MINISTERUL EDUCAȚIEI, CERCETĂRII, TINERETULUI ȘI SPORTULUI
UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

DOCUMENTATIE TEMA 3

ORDER MANAGEMENT

Cociubei Antonia-Rahela
Grupa 302210
Profesor Marcel Antal

Cuprins

1. Obiective

1.1. Obiectiv Principal

Obiectivul acestui proiect a fost de a simula o aplicatie de managemnet a unui magazin/depozit. Clinetii, produsele si comenzile efectuate sunt înregistrati in baza de date. Administratorul depozitului avand posibilitatea de a modifica anumite caracteristici ale acestora. Dar si de a adauga marfa noua sau a actualiza stocul, ori a manageria comezile plasate.

1.2. Obiectiv Secundar

"Consider an application OrderManagement for processing customer orders for a warehouse. Relational databases are used to store the products, the clients and the orders. Furthermore, the application should be structured in packages using a layered architecture presented in the support material and should use (minimally) the following classes:

- Model classes - the data models of the application
- Business Logic classes – implement the application logic
- Presentation classes – implement the user input/output
- Data access classes - implement the access to the database

The application should allow processing commands from a text file given as argument, perform the requested operations, save the data in the database, and generate reports in pdf format. Other classes and packages can be added to implement the full functionality of the application."

2. Analiza Problemei

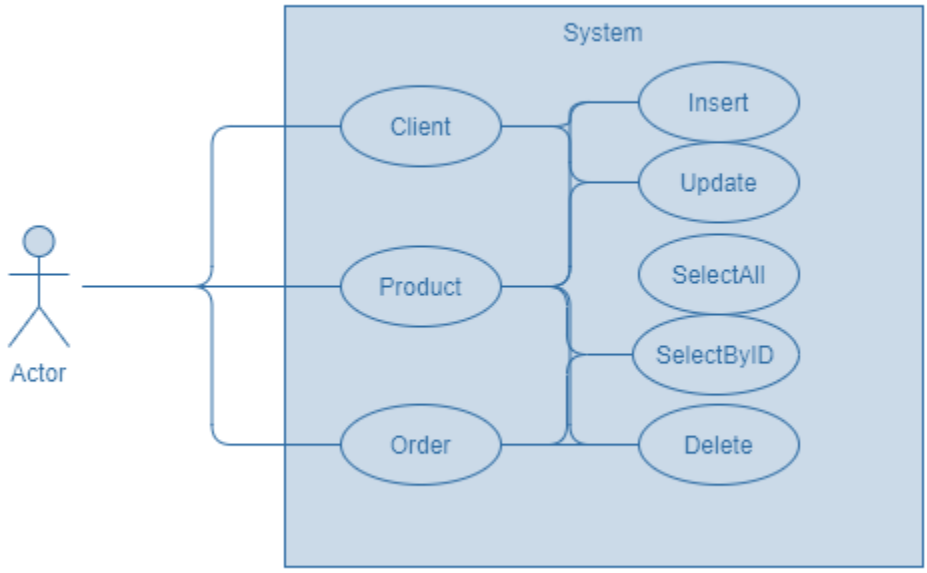
Modelarea problemei a fost facuta in mare parte dupa exemplul prezentat. A fost nevoie de a face 3 tabele fiecare cu cate un id PK. Dupa care o clasa care ne ajuta sa ne conectam la baza de date fiind apelata de fiecare daca cand vom avea nevoie sa realizam operatii pe aceasta.

1) **Model** (contine clasele: Client, Products, Order), care contine clase asociate cu tabelele din baza de date. Important atributele fiecarei clase trebuie sa sie exact ca in baza de date.
2) **DAO** (contine clasele: AbstractDAO,ClientsDAO, ProductsDAO, OrderDAO), unde sunt implementate query-urile pentru fiecare actiune din fiecare clasa. Pentru a evita duplicarea de cod am folosit o clasa abstracta care primeste un perametru generic. Iar fiecare clasa va extinde aceasta clasa abstracta. Aceasta abordare a necesitat umpic mai mult timp de gandire dar in final vom putea adauga un numar nelimitat clase fara a fi nevoie de modificari majore la partea aceasta de cod.
3) **BLL** (contine clasele: ClientsBLL, ProductsBLL, OrderBLL), care este responsabil cu definirea concreta a actiunilor pentru fiecare clasa. De aici sunt apelate metodele din Dao, si aici sunt

generate reporturile pentru fiecare clasa in parte. Aici in mmod special am generat si factura clientului in cazul unei noi comenzi.
Validators are rolul de a valida datele introduse de catre utilizator. In cadrul acestui proiect am folosit doar pentru email Dar daca un administrator ar avea nevoie de mai multe date de la clienti cum ar fi numarul de telefon ar putea valida si acest lucru.

4) **Presentation** contine clasele controller si view. Am creat mai multe clase de view pentru a avea un cod mai structurat. In primul rand am creat interfata principala in care operatorul are 3 opriuni și adică, de a realiza operatii pe unul din cele 3 tabele. Dupa alegerea unui tabel o fereastra îi va aparea pe ecran. Pentru fiecare vedere am implementat un nou view deoarece nu fiecare tabel are un numar egal de coloane astfel trebuia personalizat pentru fiecare tabel. Dupa care in controller le-am legat pe toate intre ele. Controller (care se ocupa de parsarea fisierului de intrare si extragereadatelor de intrare si apleleaza metodele din clasele aferente actiunii dorite din BLL) si start (care este main-ul proiectului, in care se apeleaza constructorul pentru Controller).
5) **Conection Factory** realizeaza conectarea utilizatorului la baza de date. Aceasta clasa ne-a fost data direct implementata de catre coordonatorii proiectului. Aceasta clasa este apelata in DAO deoarece aceasta clasa se ocupa cu interogarile asupra bazei de date

3. Cazuri de Utilizare



Operatii asupra bazei de date:

1. Insert

1. Client: “Insert client: nume prenume, adresa,email”
2. Product: “Insert product: nume, cantitate, pret”
3. Order: “Order:id produs, id client, cantitate”

“nume prenume” se refera la numele si prenumele clientului.

2. Delete

1. Client: “Delete clinet: id”
2. Product: “Delete product: id”
3. Product: “Delete order: id”

3. Edit

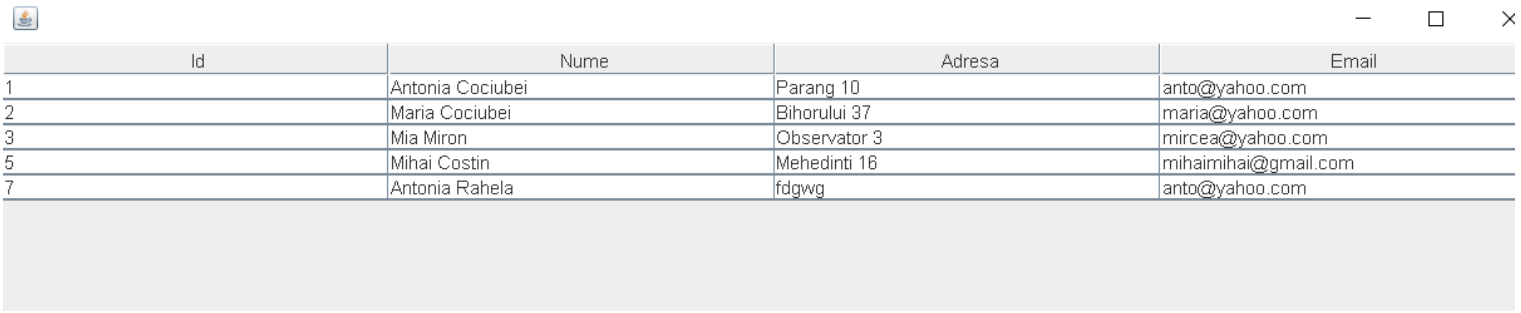
In acest caz se va introduce caracteristicile care sunt de dorit pentru a fi schimbate iar in restul locurilor se introduce null. In cazul numerelor trebuie introdus ceva pentru a evita eventualele erori de procesare.

4.View all

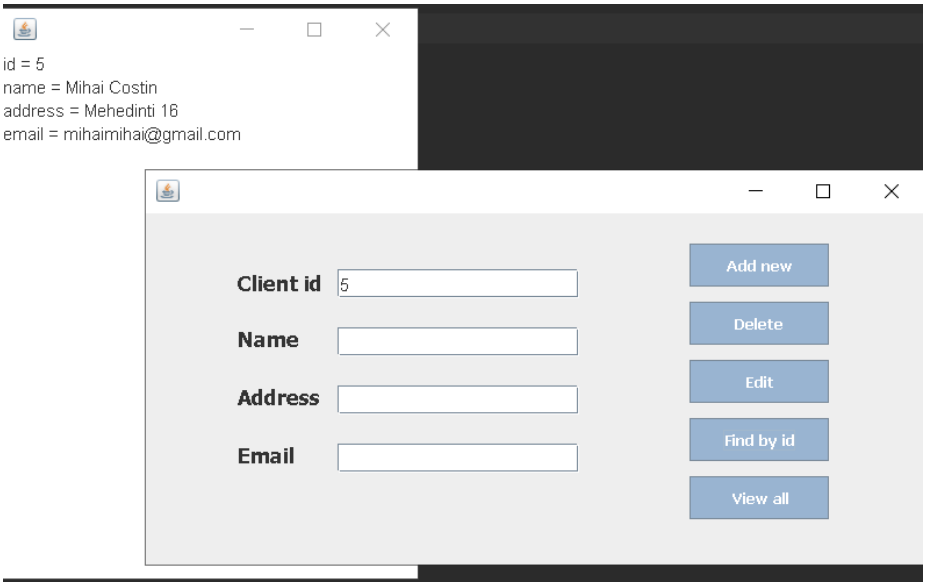
Aceasta optiune da posibilitatea administratorului sa vizualizeze toti clientii inregistrati din baza de date. Aceasta afisare se face sub forma de tabel deoarece fiind multe date o afisare ne structurata ar ingreuna mult citirea datelor. Putem vizualiza si tot stocul din acest depozit vazand produsele care urmeaza sa nu le mai avem disponibile in scurt timp. Sau la sfarsit de zi sa centralizam datele despre incasari.

5.Find by Id

Utilizand aceasta metoda putem gasi un anumit client sau un anumit produs pentru a vedea de exemplu stocul disponibil si a comanda mai multe produse in cazul unui stoc redus. Sau in cazul comenzilor putem vedea o anumita comanda daca un client are nevoie de factura ca a pierdut-o putem introduce doar numarul acesteia si va fi gasita. Intr-o implementare ulterioara s-ar putea introduce un nou tabel in care sa fie facturile fiecarui client astfel ne mai fi necesar sa tinem minte numele facturii. Sau o metoda care ar afisa toate facturile unui client.

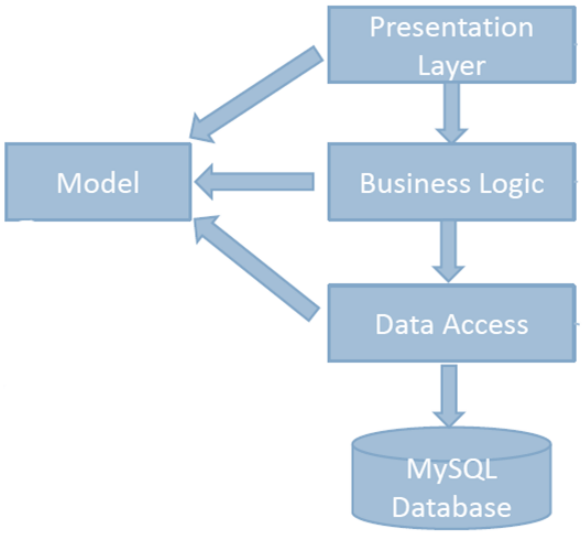


Id	Nume	Adresa	Email
1	Antonia Cociubei	Parang 10	anto@yahoo.com
2	Maria Cociubei	Bihorului 37	maria@yahoo.com
3	Mia Miron	Observator 3	mircea@yahoo.com
5	Mihai Costin	Mehedinti 16	mihaimihai@gmail.com
7	Antonia Rahela	fdgwg	anto@yahoo.com



4. Proiectare

Proiectarea claselor a fost facuta pe modelul Layerd Architecture. Astfel, fiecare pachet are un rol prestabilit de dinainte si in proiectare acestora, nu s-a sarit peste nivele. Fiecare nivel face apeluri doar catre nivelul de “sub” el.



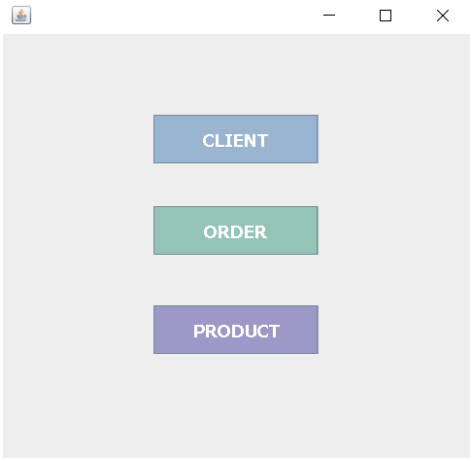
Pachetul DAO (Data Acess)

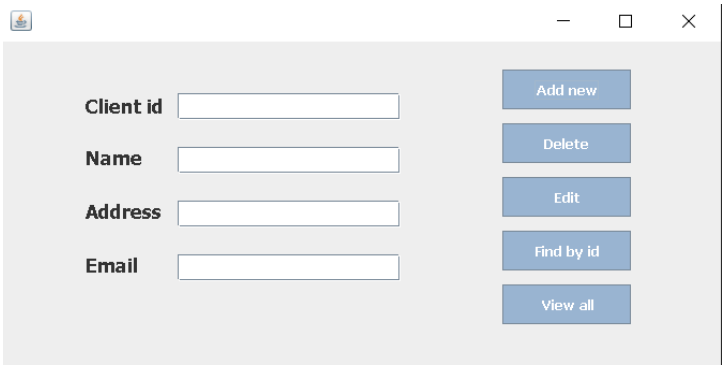
Are in componența mai multe clase și anume AbstractDAO clasa principala dupa care inca 3 clase care o exind (ClientDAO, ProductDao, OrderDAO) aceasta clasa are mai multe metode. Dar am putea sa o impartim in 2. O prima parte ne genereaza String-ul care va fi rulat pe baza de date, in cazul in care nu stim de exemplu ce id ar dori BDA sa caute am introdus „ ? “ care indica faptul ca aclo va fi inserat un element. Sau in cazul unei inserari in tabel nu stim cate coloane are respectivul tabel si cate ar dori utilizatorul sa completeze din ele asa ca acest string a fost creeat dupa ce ne-a fost dat obiectul de inserat. In partea a 2 a din clasa ne conectam la baza de date completam toate locurile marcate anterior cu „ ? “ dupa care executam instructiunea asupra bazei de date.

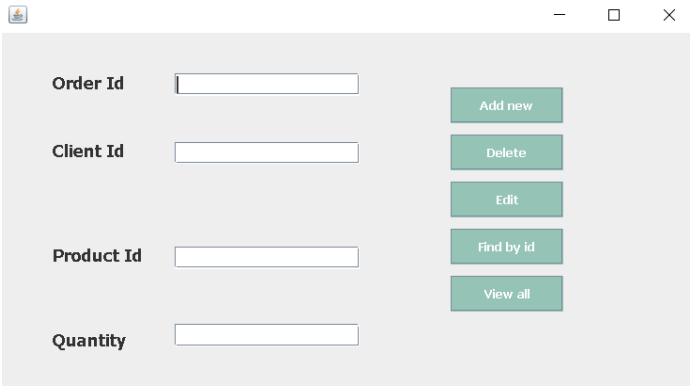
Pentru ca in final sa extindem aceasta clasa pentru orice clasa care reprezinta un tabel. Facand dezvoltarea ulterioara mult mai usoara.

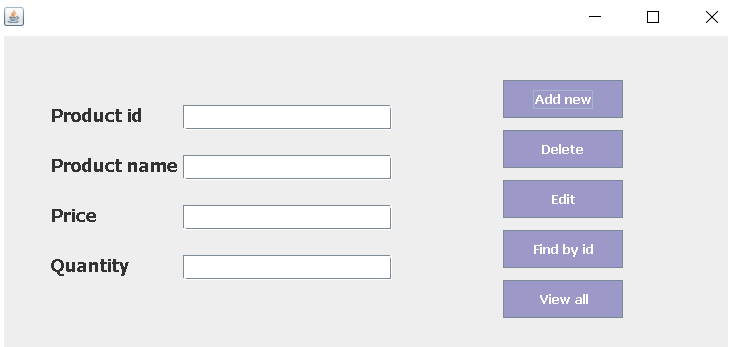
Pachetul Model

Model (contine clasele: OrderClient, Products, Order), care contine clase asociate cu tabelele din baza de date. Important atributele fiecarei clase trebuie sa sie exact ca in baza de date. Aceasta clasa are doar niste gettere si settere si 2-3 construcori.









Pachetul BLL (Business Logic)

ClientBll are 2 parametrii. O lista de validatori care poate avea un numar nelimitat in functie de implementarea acestor validatori. Si un obiect de tipul ClientDAO deoarece aceasta clasa realizeaza apelurile asupra BD.

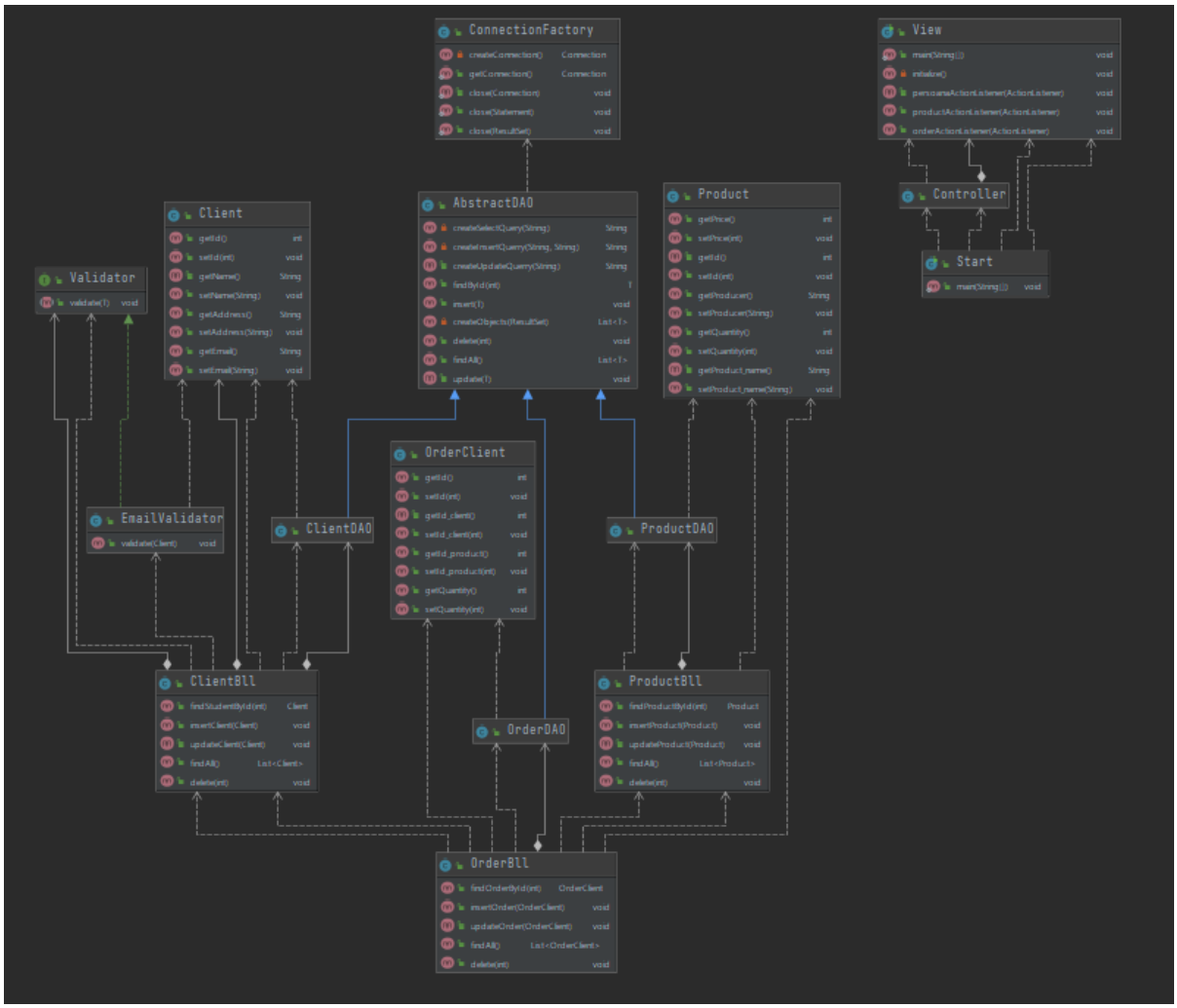
OrderBLL are 3 parametrii un File Writter un obiect OrderDAO si un int static cu rol de contor pentru factura. Aceasta clasa nu a avut nevoie de validator dar o dezvoltare ulterioara ar putea include validatori de exemplu pentru suma minima pentru care se poate vinde sau multe altele.

ProductBll are un singur parametru si anume un obiect de tipul OrderDAO care ne ajuta sa apelam metodele extinse de aceasta clasa.

De asemenea, in acest pachet, avem un subpachet numit Validators, in care avem clasele pentru realizarea validarii email-ului,in cazul de fata, cat si pentru implemnetarea ulterioara a unor noi functionalitati de valdare a datelor citite din interfata utilizator.

Pachetul presentation

Acest pachet are toate clasele referitoare la vizualizare. Un controller care realizeaza conexiunea intre toate clasele implementate in functie de optiunile alese de catre utilizator. Am creat mai multe clase de view pentru a avea un cod mai structurat. In primul rand am creat interfata principala in care operatorul are 3 opriuni și adică, de a realiza operatii pe unul din cele 3 tabele. Dupa alegerea unui tabel o fereastra îi va aparea pe ecran. Pentru fiecare vedere am implementat un nou view deoarece nu fiecare tabel are un numar egal de coloane astfel trebuia personalizat pentru fiecare tabel. Dupa care in controller le-am legat pe toate intre ele. Controller (care se ocupa de parsarea fisierului de intrare si extragereadatelor de intrare si apleleaza metodele din clasele aferente actiunii dorite din BLL) si start (care este main-ul proiectului, in care se apeleaza constructorul pentru Controller).



5. Implementare

Pachetul connection:

1. Clasa ConnectionFactory implementeaza urmatoarele metode:

   - createConnection () care realizeaza legatura la baza de data

   - getConnection () care returneaza conexiunea curenta cu baza de date

   - close (Connection) intrerupe conexiunea cu baza de date

   - close (Statement) inchide un statement primit ca parametru

Pachetul model:

1. Clasa Clients:

   - constructori getter and setter Deoarece fiecare atribut este privat

2. Clasa Products:

- constructori getter and setter

3. Clasa Order:

- constructori getter and setter

Pachetul dao:

1. Clasa ClientsDAO:

- createSelectQuery care imi genetreaza stringul pentru select. Are un parametru care reprezinta in functie de ce facem acest select.

-CreateInsertQuery care imi genetreaza stringul pentru select. Are un doi parametrii care reprezinta ce vrem sa inseram iar cel de-al 2 lea ne marcheaza cu ? locul unde va trebui introduse valorile

-createUpdateQuery este similar cu cele 2. Are si acesta doar un parametru.

-findById primeste un id dupa care se conecteaza la baza de date apeleaza createSelectQuery cu stringul id dupa care inlocuieste ? cu valoarea id ului introdus de catre utilizator.

-insert similar cu metoda de findById doar ca acesta primeste un obiect de tipul T Initial imi creez un string care reprezinta coloanele tabelului unde se doreste acest insert dar in acelas timp imi generez si un String cu ? reprezentand locul in care voi iintroduce elementele. Dupa care apelez CreateInsertQuery care imi va genera Stringul cu toate elementele. Completez spatiile marcate dupa care rulez instructiunea asupra tabelului.

-Delete realizeaza stergerea dupa un id. Stringul in realizez direct in metoda deoarece nu e foarte complicat si nu e nevoie de a crea o noua metoda pentru acesta. Dupa care similar ca la celelalte metode completez ? cu id si rulez scriptul.

-update este similar cu cea de insert doar modul in care imi generez acest string difera fiind nevoie sa scriu coloana = value

-findall similar cu findById doar ca nu mai am nevoie de un id ci voi returna o lista de obiecte care in final sa fie procesata si afisata sub forma de tabel.

6. Concluzii

Proeictul acesta a fost unul provocator. Desi am mai folosit diverse tipuri de baza de date si de platforme, aceasta a fost pentru prima data cand am lucrat cu MySql Workbanch. A fost un proiect care m-a fascinat si mi-a placut sa lucrez cu baze de date si sa descopar noi functionalitati ale limbajului Java, dar si a IDE-ului InteliJi.

Am apreciat mult ca ne-a fost ne-au fost puse la dipozitie template-uri de cod detaliate. Acest lucru mi-a usurat foarte mult intelegere modului in care se implementeza o arhitectura Layered, cat si tehnicile de reflectie.

7. Rezultate

Pentru testatra programului am folosit mai multe interogari. Aplicatia functioneaza relativ bine.

Este loc pentru multe imbunatatiri, dar este un proiect foarte practic, cu multe aplicabilitati in practica reala.

8. Bibliografie

https://dsrl.eu/courses/pt/materials/A3_Support_Presentation.pdf

https://gitlab.com/utcn_dsrl/pt-reflection-example

https://www.oreilly.com/library/view/software-architecture-patterns/9781491971437/ch01.html

https://www.jetbrains.com/help/idea/working-with-code-documentation.html#troubleshoot

https://stackoverflow.com

https://www.programiz.com/java-programming/reflection