stream-uri iau toate comenzile si adaug in HashMap-ul creat produsul cu numarul de aparitii. Dupa aceea filtrez produsele din hashMap astfel incat sa iau produsele cu numarul de aparitii mai mare decat parametrul times. Dupa aceea scriu intr-un nou fisier raportul.

```
public void reportThree(int minNumber, int minValue){  
public void reportFour(int year, int month, int day){
```

Aceste metode se executa asemanator cu metoda pentru raportul doi, avand doar alte criterii de filtrare

Clasa mai are si metode pentru adaugarea, stergerea sau modificarea unui produs de catre administrator, acestea fiind niste metode simple executate pe hashMap.

## PACHETUL DAL

### Clasa FileSplit

Contine doar metoda split() care foloseste stream-uri pentru creerea obiectelor de tip BaseProduct luate dintr-un fisier CSV. Ideea principala este sa sar peste prima linie care reprezinta header-ul tabelului, iar apoi sa despart fiecare linie in Stringuri, despartite de virgula.

### Clasa Serializator

Contine metode pentru serializarea si deserializarea utilizatorilor, comenzilor si produselor. Ideea serializarii este ca de fiecare data cand inchid aplicatia, valorile sa fie salvate in fisiere astfel incat sa nu se piarda datele. Cand deschid iar aplicatia folosesc deserializarea pentru luarea datelor din fisiere.

## PACHETUL PRESENTATION

### Clasa Controller

Initializeaza celelalte controlere. Acest controller este pentru registerGUI si mainGUI. Cand se apasa butonul de login din mainGUI se verifica daca user-ul este client, admin sau angajat prin verificarea rolului. In functie de ce rol are se va deschide fereastra specifica utilizatorului. La apasarea butonului register din registerGUI se va crea un nou utilizator, luand valorile din field-uri, apoi se va adauga in lista de useri si se va serializa.

### Clasa AdminController

Contine ActionListeners pentru butoanele din interfata administratorului. Pentru butoanele delete, add si modify se executa instructiunile pe HashMap-ul de produse, iar apoi se apeleaza

metodele specific din DeliveryService. La apasarea butonului “+” se adauga produsul selectat intr-o lista de produse care vor compune produsul compus. La apasarea butonului “Generate Reports” se instantiaza controller-ul pentru rapoarte.

#### Clasa ClientController

Contine ActionListeners pentru butoanele din interfata clientului. La apasarea butonului clear se sterge lista de produse din cosul cumparatorului. La apasarea butonului “delete from cart” se sterge produsul selectat din lcosul de cumparaturi. La apasarea butonului order se creeaza o noua comanda care va avea id-ul numarul de comenzi curente + 1. Cu noua comanda se apleaza metoda de createOrder din DeliveryService. La apasarea butonului search se iau valorile din textField-uri si se salveaza intr-o lista de MenuItems lista returnata de metoda filterProducts din DeliveryService.

#### Clasa ReportController

la din interfata reportsGUI valorile din textField-uri, iar in functie de butonul apasat genereaza raportul specific numarului butonului, apeland metoda specifica din DeliverySevice.

#### Clasa EmployeeControler

Are o metoda de update care primeste ca parametrii un Observable si un obiect si notifica angajatul despre noile comeni efectuate.

## 4.2. Graphical User Interface

Pentru realizarea interfetei grafice am implementat 6 clase diferite.

**Clasa MainGUI** initiaza pagina principala de unde utilizatorul se va loga cu numele de utilizator si parola in cazul in care are deja un cont creat, altfel apasa pe butonul de register pentru a-si creea un nou cont.



**Clasa RegisterGUI** va avea TextField-uri pentru crearea unui nou cont. Utilizatorul v-a selecta pentru ce cont se inregistreaza , alegant din ComboBox dintre variantele “Client”, “Admin” si “Employee”, apoi dupa apasarea butonului “Register”, un nou cont va fi creat si salvat prin serializarea in fisierul de utilizatori.

REGISTER

### Create account

First name

Last name

Email address

Username

Password

Mobile number

client

Register

**Clasa ClientGUI** contine un JTable care contine toate produsele din meniu. In cazul in care clientul vrea sa faca o comanda apasa pe un produs din tabel si apasa butonul “Add to cart”. Daca vrea sa efectueze comanda apasa butonul “Make order”. De asemenea, clientul poate filtra produsul dupa anumite criterii.

CLIENT

### SEARCH FOR PRODUCTS

Title

Min Price

Max Price

Min Rating

Max Rating

Min Calorie

Max Calorie

Min Protein

Max Protein

Min Fat

Max Fat

Min Sodium

Max Sodium

Search

Add to cart

Beef and Carrot Stew with Dark Beer

The Ultimate Bolognese Sauce

Chicken Burgers with Lemon and Tarragon

Basmati Rice with Summer Vegetable Salad

Bacon and Cashew Caramel Corn

Make Order

Clear cart

Title	Rating	Calories	Protein	Fat	Sodium	Price
Candie...	3.75	305	4	16	54	38
Beef a...	4.375	647	37	54	325	73
Brandy...	0.0	278	1	12	15	42
Aunt T...	5.0	377	3	24	201	65
Scallop...	4.375	304	34	12	1809	23
Savory...	1.25	105	5	7	130	81
Broiled...	4.375	423	6	44	863	82
Poach...	4.375	716	46	53	1238	99
Mustar...	0.0	210	1	23	73	30
The Ult...	3.75	552	5	8	85	52
Butter...	4.375	438	9	25	65	25
Spicy ...	3.75	202	19	8	815	69
Lobste...	3.75	838	15	61	316	30
Sautée...	3.125	1218	46	110	2913	54
Sweet ...	2.5	134	1	1	2919	12
Sautée...	3.125	148	4	10	55	72
Banan...	3.75	669	11	46	420	27
Chicke...	3.75	445	24	38	662	75
Butters...	4.375	437	4	31	346	82
Basma...	4.375	656	24	32	202	34
Crudite...	3.75	563	6	50	314	46
Grog ...	0.0	173	1	0	3	28
Chicke...	3.75	246	12	21	50	39
Pear a...	4.375	681	19	40	1162	96
Bacon ...	4.375	298	4	14	309	10
Napa ...	4.375	103	3	6	204	21
Strawb...	3.75	207	2	5	33	26

**Clasa AdminGUI** contine un JTable care contine toate produsele accesate dintr-un fisier CSV. La apasarea butonului “Add base product” se adauga un nou produs de baza in meniu. La apasarea

butonului “Delete product” se sterge produsul selectat din tabel. La apasarea butonului “Modify product” se modifica un produs selectat. In cazul in care administratorul vrea sa genereze rapoartele, va apasa pe butonul “Generate reports” si se va deschide o noua fereastra. Daca administratorul vrea sa creeze un produs compus, va apasa pe un produs din tabel, apoi pe butonul de “+” si va repeat pasii pana adauga toate produsele dorite in lista, apoi va scrie in JTextField-uri numele si rating-ul noului produs.

The screenshot shows the 'ADMINISTRATOR' window. On the left is a sidebar with the following buttons: 'Add base product', 'Delete product', 'Modify product', 'Generate reports', and 'New composite product'. Below these is a '+' button and two input fields labeled 'Composite product name' and 'Composite product rating'. On the right is a table with the following columns: Title, Rating, Calories, Protein, Fat, Sodium, and Price. The table contains 25 rows of product data.

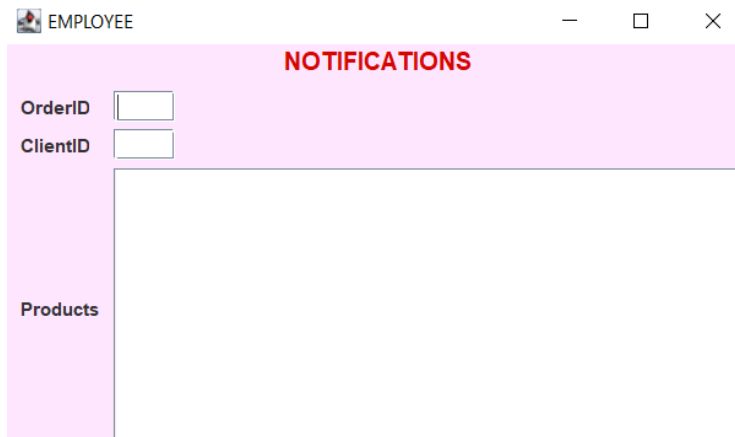
Title	Rating	Calories	Protein	Fat	Sodium	Price
Candie...	3.75	305	4	16	54	38
Beef a...	4.375	647	37	54	325	73
Brandy...	0.0	278	1	12	15	42
Aunt T...	5.0	377	3	24	201	65
Scallo...	4.375	304	34	12	1809	23
Savory...	1.25	105	5	7	130	81
Broiled...	4.375	423	6	44	863	82
Poach...	4.375	716	46	53	1238	99
Mustar...	0.0	210	1	23	73	30
The Ult...	3.75	552	5	8	85	52
Butter...	4.375	438	9	25	65	25
Spicy ...	3.75	202	19	8	815	69
Lobste...	3.75	838	15	61	316	30
Sauté...	3.125	1218	46	110	2913	54
Sweet ...	2.5	134	1	1	2919	12
Sauté...	3.125	148	4	10	55	72
Banan...	3.75	669	11	46	420	27
Chicke...	3.75	445	24	38	662	75
Butters...	4.375	437	4	31	346	82
Basma...	4.375	656	24	32	202	34
Crudite...	3.75	563	6	50	314	46
Grog	0.0	173	1	0	3	28
Chicke...	3.75	246	12	21	50	39
Pear a...	4.375	681	19	40	1162	96
Bacon ...	4.375	298	4	14	309	10
Napa ...	4.375	103	3	6	204	21
Strawb...	3.75	207	2	5	33	26

Clasa **ReportsGUI** contine TextField-uri si Butoane pentru generarea de rapoarte de catre utilizator.

The screenshot shows the 'REPORTS' window. It contains several sections for generating reports:

- Start hour** and **End hour** input fields, followed by a **Generate report** button.
- Products ordered more than** input field followed by **times**, followed by a **Generate report** button.
- Min number of orders** and **Min value of order** input fields, followed by a **Generate report** button.
- Date of order:** with three dropdown menus showing '2018', '1', and '1', followed by a **Generate report** button.

**Clasa EmployeeGUI** reprezinta interfata pentru angajat. In aceasta interfata, angajatul v-a putea vedea, prin notificari, comenzile care au fost efectuate.



## 5. Rezultate

Pentru testare am creat 5 clienti, un admin si un angajat. Rezultatele rapoartelor generate de administrator:

-----  
Raport1

OrderID: 7

ClientID: 1

Products: Aunt Tom's Italian Cream Cake , Butter-Sugar Crepes , Date: 2022-05-15T14:31:13.593324100

OrderID: 6

ClientID: 1

Products: Brandy Alexander II , Sweet and Sour Pickles , Date: 2022-05-15T10:39:56.232607400

OrderID: 3

ClientID: 1

Products: Brandy Alexander II , Mustard Tarragon Butter , Date: 2022-05-15T10:28:49.188422400

OrderID: 2

ClientID: 1

Products: Brandy Alexander II , Date: 2022-05-15T10:28:39.257091

OrderID: 4

ClientID: 1

Products: Poached Sockeye Salmon with Mustard Herb Sauce , Date: 2022-05-15T10:29:11.144480700

OrderID: 0

ClientID: 2

Products: Mustard Tarragon Butter , Date: 2022-05-15T10:26:38.386801600

OrderID: 5

ClientID: 1

Products: Scallop and Shrimp Creole , Lobster with Roasted Garlic-Potato Salad and Coleslaw , Date: 2022-05-15T10:36:07.792660700

---

#### Raport2

Products: Brandy Alexander II  
Banana Coconut Crunch Cake

---

#### Raport3

Clients: Id: 1  
Name: Emma Szakacs

---

#### Raport4

Product: Grog  
Number: 1  
Product: Mustard Tarragon Butter  
Number: 1  
Product: Butter-Sugar Crepes  
Number: 1  
Product: Brandy Alexander II  
Number: 2  
Product: Banana Coconut Crunch Cake  
Number: 2

De asemenea, o factura generate arata in felul urmatoar:

#### Bill

First Name: Emma  
Last Name: Szakacs  
Email: emma@yahoo.com  
Phone number: 0742707835

Products:  
Butter-Sugar Crepes Price: 25  
Total price: 25  
Date: 2022/5/15 at 22:14:29

## ***6. Concluzii***

Prin urmare, acest proiect a fost util pentru aprofundarea cunostintelor acumulate la limbajul de programare Java. Am invatat cum sa folosesc Lamba Expressions, Stream-uri, serializarea si HashMap-uri.

Ca dezvoltare ulterioara s-ar mai putea lucra la interfata grafica. De asemenea, implementarea interfetei pentru angajat s-ar putea imbunatatii astfel incat sa ii apara notificari in continuu despre comenzile care au fost date de catre client, iar notificarile sa dispara odata ce au fost vazute.

## **7. Bibliografie**

- Pentru serializare:

[https://www.tutorialspoint.com/java/java\\_serialization.htm](https://www.tutorialspoint.com/java/java_serialization.htm)

- Pentru lamba expressions:

<https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>

<https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html>

<https://howtodoinjava.com/java8/java-stream-distINCT-examples/>

- Pentru HashMap:

<https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html>