

# DOCUMENTATIE

TEMA *NUMARUL\_TEMEI*

NUME STUDENT: Calin Catalin Iacob  
GRUPA: 30222

## CUPRINS

1.	Obiectivul temei.....	3
2.	Analiza problemei, modelare, scenarii, cazuri de utilizare .....	3
3.	Proiectare .....	4
4.	Implementare .....	12
5.	Rezultate .....	20
6.	Concluzii.....	21
7.	Bibliografie .....	21

## 1. Obiectivul temei

*Principalul obiectiv al temei este de a construi si a implementa o aplicatie pentru managementul comenzilor clientilor pentru un depozit.*

*Obiective secundare:*

- *Analiza problemei si indentificarea cerintelor – Capitolul 2 - Analiza problemei, modelare, scenarii, cazuri de utilizare.*
- *Proiectarea aplicatiei de management a comenzilor – Capitolul 3 - Proiectarea*
- *Implementarea aplicatiei de management a comenzilor – Capitolul 4 – Implementare.*
- *Testarea aplicatiei de management a aplicatiei – Capitolul 5 - Rezultate*

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

*Aceasta aplicatie presupune constuirea a trei interfete grafice pentru a mentine numarul de produse din depozit, clienti, cat si comenzile clientilor. Fiecare interfata trebuie sa aiba posibilitatea de a aduga, sterge sau edita un produs, client, comanda.*

*Cerinte functionale:*

- *Aplicatia trebuie sa permita utilizatorului sa adauge un client nou;*
- *Aplicatia trebuie sa permita utilizatorului sa stearga un client;*
- *Aplicatia trebuie sa permita utilizatorului sa editeze datele unui client;*
- *Aplicatia trebuie sa permita utilizatorului sa adauge un produs nou;*
- *Aplicatia trebuie sa permita utilizatorului sa stearga un produs;*
- *Aplicatia trebuie sa permita utilizatorului sa editeze datele unui produs;*
- *Aplicatia trebuie sa permita utilizatorului sa adauge o comanda noua;*
- *Aplicatia trebuie sa permita utilizatorului sa stearga o comanda;*
- *Aplicatia trebuie sa permita utilizatorului sa editeze datele unei comenzi;*

*Use-case: adugarea unui client:*

*Actor principal: utilizatorul*

*Scenariu principal:*

- *Utilizatorul porneste aplicatia;*
- *Apasa butonul “Adauga client”, se deschide interfata pentru manipularea datelor pentru clienti;*
- *Pentru a aduga un client nou trebuie introduse datele urmatoare: Nume , Adresa;*
- *Utilizatorul apasa butonul adauga, iar clientul cu datele introduse de utilizator va aparea in tabelul cu clienti;*

*Use-case: adagarea unui produs:*

*Actor principal: utilizatorul*

*Scenariu principal:*

- *Utilizatorul porneste aplicatia;*
- *Apasa butonul “Adauga produs”, se deschide interfata pentru manipularea datelor pentru produse;*

- Pentru a aduga un produs nou trebuie introduse datele urmatoare: Nume , Pret, Stoc;
- Utilizatorul apasa butonul adauga, iar produsul cu datele introduse de utilizator va aparea in tabelul cu produse;

*Use-case: adagarea unei comenzi:*

*Actor principal: utilizatorul*

*Scenariu principal:*

- Utilizatorul porneste aplicatia;
- Apasa butonul “Adauga comanda”, se deschide interfata pentru manipularea datelor pentru comenzi;
- Pentru a aduga o comanda noua trebuie introduse datele urmatoare: Client ID , Produs ID, Cantitate;
- Utilizatorul apasa butonul adauga, iar comanda cu datele introduse de utilizator va aparea in tabelul cu comenzi;

*Use-case: sterrgerea unui produs/client si unei comenzi*

*Actor principal: utilizatorul*

*Scenariu principal:*

- Utilizatorul porneste aplicatia;
- Apasa butonul “Adauga Produs/Adauga client/Adauga comanda”, se deschide interfata pentru manipularea datelor pentru tipul selectat;
- Utilizatorul apasa pe iconita de stergere din tabel din dreptul itemului pe care doreste sa-l stearga, iar respectivul item dispare din tabel, iar tot odata este sters si din baza de date;

•

*Use-case: editarea unui produs/client si unei comenzi*

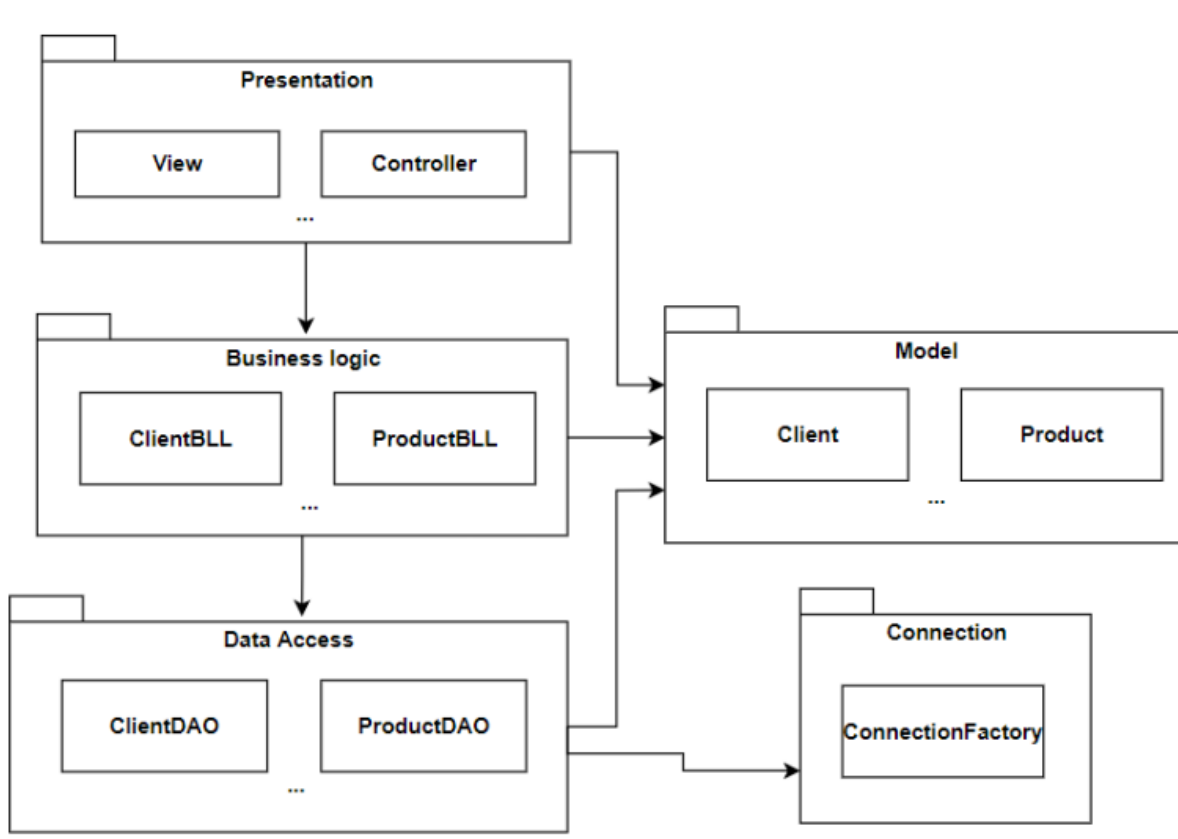
*Actor principal: utilizatorul*

*Scenariu principal:*

- Utilizatorul porneste aplicatia;
- Apasa butonul “Adauga Produs/Adauga client/Adauga comanda”, se deschide interfata pentru manipularea datelor pentru tipul selectat;
- Utilizatorul apasa pe iconita de editare din tabel din dreptul itemului pe care doreste sa-l editeze;
- Modifica datele dorite;
- Apsa butonul “Salveaza”, iar datele sunt salvate atat in tabelu cu itemi cat si in baza de date;

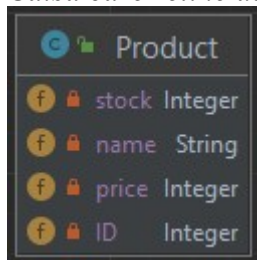
### **3. Proiectare**

*Diagrama de clase imaprta pe pachete:*



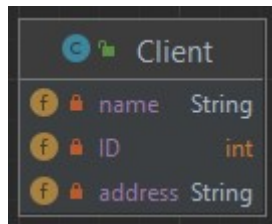
### 3.1 Clasa Product

- Clasa care retine datele pentru un produs din depozit: ID, nume, pret, stoc.



### 3.2 Clasa Client

- Clasa care retine datele pentru un client al depozitului: ID, nume, adresa.



### 3.3 Clasa Order

- Clasa care retine datele pentru o comanda: Id, client Id, produs Id, cantitate, pretul total al comenzi.

Order		
f	total	Integer
f	clientID	Integer
f	ID	Integer
f	purchaseDate	Date
f	amount	Integer
f	productID	Integer

### 3.4 Clasa Bill

- Clasa folosit pentru a crea o factura noua;

Bill		
f	fileWriter	FileWriter

### 3.5 Clasa AbstractDAO

- Clasa generica folosita pentru accesele datelor din baza de date

AbstractDAO<T>		
f	type	Class<T>
f	LOGGER	Logger
m	AbstractDAO()	
m	createSelectQuery()	String
m	update(T)	T
m	findById(int)	T
m	createObjects(ResultSet)	List<T>
m	delete(int)	T
m	findAll()	List<T>
m	populateTable(List<T>, TableView<T>)	void
m	createSelectQuery(String)	String
m	insert(T)	T

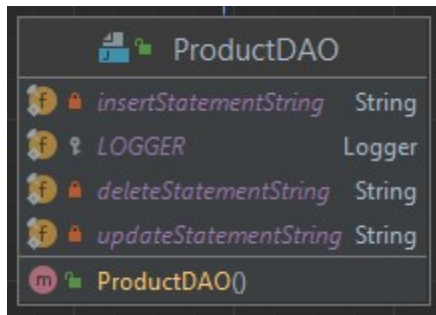
### 3.6 Clasa ClientDAO

- Clasa folosita pentru accesul datelor pentru clientii din baza de date

ClientDAO		
f	LOGGER	Logger
f	insertStatementString	String
f	updateStatementString	String
f	deleteStatementString	String
m	ClientDAO()	

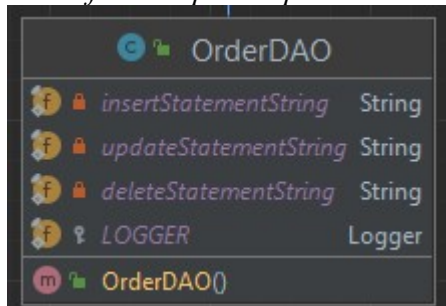
### 3.7 Clasa ProductDAO

- Clasa folosita pentru prelucrarea datelor legate de produse din baza de date



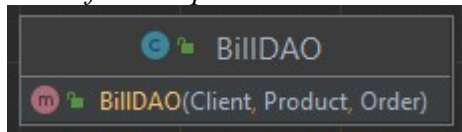
### 3.8 Clasa OrderDAO

- Clasa folosita pentru prelucrarea datelor legate de comenzi din baza de date



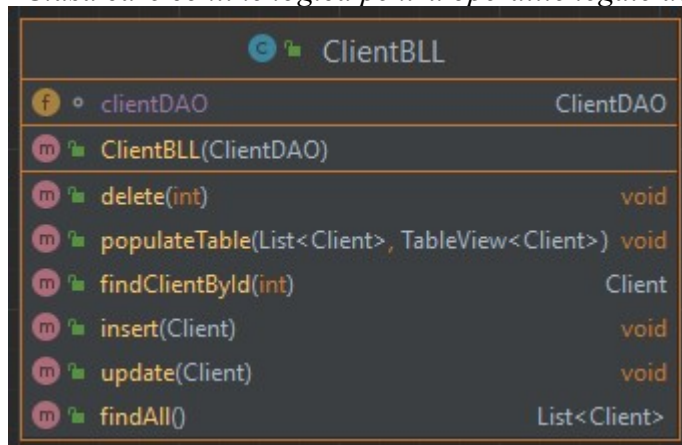
### 3.9 Clasa BillDAO

- Clasa folosita pentru construirea unei facturi.



### 3.10 Clasa ClientBLL

- Clasa care contine logica pentru operatiile legate de clienti



### 3.11 Clasa ProductBLL

- Clasa care contine logica pentru operatiile legate de produse

ProductBLL		
f	productDAO	ProductDAO
m	ProductBLL(ProductDAO)	
m	delete(Product)	void
m	update(Product)	void
m	populateTable(List<Product>, TableView<Product>)	void
m	findProductById(int)	Product
m	findAll()	List<Product>
m	insert(Product)	void

### 3.12 Clasa OrderBLL

- Clasa care contine logica pentru operatiile legate de comenzile clientilor

OrderBLL		
f	orderDAO	OrderDAO
m	OrderBLL(OrderDAO)	
m	update(Order)	void
m	populateTable(List<Order>, TableView<Order>)	void
m	findOrderById(int)	Order
m	delete(Order)	void
m	insert(Order)	void
m	findAll()	List<Order>

### 3.13 Clasa BillBLL

- Clasa care contine logica pentru crearea unei noi comenzi

BillBLL		
f	billDAO	BillDAO
m	BillBLL(BillDAO)	

### 3.14 Clasa ConnectionFactory

- Clasa care conecteaza aplicatia la baza de date



ConnectionFactory		
f	DRIVER	String
f	PASS	String
f	singleInstance	ConnectionFactory
f	LOGGER	Logger
f	DBURL	String
f	USER	String
m	createConnection()	Connection
m	getConnection()	Connection
m	close(Connection)	void
m	close(ResultSet)	void
m	close(Statement)	void

### 3.15 Clasa MainController

- Clasa controller pentru interfata unde se selecteaza operatia dorita aduga client/ adauga produs/ adauga comanda.

MainController		
f	addClientButton	Button
f	addOrderButton	Button
f	addProductButton	Button
m	addClient(MouseEvent)	void
m	addProduct(MouseEvent)	void
m	addOrder(MouseEvent)	void

### 3.16 Clasa ClientController

- Clasa controller pentru interfata in care se prelucraeaza datele legate de clienti

ClientController		
f	editColumn	TableColumn<Client, String>
f	nameField	TextField
f	nameColumn	TableColumn<Client, String>
f	clientIDColumn	TableColumn<Client, Integer>
f	saveButton	Button
f	view	View
f	addressField	TextField
f	observableList	ObservableList<Client>
f	addressColumn	TableColumn<Client, String>
f	client	Client
f	addButton	Button
f	clientTable	TableView<Client>
f	cancelButton	FontAwesomeIconView
m	setError()	void
m	saveButton(MouseEvent)	void
m	validate()	boolean
m	addButton(MouseEvent)	void
m	refresh()	void
m	cancelButton(MouseEvent)	void
m	loadData()	void
m	initialize(URL, ResourceBundle)	void

### 3.17 Clasa ProductController

- Clasa controller pentru interfata in care se prelucreaza datele legate de produse

ProductController		
f	productTable	TableView<Product>
f	productIDColumn	TableColumn<Product, Integer>
f	stockField	TextField
f	priceField	TextField
f	priceColumn	TableColumn<Product, Integer>
f	editColumn	TableColumn<Product, String>
f	cancelButton	FontAwesomeIconView
f	view	View
f	addButton	Button
f	nameField	TextField
f	saveButton	Button
f	product	Product
f	nameColumn	TableColumn<Product, String>
f	stockColumn	TableColumn<Product, Integer>
f	observableList	ObservableList<Product>
m	cancelButton(MouseEvent)	void
m	initialize(URL, ResourceBundle)	void
m	validate()	boolean
m	setError()	void
m	saveButton(MouseEvent)	void
m	refresh()	void
m	loadData()	void
m	addButton(MouseEvent)	void

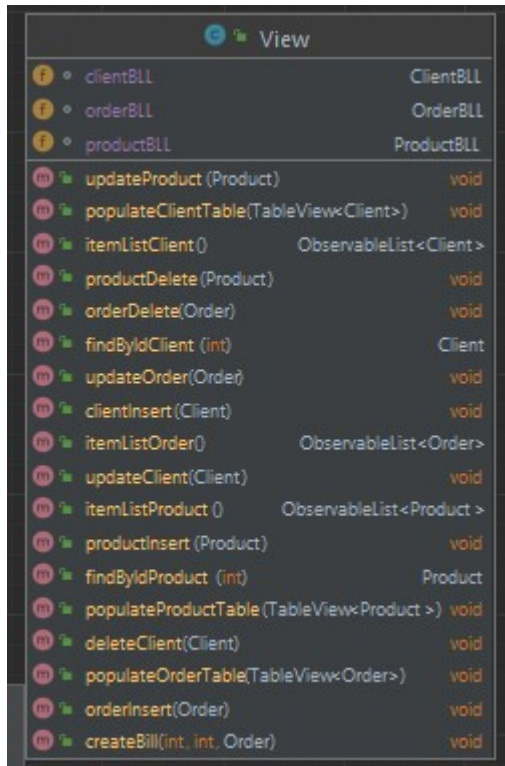
### 3.18 Clasa OrderController

- Clasa controller pentru interfata in care se prelucreaza datele legate de comenzi.

OrderController	
editCol	TableColumn<Order, String>
productIDColumn	TableColumn<Product, Integer>
productTable	TableView<Product>
client	Client
order	Order
clientObservableList	ObservableList<Client>
clientIDField	TextField
addressColumn	TableColumn<Client, String>
amountField	TextField
orderIDColumn	TableColumn<Order, Integer>
orderObservableList	ObservableList<Order>
amountColumn	TableColumn<Order, Integer>
stockColumn	TableColumn<Product, Integer>
errorText	Text
purchaseDateColumn	TableColumn<Order, Date>
observableListProduct	ObservableList<Product>
clientIDColumn	TableColumn<Client, Integer>
addButton	Button
product	Product
clientTable	TableView<Client>
nameProductColumn	TableColumn<Product, String>
totalColumn	TableColumn<Order, Integer>
nameClientColumn	TableColumn<Client, String>
clientIDOrderColumn	TableColumn<Order, Integer>
priceProductColumn	TableColumn<Product, Integer>
saveButton	Button
productIDField	TextField
cancelButton	FontAwesomeIconView
productIDOrderColumn	TableColumn<Order, Integer>
view	View
orderTable	TableView<Order>
validate()	boolean
loadData()	void
saveButton(MouseEvent)	void
refresh()	void
setError()	void
addButton(MouseEvent)	void
getClientTableElement(MouseEvent)	void
cancelButton(MouseEvent)	void
getProductTableElement(MouseEvent)	void
initialize(URL, ResourceBundle)	void

### 3.19 Clasa View

- Clasa folosita pentru accesul datelor care trebuie sa fie afisate pe cele 3 intrefete pentru Client/ Produs /Comanda.



## 4 Implementare

4.1 Clasa Client: retine attributele unui client.

Atribute:

- ID: int – idul unic al clientului;
- name: String - numele clientului;
- address: String – adresa clientului;

Metode: getter si setter;

4.2 Clasa Product: retine attributele unui produs

Atribute:

- ID: Integer – idul unic al unui produs;
- Name: String – numele produsului;
- Stock: Integer – numarul de produse de tipu; respectiv din depozit;

Metode: getter si setter;

4.3 Clasa Order: clasa care retine attributele unei comenzi

Atribute:

- ID: Integer – idul comenzi;
- clientID: Integer – idul clientului caruia ii apartine comanda;
- productid: Integer – idul produsului cumparat de client;
- purchaseDate: Date – data achizitionari produsului de catre client;
- amount: Integer – cantitate de produse cumparate;
- total: Integer – suma totala care o are de achitat;
- Metode: getter si setter;

#### 4.4 Clasa Bill: clasa folosita petru a crea o noua comanda

*Atribute: -fileWriter: FileWriter: fisierul txt care va reprezenta factura;*

*Metode: getter si setter;*

#### 4.5 Clasa BillBLL: clasa folosita pentru crearea unei facturi

*Atribute: - billDAO: BillDAO*

*Metode: constructor;*

#### 4.6 Clasa ClientBLL: clasa care contine operatile pentru datele legate de clienti

*Atribute: - clientDAO: ClientDAO – instanta pentru accesul datelor pentru clienti*

*Metode:*

- *findClientById – cauta un client dupa ID-ul sau;*
- *findAll – returneaza o lista cu toti clienti din baza de date;*
- *insert – insereaza un client nou in baza de date;*
- *update – editeaza un client din baza de date;*
- *delete – sterge un client din baza de date;*
- *populateTable – populeaza tabelul de clienti in interfata cu utilizatorul;*

#### 4.7 Clasa ProductBLL: clasa care contine operatile pentru datele legate de produse

*Atribute: - productDAO: ProductDAO – instanta pentru accesul datelor pentru produse*

*Metode:*

- *findProductById – cauta un produs dupa ID-ul sau;*
- *findAll – returneaza o lista cu toate produsele din baza de date;*
- *insert – insereaza un produs nou in baza de date;*
- *update – editeaza un produs din baza de date;*
- *delete – sterge un produs din baza de date;*
- *populateTable – populeaza tabelul de produse in interfata cu utilizatorul;*

#### 4.8 Clasa OrderBLL: clasa care contine operatile pentru datele legate de comenzi

*Atribute: - orderDAO: OrderDAO – instanta pentru accesul datelor pentru comenzi*

*Metode:*

- *findOrderById – cauta o comanda dupa ID-ul sau;*
- *findAll – returneaza o lista cu toate comenzile din baza de date;*
- *insert – insereaza o comanda nou in baza de date;*
- *update – editeaza o comanda din baza de date;*
- *delete – sterge o comanda din baza de date;*
- *populateTable – populeaza tabelul de comenzi in interfata cu utilizatorul;*

#### 4.9 Clasa AbstractDAO: clasa care continue metodele generice de acces la datele din baza de date

*Atribute:*

- *LOGGER: Logger;*
- *type: Class<T> - typul clasei care acceseaza metodele;*

*Metode:*

- *createSelectQuery(String field) – creaza o interogare pentru baza de date pentru a cauta un element dupa un anumit field;*
- *createSelectQuery() – creaza o interogare pentru a returna toate elementele dintr-un tabel;*
- *findAll – returneaza o lista cu toate elementele dintr-un tabel;*

- *findByld* – cauta un element dupa ID-ul sau;
- *createObjects* – creaza obiectele din rezultatul dat dupa executarea interogari;
- *delete* – sterge un element din baza de date;
- *insert* – insereaza un nou element in baza de date;
- *update* – updateaza un element din baza de date;
- *populateTabel* – populeaza un tabel cu elemente de tip “type”;

4.10 Clasa ClientDAO: clasa care continue interogariile pentru accesul datelor din baza de date pentru clienti

*Atribute:*

- *LOGGER: Logger;*
- *insertStatementString: String* – interogarea pentru inserare
- *deleteStatementString: String* – interogarea pentru stergere
- *updateStatementString: String* – interogarea pentru editare

4.11 Clasa ProductDAO: clasa care continue interogariile pentru accesul datelor din baza de date pentru produse

*Atribute:*

- *LOGGER: Logger;*
- *insertStatementString: String* – interogarea pentru inserare
- *deleteStatementString: String* – interogarea pentru stergere
- *updateStatementString: String* – interogarea pentru editare

4.12 Clasa OrderDAO: clasa care continue interogariile pentru accesul datelor din baza de date pentru comenzi

*Atribute:*

- *LOGGER: Logger;*
- *insertStatementString: String* – interogarea pentru inserare
- *deleteStatementString: String* – interogarea pentru stergere
- *updateStatementString: String* – interogarea pentru editare

4.13 Clasa ClientController

*Atribute:*

- *addButton*
- *addressField*
- *addressColumn*
- *cancelButton*
- *clientIDColumn*
- *editColumn*
- *nameField*
- *clientTabel*
- *saveButton*
- *client*
- *observableList*
- *view*

*Metode:*

- *saveButton: salveaza modificarile aduse asupra unui client din baza de date;*
- *addButton: aduga un nou client in baza de date;*
- *cancelButton: trece la interfata principala;*
- *initialize: initializeaza interfata*

- *loadData*: incarca datele pentru interfata
- *refresh*: reimprospateaza datele interfetei
- *setError*: afiseaza utilizatorului o eroare in cazul in care ceva nu este bine

#### 4.14 Clasa ProductController

*Attribute:*

- *addButton*
- *priceField*
- *nameColumn*
- *cancelButton*
- *productIDColumn*
- *editColumn*
- *nameField*
- *productTabel*
- *saveButton*
- *product*
- *observableList*
- *view*
- *stockColumn*
- *priceColumn*
- *stockField*

*Metode:*

- *saveButton*: salveaza modificarile aduse asupra unui client din baza de date;
- *addButton*: aduga un nou client in baza de date;
- *cancelButton*: trece la interfata principala;
- *initialize*: initializeaza interfata
- *loadData*: incarca datele pentru interfata
- *refresh*: reimprospateaza datele interfetei
- *setError*: afiseaza utilizatorului o eroare in cazul in care ceva nu este bine

#### 4.15 Clasa OrderController

*Attribute:*

- *addButton*
- *nameColumn*
- *cancelButton*
- *productIDColumn*
- *editColumn*
- *nameField*
- *productTabel*
- *saveButton*
- *product*
- *view*
- *stockColumn*
- *priceColumn*
- *stockField*
- *productIDField*
- *clientIDField*
- *addressColumn*

- *amountColumn*
- *clientIDColumn*
- *clientIDOrderColumn*
- *clientTabel*
- *errorText*
- *nameClientColumn*
- *amountField*
- *orderTabel*
- *priceProductTabel*
- *totalColumn*
- *client*
- *observableListProduct*
- *observableListClient*
- *observableListOrder*

*Metode:*

- *saveButton*: salveaza modificarile aduse asupra unui client din baza de date;
- *addButton*: aduga un nou client in baza de date;
- *cancelButton*: trece la interfata principala;
- *initialize*: initializeaza interfata
- *loadData*: incarca datele pentru interfata
- *refresh*: reimprospateaza datele interfetei
- *setError*: afiseaza utilizatorului o eroare in cazul in care ceva nu este bine

#### *4.16 GUI*

- pentru construirea interfetelor grafice am folosit JavaFX SceneBuilder si CSS pentru anumite efecte asupra componentelor;

*Interfata pricipala:*





*Interfata pentru prelucrarea clientilor:*










Nume

Pret

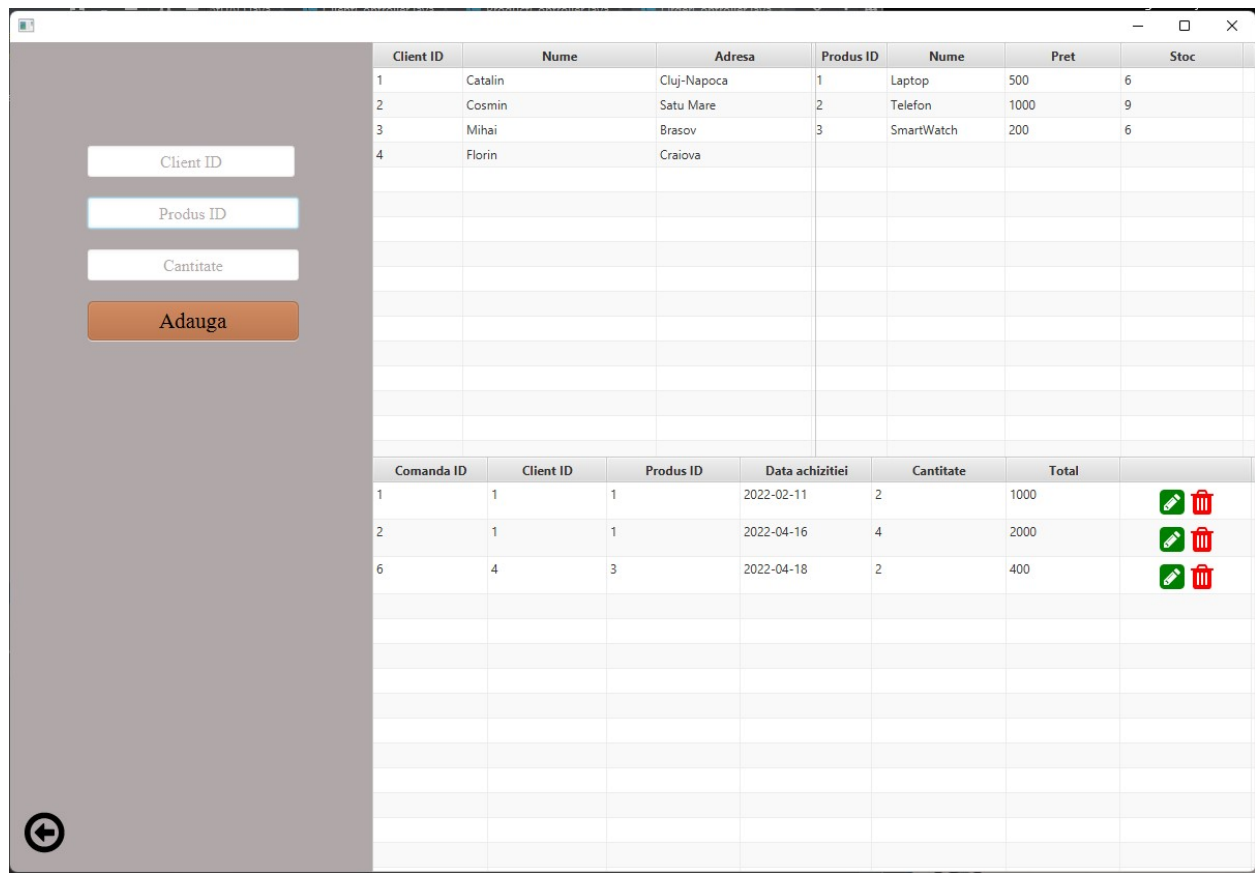
Stoc

Adauga



Produs ID	Nume	Pret	Stoc	
1	Laptop	500	6	 
2	Telefon	1000	9	 
3	SmartWatch	200	6	 

Interfata pentru preluarea comenzilor:



## 5 Resultate

*Scenarii de utilizare:*

### Scenariu 1: Adaugarea unui client nou

- Se apas butul de run, porneste aplicatia de simulare;
- Apare GUI-ul principal;
- Apasa butonul :Adauga client”
- Introducem datele pentru noul client;
- Apasa butonul “Adauga”;
- Noul client apare in tabel;

### Scenariu 2: Adaugarea unui client esuata

- Se apas butul de run, porneste aplicatia de simulare;
- Apare GUI-ul principal;
- Apasa butonl “Adauga client”
- Introducem datele pentru noul client, dar nu pe toate, ori nici una;
- Apasa butonul “Adauga”;
- Campurile necompletate vor aparea cu rosu, iar clientul nou nu este adugat;

### Scenariu 3: Editarea unui client

- Se apas butul de run, porneste aplicatia de simulare;

- Apare GUI-ul principal;
- Apasa butonul "Adauga client"
- Apasam iconita "Creion" sin dreptul clientului caruia vrem sa-l editam datele;
- Introducem noile datele pentru noul client;
- Apasa butonul "Salveaza";
- Noile date apare in tabel;

*Scenariu 4: Stergere unui client din baza de date*

*Se apasa butonul de run, porneste aplicatia de simulare;*

- Apare GUI-ul principal;
- Apasa butonul "Adauga client"
- Se apasa iconita "cos de gunoi", din dreptul clientului pe care dorim sa-l stergem;
- Clientul dispare din tabel;

*Acelas lucru se face si pentru produse si comenzi;*

## 6 Concluzii

*Concluzia finala este ca am implementat o aplicatie de management pentru un depozit, care continue 4 interfete, pentru clienti, produse si comenzi, foarte usor de folosit de catre oricine. Referitor la cunostine acumulate, am recapitulate bazele de date(MySQL), am invatat sa folosesc iconitele in JavaFX, am invatat tehnica de reflexive si am folosit JavaDoc. La parte de dezvoltare as imbunati interfețele grafice.*

## 7 Bibliografie

- 1 .....[https://dsrl.eu/courses/pt/materials/A3\\_Support\\_Presentation.pdf](https://dsrl.eu/courses/pt/materials/A3_Support_Presentation.pdf)