DOCUMENTATIE

TEMA *3*

NUME STUDENT: HIRTESCU CIPRIAN-GABRIEL
GRUPA:30225

CUPRINS

1. Obiectivul temei

Obiectivul principal al temei este de a dezvolta o aplicatie pentru management-ul produselor aflate intr-un depozit si atribuirea acestora la diferiti clienti , atribuire garantata pe baza plasarii unei comenzi. Pentru mentinerea evindentei asupra datelor cu privire la clienti , produse si comenzi se recurge la folosirea bazei de date SQL , fiind cea mai potrivita pentru acest caz. Totodata sub-obiectivele se remarca prin analiza problemei si identificarea cerintelor , design-ul aplicatiei de management si testarea acesteia. Aceste informatii vor fi detaliate in capitolele urmatoare.

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

Un prim caz de utilizare ar putea fi cel in care se doreste executarea unor operatiuni pe tabela asociata clientilor . Operatiile posibile sunt : adaugarea , stergerea , actualizarea unui client si vizualizarea tuturor clientilor . Toate acestea au ca utilizator al aplicatiei un angajat care prin introducerea in campurile asociate atributelor unui client poate modifica continutul tabelei ,daca argumentele sunt verificare cu succes, bineinteles . Altfel va fi afisat un pop-up de informare asupra evenimentului ce a condus la indisponibilitatea de executie a operatiei .
Un alt caz de utilizare , care este identic cu cel de mai sus prezentat este legat de operatiile posibile pe tabela ce tine evidenta produselor. Toate instructiunile sunt identice , diferenta majora fiind data de parametrii dati la intrare astfel incat sa se execute opratiunea cu succes.
Pentru ambele cazuri se observa ca in eventualitatea in care se introduce o valoare negativa pentru orice camp ce corespunde la un atribut al tabelei respective , operatiunea nu se va putea efectua , conducand la o revenire la un pas anterior si aparitia unui pop-up de informare asupra datelor eronate care au condus la un astfel de eveniment.
Ultimul caz de utilizare este dat de posibilitatea de a crea o conexiune intre client si produs prin introducerea evenimentului de plasare a unei comenzi. Astfel ca , angajatul cunoaste clientul caruia i se va atribui comanda , si adauga o noua inregistrare in tabelul asociat comenzilor , urmat de actualizarea stocului de marfa de tipul obiectului cumparat. In cazul in care se introduce o comanda care contine mai multe produse decat sunt in stoc , este evident ca o astfel de comanda nu se poate face , ceea ce conduce la imposibilitatea plasarii respectivei comenzi si revenirea la starea anterioara.
Mai departe , se pot defini urmatoarele tipuri de cerinte : functionale si non-functionale.

In cazul cerintelor functionale sunt prezente urmatoarele posibilitati:

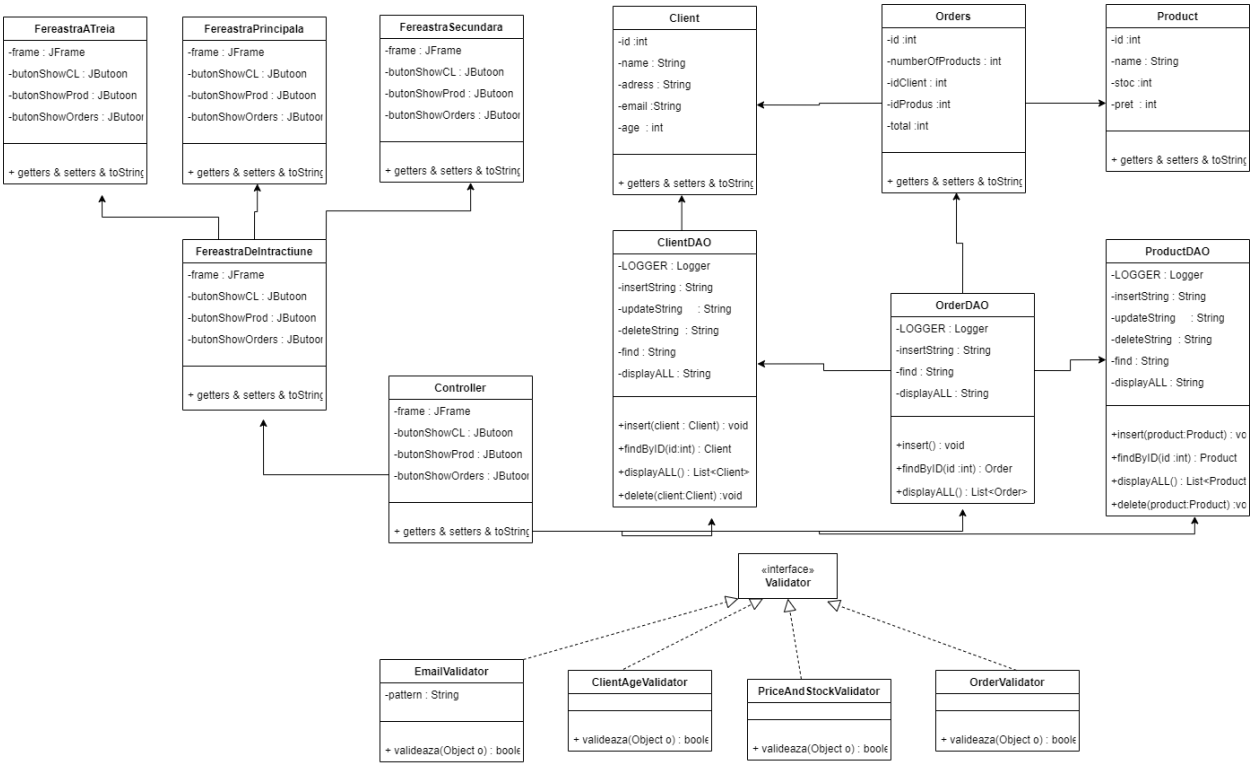
1. Aplicatia ar trebui sa permita angajatului sa adauge un nou client/produs/comanda
2. Aplicatia ar trebui sa permita angajatului sa poata executa modificari asupra unui camp din una din tabelele asociate clientului/produsului/comenzii.
3. Aplicatia ar trebui sa permita stergerea nui client/produs/comenzi.
4. Aplicatia ar trebui sa verifice corectitudinea datelor de intrare si sa evidentieze faptul ca acestea au fost introduse in mod eronat , daca este cazul.
5. Aplicatia ar trebui sa scada din stocul produsului introdus intr-o comanda , cantitatea din aceasta , in cazul in care acest lucru se poate face.

In cazul cerintelor non-functionale se observa urmatoarele:

1. Aplicatia ar trebui sa fie intuitiva si usor de folosit de catre utilizator (angajat)
2. Aplicatia ar trebui sa faca modificari la nivelul bazei de date ce tine evidenta tuturor obiectelor.
3. Aplicatia ar trebui sa execute toate operatiile in mod rapid si corect.

3. Proiectare

Proiectarea OOP a aplicatiei presupune divizarea problemei propuse in mai multe sub-obiective studiate separat in cadrul obiectelor ce mai departe sunt introduse in pachete corespunzatoare. In primul rand , pentru identificarea unui client este nevoie de un id , nume , adresa, email si varsta , ceea ce intregeste clasa Client. Pentru identificarea unui produs este necesar sa se cunoasca id-ul produsului , numele acestuia , pretul si cantitatea aflata in stoc . Iar pentru plasarea unei comenzi trebuie sa se introduca un id al comenzii , id – ul clientului care a plasat comanda , id -ul produsului asociat , cantitatea de produse de tipul respectiv ce au fost achizitionate si totalul de plata pentru comanda curent plasata . Toate aceste detalii sunt absolut necesare in vederea construirii unei astfel de aplicatii .



4. Implementare

Partea de interfata este constituita dintr-un numar de patru ferestre ce apar succesiv o data ce sunt apelate. Exista o fereastra principala de interactiune cu utilizatorul , cu ajutorul careia se poate face schimbarea de la o fereastra la alta astfel incat angajatul sa isi poata face atributiile . Pe langa aceasta exista cate o fereastra pentru fiecare dintre cele trei tabele aflate in baza de date asociate clientulilor , produselor si comenzilor. In fereastra corespunzatoare clientilor exista posibilitatea inserarii de date in casutele destinate acestui scop , fiecare avand de asemenea o eticheta astfel incat sa se stie clar unde se vor duce informatiile introduse , iar mai departe la apasarea unuia din butoanele prezenta in interfata datele vor fi transmise mai departe , validate , iar in caz afirmativ introduse in baza de date.
Aceleasi operatii sunt prezente si pentru produse si comenzi . In fereastra comenzi pentru a fi mult mai vizibila modificarea pe tabela de produse o data ce o comanda a fost introdusa cu succes , s-au adaugat inca doua locatii de vizualiare a produselor si a clientilor , astfel facilitandu-se timpul in care se vor face operatiunile de inserare .
Pentru partea de inserare in tablele memorate in baza de date si modelare a acestora, clasele din pachetul DataAcces intrunesc toate aceste atributii.
Detalierea este urmatoarea:

1. ClientDAO va efectua operatiile de inserare stergere si actualizare asupra datelor preluate din partea anterioara de interactiune cu utilizatorul. Pe langa toate acestea exista inca doua functionalitati si anume vizualizarea clientului cu un anumit id , din baza de date si multimea de clienti din baza de date.
2. ProductDAO va realiza operatii de acelasi tip , de data aceasta pe tabela de produse , functionalitatea este similara.
3. OrdersDAO se va ocupa de operatii de forma : preluarea multimii de comenzi , preluarea unei comenzi cu un id cunoscut si adaugare a unei comenzi de la un client x la un produs y , calculand totodata si totalul de plata pentru comanda respectiva.

Pentru partea de logica a operatiilor , clasele din pachetul BusinessLogic sunt responsabile pentru validarea informatiilor primite de la utilizator si trimiterea spre executie a acestora in vederea obtinerii unui rezultat ce va fi intors mai departe la utilizator , in caz de succes , actualizare a bazei de date , in caz de esec , exceptie urmata de un pop-up de informare.
Mai detaliat, in cazul clasei ClientBLL avem la dispozitie o serie de validatori stocata intr-o lista de variabile generice , ce vor verifica datele si mai departe vor compara continutul din fiecare obiect cu valorile prestabilite , valide pentru un posibil client .Validatorii sunt urmatorii: validator pentru email : foloseste un pattern in care daca string-ul introdus contine un numar de caractere urmat de un numar de cifre (0,*) la care se adauga caracterul special @ si extensia . sir_de_caractere ; un validator de varsta in care se mentioneaza ca un client cu varsta cuprinsa intre 14 si 70 de ani este valid . Pentru clasa ProductBLL , de asemenea va contine un validator pentru verificarea evenimentului daca utilizatorul a introdus un pret si o cantitate aflata in stoc , pozitive , caz in care produsul va putea fi introdus in tabel. Pentru clasa OrderBLL se observa prezenta unui validator ce reprezinta o verificare a cantitatii introduse de utilizator , mai departe urmand sa se incerce introducerea noilor date in tabel.

Se mentioneaza faptul ca toate informatiile primite din interfata vor fi validate de catre metodele ce implementeaza interfata validator , urmand ca o data ce au fost validate sa fie introduse in baza de date corespunzatoare tabelei respective . De exemplu pentru product inserarea se valideaza o data ce are nume id , alaturi de o cantitate si un pret numere pozitive , altfel se va arunca o exceptie care sa informeze utlizatorul ca datele introduse nu sunt corespunzatoare , iar in cazul in care ID – ul produsului se repeta va aparea o exceptie care sa mentioneze acest lucru. Pentru client analog se vor prelua informatiile din partea utilizatorului din casutele corespunzatoare locatiilor din tabele unde se vor plasa informatiile . Validarea se bazeaza pe trei pasi : primul ar fi ca acesta sa aiba o varsta cuprinsa intre 14-70 de ani, un email ce se valideaza pe baza pattern-ului , id -ul unic si un nume.

5. Rezultate

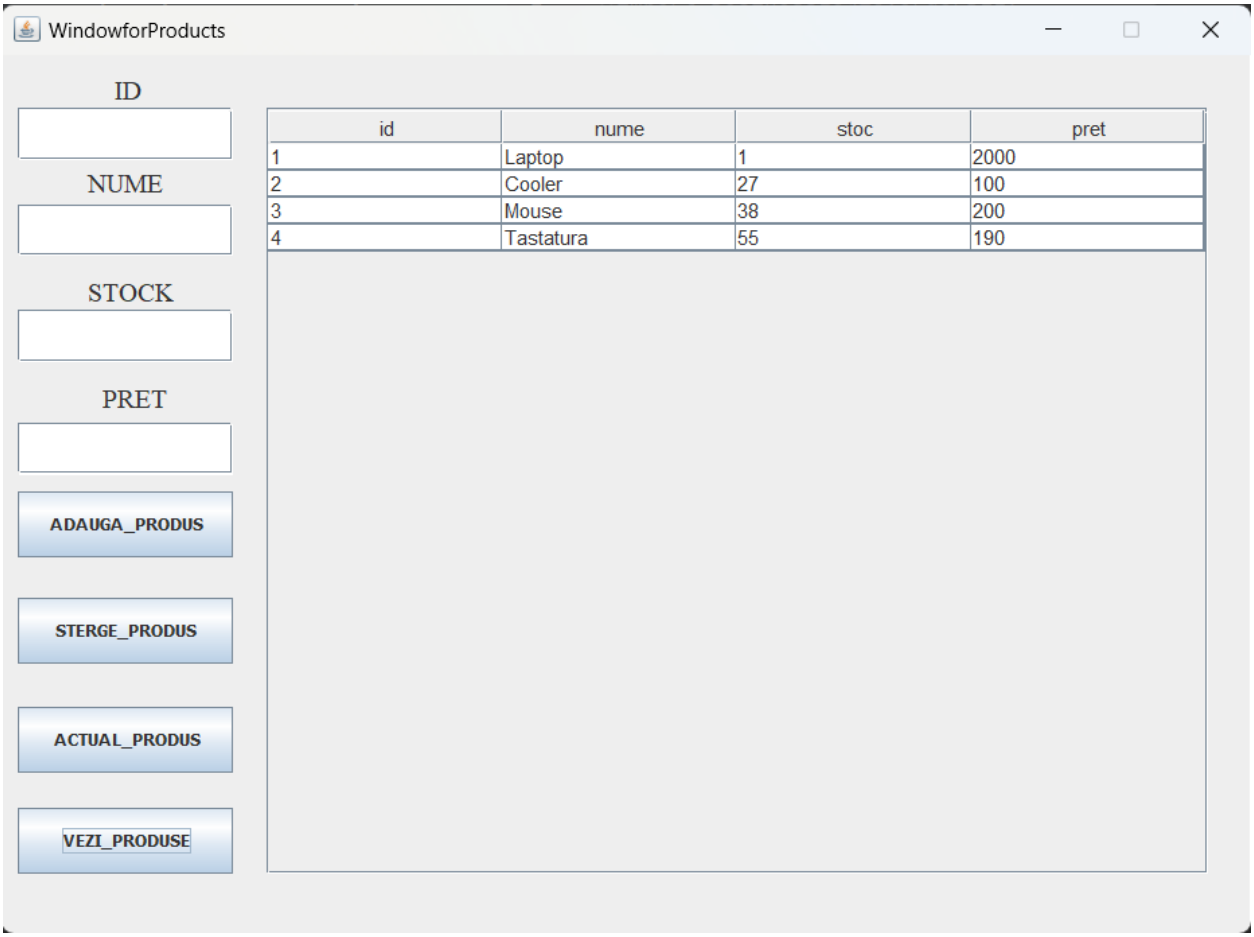
In urma testarii cu Junit s-a constatat ca operatiunile de inserare stergere actualizare si preluare date din tabelel se executa in mod corect , conform cerintei. Iar metodele ce arunca exceptii sunt invalidate , fapt asteptat. Folosind metodele deja validate s-a putut face apel la rezultatele obtinute de la acestea in vederea verificarii altora , de exemplu in cazul verificarii evenimentului daca un produs se afla in tabel . Inainte de acest lucru trebuie inserat obiectul in tabel , dupa care sa se caute operatii dezvoltate de findID si de inserProduct , anterior validate , au putut face posibila validarea metodei curente.
Testarea a constat in verificarea metodelor din clasele de data acces : ClientDAO, ProductDAO, OrderDAO pentru inserare s-a considerat ca se va folosi validarea .

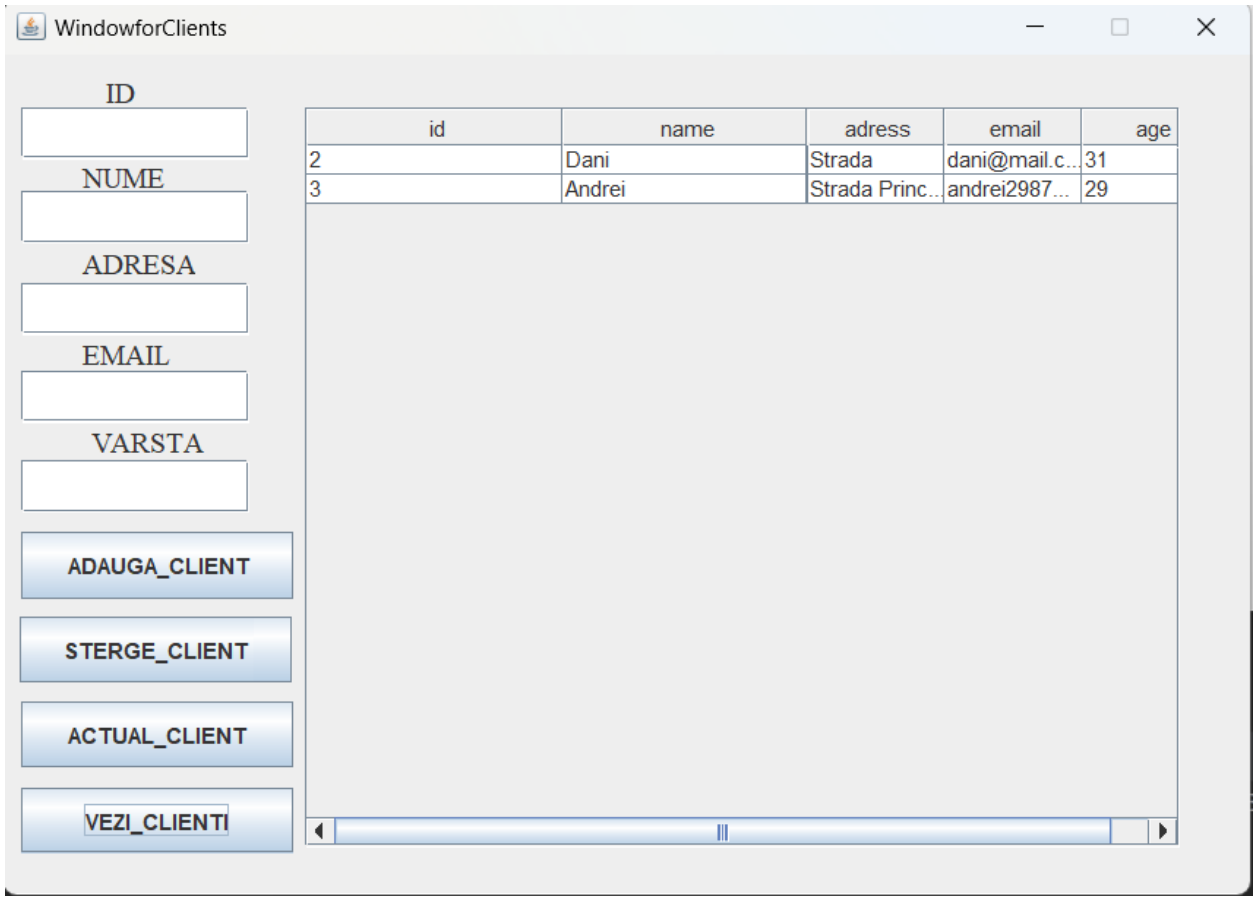
6. Concluzii

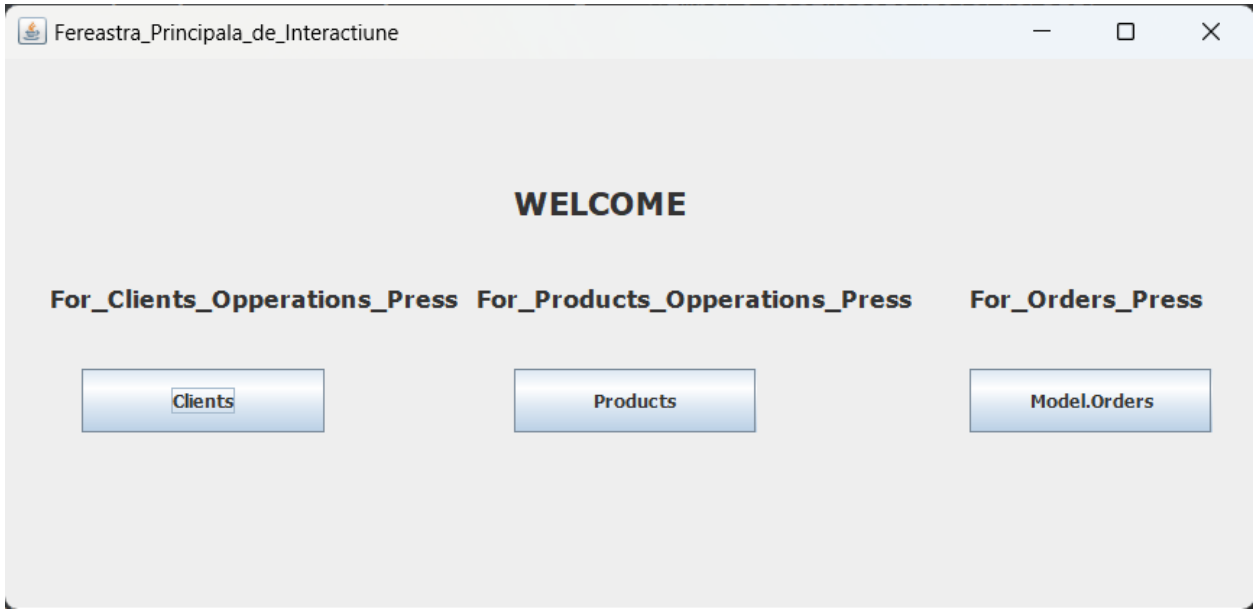
In urma rezolvarii temei am invatat despre Layered Architecture , si generice , restul informatiilor si resurselor necesare implementarii solutiei cum ar fi cunoasterea lucrului cu bazele de date si executarea unor interogari in vederea obtinerii de informatii relevante .
Ca si dezvoltari ulterioare se poate observa ca pe partea de generice o metoda mai eficienta ar fi crearae unei clase abstracte care sa poata executa operatiunile comune de tip insert select update si delete , diferenta fiind numarul de argumente ce vor fi preluate.

7. Bibliografie


Place_an_Order
Produse
id nume stoc pret
1 Laptop 1 2000
2 Cooler 27 100
3 Mouse 38 200
4 Tastatura 55 190
Clienti
id name adress email age
2 Dani Strada dani@mail.com 31
3 Andrei Strada Principala andrei2987@mail.com 29
Comenzi
id numberOfProducts idClient idProdus total
1 1 2 1 2000
2 1 2 3 200
Cantitate
Refresh
PLASEAZA_COMANDA
Checkout


WindowforProducts
ID
NUME
STOCK
PRET
ADAUGA_PRODUS
STERGE_PRODUS
ACTUAL_PRODUS
VEZI_PRODUSE
id
nume
stoc
pret
1
Laptop
1
2000
2
Cooler
27
100
3
Mouse
38
200
4
Tastatura
55
190


WindowforClients
ID
NUME
ADRESA
EMAIL
VARSTA
ADAUGA_CLIENT
STERGE_CLIENT
ACTUAL_CLIENT
VEZI_CLIENTI
id
name
adress
email
age
2
Dani
Strada
dani@mail.c...
31
3
Andrei
Strada Princ...
andrei2987...
29


Fereastra_Principala_de_Interactiune
WELCOME
For_Clients_Opperations_Press
For_Products_Opperations_Press
For_Orders_Press
Clients
Products
Model.Orders