Assignment 1

Analysis and Design Document

Student: Dulău Marius-Cristian

**Group: 30235**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 3

2. Use-Case Model 4

3. System Architectural Design 6

4. UML Sequence Diagrams 9

5. Class Design 10

6. Data Model 13

7. System Testing 16

8. Bibliography 22

1. Requirements Analysis

# Assignment Specification

Folosiți JAVA/C# API pentru a proiecta și implementa o aplicație pentru livrarea mâncării. Aplicația are 2 tipuri de utilizatori (un utilizator obișnuit care reprezintă un clinet și un user administrator) care au nevoie de un cont (username și parolă) pentru a folosi aplicația.

# Functional Requirements

Clientul poate efectua următoarele operații:

* Adăugare/modificare/vizualizare a informațiilor personale (nume, adresă, număr telefon, etc.);
* Creare/modificare/ștergere/vizualizarea coșului de cumpărături (produse, suma totală de bani, tipul plății etc.);
* Alegerea tipului de plată;
* Vizualizarea istoricului de comenzi.

Administratorul poate efectua următoarele operații:

* CRUD pe informațiile clientului;
* CRUD pe meniu;
* Generare raport pentru o anumită perioadă ce conține activitățile effectuate de client și setarea acestuia ca și client loial (5% reducere)

.

# Non-functional Requirements

Constrângerile aplicației sunt:

* Datele vor fi stocate într-o bază de date;
* Folosirea Design Pattern-ului Layers (design arhitectural) pentru a organiza aplicația;
* Folosirea unui domeniu logic pattern (transaction script sau domeniul modelului) / un data source hybrid (table module, active record) și un data source pure pattern (table data gateway, row data gateway, data mapper) cel mai potrivit pentru aplicație;
* Toate intrările aplicației vor fi validate înainte de stocarea acestora în baza de date.

2. Use-Case Model

**Use case:** Comandă produse

**Level:** user-goal level

**Primary actor:** Client

**Main success scenario:** Clientul accesează aplicația prin intermediul contului creat de administrator. El introduce datele contului în fereastra de login. Dacă datele sunt corecte, va putea efectua comenzi.

**Extensions:** După autentificare clientul poate vizualiza produsele, datele personale și istoricul comenzilor. Când vizualizează datele are dreptul de a le modifica. Pentru a se înregistra modificarea, el trebuie să apese pe un buton pentru a se efectua această operație. Totodată, el își poate șterge contul doar cu un click pe un alt buton. La vizualizarea produselor el poate selecta cantitatea pe care vrea să o pună în coșul de cumpărături. Dacă se decide că nu dorește o anumit fel de mâncare poate renunța la el. Pentru a se efectua comanda, clientul trebuie să specifice modalitatea de plată. La vizualizare istoricului, clientului îi este prezentat toate cumpărăturile efectuate.

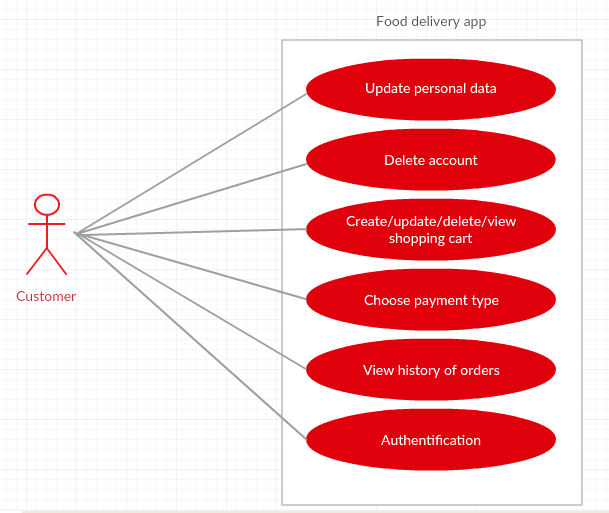


Figura 1. Use case client

**Use case:** Administrare aplicație

**Level:** user-goal level

**Primary actor:** Administrator

**Main success scenario:** administratorul accesează aplicația prin intermediul unui cont standard. El introduce datele contului în fereastra de login. Dacă datele sunt corecte, va putea efectua modificări asupra conținutului bazei de date.

**Extensions:** După autentificare administratorul poate vizualiza clienții, să modifice datele acestora, să creeze conturi noi, să adauge/modifice produse, să seteze un client ca fiind loial.

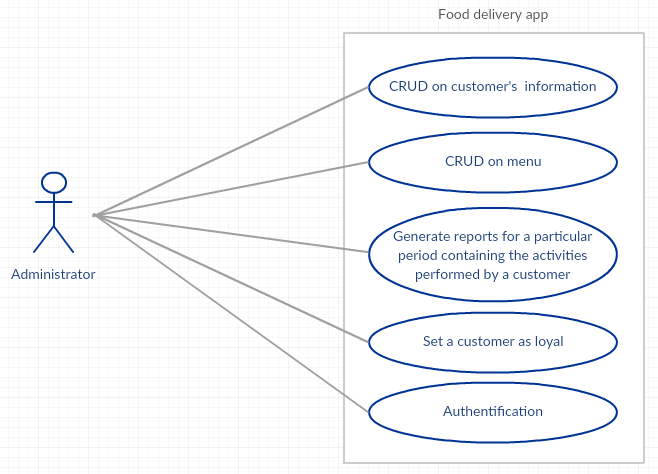


Figura 2. Use case administrator

3. System Architectural Design

**3.1 Architectural Pattern Description**

Un model arhitectural exprimă o schemă fundamentală de organizare structurală pentru sistemele software. Un exemplu de pattern arhitectural este design pattern-ul Layers. Design pattern-ul Layers ajută la structurarea aplicațiilor care pot fi descompuse în grupuri de subcerințe. Cele mai importante nivele ale acestui pattern sunt: Application Layer, Business Layer și Data Layer. Layers nu se limitează doar la 3 nivele, în practică folosindu-se mai multe (arhitectură pe N nivele) .

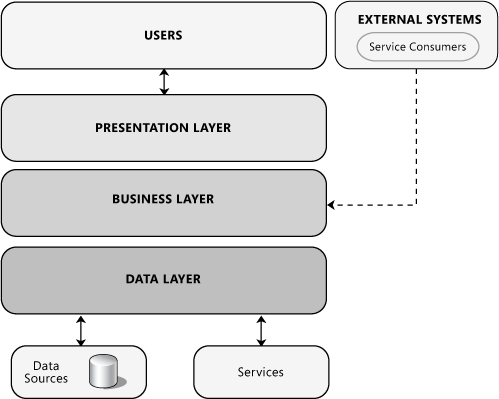


Figura 3. Arhitectură de tip layer

Constrângerile utilizării acestui pattern sunt:

* Modificările codului sursă nu ar trebui să se propage în sistem;
* Interfețele trebuie să fie stabile;
* Părțile componente ale sistemului ar trebui să poată fi schimbate;
* Responsabilități similare ar trebui grupate pentru a ajuta la înțelegerea și mentenanța proiectului;
* Componentele complexe necesită o descompunere suplimentară.

Beneficii:

* Reutilizarea nivelelor;
* Convertibilitatea;
* Suport pentru standardizare.

Dezavantaje:

* Eficiență redusă;
* Creșterea efortului de proiectare;
* Dificultatea de a stabili granularitatea corectă a nivelelor;

Transaction script se referă la verificate datelor pentru a nu exista deja în baza de date. Dacă acestea nu există deja, ele trebuie validate, iar în cazul unor date numerice trebuie efectuate calculele pentru câmpurile dintr-un tabel. La final aceste date vor fi inserate în baza de date.

Table Module este conceptul prin care o clasă din model are ca și variabile de instanță câmpurile tabelului pe care îl reprezintă. Table data gateway prezintă conceptul de creare de clase care să efectueze operații pe o bază de date.

**3.2 Diagrams**

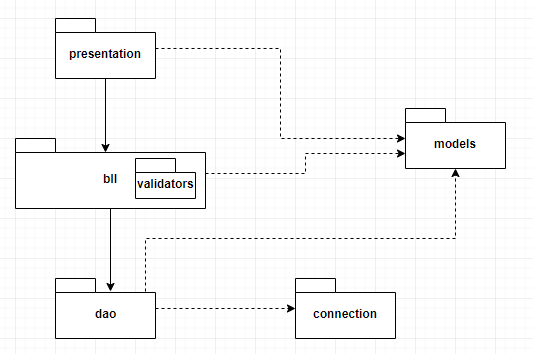


Figura 4. Diagrama de pachete

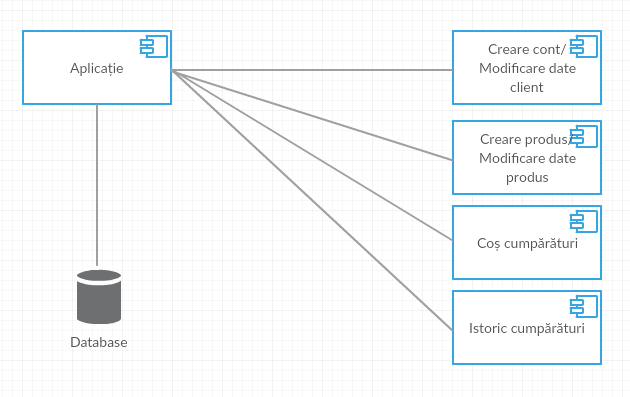


Figura 5. Digrama de componente

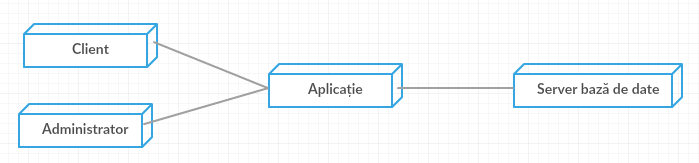


Figura 6. Diagrama de deployment

4. UML Sequence Diagrams

Un scenario relevant este efectuarea unei comenzi. Clientul trebuie să se autentifice. După autentificare, el trebuie să selecteze unul sau mai multe produse cu cantitatea aferentă. După efectuarea comenzii, clientul poate verifica istoricul comenzilor, iar apoi se deautentifică.

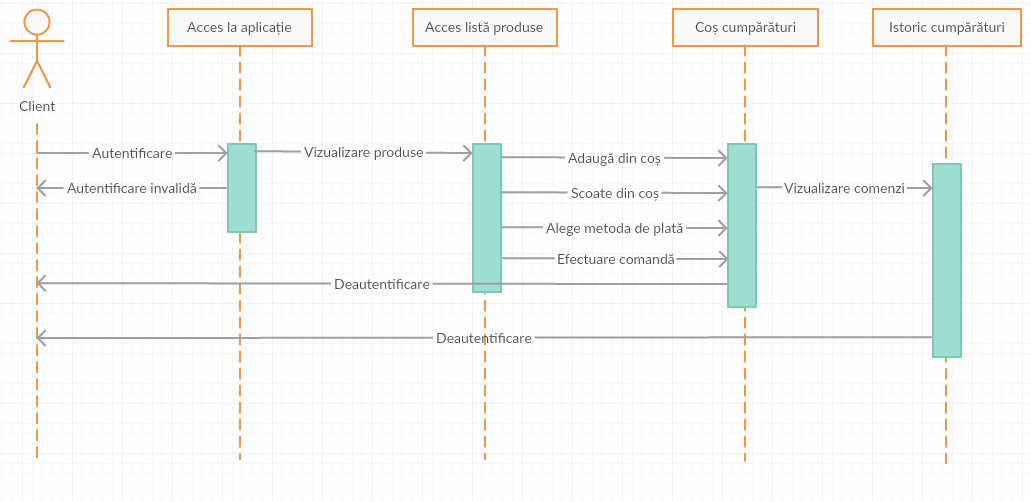


Figura 7. Diagrama de secvență

5. Class Design

**5.1 Design Patterns Description**

Design pattern-urile utilizate în cadrul acestui proiect sunt Layers, domain logic pattern(transaction script), data source hybrid pattern (table module) și data source pure pattern (table data gateway).

Transaction script se referă la verificate datelor pentru a nu exista deja în baza de date. Dacă acestea nu există deja, ele trebuie validate, iar în cazul unor date numerice trebuie efectuate calculele pentru câmpurile dintr-un tabel. La final aceste date vor fi inserate în baza de date. De exemplu, pentru inserarea datelor unui client trebuie să se verifice dacă cnp-ul este valid, dăcă toate câmpurile au fost completate, dacă numarul de telefon are 10 cifre, dacă email-ul este valid. Pentru validarea datelor s-au creat clase care implementează o interfață generică.

Table Module este conceptul prin care o clasă din model are ca și variabile de instanță câmpurile tabelului pe care îl reprezintă.

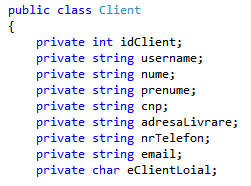


Figura 8. Clasa corespunzătoare tabelului

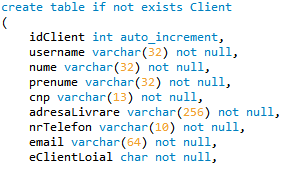


Figura 9. Tabelul după care se construiește clasa

Table data gateway prezintă conceptul de creare de clase care să efectueze operații pe o bază de date. Pentru un client operațiile relevante sunt inserarea, actualizarea, ștergerea, căutarea după numele de utilizator și validarea datelor.

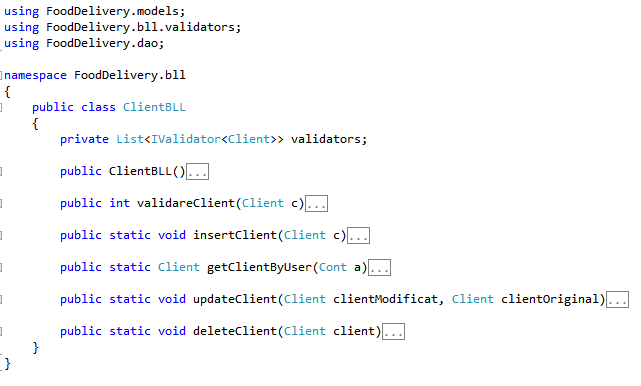


Figura 10. Metode pentru implementarea table data gateway

Pe lângă aceste patter-uri s-a folosit și Singleton pentru a se crea o singură instanță a clasei responsabilă cu conexiunea la baza de date.

**5.2 UML Class Diagram**

În pachetul de model avem următoarele clase: CategoriiProdus, Client, Comanda, Cont, DetaliiComanda și Produs. Aceste clase sunt independente între ele, nu moștenesc alte clase. În pachetul bll avem o clasă pentru fiecare clasă din model pentru a trimite datele la clasele ce efectuează operații cu baza de date. Pachetul bll are inclus un subpachet ce conține validatorii pentru anumite câmpuri din unele clase. Clasele din pachetul bll folosesc clasele din pachetul dao pentru efectuarea propriu-zisă a operațiilor cu baza de date. Clasele din dao au nevoie de o conexiune la baza de date. Clasa responsabilă pentru conexiunea cu baza de date. Această clasă este o clasă de tip Singleton.

Pentru interfața grafică se folosesc clase stocate în pachetul presentation.

Prin împărțirea claselor pe pachete se utilizează pattern-ul Layers.

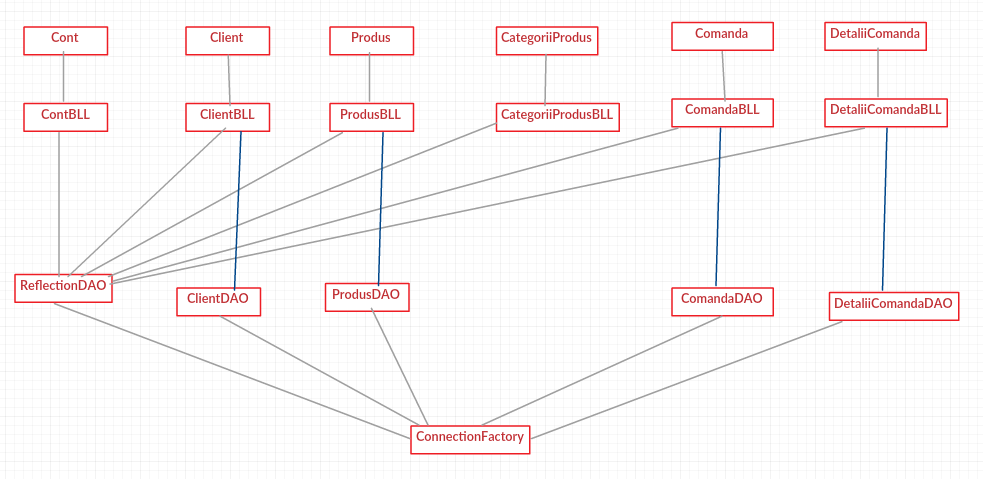


Figura 11. Diagrama UML a claselor sistemului

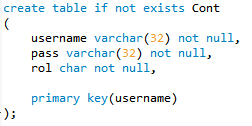
6. Data Model

În cadrul proiectului, clasele din model se mapează pe tabelele din baza de date. Ca și metode, aceste clase au gettere și settere. Clasele din Business Logic se ocupă cu trimiterea claselor model la clasele din pachetul DAO.

Clasa Cont are următoarele variabile de instanță:

* private string username;
* private string pass;
* private char rol;

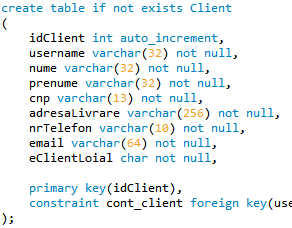
Tabelul corespunzător clasei Cont este următorul:



Clasa Client are următoarea structură la variabilele de instanță:

* private int idClient;
* private string username;
* private string nume;
* private string prenume;
* private string cnp;
* private string adresaLivrare;
* private string nrTelefon;
* private string email;
* private char eClientLoial;

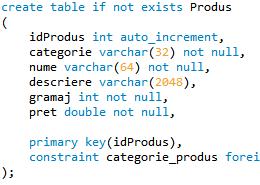
Tabelul corespunzător clasei Client este următorul:



Clasa Produs are următoarele variabile de instanță:

* private int idProdus;
* private string categorie;
* private string nume;
* private string descriere;
* private int gramaj;
* private double pret;

Variabilele de instanță ale clasei Produs sunt câmpurile tabelului Produs din baza de date.

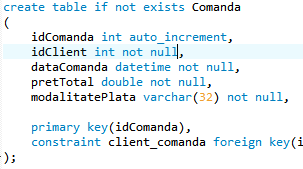


Clasa CategoriiProdus are doar o variabilă de instanță: string categorie. Tabelul CategoriiProdus din baza de date are doar un câmp, la fel ce și clasa care se mapează.

Clasa Comandă are 5 variabile de instanță:

* private int idComanda;
* private int idClient;
* private string dataComanda;
* private double pretTotal;
* private string modalitatePlata;

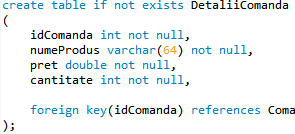
În baza de date tabelul Comanda are următoarea structură:



Ultima clasă din pachetul BLL, DetaliiComanda, are doar 4 variabile de instanță:

* private int idComanda;
* private string numeProdus;
* private double pret;
* private int cantitate;

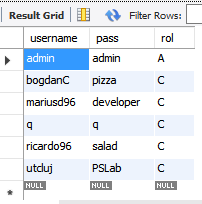
În tabelul DateliiComanda din baza de date are aceleasși câmpuri pe care le are și clasa DetaliiComanda :



7. System Testing

Testarea aplicației s-a realizat prin verificarea datelor din baza de date cu cele din interfața grafică. În continuare se prezintă scenariul pentru testarea creării unui cont, modificarea datelor acestuia, crearea unei comenzi de clientul nou creat și vizualizarea istoricului de cumpărături. Având clasă care utilizează metoda de reflection, testarea s-a bazat mai mult pe integrare, validare și partiționare.

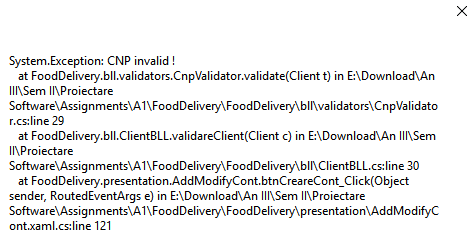
Vom crea contul cu username = a și pass = a. Inițial tabela conturilor din baza de date arată în felul următor:



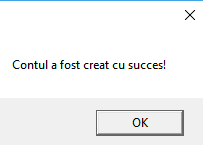
Din contul administratorului cream un cont.

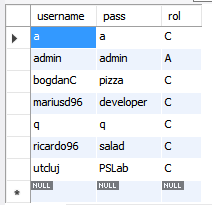


Contul din imaginea de mai sus e invalid din cauza codului numeric personal. La încercarea de introducere a acestui cont sistemul va arunca o excepție.

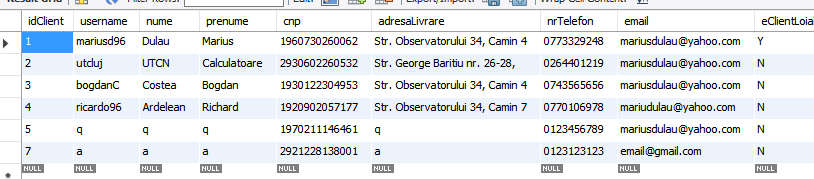


Dacă modificăm cnp-ul să fie ok, contul și clientul vor fi introduși în baza de date. Sistemul va da un mesaj că datele au fost introduse cu succes în baza de date.





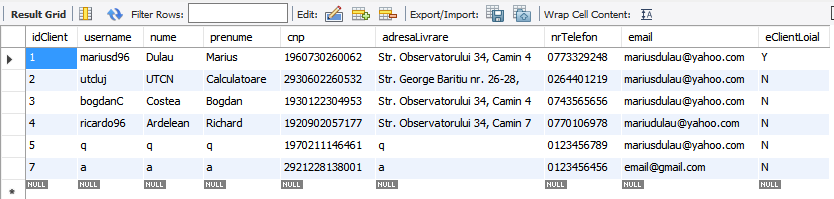
Clientul nou inserat are idClient = 7;



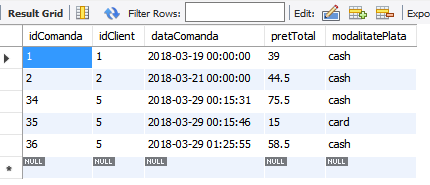
Din contul clientului încercăm să modificăm numărul de telefon.



Numărul de telefon nou introdus din imaginea de mai sus este valid (are 10 cifre), deci va fi actualizat în baza de date.

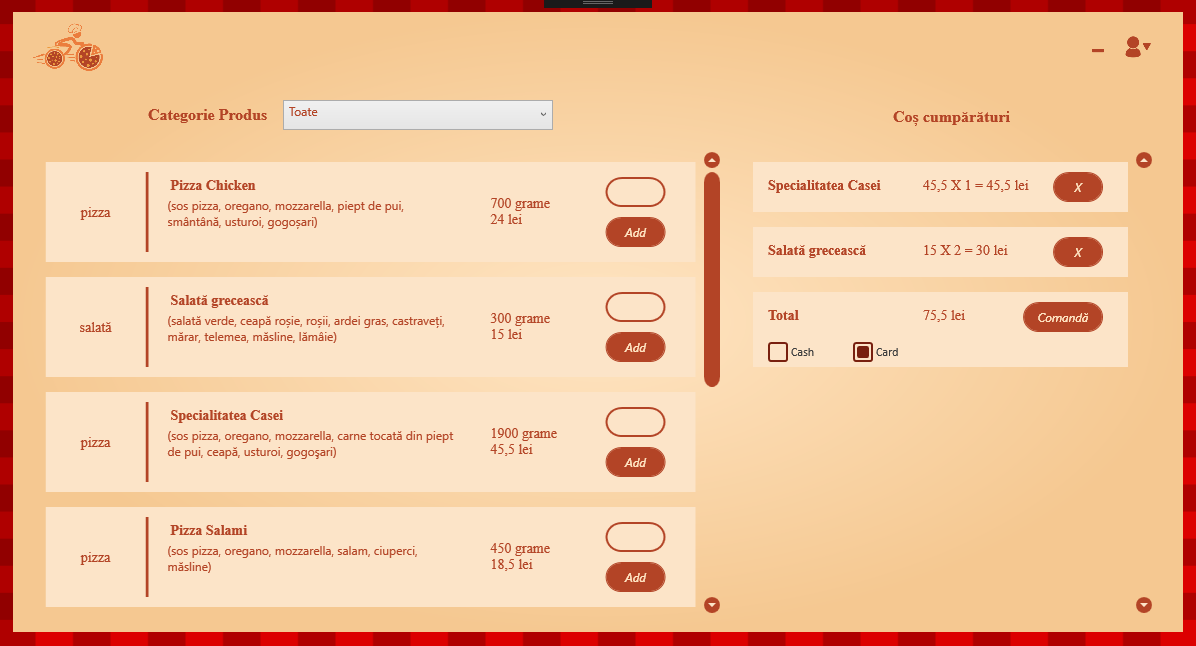


Datele din tabelele Comanda și DetaliiComandă sunt următoarele:

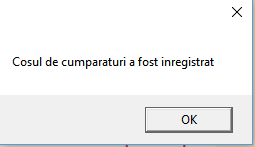




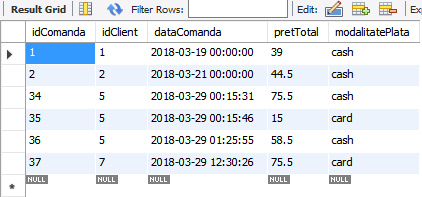
Datele introduse vor fi cele din figura de mai jos.



Datele din imaginea de mai sus sunt corecte, este selectată opțiunea de plată, deci datele sunt introduse în baza de date. Va apărea un mesaj care indică inserarea corectă.

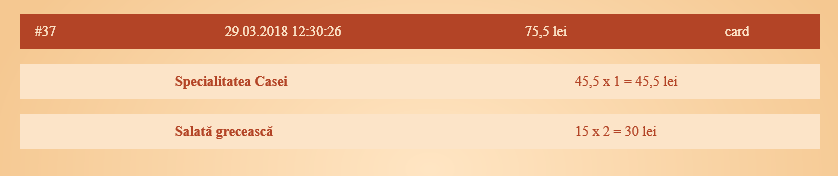


În MySql Workbench s-a verificat popularea tabelelor cu datele din interfața grafică. În imaginile de mai jos idClient este 7, iar idComanda este 37.





În interfața grafică, la vizualizarea istoricului de cumpărături se observă că datele preluate sunt cele din baza de date.



8. Bibliography

<http://users.utcluj.ro/~dinso/PS2018/Lectures/> (Curs 3)

<https://www.youtube.com/watch?v=fnsU9cqcY3I>

<https://www.youtube.com/watch?v=v_bYm091w3g>

<https://www.youtube.com/watch?v=F6suKxChkk8>

<https://dotnetcademy.net/Learn/4/Pages/6>

<https://stackoverflow.com/questions/350027/setting-wpf-image-source-in-code>

<https://www.youtube.com/watch?v=kF0jleGpH60>

<https://stackoverflow.com/questions/34403009/how-to-clear-the-contents-of-a-canvas-in-wpf>

<https://msdn.microsoft.com/en-us/library/system.windows.documents.textelement.fontstretch(v=vs.110).aspx>

<https://stackoverflow.com/questions/6901070/getting-selected-value-of-a-combobox>

<https://stackoverflow.com/questions/11560465/parse-strings-to-double-with-comma-and-point>