***TEMA 3***

***ORDERS MANAGEMENT***

***Materie: Tehnici de Programare***

***Elev: Pavel Mădălina- Denisa***

***Grupa: 30223***

***Specializare: CTI RO, an 2***

*Cuprins*

[*Obiectivul temei* 2](#_Toc131095220)

[*Analiza problemei, modelare, scenarii, cazuri de utilizare* 3](#_Toc131095221)

[*Proiectare* 4](#_Toc131095222)

[*Implementare* 5](#_Toc131095223)

[*Rezultate* 6](#_Toc131095224)

[Concluzii 9](#_Toc131095225)

[Bibliografie 10](#_Toc131095226)

# *Obiectivul temei*

Obiectivul proiectului este dezvoltarea unei aplicații de gestionare a comenzilor pentru un depozit. Este necesara utilizarea bazelor de date relationale pentru a stoca produsele, clienții și comenzile. Aplicația va fi proiectată conform pattern-ului arhitectural stratificat si va cuprinde principalele tipuri de clase:

* Clasele Model - reprezintă modelele de date ale aplicației.
* Clasele Business Logic - conțin logica aplicației.
* Clasele Presentation - sunt clase legate de interfața grafică (GUI).
* Clasele Data Access - sunt clase care conțin accesul la baza de date.

Această arhitectură stratificată permite o separare clară a responsabilităților și promovează modularitatea și reutilizarea codului.

# *Analiza problemei, modelare, scenarii, cazuri de utilizare*

Un concept extrem de important pe care l-am utilizat in acest proiect a fost “reflection”. Acest termen de refera la generalizarea metodelor si a claselor, in asa fel incat sa se evite scrierea codului duplicat. Acest lucru a fost facut prin crearea clasei AbstractDao, in care au fost implementate toate operatiile care tin de baza de date si care ulterior au fost accesate prin intermediul anumitor obiecte.

În procesul de analiză a problemei și identificare a claselor și funcționalităților acestora, programarea orientată pe obiect ne permite să implementăm o aplicație folosind informații de suprafață.

Impreuna cu cunoasterea paradagimelor programarii orientate pe obiect mai sunt necesare si cunoasterea unor minime tehnici de programare a bazelor de date mySQL.

O **bază de date**, uneori numită și **bancă de date** (abreviat **BD**), reprezintă o modalitate de stocare a unor informatii și date pe un suport extern (un dispozitiv de stocare), cu posibilitatea extinderii ușoare și a regăsirii rapide a acestora.

Cel mai răspândit tip de baze de date este cel relațional, în care datele sunt memorate în tabele. Pe lânga tabele, o bază de date relațională mai poate conține: indecși, proceduri stocate, declanșatori, utilizatori și grupuri de utilizatori, tipuri de date, mecanisme de securitate și de gestiune a tranzacțiilor etc.

Exemplu de use-case:

Initial, utilizatorul trebuie sa aleaga ce tip de operatiune doreste sa efectueze:

* Operatii pe produse(adaugare, stergere, actualizare sau vizualtizare produse);
* Operatii pe clienti(adaugare, stergere, actualizare sau vizualizare clienti);
* Efectuarea unei comenzi.

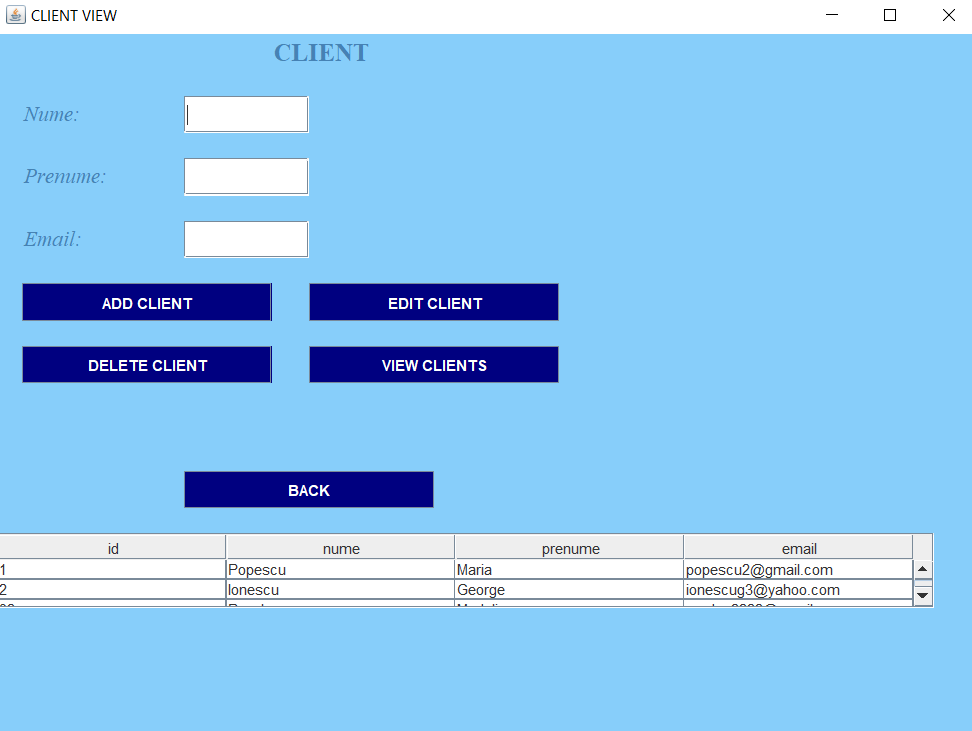
In cazul efectuarii unei comenzi, utilizatorul alege unul dintre clientii existenti, un produs existent, introduce in caseta de text cantitatea dorita si apasa pe butonul “ADD ORDER”.

Dupa apasarea butonului in caseta de text corespunzatoare totalului se va afisa suma totala de plata si va aparea un mesaj de confirmare a plasarii comenzii.

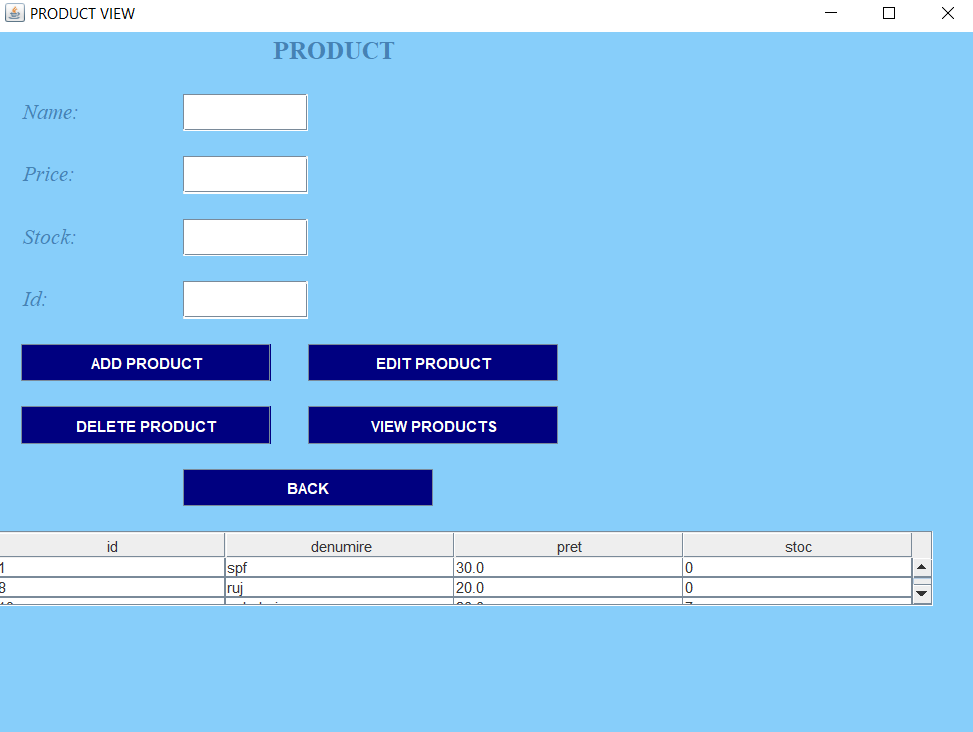
1. Alegerea operatiunii:



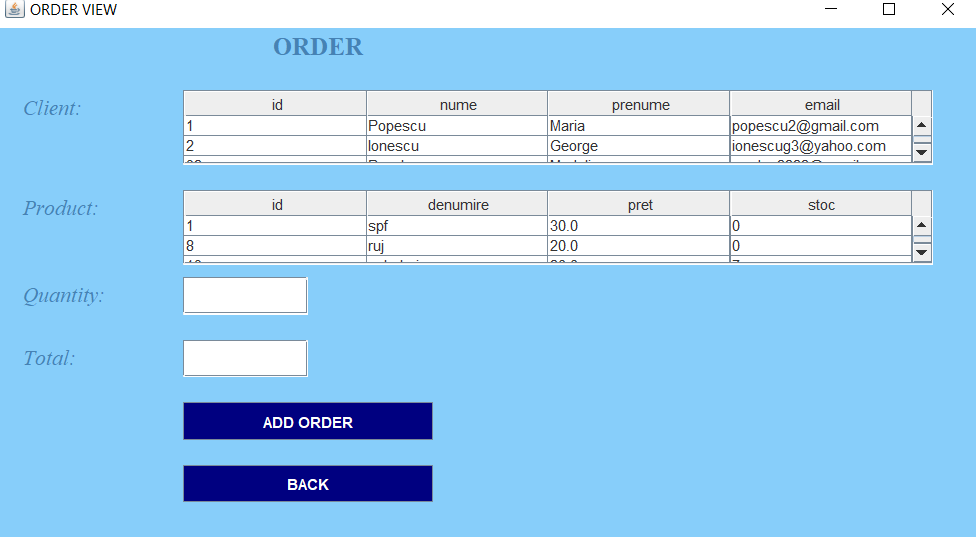
1. Interfata pentru Client:



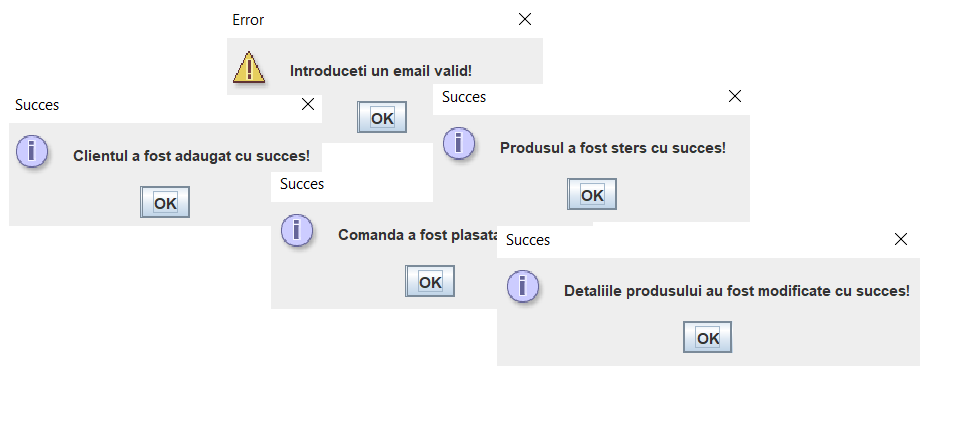
1. Interfata pentru Produs:



1. Plasarea unei comenzi:



In cadrul fiecarei interfete am utilizat si casete de dialog, pentru a semnaliza executia cu succes sau esecul unei operatiuni:



# *Proiectare*

Pentru proiectarea acestei aplicații am folosit design-pattern-ul Model-BusinessLogic-Presentation-DataAccess-Connection, structurându-mi astfel proiectul în 5 pachete:

* *Model:* acest pachet conține clasele **Client** ,**Product, Order si Bill**, care constituie practic baza proiectului, unde sunt „proiectate” obiectele de fiecare tip.
* *BusinessLogic(BLL)*:acest pachet contine clasele **ClientBLL, ProductBLL, OrderBll** dar si o clasa de validare, **EmailValidator.**
* *Presentation:* acest pachet contine clase care conecteaza interfata grafica de logica din spate a programului
* *DAO*: in cadrul claselor din acest pachet se efectueaza operatiile CRUD pe baza de date. Aici practic se utilizeaza informatiile primate de la interfata si se manipuleaza.
* *Connection*:aici avem o singura clasa, cea in care se face conexiunea cu baza de date. Aceasta a fost facut prin adaugarea unei dependinte in **pom.xml**.

# *Implementare*

Cele mai importante clase din acest proiect sunt **AbstractDao** si **Controller.** In cadrul clasei abstracte se utilizeaza, asa cum am mai mentionat, conceptul de “reflexie”, care consta in generalizarea codului pentru a evita scrierea duplicata a acestuia. Aici sunt effectuate toate opertiile pe baza de date, tot ce inseamna manipularea tabelelor.

Pe de alta parte, in clasa Controller sunt legate toate actiunile care au loc in interfata(apasare de buton, selectare rand din tabel), de o anumita clasa/metoda existenta in proiect. Practic, astfel se face aceasta legatura intre interfata si codul din spate.

# 

# Concluzii

În concluzie, proiectul Orders Management utilizează o arhitectură stratificată și baze de date relaționale pentru a gestiona comenzile clienților într-un depozit. Folosind modele de date, logica de afaceri și interfața grafică, proiectul permite o gestionare eficientă a comenzilor și manipularea datelor într-un mod flexibil. Prin aplicarea conceptului de reflection, se obține o abordare dinamică și ușurință în adaptarea și extinderea funcționalității aplicației. În ansamblu, proiectul oferă o soluție bine structurată și scalabilă pentru gestionarea comenzilor într-un mediu de depozit.

# Bibliografie

<https://dsrl.eu/courses/pt/>

<https://www.javatpoint.com/java-jscrollpane>

<https://stackoverflow.com/questions/2385701/regular-expression-for-first-and-last-name><https://dev.mysql.com/doc/workbench/en/wb-admin-export-import-management.html>

<https://stackoverflow.com/questions/3661413/how-to-cast-an-object-to-an-int>