Universitatea Tehnica din Cluj-Napoca

Facultatea de Automatica si Calculatoare

Management Application

Elev: Rosca Ion-Sergiu

Grupa: 30225

Cuprins

1.Obiectivul temei

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

3.Proiectare

4.Implementare

5.Rezultate

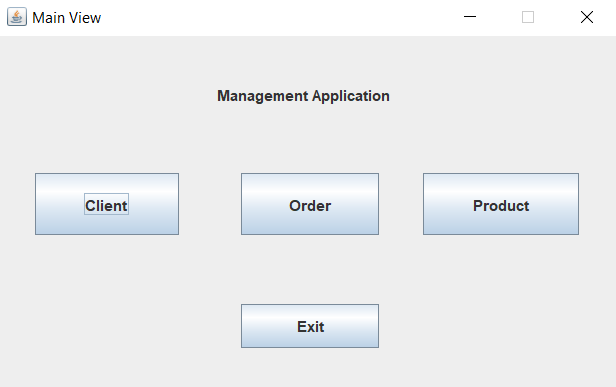
6.Concluzii

7.Bibliografie

1.Obiectivul temei

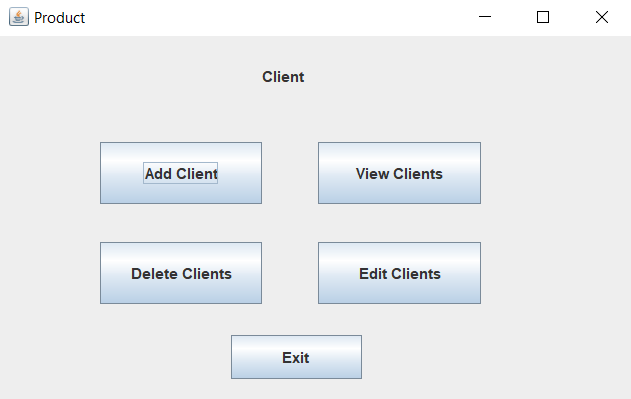
Se cere proiectarea si implementarea unei aplicatii de management pentru gestionarea rapida si eficienta a unor, produse, clienti si comenzi ale unui depozit, unde scrierea manuala poate dura mult, fiind si dificila. Clientii pot fii introdusi intr-o baza de data prin intermediul aplicatiei implementate. Utilizatorul poate si sa modifice, sa elimine sau sa vizualizeze clientii. Asemenea clientilor, si pentru produse se pot efectua operatiile de adaugare, editare, eliminare si vizualizare a produselor. Din aplicatia implementata utilizatorul poate plasa comenzi cu produsele existente in baza de date. Pentru efectuarea unei comenzi prima data se va alege clientul care va plasa comanda, dupa ce clientul a fost selectat, poate adauga produse in cos, ulterior sa finalizeze comanda. In urma fiecarei comenzi stocul fiecarui va scadea cu cantitatea produsului respectiv adaugat de client in cos, in cazul in care cantitatea este mai mare decat stocul produsului clientul nu va fi lasat sa adauge acel produs in cos, doar daca, ulterior utilizatorul va introduce o cantitate disponibila stocului de produs. In urma plasarii cu succes a unei comenzi se va genera un bon intr-un fisier nou, cu numele Comanda + numarul bonului.txt, acolo apar detaliile clientului care efectueaza comanda, produsele alaturi de pretul fiecaruia si cantiatea acestuia, si suma totala. Necompletarea incompleta a campurilor va aduce la aparitia de erori.

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

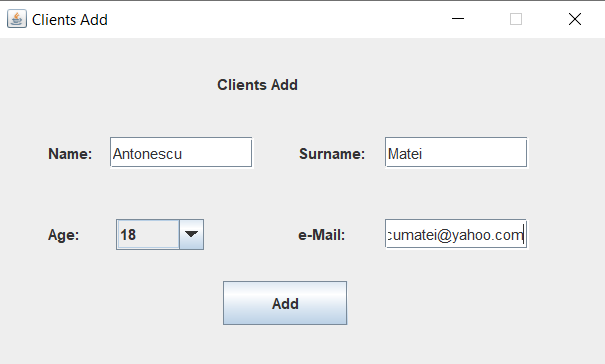
Datorita faptului ca avem operatii similare pentru clienti si produse ( adaugare, eliminare, editare, vizualizare ) am apelat la reflection techniques, construind pentru fiecare operatie doar o singura metoda pe care am apelat – o si in cazul clientilor dar si in cazlu produselor. Pentru a face acest lucru am parcurs campurile din fiecare obiect urmand sa inserez acel camp in apelul sarcinii specifice pentru baza de date. Utilizand reflection techniques am realizat o posibila dezvoltare ulterioara a aplicatiei, deoarece acele metode se pot folosi si pentru noi tabele din baza de date. Pentru implementarea inseriei unei noi linii intr-un tabel am apelat la o metoda clasica, adica cea de a insera prin intermediul unui apel al bazei de date, valorile luate din functia de inserare. Acea functie ne furnizeaza campurile necesare inserarii unui nou rand in tabel pentru orice obiect. Pentru functia de eliminare am procedat similar functiei de adaugare cu exceptia apelului al bazei de date, care difera. Pentru utilizare, am implementat o interfata usor de folosit pentru utilizator, dar, in cazu neintroducerii unor valori in toate campurile vor aparea erori. Pentru inceput interfata principala va contine 4 butoane, unul pentru clienti, unul pentru comenzi, unul pentru produse si unul pentru iesire.

Apasarea butonului Client va duce spre interfata destinata clientului, butonul Order va duce spre o lista de angajati, angajatul selectat din acea lista va putea efectua o comanda, butonul Product va duce spre interfata destinata produselor, iar prin apasarea butonului Exit aplicatia se va inchide.

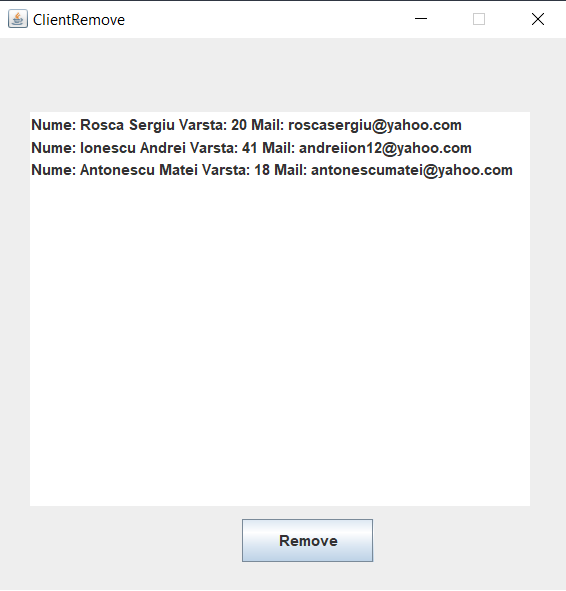
Interfata destinata clientilor va duce spre cele 4 operatii disponibile: adaugare, eliminare, editare, vizualizare.



In urma apasarii butonului Add Client utiliazatorul va fi redirectionat spre interfata destinata adaugarii unui client. In acea interfata vor aparea campuri disponibile completarii. Toate campurile trebuie completate.

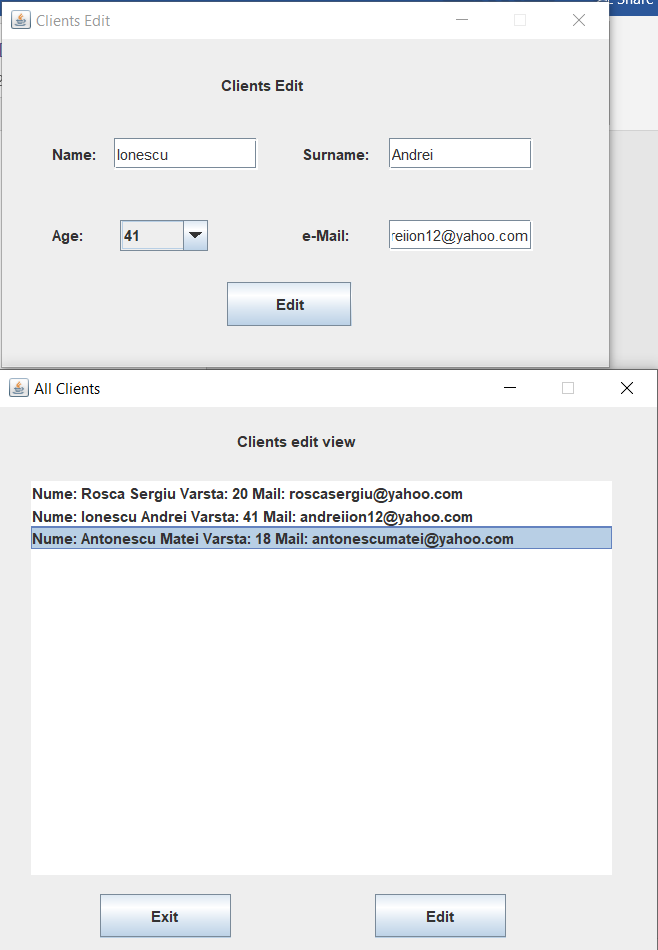


Dupa ce utilizatorul completeaza toate campurile si apasa pe butonul Add fereastra se va inchide iar clientul este introdus in baza de date. Atat numele cat si prenumele trebuie sa inceapa cu Majuscula, varsta trebuie sa fie cuprinsa intre 16 si 65 de ani iar mail – ul trebuie sa contina simbolul “@” si “.”. In cazul nerespectarii acestor reguli clientul nu va fi inserat, aceste exceptii find tratate.



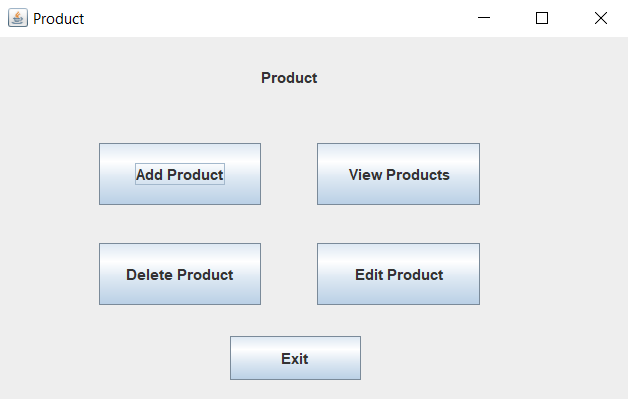
In cazul apasarii butonului Delete Client se vor afisa intr-o lista toti clientii cu detalile fiecaruia: nume, prenume, varsta si mail. Utilizatorul trebuie sa selecteze utilizatorul din lista si sa apase butonul Remove. Odata apasat butonul Remove fereastra curenta se va inchide iar clientul va fi sters din baza de date.

In cazul apasarii Edit Client va aparea o lista similara celei de mai sus. Pentru editarea unui client utilizatorul trebuie sa aleaga un client din lista, iar la apasarea butonului Edit se va deschide fereastra dedicata editarii clientilor.

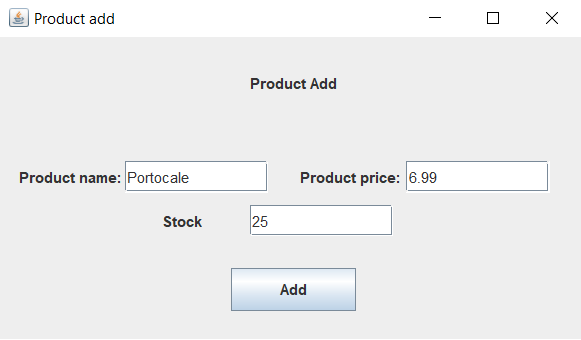


Valorile pre- introduse in campuri sunt valorile curente ale clientilor, modificarea acestora urmata de apasarea butonului edit va duce la modificarea valorilor in baza de date si la inchiderea ferestrei curente.

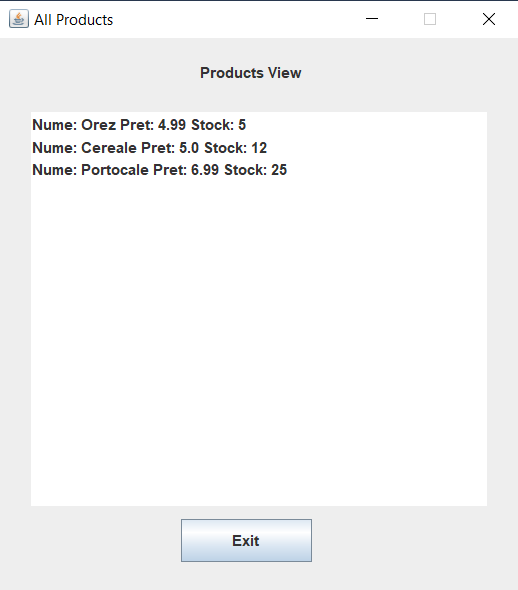
In cazul produselor se procedeaza similar obiectelor, avand aceleasi operatii: AddProduct, EditProduct, ViewAllProducts, RemoveProduct.



In cazul apasarii butonului Add Product utilizatorul va fi redirectionat spre interfata de adaugare a produsului unde va trebui sa completeze toate campurile. In cazul necompletarii acestora vor aparea erori ulterior.

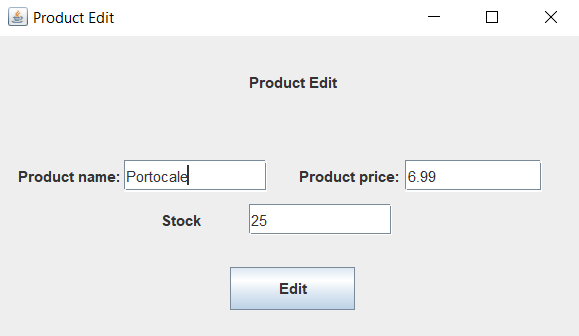


Dupa completarea campurilor si apasarea butonului Add produsul cu valorile introduse in campuri de catre utilizator se va insera in baza de date si se poate vedea imediat, la apasarea butonului View Products va aparea o lista de produse, fiecare produs fiind caracterizat prin nume produs, pret si stoc.



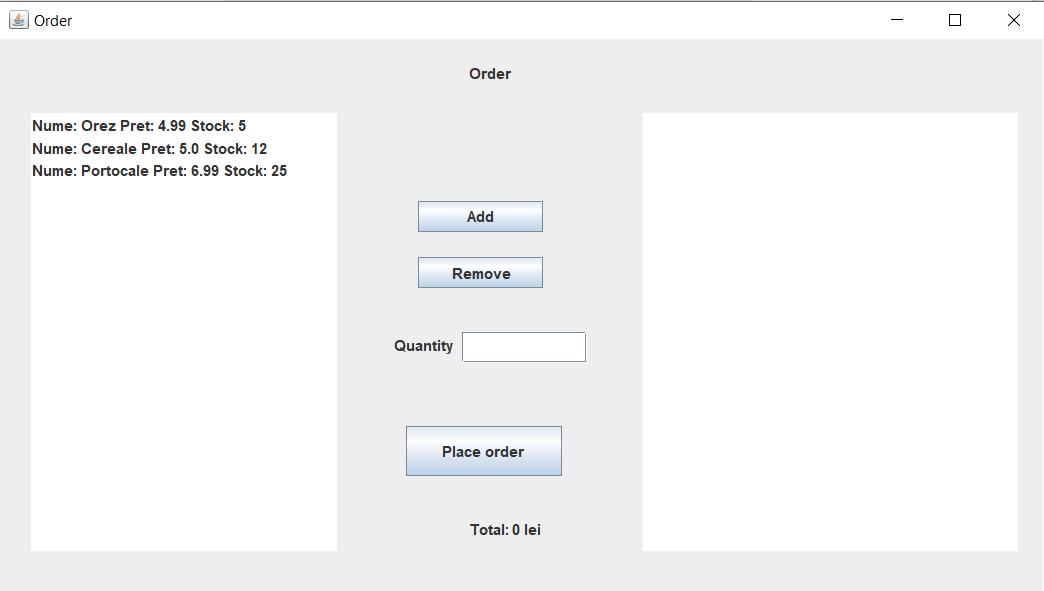
Dupa cum se poate observa produsul creat anterior din interfata destinata pentru creearea produselor a fost introdus cu succes si apare in lista cu produse. La apasarea butonului Exit fereastra se va inchide.

Pentru editarea produselor, utilizatorul va apasa butonul Edit Product care il va redirectiona spre fereastra de vizualizare a tuturor produselor. Din acea lista de produse utilizatorul poate selecta produsul dorit selectand produsul sau poate iesi din fereastra apasand butonul Exit. Odata selectat produsul dorit se va apasa butonul Edit iar fereastra dedicata editarii produselor se va deschide. In campuri valorile curente ale produselor sunt deja pre-inserate, in cazl in care utilizatorul nu doreste sa editeze toate campurile.

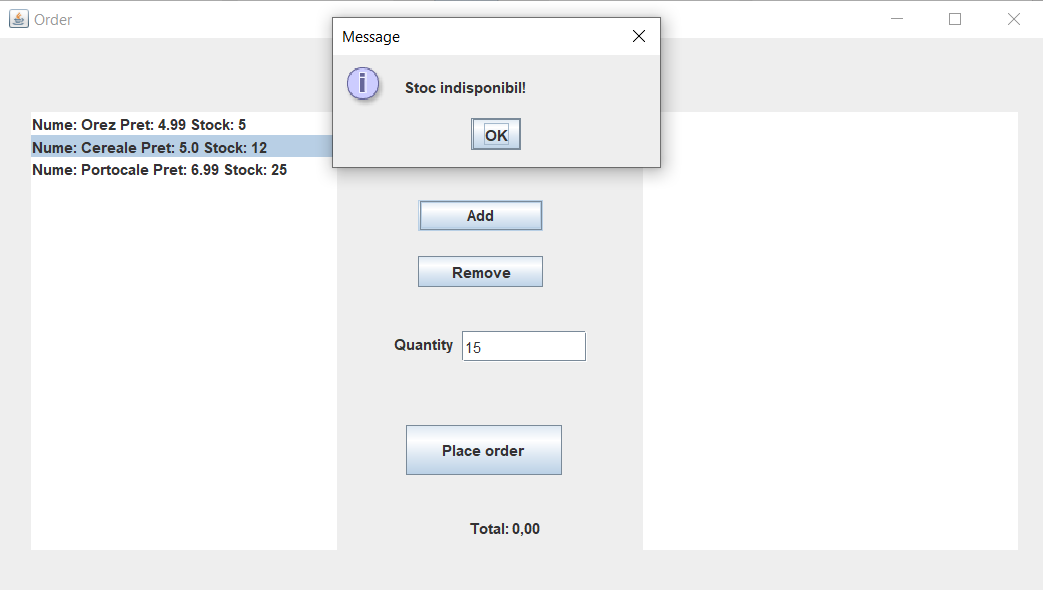


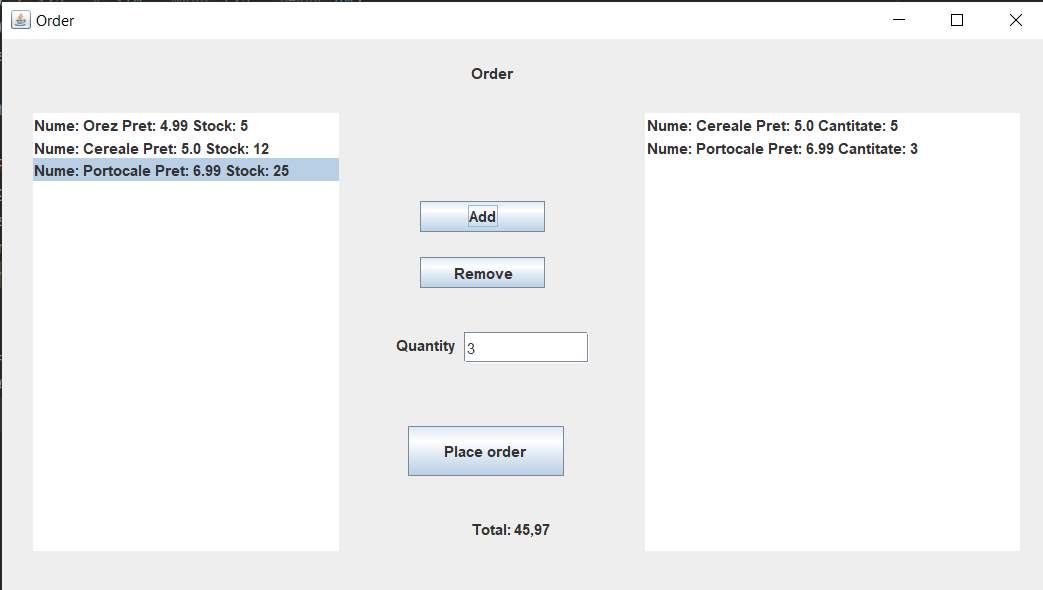
In cazul selectarii produsului din lista cu numele: Portocale, pret: 6.99, stock 25, sa deschis fereastra cu valorile produsului pre-inserate in campuri. Dupa modificarile dorite utilizatorul poate apasa butonul edit iar in baza de date, valorile produsului vor fi schimbate cu cele introduse .

Pentru plasarea unei comenzi utilizatorul va trebui sa apese butonul Order din interfata principala. In urma acestui buton utilizatorul va fi redirectionat spre lista de clienti de unde se va alege clientul care va plasa comanda.Dupa selectarea clientului se va deshide fereastra destinata plasarii comenzilor. In stanga va aparea lista de produse disponibile iar in dreapta va aparea lista produselor selectate de utilizator.

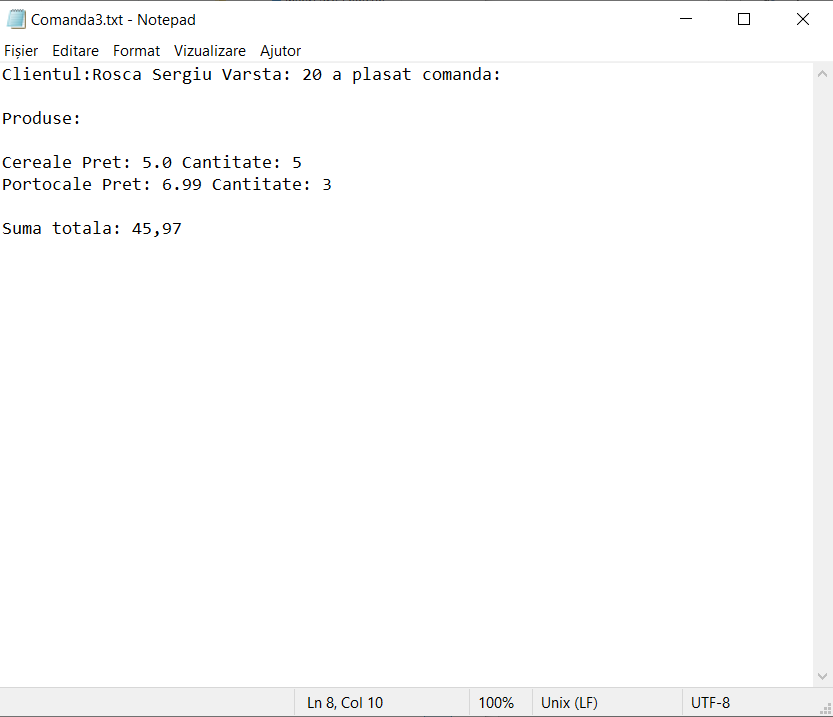


Pentru a introduce in lista pentru produsele dorite se va selecta produsul, se va introduce cantitatea si se va apasa butonul add. In cazul in care utilizatorul nu mai doreste un produs il poate scoade din lista selectand produsul si apasand butonul remove. Suma totala va fi calculata la fiecare inserare sau eliminare din lista produselor dorite. In cazul in care cantitatea selectata este mai mare decat stocul produselui, produsul u se va introduce in lista si va aparea mesajul “ Stoc insuficient”.



In cazul in care utilizatorul introduce o cantitate disponibila produsul va fi adaugat in lista produselor dorite.

Produsele selectate apar in lista produselor dorite, fiind caracterizate prin numele produsului, pretul produsului, si cantitatea introdusa a fiecarui produs. Suma se actualizeaza la fiecare operatie de adaugare si eliminare. Pentru a plasa comanda se va apasa buttonul Place order, si se va genera un fisier .txt cu numele Comanda + numarul comenzii .txt. In acel txt se vor vedea date despre client si produsele pe care le-a comandat acestea fiind caracteriazate prin nume, pret si cantitate. Tot la apasarea butonului Place order se vor modifica stocul produselor alese, astfel incat din stocul vechi se va scadea cantitatea de produs introdusa in lista.



3.Proiectare

Implementarea arhitecturii aplicatiei, s-a realizat modular fiind imartita in 6 module: Model, Business logic, Data Acces, Connection, View, Controller. In model am creea clasele pentru obiectele din aplicatia noastra, adica pentru client si produs. Atributele obiectelor sunt in aceiasi oridine si au acelasi nume ca si in baza de date pentru a putea lucra cu reflection techniques. Atributele clientului sunt numele, prenumele , mail-ul si varsta, pe cand in cazul produselor avem nume, pret si stoc. In connection realizam legatura cu baza de data, iar prin intermediul metodei getConnection() putem realiza aceasta legatura in orice clasa dorim. In view am implementat interfata cu ajutorul Java Swing, interfata prezentata mai sus. Pentru creearea metodelor am apelat la reflection techniques care a permis apelarea aceleiasi metode pentru mai multe obiecte. Metodele au fost create in AbstracDAO, din modului Data Acces. Aceste metode au fost apelate in ClientDAO si ProductDAO, fiind adaptate obiectului folosit in acel moment. Metodele apelate din AbstracDAO sunt apelate si ele la randul lor in metode in clasele ClientDAO si ProductDAO. Metodele din cele doua clase returneaza un intreg, in general id – ul obiectului asupra caruia s-a efectuat operatia. Metodele din ClientDAO si ProductDAO au fost apelate si ele la randul lor in ClientBBL si ProductBBL. In aceste doua clase se verifica metodele din ClientDAO si ProductDAO pentru a returna o valoare corecta. In cazul in care metodele nu returneaza o valoare corecta (returneaza -1) va aparea o exceptie cu un mesaj specific fiecarei metode. Spre exemplu in cazul inserarii unui client se verifica ca numele, prenumele si mail -ul sa apartina unui anumit tipar, iar varsta sa fie cuprinsa intre 16 si 65 de ani. In cazul in care metodele returneaza o valoare corecta ele vor fi apelate in interfata unde vor afisa in interfata sau modifica campuri in baza de date. Metodele din Bussines logic sunt trimise in ActionListener pentru butoanele asociate metodelor respective.

4.Implementare

Pentru implementarea claselor din Model: Client si Produs am declarar campurile, am implementat metode pentru set si get la fiecare atribut, si am implementat constructorul. In data acces am implementat clasele AbstracDAO, ClientDAO, OrderDAO, ProductDAO. In AbstracDAO am implementat metodele specifice operatiilor care se pot realiza pe obiecte, aceste metode fiin adaugare, eliminare, inserare, vizualizare, editare, si inca doua metode care returneaza campurile si valorile din coloanele tabelelor din baza de date. Pentru inserare, am apelat cele doua metode care extrag campurile si valorile acestora din baza de date, dupa care am inceput creearea interogarii specifice inserarii( insert into). Pentru continuarea creeari interogari am folosit reflection techniques pentru a extrage clasa din care apartine obiectul, clasa care are aceiasi denumire ca si tabelul din baza de date pentru a efectuarea cu usurinta a operatiilor pe obiecte fara aparitia unor erori datorate diferentei de denumire. Campurile necesare inserarii le-am inserat cu ajutorul unui for in care am extras din obiectul dat ca parametru campurile si valorile acestora. Am verificat tip ul fiecarui camp pentru a stii timpul parametrului introdus in interogare. Dupa introducerea valorilor campurilor am apelat executiea statement -ului asociat interogarii create. In urma executiei cu succes se va insera obiectul dat ca parametru in baza de date. Pentru metoda de afisare a atributului dorit am creat o metoda care primeste un obiect si un string, acel string reprezentand campul dorit extragerii din obiect. Pentru a implementa aceasta metoda am apelat metoda care returneaza campurile dintr-un tabel in functie de obiectul dat ca parametru. Creez interogarea specifica extragerii campului dorit cu inceputul specific( select ) dupa care urmeaza atributul dat ca parametru. Pentru extragerea camului din obiectul dat ca parametru parcurg campurile tabelului si le adaug interogarii. Dupa crearea interogarii, introduc valorile necesare similiar introducerii valorilor din inserare. Pentru delete procedez similiar cu inserarea si selectarea atributului, cu exceptia faptului ca la functia de delete interogarea va incepe cu “ delete”, cuvand cheie pentru a sterge. In urma executarii acestei metode se va elimina lista din tabel, linie care are aceleasi valori ca si cele din obiectul dat ca parametru. Pentru functia de view extragem din obiect campurile pe care vrem sa le vizualizam. Pentru inceput creez interogarea specifica vizualizarii element si inserez valorile necesare similar celorlalte metode.

5.Rezultate

In urma metodelor cu nume sugestiv se vor efectua operatii corespunzatoare numelui fiecarei metode. Spre exemplu pentru metoda addAccountClient se va introduce in tabelul Clients un rand nou cu valorile atributelor introduse din interfata de catre utilizator. Pentru metoda de afisare se vor afisa intr-o lista obiectele cu parametrii in afara de id – ul obiectelor. In urma executiei metodelor destinate eliminarii ibiectelor se va elimina obiectul cu atributele egale cu cele ale obiectului dat ca si parametru in functie.

6.Concluzii

In urma proiectarii si implementarii acestui proiect mi – am reimprospatat cunostiintele legate de programarea orientata pe obiecte, si am invata sa lucrez cu Reflection Techniques. Am reluat lucrul cu baza de date si cu creearea de proceduri creeate direct in IDE – ul Java.

7.Bibliografie

* <https://dsrl.eu/courses/pt/materials/A3_Support_Presentation.pdf>
* <https://dsrl.eu/courses/pt/materials/PT2021-2022_Assignment_3.pdf>
* <https://www.geeksforgeeks.org/>
* <https://stackoverflow.com/>
* <https://gitlab.com/utcn_dsrl/pt-layered-architecture>
* https://gitlab.com/utcn\_dsrl/pt-reflection-example