

**FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

***Tehnici de programare fundamentale***

***Tema 3 – Order Management***

**Matioc Bogdan Darius, 30224**

**Profesor îndrumător: David Sera**

Cuprins

[1. Obiectivul temei 3](#_Toc70002875)

[2. Analiza problemei, modelare, scenarii, cazuri de utilizare 3](#_Toc70002876)

[2.1 Analiza problemei 3](#_Toc70002877)

[2.2 Modelare 3](#_Toc70002878)

[2.3 Scenarii și cazuri de utilizare 4](#_Toc70002879)

[3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator) 5](#_Toc70002880)

[4. Implementare 5](#_Toc70002881)

[Pachetul Model 6](#_Toc70002882)

[Pachetul entities 6](#_Toc70002883)

[Pachetul dto 6](#_Toc70002884)

[Pachetul dataAcces 6](#_Toc70002885)

[Pachetul businessLogic 6](#_Toc70002886)

[Pachetul connection 13](#_Toc70002887)

[5. Rezultate 13](#_Toc70002888)

[6. Concluzii 13](#_Toc70002889)

[7. Bibliografie 13](#_Toc70002890)

# Obiectivul temei

Obiectivul acestei teme de a realiza o aplicație de order management. Aplicația trebuie să poată realiza acțiuni precum adăugarea de noi clienți, adăugarea de noi produse, plasarea unei comenzi etc.

Obiectivele secundare care dac posibilă realizarea unei astfel de aplicații sunt:

* Definirea claselor corespunzătoare
* Legarea la baza de date unde sunt stocate informațiile
* Realizarea interfeței grafice
* Preluarea datelor necesare din interfața grafică și din baza de date
* Testarea și verificarea corectitudinii implementării realizate
* Crearea de fisiere pdf

# Analiza problemei, modelare, scenarii, cazuri de utilizare

## Analiza problemei

Această aplicație are, initial, trei opțiuni diferite care pot fi utilizate. Prima dintre ele este fereastra pentru client unde utilizatorul are posibilitatea de a adăuga un client nou, de a modifica datele despre un anume client, de a șterge un client și de a vizualiza tabelul cu toți clienții. A doua fereastră disponibilă este cea pentru produs, unde utilizatorul are posibilitatea de a adăuga un produs nou, de a modifica datele despre un produs, de a șterge un produs și de a vizualiza tabelul cu toate produsele disponibile. Cea de-a treia fereastră a aplicației permite utilizatorului de a plasa o comandă pentru un client și un produs existent.

Pe langa aceste 3 fereste am mai creat inca 3. O ferestra de intrare in care utilizatorul poate alege care din cele 3 ferestre descrise anterior sa fie deschisa. Cea de a doua se deschide la creearea unei facturi, continand informatii referitoare la ultima comanda care a fost plasata. Ultima permite utilizatorului sa vizualizeze un ustoric al comenzilor.

## Modelare

Din punct de vedere al modelării, se pot distinge 3 părți principale (3 etaje). Prima parte este reprezentată de interactiunea cu interfața grafică. Utilizatorul interacționează în mod direct cu aceasta. Interfața este ușor de folosit și sugestivă, din punct de vedere al funcționalității pe care o are. Așadar utilizatorul are posibilitatea de a manipula conținutul datelor existente în baza de date după cum dorește. Poate insera date noi, le poate șterge și, de asemenea, le poate modifica și actualiza. Cea de-a doua parte este reprezentată de partea de cod aflată în spatele interfeței, care face posibilă întreaga funcționalitate a programului și înglobează algoritmii necesari pentru toate operațiile. Ultimul etaj se ocupa de interactiunea cu baza de date.

## Scenarii și cazuri de utilizare

Descriere caz de utilizare:

Dat fiind faptul că această aplicație oferă utilizatorului posibilitatea de a efectua mai multe tipuri de acțiuni o să descriu câte un caz de utilizare pentru fiecare dintre cele trei ferestre: client, produs, comandă.

Client:

1. Utilizatorul introduce datele corespunzătoare pentru un client, apoi apasă butonul pentru a adăuga acel client.
2. Utilizatorul poate modifica date in tabel iar aceste schimbari se vor efectua si asupra bazei de date la apasrea tastei enter sau la parasirea celulei.
3. Utilizatorul selecteaza un anumit rand din tabel, apoi apasă butonul pentru a șterge clientul respectiv

Toate aceste modificări efectuate asupra tabelului de clienți o să se observe în timp real pe tabelul afișat în interfața grafică.

Produs:

1. Utilizatorul introduce datele corespunzătoare pentru un produs, apoi apasă butonul pentru a adăuga acel produs.
2. Utilizatorul poate modifica date in tabel iar aceste schimbari se vor efectua si asupra bazei de date la apasrea tastei enter sau la parasirea celulei.
3. Utilizatorul selecteaza un anumit rand din tabel, apoi apasă butonul pentru a șterge produs respectiv

Toate aceste modificări efectuate asupra tabelului de produse o să se observe în timp real pe tabelul afișat în interfața grafică.

Comandă:

1. Utilizatorul selecteaza un anumit client
2. Utilizatorul adaugă produsele in cos prin dublu click pe linia respectiva
3. Pentru fiecare produs adaugat in cos se poate modifica cantitatea, iar pretul se va modifica automat
4. Utilizatorul va apasa butonul „Place the order pentru a finaliza comanda”; In acest moment se va dschide o noua fereastra care contine factura. In aceasta fereasta sunt doua butoane care ofera 2 optiuni: 1.„download” pt a descarca factura respectiva (se va deschide o noua fereastra pentru a selecta locatia unde faa fi salvat fisierul pdf), 2. Un buton pentru a se intoarce in fereastra Order
5. Utilizatorul pot selecta optiunea „Shopping history” pentru a vizualiza a putea vizualiza istoricul comenzilor

Toate aceste modificări efectuate asupra tabelului de comenzi o să se observe în timp real pe tabelul afișat în interfața grafică. De asemenea, dacă utilizatorul introduce o cantitate mai mare decât există în stoc pentru produsul în cauză, atunci o să apară un mesaj de eroare. Dacă introduce o cantitate mai mică sau egală ca cea a stocului, atunci stocul este decrementat, lucru care se poate observa în tabelul pentru produse.

# Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

Una dintre deciziile de proiectare pe care le-am luat vizavi de această aplicație constă în structurarea ei așa încât să respecte layered architecture. Prin prisma acestei structurări proiectul este bine delimitat din punct de vedere al claselor, delimitare care se bazează pe funcționalitatea pe care o au. Prin această structurare proiectul este mai lizibil și mai coerent din punct de vedere logic.

Algoritmii care stau la baza acestei aplicații de order managent sunt aceia prin intermediul cărora se realizează conexiunea la baza de date și, mai ales, aceea prin intermediul cărora sunt realizate și executate interogările care permit manipularea datelor de către utilizator. Câteva exemple de astfel de metode sunt: printClients, deleteClientById, insertClient, updateClient.

Interfața utilizator este ușor de folosit și intuitivă din punct de vedere al funcționalității. Aceasta permite personalizarea datelor de intrare și urmărirea rezultatului fără probleme.

# Implementare

Proiectul realizat respectă structura layered architecture. Așadar, o să descriu fiecare pachet.

## Pachetul Model

Pachetul *model* cuprinde clasele care descriu tabelele care se află în baza de date. Pachetul contine la randul sau 2 pachete: dto si entites. Aceste pachete contin fiecare 4 clase care sunt destul de asemanatoare.

### Pachetul entities

Pachetul entities contine clasele care sunt o reprezentare in java a tabelelor din baza de date. Toate datele rezultate in urma interogarilor bazei de date vor fi stocate in obiecte instante ale uneia dintre aceste clase. Mai precis clasele cuprinse în acest pachet sunt:

* **Client** ( - id : int, - name : String, - address : String, - email : String, - age : int, - deleted : int )
* **Product** ( - id : int, - name : String, - stock : int, - price : double, - deleted : int )
* **\_Order** ( - id : int, - idClient : int, - date : Timestamp )
* **BuyList** ( - id : int, - idOrder : int, - iProduct : int, - quantity : int, - price : double )

Aceste clase implementează metode de *set* și *get* și eventual *toString*.

### Pachetul dto

Pachetul dto contin clasele: **ClientDTO, ProductDTO, \_OrderDTO, BuyListDTO.** Aceste clase sunt folosite petru a transfera date intre etajele Buiniss Logic si Presentation. Aceste clase contin doar datele necesae pentru Presentation layer.

Aceste clase implementează metode de *set* și *get* și eventual *toString*.

## Pachetul dataAcces

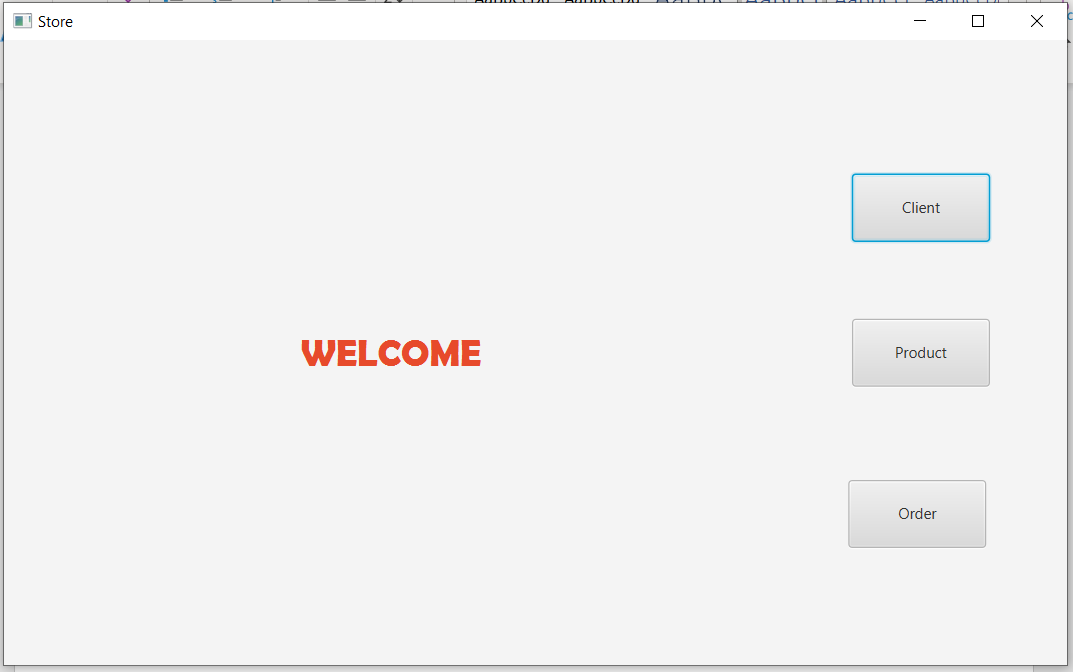
Pachetul *dataAccessLayer*  cuprinde clasele care cuprind interogările. Am realizat interfata RepositoryDAO care contine definitia antetelor de functii prin care se fac diverse operatii pe baza de date. Methode preazente in reposutory sunt: fiindAll, fiindById, findByField, add, update si delete. Toate aceste methoe au comportamentul descris in clasa generica, abstracata DAO. Aceasta clasa ofera o implementare generica a methodelor enumerate anterior si nu numai. Aceasra clasa foloseste un obiect al clasei Query, clasa care contine creare tuturor querilor utilizate in clasa DAO. Tot in acest pachet gasm si clasele ClientDAO, OrderDAO, ProductDAO si BuyListDAO clase instatiabile prin care se specifica si tiul obiectului pe care se va lucra.

## Pachetul businessLogic

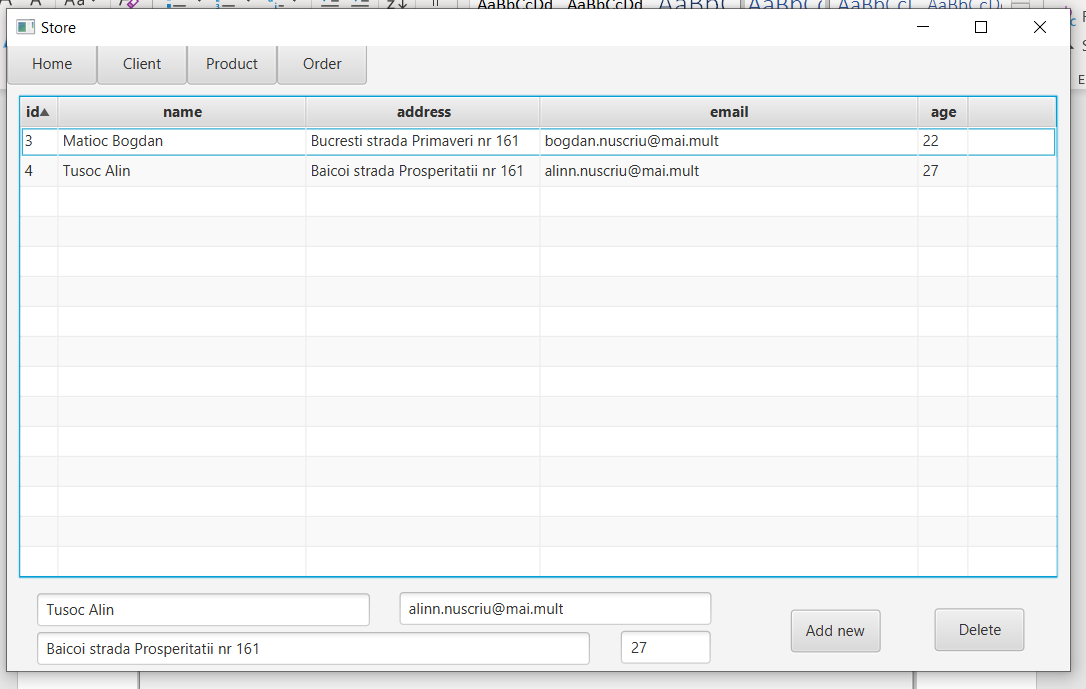
Pachetul *businessLayer* se ocupă cu preluarea și procesarea datelor provenite de la clasele din pachetul dataAccess. Așadar, fiecare clasă din pachetul model are asociată o astfel de clasă de procesare a datelor. In aceste clase se face si conversia de la un obiect entitate la un obiect dao

Pachetul *presentation* cuprinde clasele care realizează interfața grafică și clasele de control ale interfeței grafice. Aici vom gasi iarasi folosirea tehnicii reflection pentru crearea tabelelorsi popularea lor. Interfața cu care interacționează utilizatorul arată în felul următor:

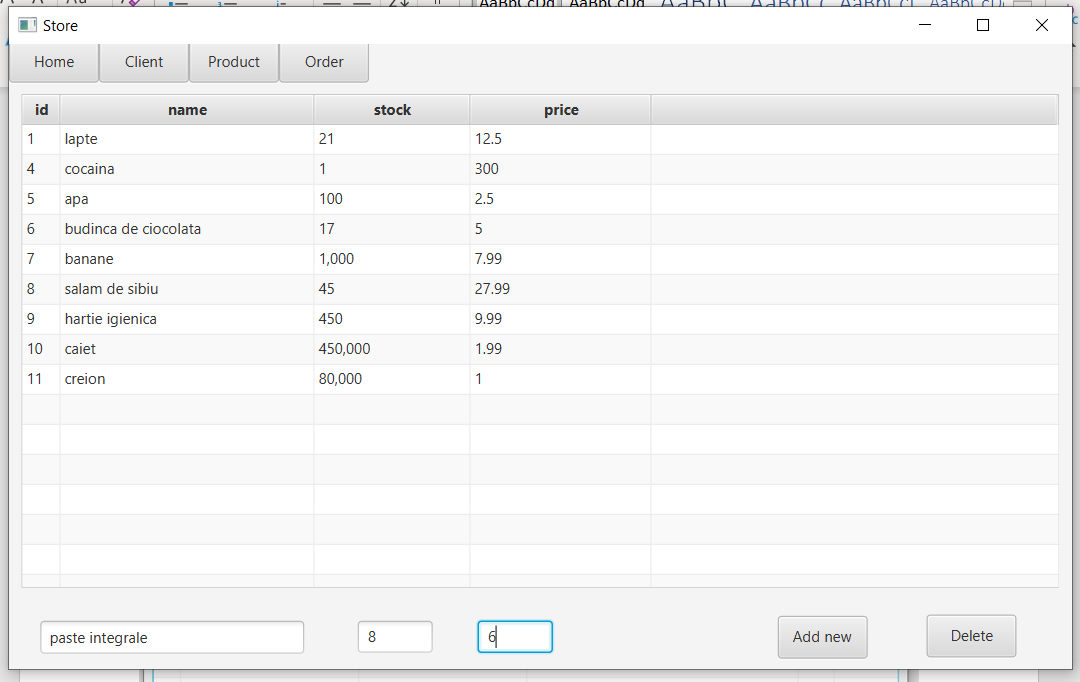
* Welcome (home) scene



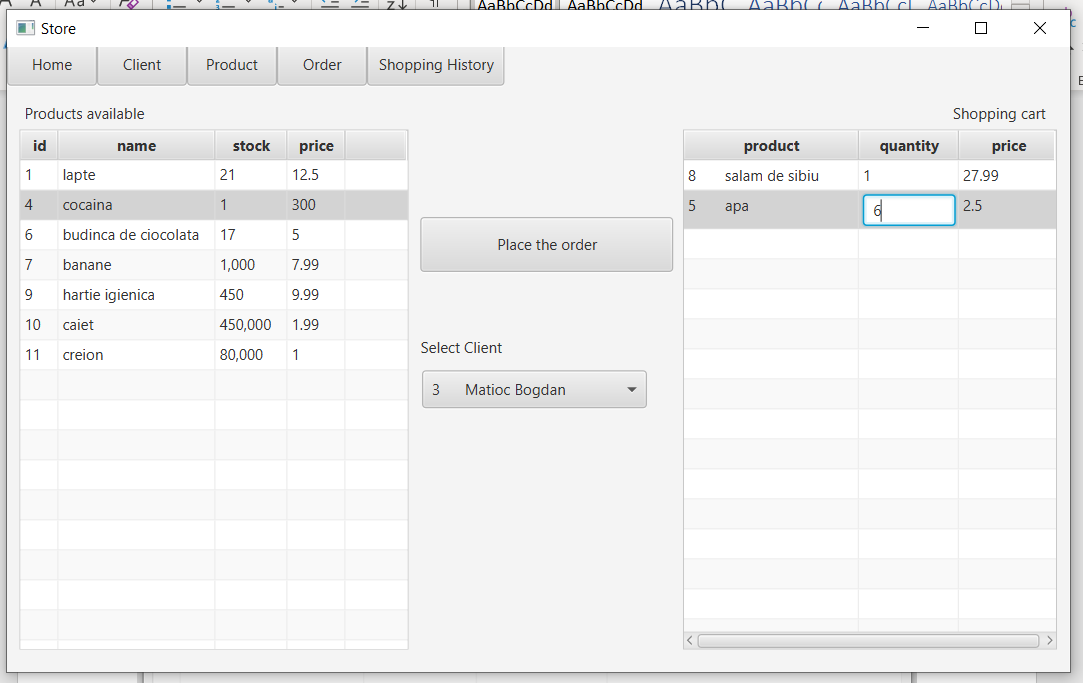
* Client scene



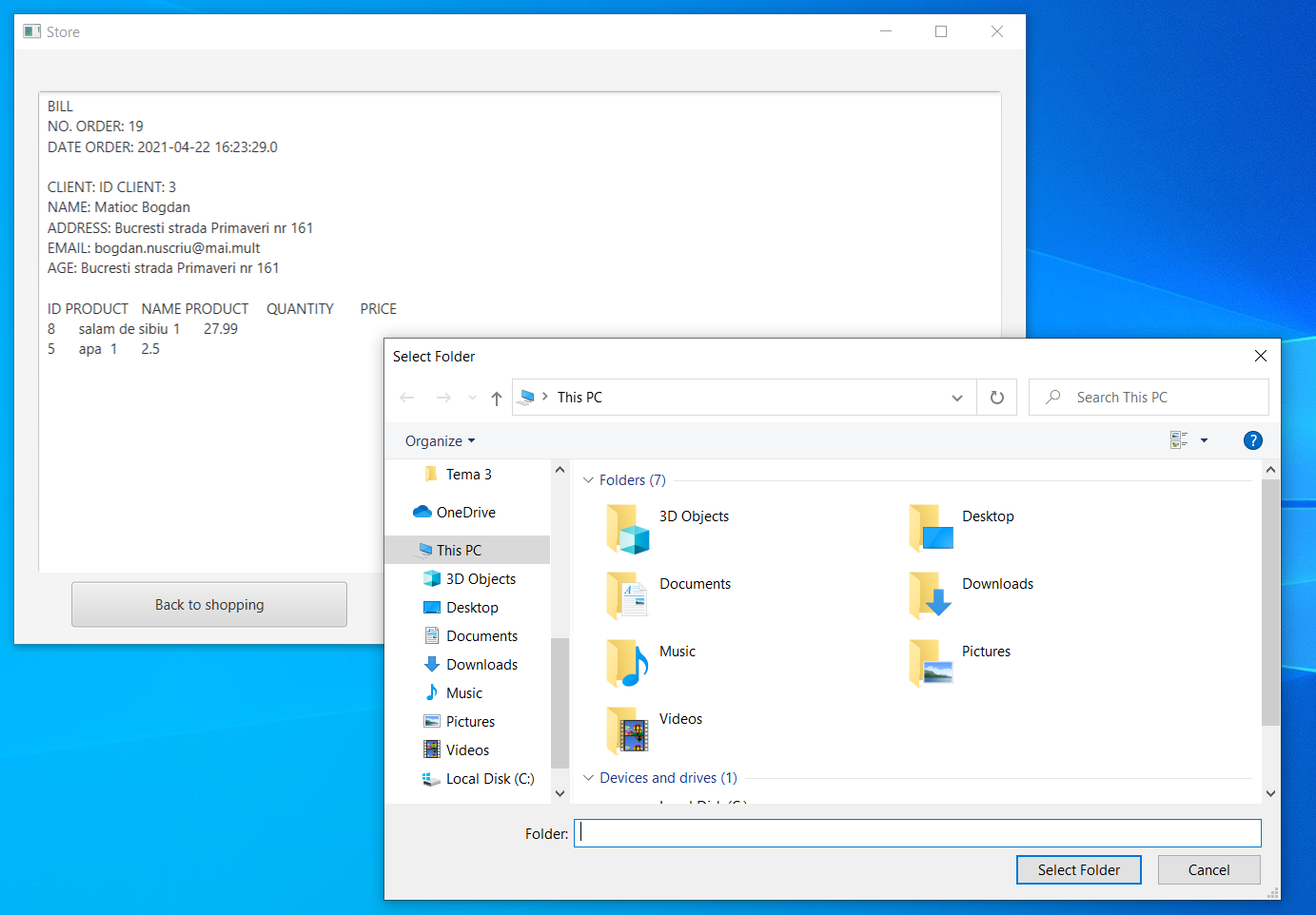
* Product scene



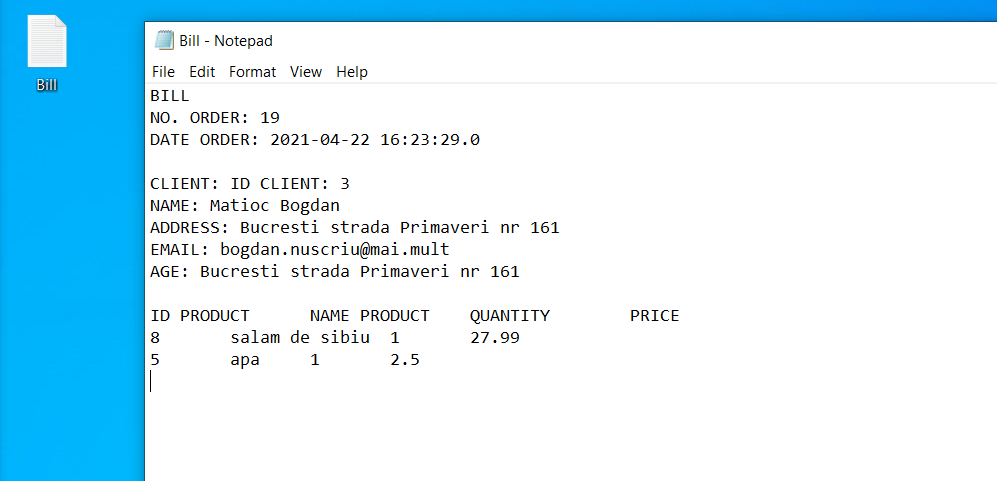
* Order scene



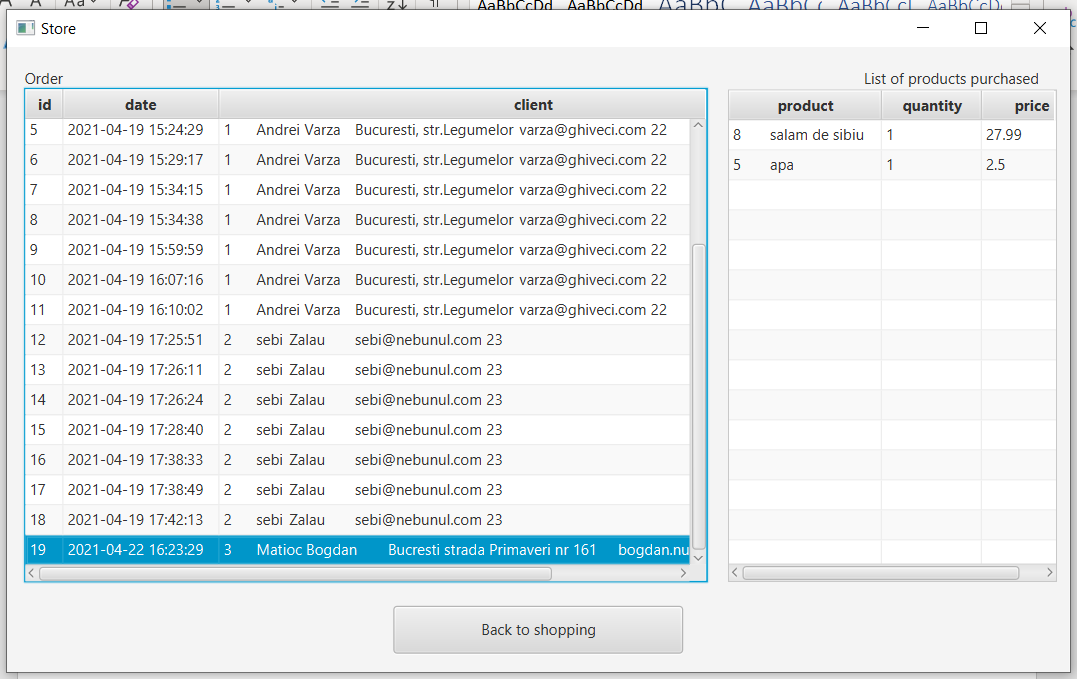
* Bill scene and chose directory window (aceasta fereastra se dechide dupa apasrea butonului download)



* Fisier creat



* Shopping History Scene



Prima fereastră de care este întâmpinat utilizatorul îl lasă să aleagă unul dintre cele trei butoane, prin intermediul cărora se va deschide o nouă fereastră dedicată clienților, produselor sau comenzilor. Dacă utilizatorul apasă pe butonul *Client* o să se deschidă cea de-a doua fereastră ilustrată în imaginea de mai sus. Odată cu deschiderea acestei ferestre utilizatorul are posibilitatea de a adăuga sau edita sau a sterge un client. Dacă utilizatorul apasă pe butonul *Product* o să se deschidă cea de-a treia fereastră ilustrată în imaginea de mai sus. Odată cu deschiderea acestei ferestre, utilizatorul are posibilitatea de a adăuga sau a edita sau sterge un produs. Dacă utilizatorul apasă butonul *Order* o să se deschidă cea de-a patra fereastră ilustrată în imaginea de mai sus. Odată cu deschiderea acestei ferestre utilizatorul are posibilitatea de a vizualiza o listă cu toți clienții existenți și toate produsele existente. De asemenea, acesta poate să plaseze o comandă prin adăugarea unui selectia unui client ,adaugarea produselor in cos si apasea buronului “place the order”. Daca utilizatorul doreste maim ult de un articol dintr-un rodus acesta poate sa mareasca acestacantitate editat fieldul quantity din tabelul afferent cosului de cumparaturi inainte de a plasa comanda. După ce o comandă este plasată se va deschide o noua scene cu informatile ultimei comezi. Prin apasarea butonului download o sa se creeze într-un fișier text o factură. În plus, utilizatorul are opțiunea de a vizualiza un istoric al comenzile plasate.

## Pachetul connection

Clasa **ConnectionFactory**

Cu ajutorul acestei clase se realizează conexiunea la baza de date.

# Rezultate

Pentru a verifica faptul că aplicația funcționează corect am realizat multiple teste și verificări. M-am asigurat că întotdeauna datele introduse în interfață corespund cu cele existente în baza de date, că fiecare ștergere sau modificare a unui client, produs sau comenzi corespund cu ceea ce se află în baza de date.

# Concluzii

Prin prisma acestei teme mi-am consolidat cunoștințele pe programare orientată pe obiect. De asemenea, am învățat cum să fac legătura cu baza de date și să îmi preiau datele. O nouă tehnică învățată a fost aceea de reflection, prin intermediul căreia am construit tabelele din interfață. Mai mult decât atât, datorită acestei aplicații am învățat despre structura de layered architecture.

Din punct de vedere al dezvoltării ulterioare aplicației i se pot aduce multe îmbunătățiri. O posibilă îmbunătățire ar fi aceea ca utilizatorul să aibă posibilitatea de a plasa într-o comandă mai mult de un singur produs. Mai mut decât atât, la plasarea comenzii utilizatorul ar putea să introducă o adresă de email, iar odată cu plasarea acestei comenzi să fie transmis un mesaj de confirmare cum că lucrurile au decurs cu succes. De asemenea, aplicația ar putea să fie îmbunătățită și printr-un mecanism de ștergere automată a unui produs din stoc, în cazul în care stocul curent al produsului respectiv este 0. Mai mult decât atât, aplicația se poate îmbunătății și prin apariția unor mesaje de avertizare semnificative în momentul în care utilizatorul vrea să introducă un client sau un produs cu un id deja existent în listele de clienți sau produse.

# Bibliografie

1. FUNDAMENTAL PROGRAMMING TECHNIQUES - ASSIGNMENT 3 – SUPPORT PRESENTATION
2. <https://gitlab.com/utcn_dsrl/pt-layered-architecture>
3. <https://gitlab.com/utcn_dsrl/pt-reflection-example>