

**UNIVERSITATEA TEHNICĂ CLUJ-
NAPOCA**
AUTOMATICĂ ȘI CALCULATOARE, AN II



Lucrare laborator – Assignment III
Aplicatie pentru procesarea comenzilor dintr-un depozit

Profesor curs: Prof. Dr. Ing. Cristina Pop
Profesor laborator: Teodor Petrican

Student: Jitaru Andrei
Grupa: 30229

CUPRINS

1. Introducere – Obiectivul temei
2. Analiza problemei
 - a. Asumpții
 - b. Modelare
 - c. Scenarii
 - d. Cazuri de utilizare
 - e. Erori
3. Proiectare
 - a. Diagrame UML
 - b. Diagrama bazei de date
 - c. Proiectare clase
 - d. Algoritmi
 - e. Pachete
 - f. Interfata Utilizator
4. Implementare
5. Rezultate
6. Concluzii și dezvoltări ulterioare

1. Introducere – obiectivul proiectului

Obiectivul acestui proiect este de a simula cat mai veridic interactiunea unui client cu o aplicatie destinata cumpararii de produse dintr-un depozit universal. In alte cuvinte, s-a dorit implementarea in limbaj de programare Java a unui model de aplicatie care se permita vizualizarea produselor din stocul depozitului, alegerea unuia dintre acestea si plasarea respectivului produs intr-o comanda finalizata prin returnarea unei chitante care sa ilustreze detalii despre comanda efectuata (numele cumparatorului, pretul comenzii, produsul cumparat). Toate aceste cerinte au fost date spre rezolvare cu scopul intelegerii anumitor principii legate de limbajul de programare Java dintre care amintim conceptul de reflexion, utilizarea bazelor de date intr-o aplicatie de tip Java, impachetarea aplicatiei conform arhitecturii 3-tier, etc.

Toate aceste aspecte enumerate anterior au fost verificate prin implementarea unei aplicatii simple care sa surprinda principalele cerinte cerute in cadrul temei. Rezolvarea temei a fost realizata sub forma unei aplicatii de tip "magazin online" care incapsuleaza mai multe concepte rudimentare. Aspectul temei se remarca prin accesibilitate si printr-un sistem de interactiune usor de utilizat care sa permita utilizatorului diverse operatii implementate la nivelul magaziei online (specificatii mult mai detaliate legate de configuratia grafica a proiectului vor fi ilustrate in capitolul 3, sub-capitolul g). Buna utilizare a aplicatiei poate fi obtinuta numai si numai prin utilizarea atat a mouse-ului cat si a tastaturii. In capitolul 2 vor fi prezentate mai multe concepte legate de interactiunea adecvata a utilizatorului cu proiectul dar si diverse principii de utilizare a acesteia. Cele amintite mai sus vor fi realizate cu scopul de a demonstra in special corecta intelegere a temei si cerintelor. Tema ofera anumite cerinte de implementare obligatorii (utilizarea conceptului de reflexion, a minim trei tabele dar si a impachetarii de tip 3-tier architecture) care au fost imbinate cu o abordare simpla si usor de inteles. Aceasta abordare va fi descrisa mai pe larg in cadrul capitolului 4. Corectitudinea operatiilor dar si demonstrarea rezultatelor vor fi prezentate succint in capitolele 5 si 6.

2. Analiza problemei

Proiectul, care în esență reprezintă o aplicație de tip **OrderManagement** realizată preponderent în scopul procesării de comenzi realizate la nivelul unui depozit, sugerează un proces complex care necesită atenție la fiecare pas. O astfel de abordare este absolut necesară deoarece se dorește înțelegerea conceptelor legate de implementarea temei. Fiecare operație executată de către aplicație este unică în felul ei și prezintă diferite caracteristici unice acompaniate de diverse excepții specifice, acestea trebuind tratate în parte cu scopul obținerii unei funcționări impecabile a programului. În anumite momente ale execuției programului vor fi analizate anumite condiții legate de introducerea corectă a datelor de intrare. Pe lângă acestea, în funcție de metoda de logare utilizatorul va obține diverse permisiuni strict legate de conceptul de utilizare.

a. Asumptii

Fiecare rulare a aplicației debutează prin apariția unei casute care are scopul de a prezenta persoanei care utilizează programul anumite restricții care îi sunt impuse de către acesta dar și o serie de pași necesari utilizării corecte a aplicației. Astfel, în momentul logării utilizatorului vom presupune că respectivul cunoaște ce atribute are (cu alte cuvinte este conștient dacă este un simplu utilizator - customer – sau este un admin – care are mult mai multe atribute). De asemenea, în cazul îndeplinirii asumției mai sus menționate, vom continua prin a presupune că utilizatorul își cunoaște username-ul și password-ul astfel încât să fie capabil de a se conecta cu succes la aplicație și de a o putea utiliza în conformitate cu atribuțiile sale. Neîndeplinirea acestor presupuneri conduce, desigur, la incapacitatea de a utiliza aplicația. Vom presupune concomitent, că odată cu buna logare a utilizatorului aceste cunoaște diferite aspecte legate de funcționarea programului (care vor fi dobândite evident prin citirea documentației – de exemplu este conștient de faptul că crearea unei comenzi poate fi realizată prin intermediul cunoașterii id-ului produsului, care la rândul lui trebuie cautat). Subcapitolele următoare vor surprinde însă diferite "sanțiuni" care vin la pachet cu nenuoasterea principiilor de utilizare.

b. Modelare

Cerinta temei si anume de a se realiza un sistem capabil de procesarea comenzilor dintr-un depozit poate fi implementata din diferite moduri, fiecare cu eficienta sa. Prima si cea mai banala metoda de implementare care ar putea fi abordata este tratarea tabelor care alcatuiesc baza de date necesara implementarii depozitului (Customer, Order, Product) in mod independent. Aceasta metoda presupune descrierea comportamentului fiecarei tabele individual, prin metode caracteristice care sa corespunda fiecareia in parte (de exemplu pentru toate cele 3 tabele ar fi necesara implementarea unei metode care sa produca adaugarea in baza de date al unui tip de obiect caracteristic tabelei respective). Desigur, rezolvarea amintita mai sus nu este dintre cele mai recomandate. Aceasta modelare a problemei prezinta diverse dezavantaje dintre care cele mai importante sunt legate de faptul ca produce o implementare greoaie, muncitoreasca care sub niciun caz nu ar putea ajuta viitorii programatori care ar dori sa dezvolte in continuare aplicatia.

Daca privim problema din punct de vedere al efortului si al resurselor utilizate atunci este evident faptul ca ar trebui sa cautam o solutie care sa permita accesul usor la orice tabela, intr-un mod cat mai general si care sa dispuna de posibilitati viitoare de dezvoltare (acompaniate de simplitate). In acest sens tehnicile de reflexie si caracterul generic al claselor ne vine in ajutor. In acest sens prin utilizarea atata a reflexiei cat si a genericelor s-a propus crearea unei clase care sa incapsule toate operatiile care pot fi afectuate la nivelul unei tabele (care ne sunt de folos, in mare parte): adaugarea intr-o tabela, stergerea dintr-o tabela, cautarea intr-o tabela, actualizarea unei tabele si nu in ultimul rand crearea unei tabele. Principiile principale legate de implementarea acestor concepte vor fi dezvoltate mult mai pe larg in capitolele care urmeaza.

c. Scenarii

O serie de etape trebuie urmate pentru ca aplicatie sa fie capabila de a returna rezultate clare si corecte. Totalitatea acestor etape dau nastere unei serii de scenarii care definesc functionalitatea programului. In acest sens, pentru a obtine un anumit rezultat, diferite evenimente trebuia sa aiba loc si o multitudine de pasi sunt necesari a fi urmati.

Vom presupune ca datele de intrare vor fi introduse in JTextField-uri. Lasarea necompletata a acestora nu este o eroare insa aplicatia nu va raspunde

sub niciun fel odata ce identifica input-uri nule.

Pasi de utilizare ai aplicatiei:

- Aparitia ferestrei de logare;
- Introducerea numelui de utilizator si a parolei in campurile dedicate (cele 2 campuri au denumiri sugestive – Username - Password);
- Alegerea modalitatii de logare din JTextField-ul "Login As" (fie ca si client, fie ca si admin);
- Optional, prin selectarea JCheckBox-ului "Show Password" se poate produce vizualizarea parolei
- In cazul logarii ca si "Client":
 - Alegerea operatiei care se doreste a se efectua din meniul din partea stanga a interfetei principale dedicate clientului logat;
 - In cazul selectarii optiunii "View Profile" ;
 - Afisarea profilului utilizatorului(id, nume, email, username si parola) ;
 - In cazul selectarii optiunii "Edit Profile"
 - Introducerea valorilor in JTextField-urile corespunzatoare datelor ce doresc a fi modificate
 - Actualizarea profilului utilizatorului si implicit a liniei din tabelul "Customers" corespunzatoare acestuia prin apasarea butonului "Update Customer"
 - In cazul selectarii optiunii "Create Order"
 - Introducerea valorilor in JTextField-urile corespunzatoare datelor ce definesc o comanda (o comanda poate fi realizata doar pentru utilizator = utilizatorul curent nu poate crea comenzi pentru alti clienti)
 - Crearea comenzii prin apasarea butonului "Add Order"
 - In cazul selectarii optiunii "View Products"
 - Afisarea tablei cu toate produsele disponibile care pot fi

cumparate de catre utilizator

- In cazul selectarii optiunii "Find Product"
 - Introducerea id-ului corespunzator produsului cautat
 - Afisarea produsului intr-un tabel prin apasarea butonului "Find"
- In cazul selectarii optiunii "View Orders"
 - Afisarea tabelului cu toate comenzile realizate de catre utilizatorul care este logat in aplicatie
- In cazul logarii ca si "Admin"
 - Alegerea operatiei care se doreste a se efectua din meniul din partea stanga a interfetei principale dedicate admin-ului logat;
 - In cazul selectarii optiunii "Customers";
 - Alegerea operatiei care se doreste a se efectua din meniul din partea stanga a interfetei dedicate lucrului cu clientii;
 - ◇ In cazul selectarii optiunii "Add Customer"
 - Introducerea valorilor in JTextField-urile corespunzatoare datelor ce vor defini un nou client
 - Crearea si adaugarea in baza de date a noului client prin apasarea butonului "Add Client"
 - ◇ In cazul selectarii optiunii "Update Customer"
 - Introducerea valorilor in JTextField-urile corespunzatoare date ce doresc a fi modificate
 - Actualizarea profilului clientului ca carui id a fost introdus in JTextField-ul "id" si implicit a liniei din tabelul "Customers" corespunzatoare acestuia prin apasarea butonului "Update"

- Customer"
- ◇ In cazul selectarii optiunii "Find Customer"
 - Introducerea id-ului corespunzator clientului cautat
 - Afisarea clientului intr-u tabel prin apasarea butonului "Find"
- ◇ In cazul selectarii optiunii "Delete Customer"
 - Introducerea id-ului corespunzator clientului care se doreste a fi sters
 - Stergerea clientului odata cu apasarea butonului "Delete"
- ◇ In cazul selectarii optiunii "View Customers"
 - Afisarea tabelei cu toti clientii din tabele "Customers"
- ◇ In cazul selectarii optiunii "Back"
 - Revenirea la meniul anterior
- In cazul selectarii optiunii "Products";
 - Alegerea operatiei care se doreste a se efectua din meniul din partea stanga a interfetei dedicate lucrului cu produsele;
 - ◇ In cazul selectarii optiunii "Add Product"
 - Introducerea valorilor in JTextField-urile corespunzatoare datelor ce vor defini un nou produs
 - Crearea si adaugarea in baza de date a noului produs prin apasarea butonului "Add Product"
 - ◇ In cazul selectarii optiunii "Update Product"
 - Introducerea valorilor in JTextField-urile corespunzatoare date ce doresc a fi modificate

- Actualizarea profilului produsului a carui id a fost introdus in JTextField-ul "id" si implicit a liniei din tabelul "Products" corespunzatoare acestuia prin apasarea butonului "Update Product"
- ◇ In cazul selectarii optiunii "Find Product"
 - Introducerea id-ului corespunzator produsului cautat
 - Afisarea produsului intr-un tabel prin apasarea butonului "Find"
- ◇ In cazul selectarii optiunii "Delete Product"
 - Introducerea id-ului corespunzator produsului care se doreste a fi sters
 - Stergerea produsului odata cu apasarea butonului "Delete"
- ◇ In cazul selectarii optiunii "View Products"
 - Afisarea tabelii cu toate produsele din tabela "Products"
- ◇ In cazul selectarii optiunii "Back"
 - Revenirea la meniul anterior
- In cazul selectarii optiunii "Create Order"
 - Introducerea valorilor in JTextField-urile corespunzatoare datelor ce definesc o comanda (comanda poate fi pusa pentru orice client)
 - Crearea comenzii prin apasarea butonului "Add Order"
- In cazul selectarii optiunii "View Orders"
 - Afisarea tabelii cu toate comenzile efectuate

d. Cazuri de utilizare

Luand in considerare faptul ca acest proiect realizeaza managementul unui depozit, la un nivel destul de ridicat, ar putea fi propus spre utilizare chiar in acest domeniu. Din moment ce dispune de o interfata grafica relativ simpla, cu o utilizare la fel de banala, se poate considera ca ar putea fi folosita fara indoiala atat de clientii cat si de managerii unui depozit.

e. Erori

Anumite evenimente pot determina aparitia unor erori. Aparitia acestor evenimente produc afisarea unei casute de dialog in care este prezentata eroarea pentru a fi mai facil de identificat.

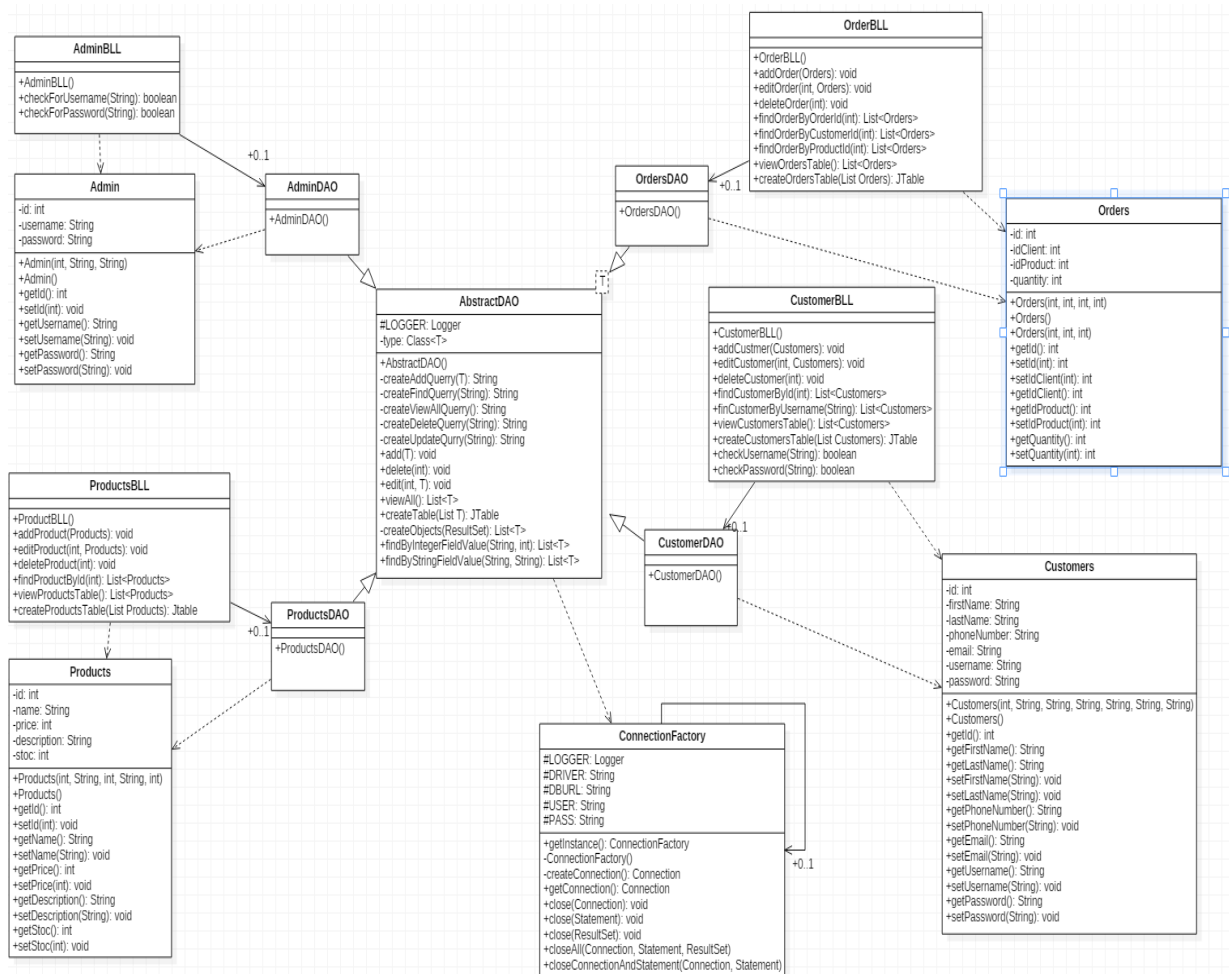
Cazuri care pot conduce la aparitia erorilor:

- Daca ne aflam in fereastra de LogIn si nu introducem valori corespunzatoare in JTextField-urile "Username" si "Password" (daca nu introducem un nume de utilizator valid caruia sa-i fie asociata, de asemenea, o parola valida);
- Daca ne aflam in fereastra de LogIn si nu alegem metoda de logare corespunzatoare numelui de utilizator si parolei introduse (daca introducem un nume de utilizator si parola corespunzatoare unui client dar ne logam ca si "Admin");
- Daca dorim sa realizam o comanda insa stocul produsului comandat este insuficient
- Daca suntem logati ca si "Client" si dorim sa cream o comanda pe id-ul altui client

3. Proiectare

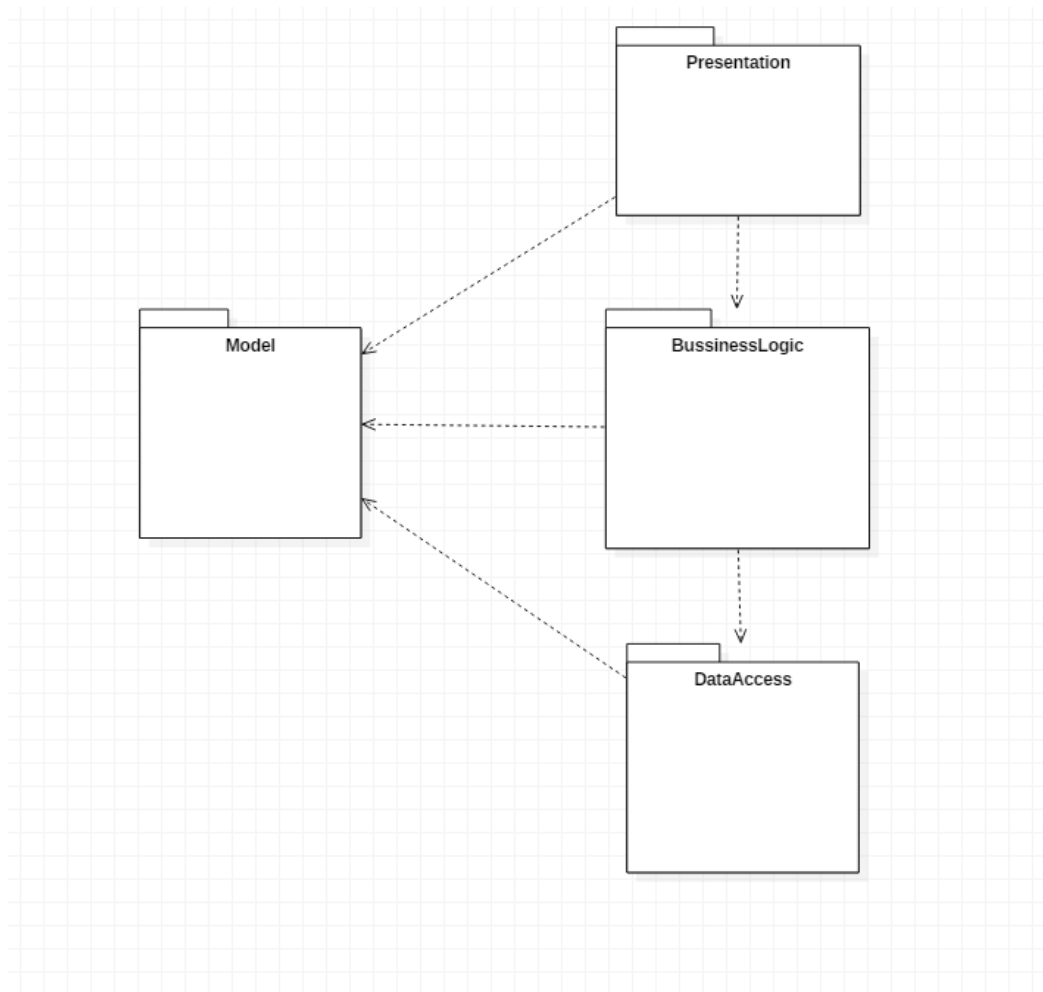
a. Diagrama de clase

În diagrama UML de mai sus sunt prezentate toate clasele, cu attributele și relațiile dintre acestea. O analiză detaliată a acestei diagrame este prezentată în capitolul 4 al acestei documentații.



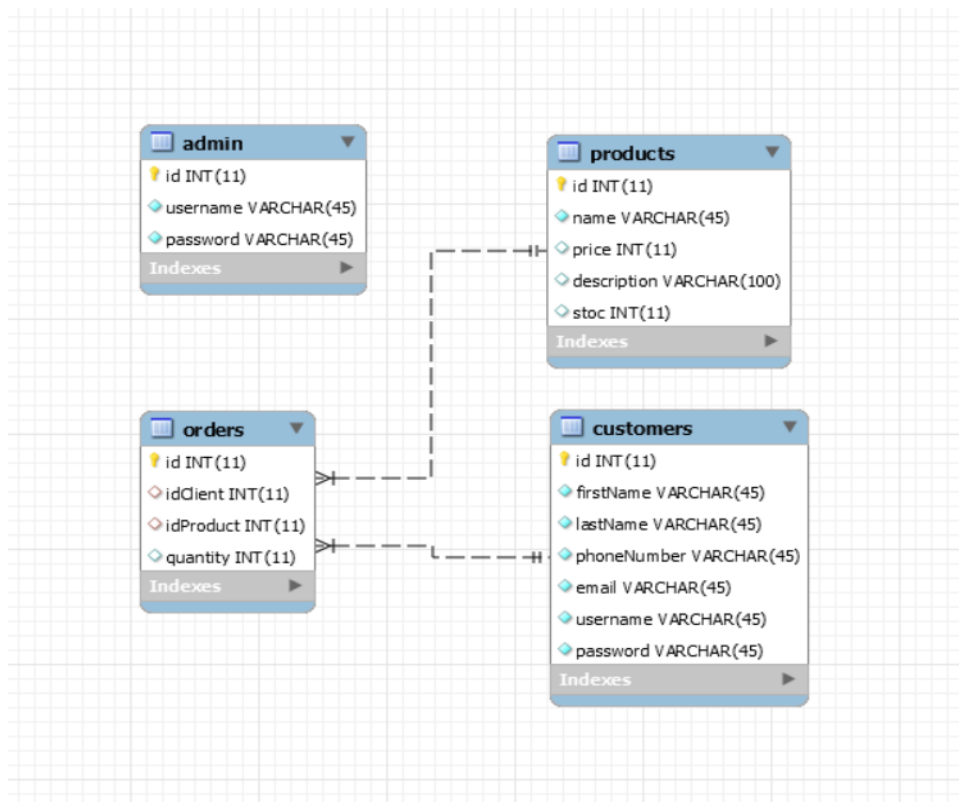
b. Diagrama de pachete

Una dintre cerintele de baza ale temei a fost utilizarea arhitecturii 3-tier in cadrul careia fiecare layer cheama functii din layer-ul de sub el. In acest sens diagrama de pachete demonstreaza utilizarea acestei arhitecturi.



c. Diagrama bazei de date

Diagrama bezei de date descrie relatiile dintre tabelele care alcatuiesc baza de date aferenta aplicatiei noastre. Este alcatuita din 4 tabele si relatiile dintre acestea.



c. **Proiectare clase**

Proiectul, ca un tot unitar, este divizat în 27 de clase care permit transpunerea cerinței și temei în limbaj de programare Java. Clasele au dimensiuni, alcătuiri și roluri variate și sunt divizate la randul lor în pachete. S-ar putea spune că la baza bunei funcționări a aplicației stă clasa "AbstractDAO" care este pivotul gestionării eficiente a bazei de date. În aceasta sunt definite

metode care construiesc query-uri care reprezinta fundatia interactiunii aplicatiei cu baza de date (ex: `createDeleteQuery(String Field)`, `createViewAllQuery(String Field)`, etc) si metode care produc efectiv aceasta interactiune prin deschiderea conexiunii, efectuarea de operatii si inchiderea respectivei conexiuni. In clasa "ConnectionFactory" sunt descrise metode care definesc efectiv conexiunea dar si lucrul cu aceasta (ex: `createConnection()`, `close(ResultSet)`, etc). Descriind mai pe larg conceptul de conexiune la baza de date se poate observa ca se realizeaza prin plasarea ei intr-un obiect de tip Singleton deoarece se doreste o singura conexiune la un moment dat.

Pentru fiecare tabela din baza de date a fost creata cate o clasa care contine attribute a caror denumire este exacta cu cea a campurilor tabelului pe care il reprezinta. Metode accesoare au fost create pentru a putea interactiona cu campurile private ale claselor. Urmatorul pas in proiectarea claselor a fost crearea a 4 clase reprezentative pentru fiecare tabel care sa extinda clasa "AbstractDAO", pentru a putea executa prin mostenire metodele definite in cadrul acesteia. Numele acestor clase este asemanator cu cel al claselor care definesc constructia tabelelor bazei de date, insa li s-a adaugat "DAO" in nume. Acestea contin doar un constructor null. Logica de lucru efectiva nu se poate realiza la nivelul acestor clase, principalul motiv fiind cerinta de a structura aplicatia sub forma 3-tier. In acest sens s-a mai creat un set de 4 clase care sa permita efectuarea logicii de lucru si ca caror nume este de asemenea asemanator cu cel al claselor care definesc tabelele insa de aceasta data li s-a adaugat "BLL" in nume. Clasele din interfata creeaza estetica aplicatiei, acestea aflandu-se in cel mai mare numar.

d. Algoritmi

În ceea ce priveste conceptul de algoritmi tema de fata nu a necesitat utilizarea unor astfel de structuri. Vom considera insa la acest capitol prezentarea query-urilor prin care se obtine informatia din baza de date.

- Metoda `private String createFindQuery(String field)` - creeaza prin intermediul unui `StringBuilder` un query de forma: `"SELECT * FROM table_name WHERE field = condition;"`
- Metoda `private String createAddQuery(T object)` - primeste ca argument un obiect de tip `T` si prin intermediul unui `String Builder` creeaza un

query de forma: "INSERT INTO table_name VALUES(value(1), ... , value(n))", unde n = numarul de coloane.

- Metoda private createViewAllQuery() - creeaza prin intermediul unui stringBuilder un query de forma "SELECT * FROM table_name;"
- Metoda private createDeleteQuery(String field) - creeaza prin intermediul unui stringBuilder un query de forma: "DELETE FROM table_name WHERE field = condition;"
- Metoda private createUpdateQuery(String Field, T object) - creeaza prin intermediul unui stringBuilder un query de forma: "UPDATE table_name SET colum_name(1) = value(1), ... , column_name(n) = value(n);"

a. Pachete

Intregul proiect este structurat pe pachete, la nivel arhitectural fiind format dupa modelul arhitecturii 3-tier. Astfel exista 4 pachete:

- BusinessLogic - care contine logica de lucru a aplicatiei si este alcatuita din clasele:
 - AdminBLL
 - CustomerBLL
 - OrderBLL
 - ProductBLL
- DataAcces – care incapsuleaza clasele care contin query-urile si conexiunea la baza de date:
 - AbstractDAO
 - ConnectionFactory
 - AdminDAO
 - CustomerDAO
 - OrderDAO
 - ProductDAO
- Model – contine clase mapate la baza de date:
 - Admin

- Customers
- Orders
- Products
- Presentation - contine clase care defineste interfata cu utilizatorul
 - AddClientWindow
 - AddProductWindow
 - Controller
 - CreateOrderWindow
 - DeleteCustomerWindow
 - DeleteProductWindow
 - FindCustomerWindow
 - FindProductWindow
 - LogInWindow
 - MainView
 - MainViewCustomer
 - UpdateCustomerWindow
 - UpdateProductWindow

e. Interfata utilizator

Interfața cu utilizatorul prezinta un aspect prietenos si relativ minimalist, astfel incat orice utilizator sa fie capabil de a-i deduce utilizarea dintr-o privire. Astfel, prima interfata este cea de logare in cadrul careia utilizatorul va alege modul de logare si va introduce datele corespunzatoare profilului propriu. In functie de metoda de logare se va deschide o noua fereastră. Daca utilizatorul se logheaza ca si admin atunci in fata acestuia va aparea o fereastră cu un meniu in

partea dreapta alcatuit din 4 butoane, primele 2 conducand la aparitia altor submeniuri, iar ultimele 2 la aparitia altor ferestre de interactiune cu aplicatia. Daca utilizatorul se logheaza ca si client atunci in fara acestuia va apare o fereastră cu un meniu in partea dreapta alcatuit din 6 butoane care prin apasare vor conduce la aparitia altor ferestre de interactiune cu aplicatia.

4. Implementare

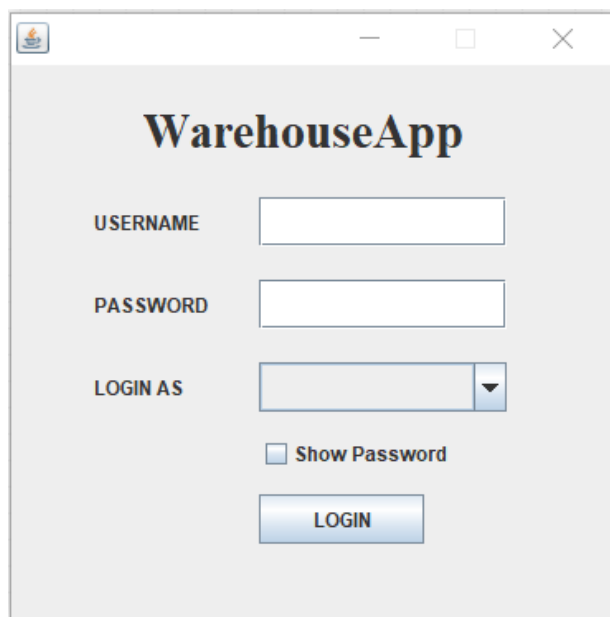
Deoarece arhitectura de pachete a acestei aplicatii este un 3-tier, fiecare clasa dintr-un pachet superior apeleaza metode dintr-o clasa dintr-un pachet inferior, cu alte cuvinte vizibilitatea este de sus in jos, nu de jos in sus. Astfel clasele din pachetul Presentation apeleaza metode din pachetul BusinessLogic, clasele din pachetul BusinessLogic apeleaza metode din clasele din pachetul DataAccess iar toate aceste 3 clase apeleaza metode din clasele din pachetul Model. Incalcarea acestor dependinte ar duce la distrugerea arhitecturii 3-tier. Astfel, in pachetul Presentation au fost create clase care sa defineasca interfata cu utilizatorul. Politica aleasa pentru interfata grafica este una simpla. S-a dorit ca utilizator sa aiba tot timpul la dispozitie un meniu cu butoane de unde sa aleaga ce operatie doreste sa execute. Astfel, au fost create 2 clase grafice care reprezinta interfetele principale care vor apare in fata utilizatorului in functie de metoda de logare (Admin sau Client). De acolo, in functie de optiunile alese din meniuri se vor deschide diverse ferestre sau panouri de interactiune cu aplicatia. Logica de lucru cu aplicatia a fost descrisa in pachetul BusinessLogic. Aici, vor avea loc toate operatiile care trebuiesc executate pentru ca aplicatia sa functioneze in parametri normali. Au fost definite metode care sa descrie interactiunea utilizatorului in functie de fiecare tabela. Accesul la baza de date este realizat prin intermediu a 6 clase. Clasa ConnectionFactory creeaza un obiect de tip Singleton care va reprezenta conexiunea efectiva a aplicatiei. Metode din cadrul acesteia vor fi utilizate in clasa AbstractDAO, pivotul muncii aplicatiei cu baza de date. Patru clase cu nume sugestive au fost

create in pachetul Model pentru de descrie tabelele din baza de date si pentru a facilita lucrul cu acestea.

5. Rezultate

Au fost executate o serie de teste pentru a exemplifica diferitele rezultate care se pot obține în urma interacțiunii utilizatorului cu interfața dar și pentru a scoate în evidență eventualele erori care pot fi afișate.

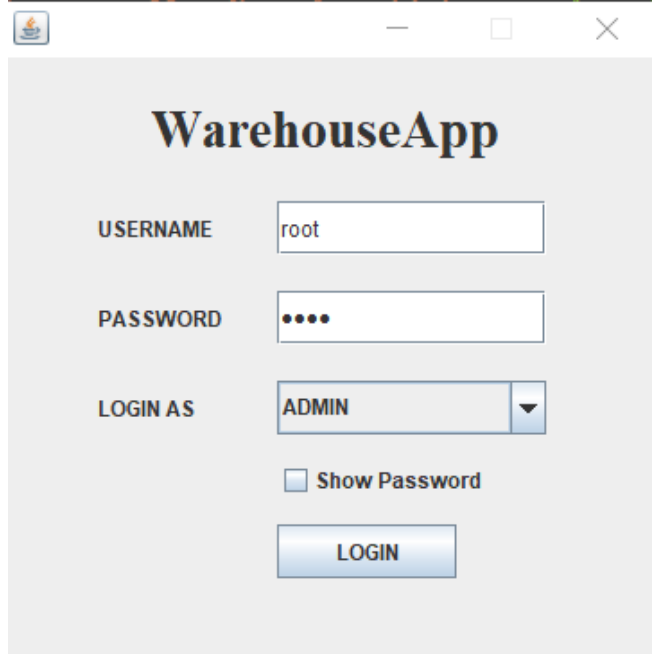
- Interfata de logare



The image shows a screenshot of a web application window titled "WarehouseApp". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area is light gray and contains the following elements:

- The title "WarehouseApp" in a large, bold, black serif font.
- A label "USERNAME" followed by a white text input field.
- A label "PASSWORD" followed by a white text input field.
- A label "LOGIN AS" followed by a white dropdown menu with a blue arrow pointing down.
- A checkbox labeled "Show Password" which is currently unchecked.
- A blue "LOGIN" button with white text.

- Logarea ca si "Admin"



A screenshot of a web application window titled "WarehouseApp". The window has a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The main content area is light gray and contains the following elements: the title "WarehouseApp" in a large, bold, black serif font; a "USERNAME" label followed by a text input field containing the text "root"; a "PASSWORD" label followed by a password input field with four black dots; a "LOGIN AS" label followed by a dropdown menu currently showing "ADMIN"; a "Show Password" checkbox which is unchecked; and a blue "LOGIN" button at the bottom.

- Logarea ca si "Client"

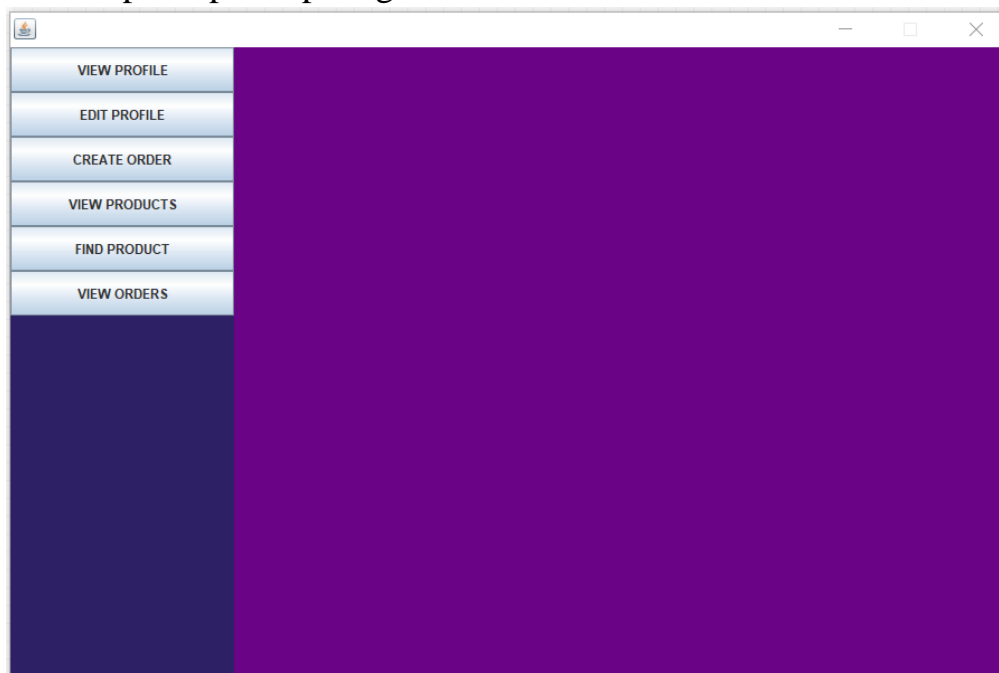


A screenshot of the same "WarehouseApp" login window, but with different values. The title bar is identical. The main content area is light gray and contains: the title "WarehouseApp" in a large, bold, black serif font; a "USERNAME" label followed by a text input field containing the text "andrei_jitaru"; a "PASSWORD" label followed by a password input field with five black dots; a "LOGIN AS" label followed by a dropdown menu currently showing "CLIENT"; a "Show Password" checkbox which is unchecked; and a blue "LOGIN" button at the bottom.

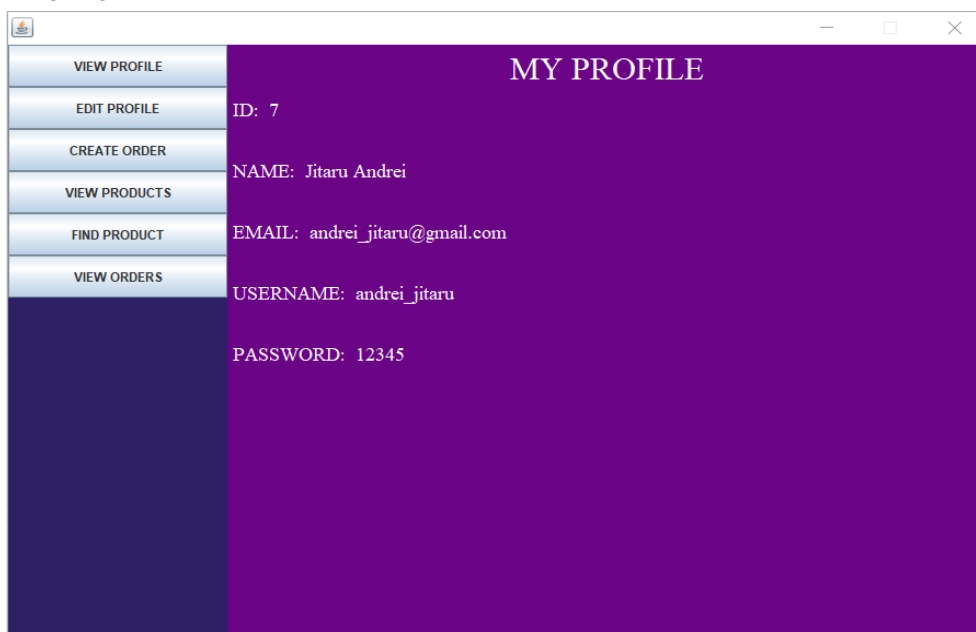
- Eroare la logare – LogIn AS



- Meniul principal dupa logarea ca si "Client"



- Meniul principal dupa logarea ca si "Client" si alegerea optiunii "View Profile"



- Meniul principal dupa logarea ca si "Client" si alegerea optiunii "Update Profile"

Dialog box for updating a customer profile. The background is purple. A sidebar on the left contains menu items: VIEW PROFILE, EDIT PROFILE, CREATE ORDER, VIEW PRODUCTS, FIND PRODUCT, and VIEW ORDERS. The dialog box is titled 'UPDATE PROFILE' and contains the following fields:

- ID: 7
- FIRST NAME:
- SECOND NAME:
- PHONE NUMBER:
- EMAIL:
- USERNAME:
- PASSWORD:

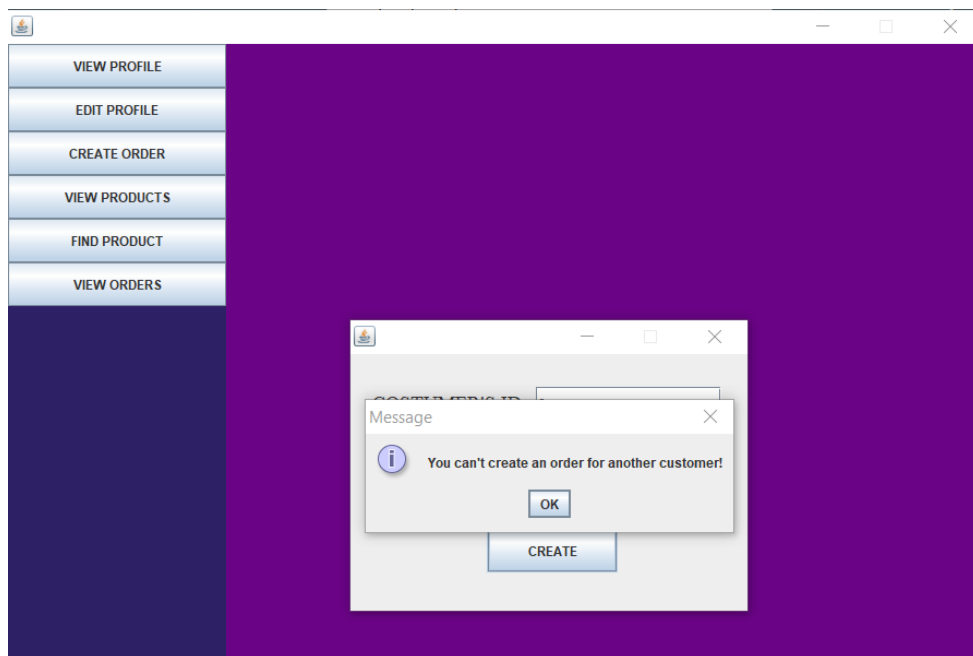
At the bottom of the dialog is a button labeled 'UPDATE COSTUMER'.

- Meniul principal dupa logarea ca si "Client" si alegerea optiunii "View Orders"

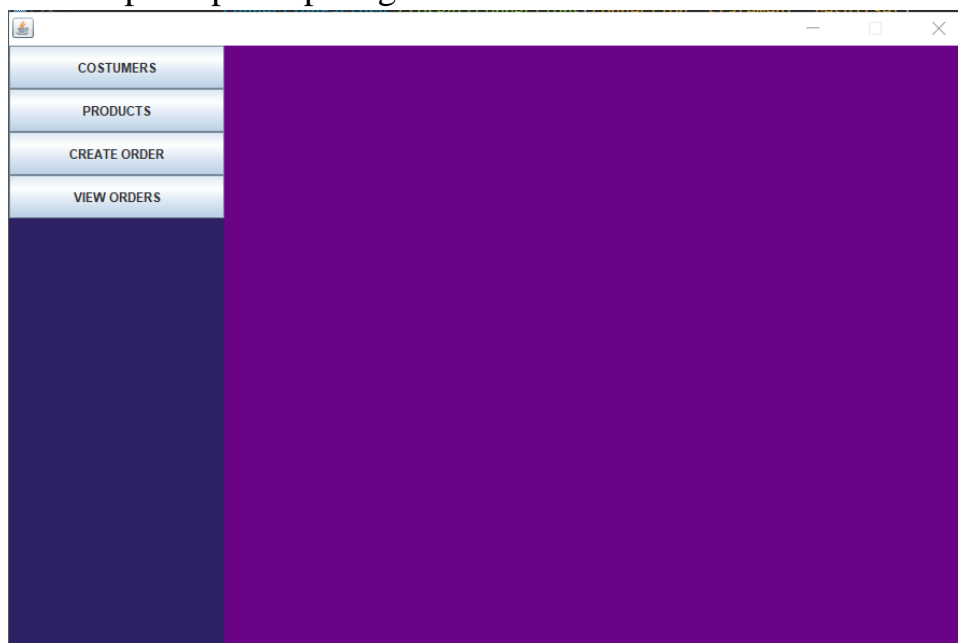
Table showing orders. The background is light gray. A sidebar on the left contains menu items: VIEW PROFILE, EDIT PROFILE, CREATE ORDER, VIEW PRODUCTS, FIND PRODUCT, and VIEW ORDERS. The table has the following data:

id	id_c	id_p	quantity
8	7	1	100
13	7	10	100
14	7	11	10

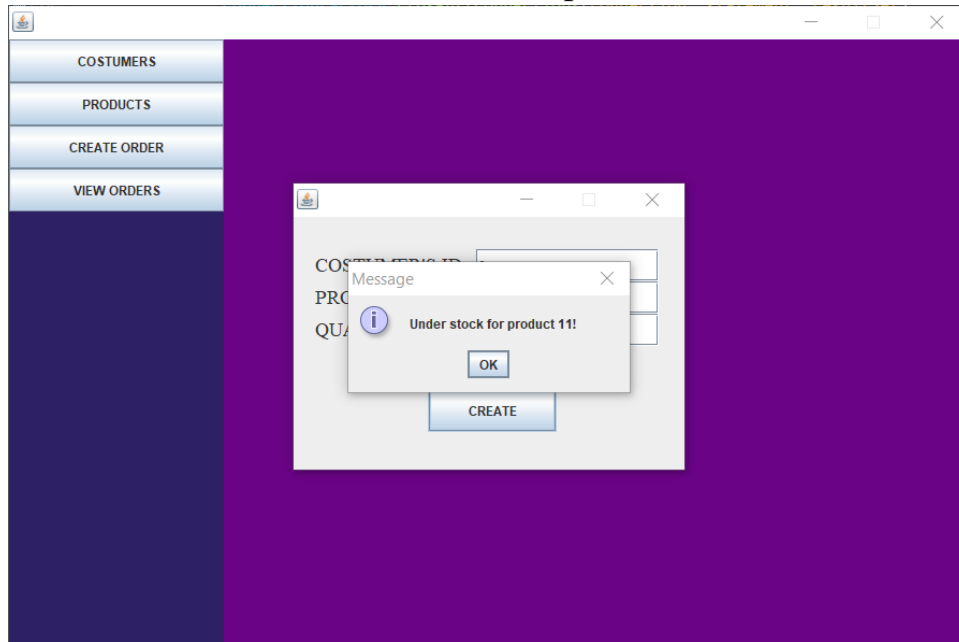
- Eroare la crearea unei comenzi cu id-ul altei persoane daca utilizatorul este logata ca si "Client"



- Meniul principal dupa logarea ca si "Admin"



- Eroare la crearea comenzi a carui produs nu detine un stoc adecvat



6. Concluzii si dezvoltari ulterioare

Sunt de parere ca aplicatia de fata ar putea fi utilizata cu usurinta de orice persoana interesate de managementul unui depozit, atat de catre admini cat si de catre clienti. In ceea ce priveste capitolul de dezvoltari ulterioare consider ca aplicatia ar avea nevoie de un nou buton in interfata de logare care sa permita crearea de conturi noi si de un buton de delogare din oricare din interfetele principale.