**DOCUMENTATIE TEMA 4**

**FOOD DELIVERY MANAGMENT SYSTEM**

**Nume prenume: Dragomir Mihai-Robert**

**Grupa: 302210**

**Profesor Laborator Assist Mitrea Dan**

Contents

[1. Cerinte Functionale 3](#_Toc476131445)

[2. Obiective 3](#_Toc476131446)

[2.1. Obiectiv Principal: 3](#_Toc476131447)

[2.2. Obective Secundare: 4](#_Toc476131448)

[3. Analiza Problemei 4](#_Toc476131449)

[4. Proiectare 5](#_Toc476131450)

[4.1. Structuri de date 5](#_Toc476131451)

[4.2. Diagrama de clase 6](#_Toc476131452)

[4.3. Algoritmi 6](#_Toc476131453)

[5. Implementare 7](#_Toc476131454)

[6. Concluzii si Dezvoltari Ulterioare 8](#_Toc476131456)

[7. Bibliografie 8](#_Toc476131457)

# Cerinte Functionale

Dezvoltati un sistem de manageriere pentru livrarea de produse alimentare in contextul unei companii de catering care sa permita urmatoarele functionalitati:

* Permite utilizatorului cu rol de administrator sa importe setul de produse ce urmeaza sa populeze meniul dintr-un fisier de tip csv.
* Permite utilizatorului cu rol de administrator sa managerieze lista de produse din meniu. Acesta are posibilitatea de adaugare, stergere sau editare a unui produs deja existent sau de creare a unui produs compus din o lista de produse de baza selectate.
* Permite utilizatorului cu rol de administrator sa genereze rapoarte pe baza intervalului de timp in care au fost realizate comenzile, a produselor comandate mai mult de un anumit numar de ori, a clientilor care au comandat de maim ult de un anumit numar de ori si a caror valoare a comenzii a deposit o anumita suma, a produselor comandate intr-o anumita data, determinand numarul de dati in care au fost comandate.
* Permite utilizatorului cu rol de client sa se logheze in ecranul principal al aplicatiei prin intermediul unui nume de utilizator si al unei parole.
* Permite utilizatorului cu rol de client sa isi creeze un cont pe baza unui nume de utilizator, al unui nume si al unei parole, daca acesta nu este deja inregistrat in cadrul aplicatiei.
* Permite utilizatorului cu rol de client sa caute produse pe baza mai multor criterii: prezenta unui cuvant cheie in titlul produsului, raiting-ul minim atribuit unui produs, numarul maxim de calorii pe care sa le contina produsul, numarul maxim de grasimi, numarul maxim de grame de sodiu, numarul minim de protein, un pret maxim de vanzare atribuit produsului.
* Permite utilizatorului cu rol de client sa depuna comenzi care sa consiste in diferite produse, pentru fiecare comanda fiind retinute data si ora plasarii comenzii. De asemenea permite generarea unei facturi care sa contina lista de produse comandate si pretul total al comenzii.
* Notifica utilizatorul cu rol de angajat de fiecare data cand o comanda a fost realizata, in vederea pregatirii ei pentru a urma sa fie livrata.

# Obiective

## Obiectiv Principal:

Propuneti, proiectati si implementati un sistem de manageriere pentru livrarea de produse alimentare in contextul unei companii de catering. Clientul poate depune comenzi pe baza produselor disponibile in oferta companiei. Sistemul ar trebui sa aiba 3 tipuri de utilizatori, care sa se logheze pe baza unui nume de utilizator si a unei parole, anume administrator, angajat obisnuit si client.

## Obective Secundare:

|  |  |  |
| --- | --- | --- |
| **Obiectiv Secundar** | **Descriere** | **Capitol** |
| Dezvoltarea de use case-uri si scenarii | Prezinta o reprezentare grafica a modului in care utilizatorul interactioneaza cu interfata grafica a sistemului de livrare a comenzilor, urmarind anumite spete si scenarii de functionare si utilizare. | 3 |
| Alegerea structurilor de date | Ofera explicatii cu privire la optiunea pentru anumite structuri de date in implementarea solutiei. | 4 |
| Impartirea pe clase | Urmareste din punct de vedere schematic modul de interactiune si asamblare a claselor in vederea functionarii corecte a aplicatei de realizare a livrarilor. | 4 |
| Dezvoltarea algoritmilor | Aduce adnotari imperios necesare asupra modului in care sunt traduse in practica, sub forma algoritmica, anumite metode auxiliare sau fundamentale in implementarea aplicatiei. | 4 |
| Implementarea solutiei | Realizeaza o introspectie asupra functionalitatii si a necesitatii fiecarei clase constituente. | 5 |

# Analiza Problemei

Diagrama de use case case:

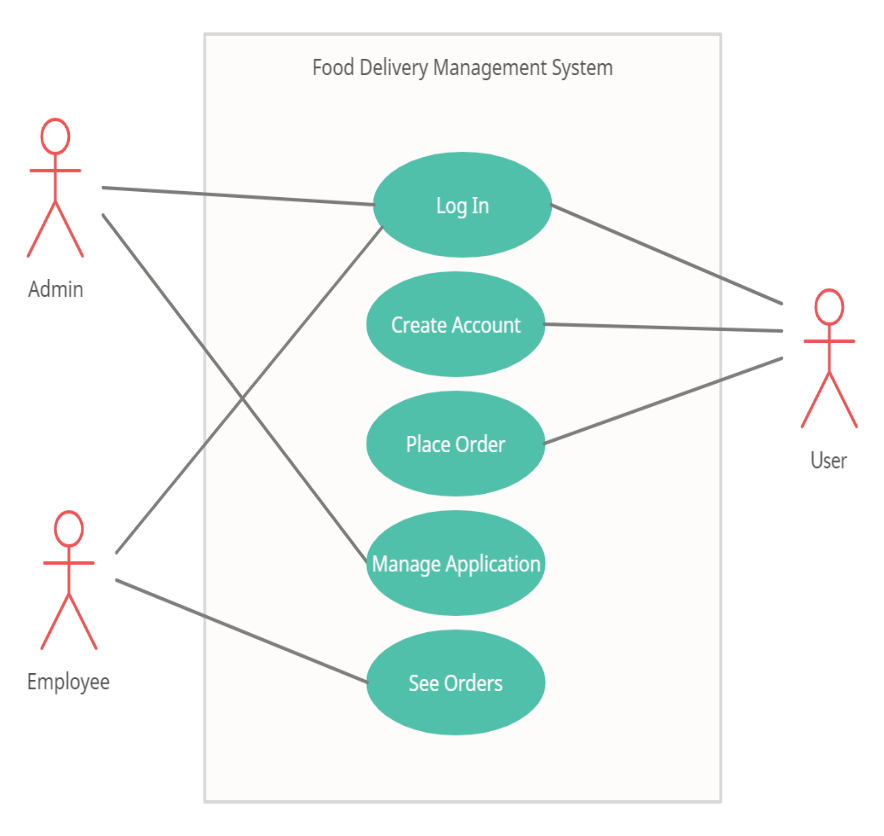


Diagrama use case aferenta logarii in sistem:

Use Case: logarea in sistem

Actor principal: user, indiferent de rolul sau de asministrator, client sau angajat

Principalul scenariu

1. Utilizatorul insereaza username-ul si parola

2. Utilizatorul apasa butonul de „Sign In”

3. Sistemul deschide o fereastra noua determinand in functie de username si parola rolul userului tocmai

logat incadrul aplicatiei, astfel incat noua fereastra sa permita doar functionalitatii aferente pozitiei pe care acesta o detine.

Scenariu alternativ

Nume de utilizator sau parola incorecta

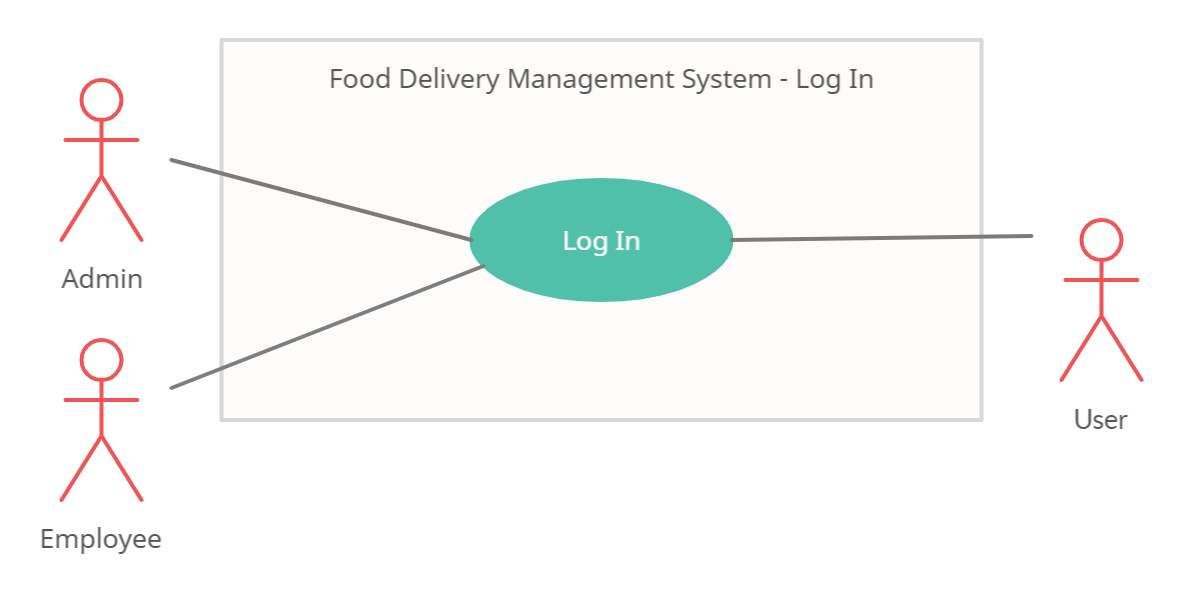
-

Utilizatorul introduce un nume de utilizator care nu concorda cu parola sau care nu exista in sistem.

-

Se realizeaza redirectionarea catre primul pas, utilzatorul avand posibilitatea apasarii unui buton de sign up care sa

Ii permita inregistrarea in sistem.



# Proiectare

## Structuri de date

Structura de date primordiala in implementarea aplicatiei o constituie HashMap, utilizat in vederea modului de stocare a perechilor de tipul Comanda – lista de produse aferenta comenzii. Avantajul principal al utilizarii acestei structurii care implementeaza o tabela de dispersie rezida in complexitatea medie de adaugare sau extragere a unor elemente din tabela, anume O(1).

O alta structura de date importanta in vederea implementarii solutiei finale o constituie TreeSet-ul. Optiunea pentru TreeSet, in detrimental unui ArrayList pentru produsele importate din fisierul csv, isi are sorgintea in redundanta stocarii datelor pe care ar fi implicat-o o structura de tip ArrayList, dat fiind faptul ca produsele din cadrul fisierului csv pot avea aparitii multiple. De asemenea, optiunea pentru TreeSet in detrimental unui ArrayList, este justificata prin caracteristicile intrinseci ale primei structuri mentionate. Astfel, desi deficitar in implementarea anumitor metode, unde ArrayList-ul ar putea fi mai benefic, ordonarea implicita a elementelor in momentul adaugarii intr-un TreeSet, l-a facut pe acesta din urma mai dezirabil.

## Diagrama de clase

Diagrama UML din randurile ce urmeaza descrie modul in care clasele utilizate in implementarea aplicateie de delivery interactioneaza intre ele.

O imagine care conține text, electronice

Descriere generată automat

## Algoritmi

In cadrul proiectului s-a urmarit implemenatarea a diferite tehnici de programare care exploateaza facilitati oferite de mediul de prgaramre java pe baza unor pattern-uri cunoscute. Pentru implementarea Observer Design Pattern s-a urmarit constituirea unei clase de tip Observer, care are drept scop auxiliar notificarea celorlalte clase de interes care implementeaza interfata Observable despre modificarile care survin la nivelul unor date relevante, in cazul de fata notificarea utilizatorului cu rol de employee in legatura cu depunerea comenzilor. In vederea atingerii conceptului de Composite Design Pattern au fost definite clasele CompositeProduct si BaseProduct care extind functionalitatea clasei MenuItem. In vederea implementarii patternului de Design By Contract au fost definite interfata IDeliveryServiceProccesing si clasa care implementeaza metodele definite in cadrul interfetei, anume DeliveryService. Pentru stocarea datelor de la o rulare la alta a aplicatiei au fost implementate conceptele de serializare si deserializare a datelor, pe baza metodelor din cadrul clasei Serializator. De asemenea, in vederea unei bune gestionari si manipularii a unui set mare de date, clasa DeliveryService uzeaza de stream-uri si de expresii lambda inerente acestora. De asemenea, in clasa LogInController a fost realizata metoda verifyLogInData care determina pe baza username-ului oferit de catre user in fereastra de logare a tpului de utilizator care urmareste lansarea aplicatiei, astfel incat, in masura in care parola concorda cu numele de utilizator, urmatoarea fereastra deschisa sa permita doar functionalitatiile particulare fiecarui tip de utilizator in parte, pastrand astfel integritatea sistemului.

# Implementare

S-a urmarit o dispersie si finalmente un mod de constructive si reprezentare stratificat al aplicatiei. La nivel practice, acest fapt este evideniat prin impartirea claselor in pachete, fiecare pache fiind afferent unui strat, astfel pachetul data corespunde Data Layer-ului, pachetul business, layerului de Business Logic, iar pachetul presentation layer-ului de presentation, cu 2 subpachete de view si controller care reduc partea de prezentare la nivel logic la impartirea claselor in clase de tip controller, care defines functionalitatea si actioneaza drept canal de comunicare intre business logic layer si view si clase “de vizualizare” aflate in pachetul view, care urmaresc sa defineasca modul in care se contureaza din punct de vedere al aspectului, interfata cu utilizatorul.

***Clasa MenuItem*** este o clasa abstracta, avand o metoda abstracta de computare a pretului, pe care clasele care o mostenesc urmeaza sa o suprascrie. Aceasta defineste la nivel de atribute principalele campuri care definesc un produs din cadrul meniului pus la dispozitie de firma de livrare.

***Clasa BaseProduct*** extinde clasa MenuItem suprascriind metoda acesteia abstracta de determinare a pretului in contextual unui produs de acest tip.

***Clasa CompositeProduct*** extinde la randul ei clasa MenuItem suprascriind metoda acesteia abstracta de determinare a pretului in contextual unui produs de acest tip. De asemenea, pornind de la definitia unui produs compus, ca fiind un produs determinat de mai multe produse obisnuite, clasa CompositeProduct defineste sub forma de atribut o lista de MenuItem-uri si determina in functie de atributele fiecarui produs din lista, atributele principale ale ale produsului compus, luand in considerare pretul, numarul de claorii, de grasimi, rating-ului fiecarui produs constituent etc.

***Clasa Role*** este de facto un Enum care defineste tipurile de utilizatori posibili in cadrul aplicatiei. Caracterul imperios oferit de aceasta clasa isi are originea in posibilitatea de distinctie intre utilizatori pe baza rolului lor, impunand deci o relatie statutara intre useri.

***Clasa User*** modeleaza conceptul unui utilizator la nivel de baza, definind atributele sale primordiale, anume parola si numele de utilizatori.

***Clasa Client*** extinde conceptul unui utilizator primar prin adaugarea unui nume si a unui identificator unic de ID setat prin intermediul unei variabile statice de tip contor.

***Clasa Order*** introduce la nivel de cod conceptul unei comenzi prin definirea atributelor sale principale. Elemental unui de identificare, anume orderID-ul este de asemenea setat cu ajutorul unei variabile statice de tip contor. Fiind un element in cadrul tabelei de dispersie, la nivelul acesteia este suprascrisa si functia de hashCode, calculate in baza ID-ului comenzii, a ID-ului clientului care a depus comanda si a datei in care a fost depusa comanda.

***Clasa DeliveryService*** implementeaza interfata IDeliveryServiceProccesing la nivelul careia sunt definite principalele operatii realizate de administrator si de utilizatori. Astfel metoda importProducts() determina importarea produselor de catre administrator din fisierul csv. Metoda addProductToMenu determina adaugarea unui obiect de tip MenuItem in lista de produse a companiei. Metoda DeleteProductFromMenu determina stergerea unui produs din lista de produse, iar metoda editProductFromMenu editarea datelor unui produs. Metoda generateTimeReport genereaza un raport in format txt, filtrand datele prin intermediul stream-urilor si pe baza expresiilor labda, a comenzilor plasate in intervalul de tip dat de startHour si endHour. Metoda generateFrequencyReport genereaza un raport in format txt, filtrand datele prin intermediul stream-urilor si pe baza expresiilor labda a produselor comandate de un numar mai mare de ori decat cel indicat prin numberTimesOrdered. Metoda generateFrequencyAndValueReport genereaza un raport in format txt, filtrand datele prin intermediul stream-urilor si pe baza expresiilor labda a clientilor care au comandat de un numar mai mare de ori decat cel indicat prin numberTimesOrdered.si a caror comanda a prsupus plata unei sume de bani mai mare decat cea indicate prin parametrul amount. Metoda generateDateReport genereaza un raport in format txt, filtrand datele prin intermediul stream-urilor si pe baza expresiilor labda a comenzilor realizate in data determinate de parametrul localDate. Metodele private filterClientsByFrequency si FilterProductsByFrequency sunt metode auxiliare, care realizeaza filtrarea dupa frecventa, utilizate in generarea raporturilor 2 si 3. Metoda createOrder determina crearea unui obiect de tip Order pe baza parametrilor primiti la apel, in timp ce metoda SearchForProduct determina cautarea pe baza anumitor criterii date prin intermediul parametrilor sai a produselor care le indeplinesc.

***Clasa FileWriter*** este utilizata in vederea scrierii in fisier, utila in generarea rapoartelor de catre administrator, dar si in generarea facturii in urma plasarii unei comenzi. Aceasta defineste metoda de scrierere in fiseier – metoda write, respective metoda care inchide fisierul – metoda close. Atributul Boolean isClosed ofera informatii despre starea fisierului in vederea serializarii si a deserializarii.

***Clasa Serializator*** este utilizata in vederea definirii metodelor prin care sunt serializate si deserializate datele in vederea pastrarii integritatii acestora din momentul inchiderii sesiunii curente si pana in momentul pornirii unei noi sesiuni.

***Clasa Main*** debuteaza prin deserializarea datelor aferente sesiunii precedente si implicit prin incarcarea lor. Aceasta defineste o instanta de tip DeliverySrvice si o fereastra de tip LogInView cu controller-ul aferent care fac posibila functionarea aplicatiei. Aprioric incheierii sesiunii datele noi, procesate pana in acest moment sunt serializate.

# Concluzii si Dezvoltari Ulterioare

Finalmente, sistemul de administrare al comenzilor pentru o firma de catering isi demonstreaza utilitatea printr-o paralela iminenta cu alte aplicatii de food delivery high-level indispensabile in ziua de azi. Chiar daca functionalitatea sa este limitata, modul in care acesta simuleaza anumite functionalitati prezente si in aplicatii de o avengura mai mare il face o unealta foarte viabila, cu potential ulterior de progres.

Ca dezvoltari ulterioare ce pot fi aduse aplicatiei s-ar putea inscrie posibilitatea de delogare in fereastra principala, dezvoltarea conceptului de Observer-Observable prin notificarea clientului care a depus o anumita comanda despre statusul ei si de timpi aproximativi de livrare. Desigur, aceasta din urma presupune dezvoltarea conceptului de Order prin adaugarea de noi attribute si metode. Nu in ultimul rand, ar putea fi extinse capacitatile administratorului prin conferirea acestuia de noi actiuni pe care sa le realizeze.

# Bibliografie

1. <https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>
2. <https://www.w3schools.com/java/java_files_create.asp>
3. <https://stackoverflow.com/questions/9223905/python-timestamp-from-day-month-year>
4. <https://www.codejava.net/java-se/swing/jtable-popup-menu-example>
5. <https://www.codejava.net/java-se/swing/how-to-handle-mouse-clicking-event-on-jtable-column-header>
6. <https://www.geeksforgeeks.org/how-to-serialize-arraylist-in-java/>