MINISTERUL EDUCAȚIEI ȘI CERCETĂRII
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE
DEPARTAMENTUL CALCULATOARE

TEHNICI DE PROGRAMRE FUNDAMENTALE

TEMA 4

Gestiunea sistemului unei companii de catering

Dragoș-Bogdan Lazea, grupa 30223

An universitar: 2020 – 2021

1. Obiectivul temei

Obiectivul principal al acestei teme îl reprezintă proiectarea și implementarea unei aplicații care să simuleze funcționarea sistemului unei companii de catering care permite conectarea a trei tipuri de utilizatori (administrator, angajat, client), fiind dedicată plasării și onorării unor comenzi de produse alimentare și dispunând de o interfață grafică intuitivă pentru a facilita interacțiunea cu orice tip de utilizator.

Obiectivele secundare ale aceste teme sunt prezentate în tabelul de mai jos:

Obiectiv secundar	Descriere	Secțiune
1. Analiza problemei	- presupune analiza și delimitarea tuturor cerințelor pe care aplicația trebuie să le îndeplinească;	2. Analiza problemei, modelare, scenarii, cazuri de utilizare
2. Identificarea scenariilor	- presupune identificarea și analiza tuturor scenariilor în care s-ar putea regăsi aplicația în funcție de valorile de intrare introduse de către utilizator (cazuri limită, excepții, funcționare corespunzătoare);	2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Identificarea cazurilor de utilizare	- presupune identificarea situațiilor în care aplicația își justifică funcționalitatea;	2. Analiza problemei, modelare, scenarii, cazuri de utilizare
4. Proiectarea propriu-zisă	- presupune parcurgerea etapelor de sintetizare și proiectare a aplicației conform paradigmei programării orientat pe obiecte, prin realizarea de diagrame UML, proiectarea claselor, imaginarea interfeței grafice cu utilizatorul etc.	3. Proiectare
5. Implementarea aplicației	- presupune dezvoltarea propriu zisă a claselor și metodelor aferente acestora în cod scris în limbajul de programare Java;	4. Implementare
6. Testarea aplicației implementate	- presupune testarea funcționalității aplicației în toate situațiile posibile de funcționare sintetizate în cadrul cazurilor de utilizare;	5. Rezultate

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

a. Analiza problemei

Problema de soluționat este reprezentată de faptul că în realitatea actuală, s-a înregistrat o creștere semnificativă a politicilor de livrare la domiciliu, cu precădere în industria restaurantelor, fapt puternic influențat de contextul pandemic, care aduce necesitatea dezvoltarea unor sisteme de plasare a comenzilor de către clienți și receptarea acestora de către angajați în vederea onorării cererii și a livrării produselor dorite la domiciliul cumpărătorului.

Prin urmare, a fost identificat, în urma analize problemei menționate, următorul cadru de cerințe funcționale:

- Aplicația trebuie să poată permite conectarea a trei tipuri de utilizatori;
- Aplicația trebuie să informeze utilizatorul dacă inputul introdus este invalid, i. e. pentru câmpurile unde se așteaptă valori numerice se introduc și alte caractere, intervalul orar este invalid etc.;
- Aplicația trebuie să poată permite fiecărui utilizator de tip administrator efectuarea operațiilor de bază asupra produselor de care compania dispune: importarea setului inițial de produse, adăugarea de noi produse, ștergerea produselor existent, editarea produselor existente, creare de produse compuse, generarea unor rapoarte în vederea conceperii unor statistici de vânzare-cumpărare;
- Aplicația trebuie să permită fiecărui utilizator de tip client: crearea unui nou cont, conectarea cu un cont deja existent, vizualizarea tuturor produselor componente, plasarea unei comenzi și filtrarea produselor disponibile după diverse criterii de selecție;

- Aplicația trebuie să permită fiecărui utilizator de tip angajat vizualizarea tuturor comenzilor plasate în vederea onorării acestora și notificarea fiecărui angajat la plasarea unei noi comenzi;

Cadrul cerințelor nefuncționale conturate în jurul problemei de rezolvat este:

- Aplicația trebuie să dispună de o interfață grafică intuitivă.
- Aplicația trebuie să fie ușor de utilizat pentru orice tip de utilizator, indiferent de domeniul de activitate al acestuia.

b. Modelare

Aplicația a fost proiectată pe modelul unui sistem cu două intrări și două ieșiri, intrările fiind reprezentate de numele de utilizator și parola asociată contului, informații necesare la conectarea în aplicație, în timp ce ieșirile sunt reprezentate de factura generată în urma plasării și înregistrării comenzii spre a fi onorate, dar și vizualizarea informațiilor prin intermediul interfeței grafice.

Modelul problemei este astfel translatat într-un model orientat pe obiecte, ce are la bază clasele User, MenuItem, BaseProduct, CompositeProduct, Order și DeliveryService prin intermediul cărora se vor defini și transpune în modelul obiectual intrările, ieșirile și funcționalitățile asociate aplicației, definite pe baza modelului din lumea reală.

c. Cazuri de utilizare și scenarii posibile

Diagrama cazurilor de utilizare pentru utilizatorul de tip Administrator este prezentată mai jos:

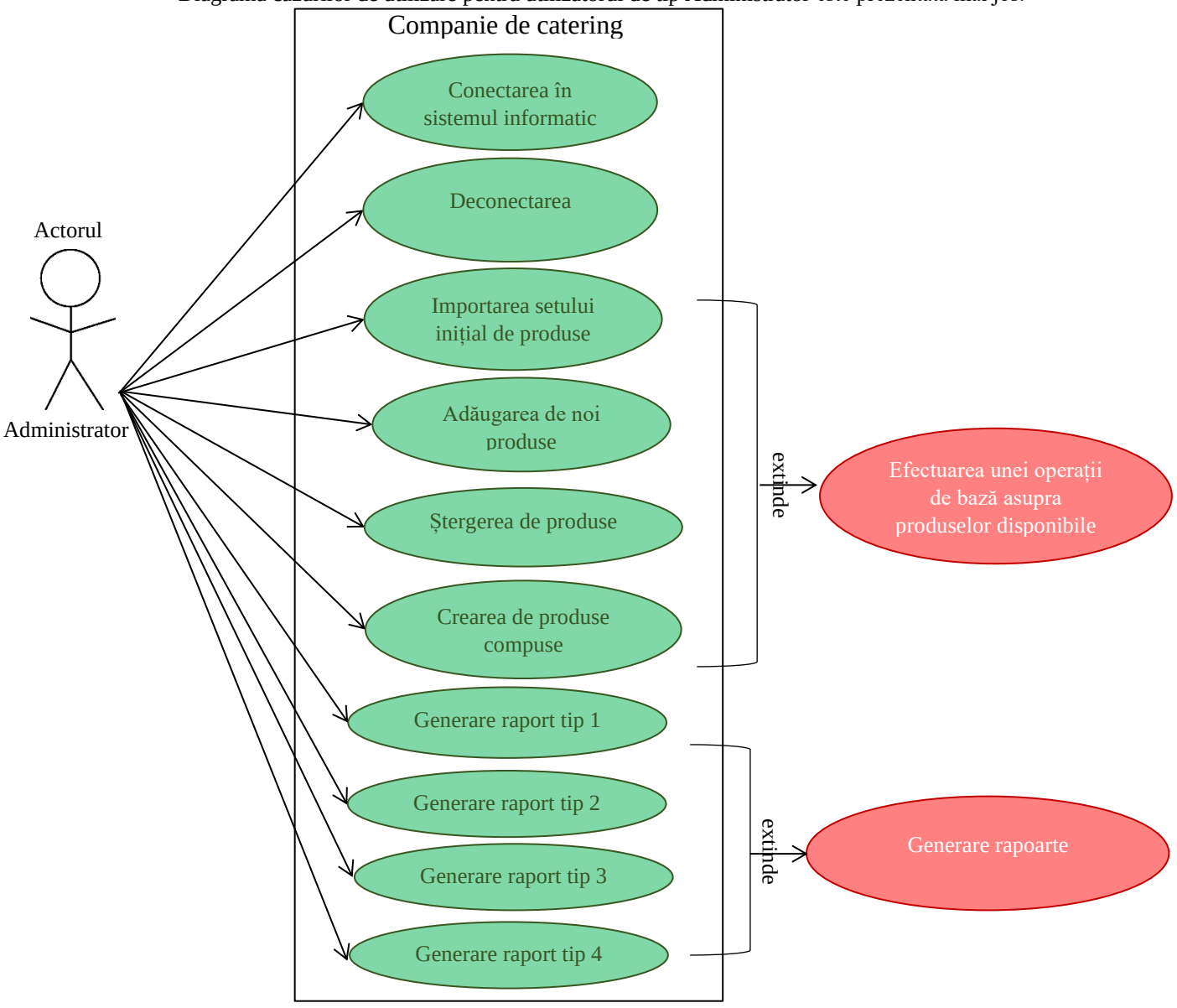


Diagrama cazurilor de utilizare pentru utilizatorul de tip Client este prezentată mai jos:

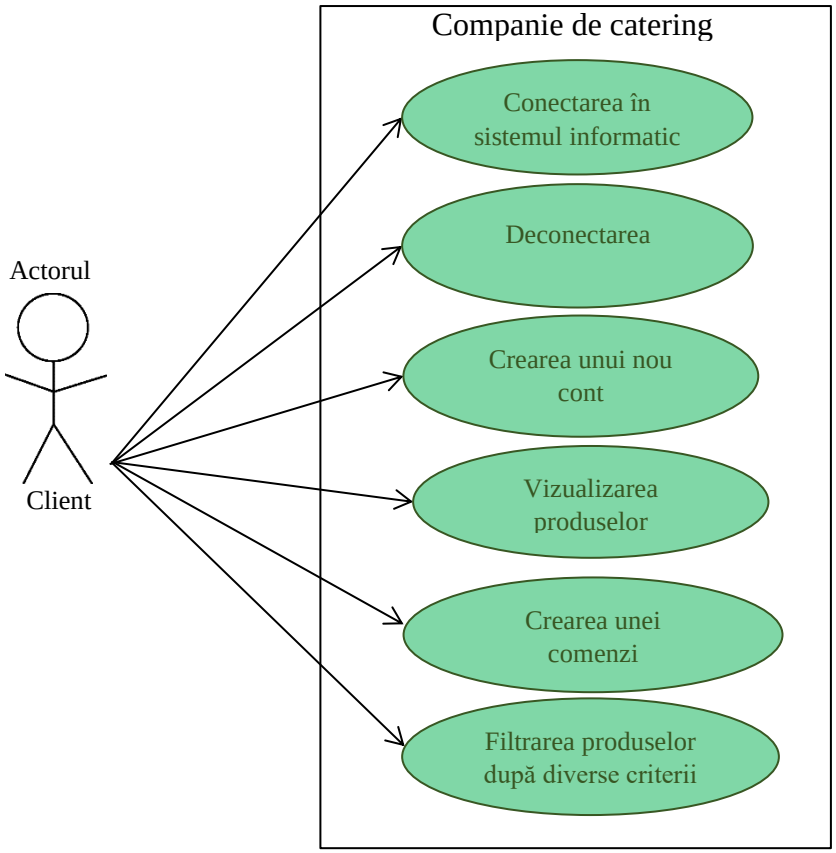
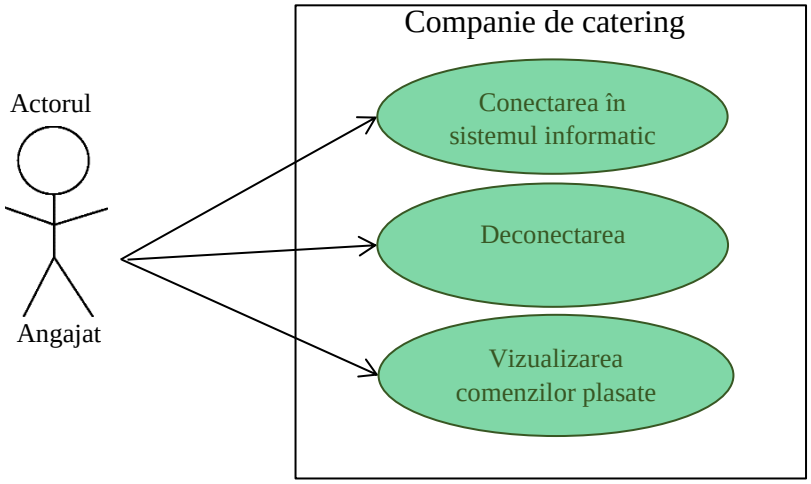


Diagrama cazurilor de utilizare pentru utilizatorul de tip Angajat este prezentată mai jos:



Cazurile de utilizare sunt similare în ceea ce privește fluxul pașilor executați de către utilizator și cel al răspunsurilor furnizate de aplicația implementată, motiv pentru care se va dezvolta o singură descriere generală a pașilor efectuați atât în scenariul de succes, cât și în scenarii limită, descriere valabilă pentru toate cazurile de utilizare surprinse în cadrul cerințelor funcționale identificate de către proiectant.

Actorul principal: Utilizatorul (Administrator/Client/Angajat)
Scenariul de succes:

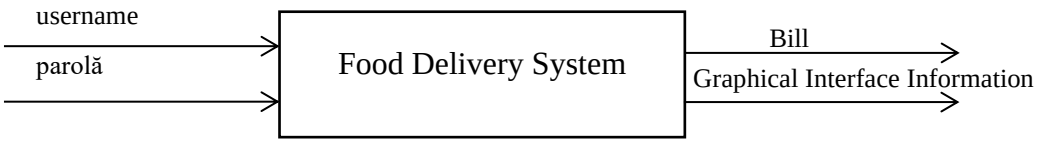
1. Utilizatorul introduce corect informațiile necesare funcționalității dorite în câmpurile de text prezente în interfața grafică sau selectează din tabelele furnizate în interfața grafică a aplicației informațiile utile și necesare funcționalității în cauză.
2. Utilizatorul apasă butonul asociat funcționalității pe care dorește să o realizeze aplicația.
3. Aplicația permite vizualizarea informațiilor asociate funcționalității în interfața grafică sau generează un fișier de ieșire corespunzător rezultatelor efectuării operației dorite de către utilizator. La apăsarea butonului are loc, de asemenea, serializarea informațiilor din cadrul aplicației și salvarea acestora în fișierul corespunzător, necesar ulterior deserializării.

Scenarii alternative (posibile erori): Informațiile introduse de utilizator sunt incorecte sau inconsistente din punctul de vedere al funcționalităților implementate.

1. Utilizatorul este notificat prin apariția unei ferestre cu un mesaj de eroare corespunzător excepției generate.
2. Scenariul se întoarce la pasul de introducere a informațiilor necesare realizării funcționalității dorite în momentul curent de către utilizatorul conectat în cadrul aplicației.

3. Proiectare

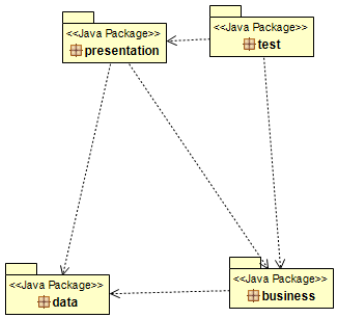
Primul nivel de proiectare îl reprezintă imaginea de ansamblu asupra aplicației prin realizarea schemei bloc a aplicației (cutia neagră a sistemului), care poate fi vizualizată mai jos.



Următoarea etapă a proiectării este reprezentată de divizarea aplicației în subaplicații cu funcționalități specializate. Subsistemele componente ale aplicației fiind organizate conform unui pattern arhitectural de tip **Layered Architecture**, compus din: **data**, **business**, **presentation** și **test,** următorul nivel de divizare al aplicației putând fi identificat în diagrama de pachete prezentată mai jos.

Astfel, subaplicația definită în **data** face posibilă accesarea informației stocate în fișierul de intrare asociat aplicației și generarea fișierelor de ieșire corespunzătoare funcționalităților de tip *scriere rezultate în fișier*, iar **business** definește operațiile ce se pot efectua produselor existente folosind modelul obiectual descris în cadrul aplicației, conținând logica operațiilor definite asupra obiectelor transpuse din soluția problemei reale. Pachetul **presentation** redă conținutul modelului, fiind responsabilă de realizarea interfeței grafice cu utilizatorul, iar **test** este responsabil de pornirea efectivă a aplicației prin instanțierea obiectelor necesare într-o metodă de tip main.

Astfel, aplicația va fi compusă din cele patru componente menționate mai sus, formate la rândul lor din subcomponente, interconectate astfel încât să formeze un ansamblu unitar, fapt ce poate fi observat în următoarea diagramă de pachete.



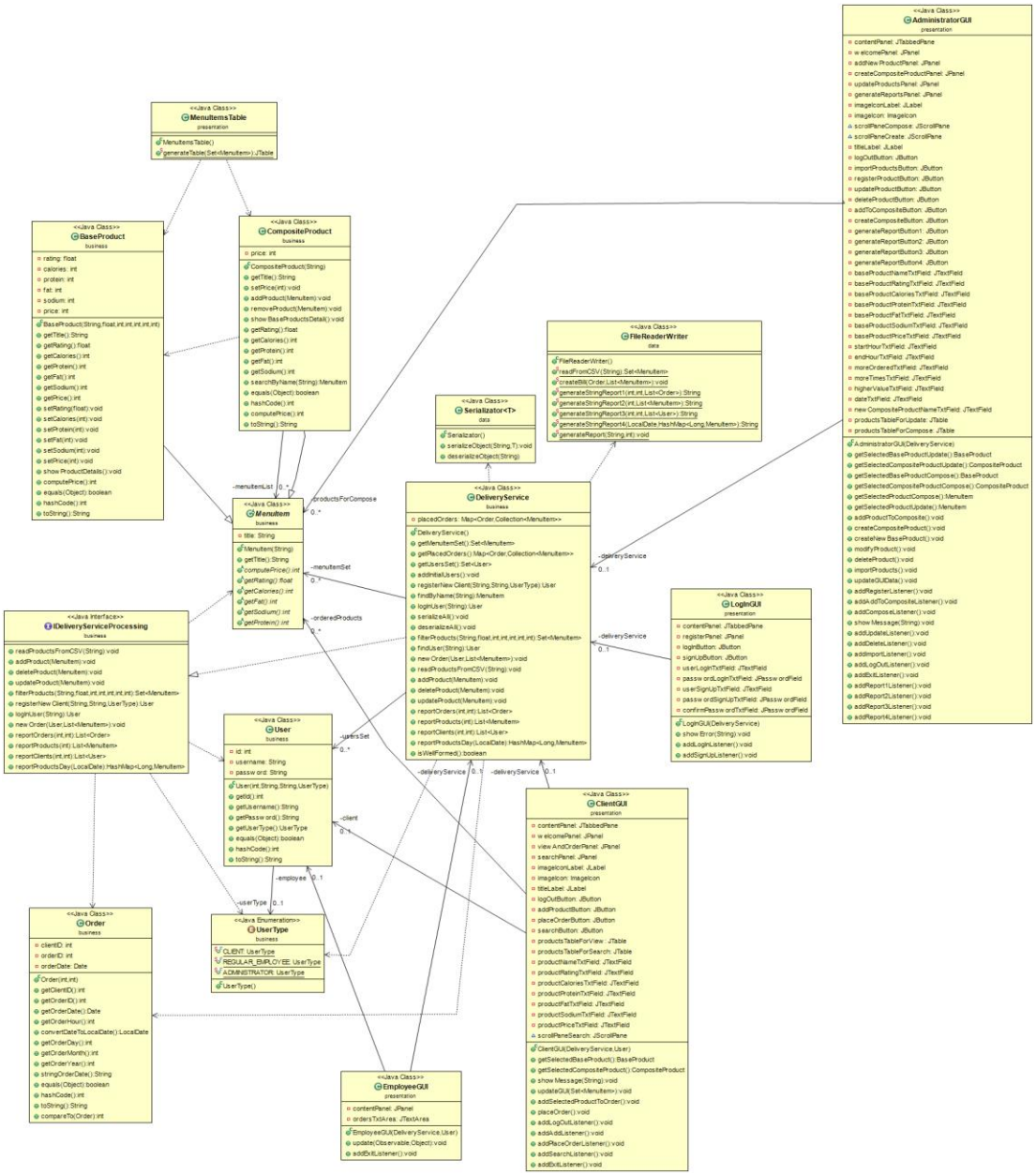
Trecând la cea de-a treia etapă de proiectare, se continuă descompunerea problemei și astfel se definesc cele mai de jos unități din descompunerea aplicației conform paradigmei programării orientate pe obiecte (POO), și anume clasele definite de dezvoltator.

Astfel, după cum a fost menționat anterior, fiecare pachet identificat în descompunerea aplicației are o specializare distinctă, aceasta fiind implementată propriu zis prin intermediul claselor componente ale pachetului.

Descrierea tuturor claselor va fi detaliată în secțiunea dedicată implementării aplicației, împreună cu metodele importante și definitorii fiecărei clase din componența proiectului. Dintre clasele și interfețele implementate se evidențiază clasa DeliveryService și interfața IDeliveryServiceProcessing, responsabile de efectuarea operațiilor propriu zise pe care se fundamentează funcționalitățile aplicației pentru cele trei tipuri de utilizatori definiți. Descrierea metodelor asociate acestora se va putea regăsi în secțiunea Implementare a prezentei descrieri.

Dintre algoritmii utilizați la proiectarea soluției, relevanți pentru analiză în contextul dezvoltării aplicației se dovedesc algoritmii pentru filtrarea informațiilor după diverse criterii, având scopuri diferite. Acești algoritmi au fost construiți folosind stream-uri bazate pe expresii lambda formate din predicate.

În cele ce urmează va fi expusă diagrama UML de clase ale aplicației, denumirile acestora fiind sugestive și inspirate din domeniul problemei. Punctul de plecare al diagramei de clase este reprezentat de schița diagramei de clase realizate în prezentarea suport atașată temei, însă aceasta a fost completată cu clasele și interfețele necesare, identificare în procesul de proiectare a sistemului de gestiune a companiei de catering.



4. Implementare

4.1. Clase, interfețe și metode definite

Denumirile claselor, ale variabilelor instanță și ale metodelor ce urmează a fi sintetizate și prezentate în cele ce urmează sunt sugestive, nefiind necesară explicitarea semnificației acestora.

În cele ce urmează se va explica utilitatea fiecărei clase implementate:

Package data

Class Summary

Class	Description
FileReaderWriter	Clasa pentru scrierea si citirea din fiseiere
Serializator<T>	Clasa generica pentru serializator

Package business

Interface Summary

Interface	Description
IDeliveryServiceProcessing	Interfata pentru descrierea functionalitatilor posibile pentru utilizatorii de tip Administrator si Client

Class Summary

Class	Description
BaseProduct	Clasa pentru produs de baza
CompositeProduct	Clasa pentru produs compus
DeliveryService	Clasa pentru implementarea functionalitatilor din interfata IDeliveryServiceProcessing
MenuItem	Clasa abstracta pentru produs din meniu
Order	Clasa pentru comanda
User	Clasa pentru utilizator

Enum Summary

Enum	Description
UserType	Enumerare pentru tipurile de utilizator

Package presentation

Class Summary

Class	Description
AdministratorGUI	Fereastra pentru Administrator
ClientGUI	Fereastra pentru Client
EmployeeGUI	Fereastra pentru angajat
LogInGUI	Fereastra de LOG IN si SIGN UP
MenuItemsTable	Clasa pentru tabelul cu produse

Package test

Class Summary

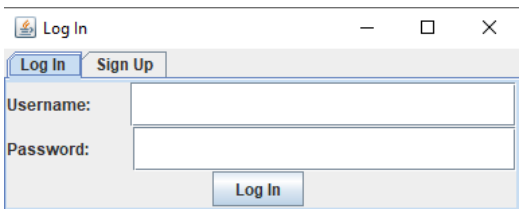
Class	Description
MainClass	Clasa pentru pornirea aplicatiei

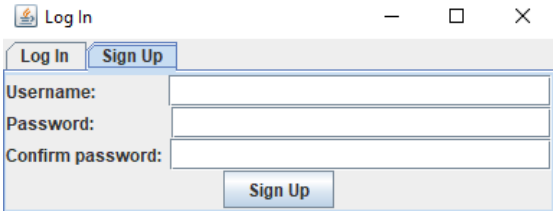
Descrierea implementării pentru toate metodele aferente fiecărei clase poate fi consultată în cadrul JavaDoc, aceasta putând fi regăsită în directorul javadoc din folderul proiectului.

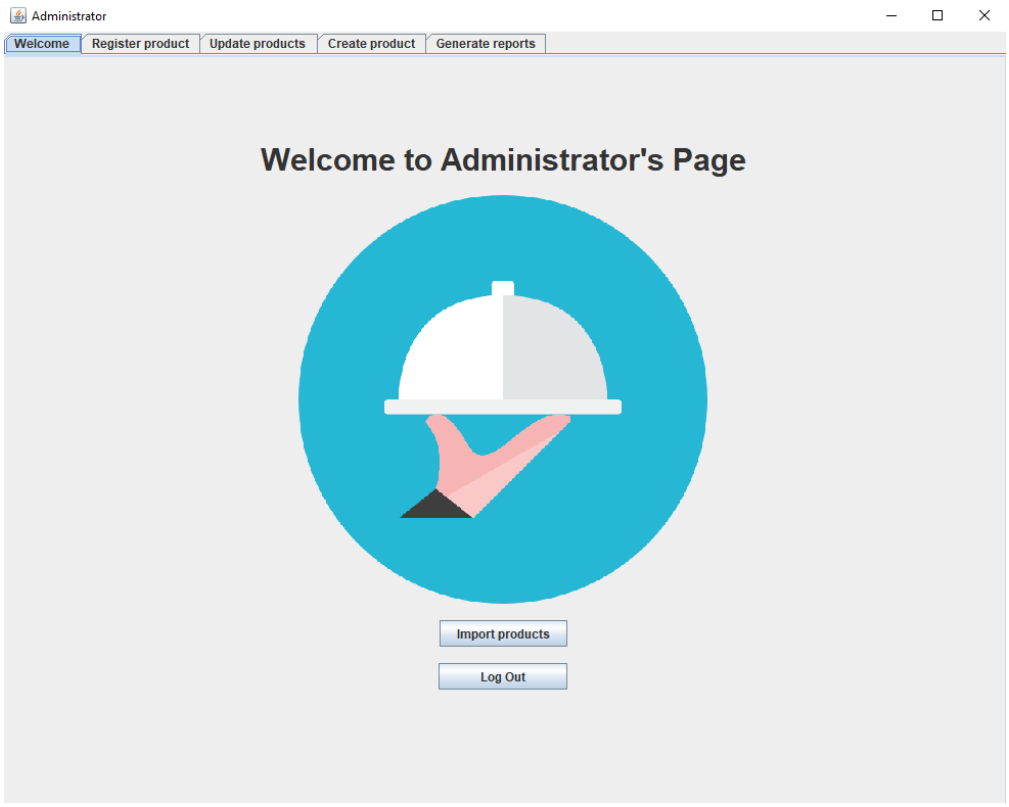
4.2. Interfața grafică cu utilizatorul (GUI)

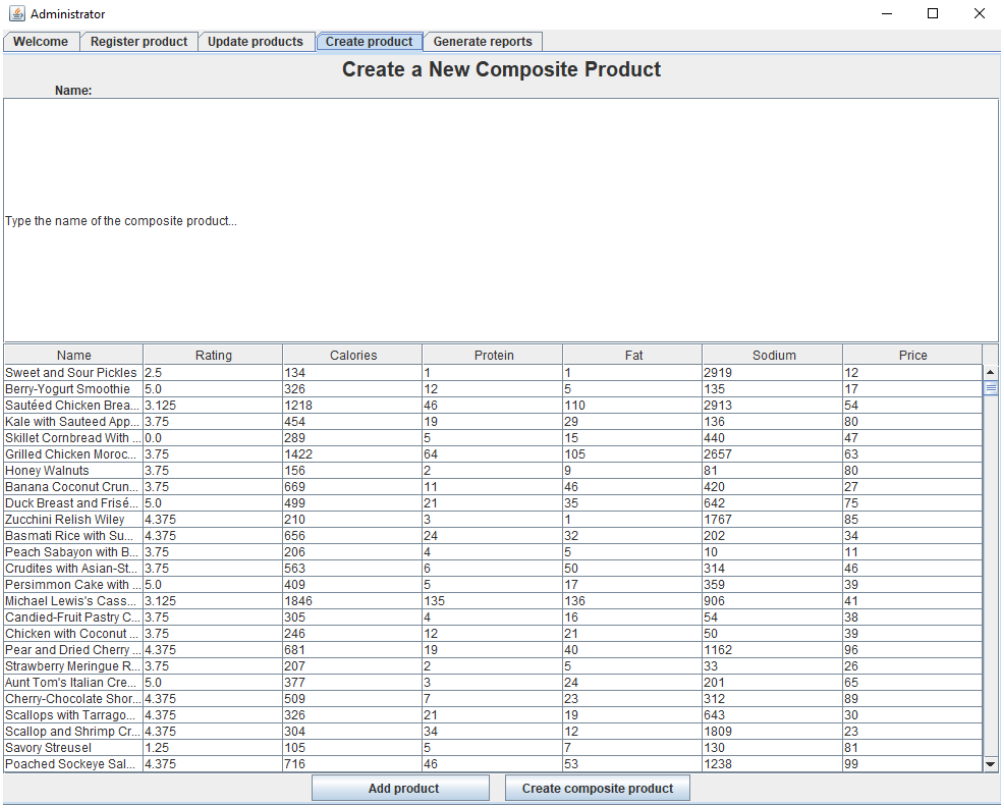
Aplicația dispune de o interfață grafică intuitivă, dezvoltată folosind Swing și fiind construită pe baza a patru JFrame: o fereastră dedicată conectării utilizatorilor în cadrul sistemului implementat, o a doua funcționalitățile utilizatorului de tip Administrator, având câte o filă dedicată fiecărui tip de funcționalitate implementată, o a treia fereastră pentru facilitarea realizării operațiilor permise utilizatorilor de tip Client, cu filele atașate fiecărui tip de funcționalitate și o a patra fereastră, dedicată utilizatorilor de tip Angajat, care permite vizualizarea comenzilor plasate de către clienți.

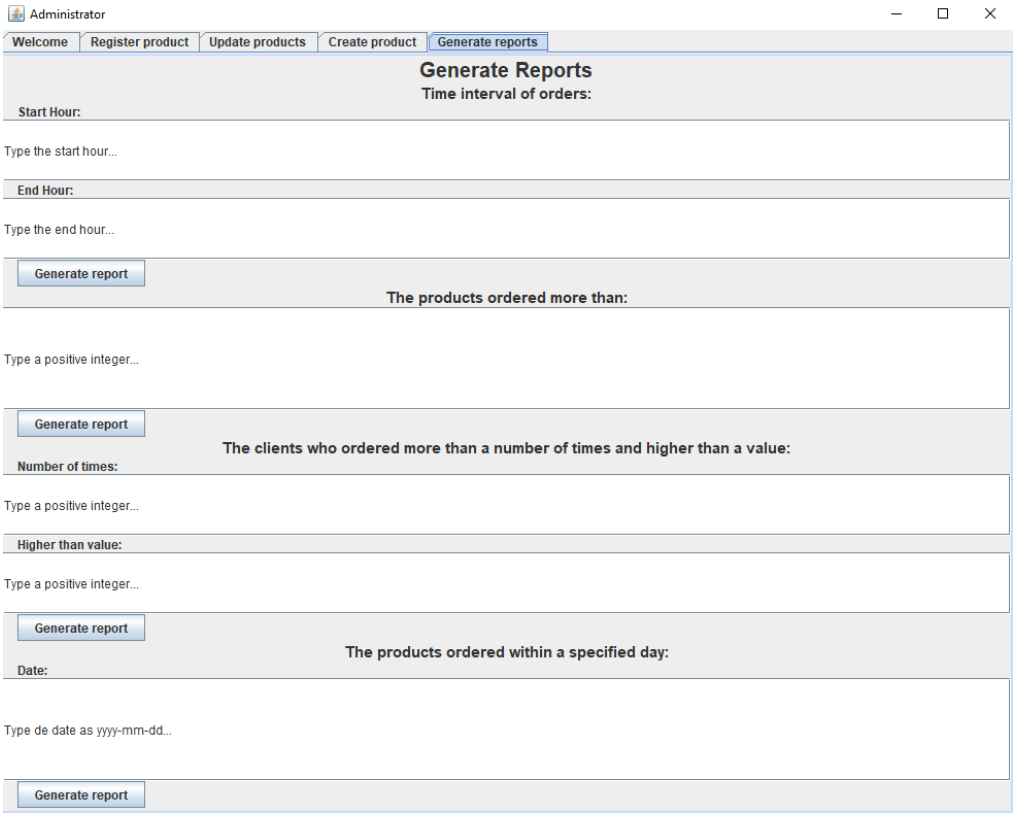
Ferestrele dezvoltate un panou principal de tip JTabbedPane în care au fost adăugate diferite subpanouri, implementate folosind JPanel. Astfel, interfața grafică are în componența sa câmpuri de text editabile pentru introducerea informațiilor în vederea înregistrării a unui nou client sau produs de către utilizator, a furnizării informațiilor de generare a rapoartelor de către Administrator și setării criteriilor de filtrare de către clientul conectat. Etichetele sunt realizate folosind JLabel, în timp ce câmpurile de text au la bază componente de tip JTextField. De asemenea, în interfața grafică a aplicației se pot regăsi numeroase butoane butoane, realizate folosind componente JButton. Tabelele ce conțin produsele disponibile au fost plasate în panouri de derulare de tip JScrollPanel pentru a permite vizualizarea acestora în întregime. Câteva dintre componentele interfeței grafice pot fi vizualizate în figurile de mai jos:

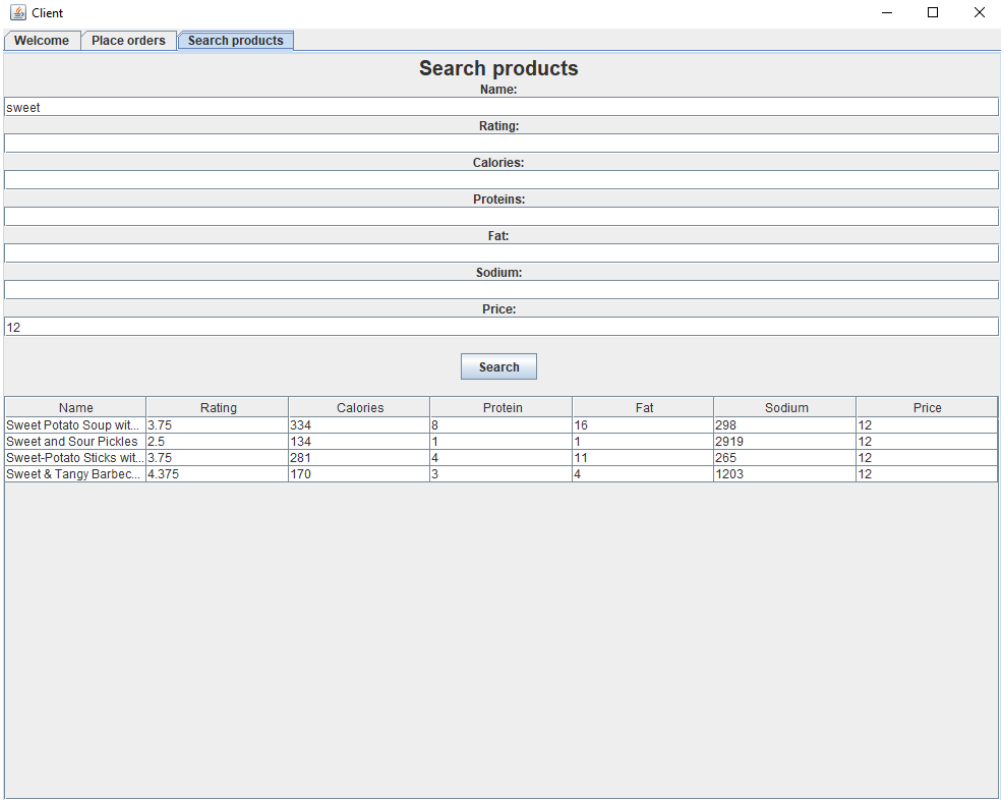


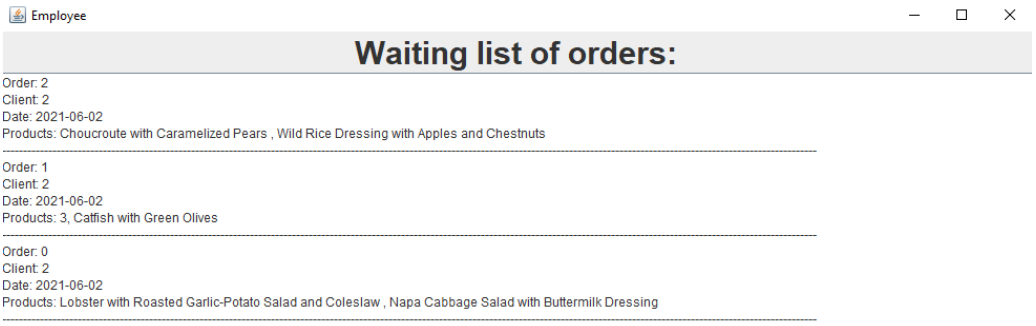




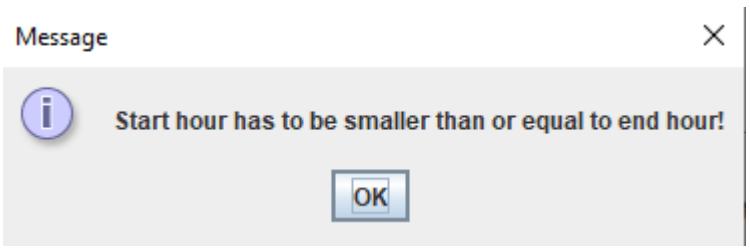

Administrator
Welcome
Register product
Update products
Create product
Generate reports
Create a New Composite Product
Name:
Type the name of the composite product...
Name
Rating
Calories
Protein
Fat
Sodium
Price
Sweet and Sour Pickles 2.5 134 1 1 2919 12
Berry-Yogurt Smoothie 5.0 326 12 5 135 17
Sautéed Chicken Brea... 3.125 1218 46 110 2913 54
Kale with Sauteed App... 3.75 454 19 29 136 80
Skillet Cornbread With ... 0.0 289 5 15 440 47
Grilled Chicken Moroc... 3.75 1422 64 105 2657 63
Honey Walnuts 3.75 156 2 9 81 80
Banana Coconut Crun... 3.75 669 11 46 420 27
Duck Breast and Frisé... 5.0 499 21 35 642 75
Zucchini Relish Wiley 4.375 210 3 1 1767 85
Basmati Rice with Su... 4.375 656 24 32 202 34
Peach Sabayon with B... 3.75 206 4 5 10 11
Crudites with Asian-St... 3.75 563 6 50 314 46
Persimmon Cake with ... 5.0 409 5 17 359 39
Michael Lewis's Cass... 3.125 1846 135 136 906 41
Candied-Fruit Pastry C... 3.75 305 4 16 54 38
Chicken with Coconut ... 3.75 246 12 21 50 39
Pear and Dried Cherry ... 4.375 681 19 40 1162 96
Strawberry Meringue R... 3.75 207 2 5 33 26
Aunt Tom's Italian Cre... 5.0 377 3 24 201 65
Cherry-Chocolate Shor... 4.375 509 7 23 312 89
Scallops with Tarrago... 4.375 326 21 19 643 30
Scallop and Shrimp Cr... 4.375 304 34 12 1809 23
Savory Streusel 1.25 105 5 7 130 81
Poached Sockeye Sal... 4.375 716 46 53 1238 99
Add product
Create composite product


Administrator
Welcome
Register product
Update products
Create product
Generate reports
Generate Reports
Time interval of orders:
Start Hour:
Type the start hour...
End Hour:
Type the end hour...
Generate report
The products ordered more than:
Type a positive integer...
Generate report
The clients who ordered more than a number of times and higher than a value:
Number of times:
Type a positive integer...
Higher than value:
Type a positive integer...
Generate report
The products ordered within a specified day:
Date:
Type de date as yyyy-mm-dd...
Generate report





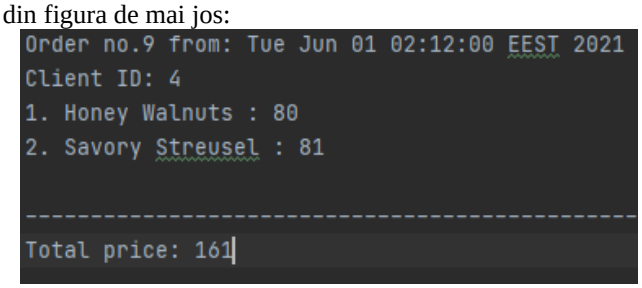
Aplicația semnalizează o eroare prin apariția unei ferestre cu un mesaj specific în cazul în care utilizatorul introduce date invalide. Ferestrele ce afișează mesajele de eroare au următorul design:



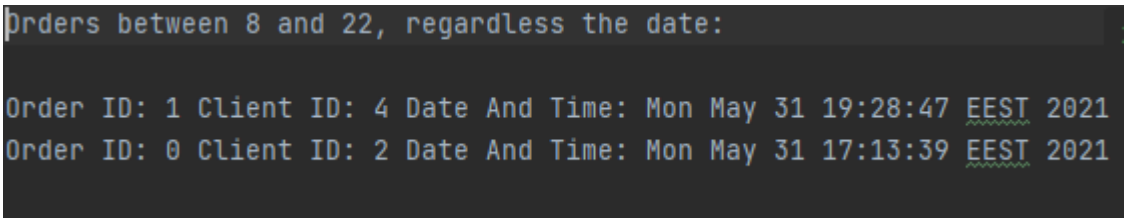
Interfața grafică facilitează folosirea simulatorului de către orice tip de utilizator și oferă o interacțiune mai bună a utilizatorului modelul obiectual al soluției.

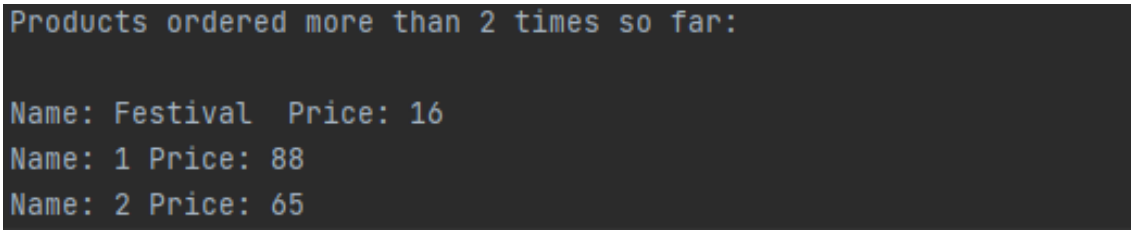
5. Rezultate

Testarea aplicației a fost realizată astfel încât să acopere toate cazurile de utilizare, atât în scenariul de succes, cât și în cel de eșec. Aplicația a reacționat corect, având comportamentul așteptat în toate cazurile de testare furnizate, a respectat cadrul de cerințe funcționale impus prin specificația problemei, în fișierul .txt generat putând fi regăsite pentru fiecare comandă informațiile referitoare la clientul care a plasat comanda, informațiile referitoare la produsul comandat și totalul de plată, după cum se poate observa din figura de mai jos:

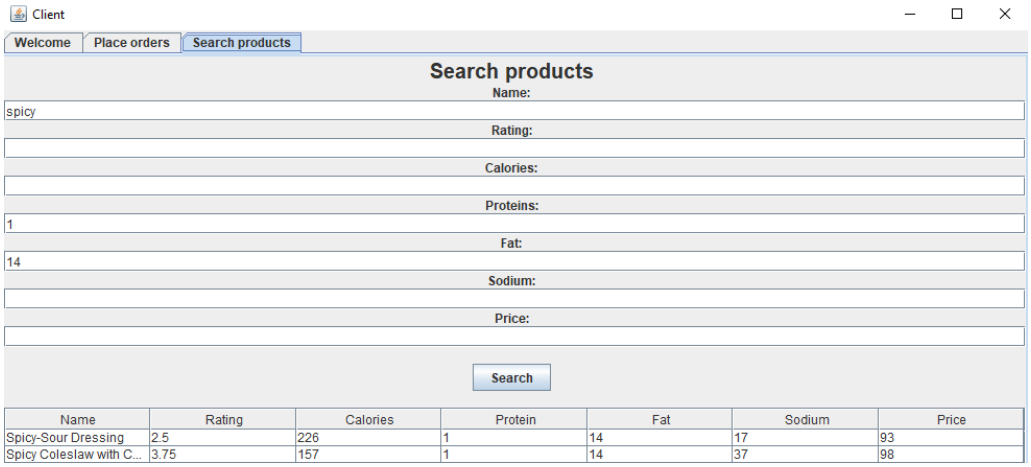


De asemenea, fișierele conținând rapoartele generate de către administrator au reflectat corectitudinea comportamentului aplicației în salvarea datelor prin serializare și încărcarea acestora prin intermediul deserializării, rapoartele furnizând informații corecte și relevante din punct de vedere statistic cu privire la clienți, comenzi, produse, fapt ce poate fi observat în figurile de mai jos:





Nu în ultimul rând, și operațiile care au ca rezultat afișarea unor informații în interfața grafică asociată aplicației au dovedit corectitudine în comportament pe toate cazurile de testare furnizate, după cum se poate vizualiza din figura următoare:



Testarea proiectului s-a realizat pe un set larg de teste, aplicația având un răspuns pozitiv la multitudinea de cazuri de test furnizate, printre acestea numărându-se, de asemenea, situații de tip excepție care au fost interceptate și tratate corect. De asemenea, la fiecare pas de testare s-a verificat concordanța rezultatelor scrise în fișierul ce conține factura atașată comenzii cu cele reținute prin intermediul serializării.

6. Concluzii

Concluzionând, se poate afirma că această temă vine să soluționeze problema necesității dezvoltării unui sistem informatic de gestionare a firmelor ce au drept obiect de lucru livrarea de produse alimentare, fiind implementată conform paradigmei programării orientate pe obiect și proiectându-se ca o mini-aplicație ce poate fi folosită de orice tip de utilizator, dispunând de un pattern arhitectural care permite separarea modelului obiectual, de interfața grafică cu care utilizatorul interacționează, fiecare nivel al logicii interne fiind bine delimitat. Pentru dezvoltatorul proiectului, implementarea acestuia s-a dovedit utilă, având drept rezultat o mai bună însușire a manipulării și dezvoltării aplicațiilor Object Oriented, prin exersarea implementării practice a conceptelor specifice acestei paradigme de programare, dar și acumularea de noi cunoștințe referitoare la salvarea datelor prin intermediul serializării, a utilității stream-urilor și a expresiilor lambda în filtrarea și selectarea obiectelor după diverse criterii și a elementelor de grafică și vizualizare în dezvoltarea unei aplicații susținute de o interfață grafică cu utilizatorul.

Printre posibilele dezvoltări ulterioare ale aplicației se numără afișarea unei ferestre ce conține detaliile și produsele componente ale unui produs compus în momentul selectării acestuia în vederea plasării unei comenzi, dar și posibilitatea de a renunța la comanda plasată într-un anumit interval de timp de la înregistrarea acesteia.

7. Bibliografie

- ASSIGNMENT_4_SUPPORT_PRESENTATION.pdf (furnizat pe Microsoft Teams)
- Tema4.pdf (furnizat pe Microsoft Teams)
- Cursuri și lucrări de laborator de la disciplina Programare Orientată pe Obiecte, studiată în semestrul I al anului universitar 2020-2021
- https://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html
- https://www.tutorialspoint.com/java/java_serialization.htm
- https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html
- https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html
- https://javarevisited.blogspot.com/2011/02/how-hashmap-works-in-java.html#axzz6wacloiOB