

UNIVERSITATEA TEHNICĂ „Gheorghe Asachi” din IAȘI  
FACULTATEA DE AUTOMATICĂ ȘI CALCULATOARE  
SPECIALIZAREA: Calculatoare și tehnologia informației

Disciplina Baze de Date

# **Implementare practică a unui magazin online de mobilă**

Coordonator științific  
Prof. Mironeanu Cătălin

Student,  
Imbrea Daniel  
Grupa 1306B

## **Scopul proiectului**

Proiectul își propune crearea unei aplicații desktop de tip N-tier care permite interacțiunea cu un magazin online, atât din perspectiva unui utilizator obișnuit, cât și din cea a unui administrator.

## **Prezentare generală**

În era digitală este crucial ca orice brand să aibă o prezență online, cu atât mai mult pentru a facilita vânzarea a cât mai multor produse. În acest scop este necesară o aplicație care să ofere potențialului client toate informațiile pe care și le-ar putea dori despre un anumit produs, cât și facilitarea procesului de a comanda respectivul produs.

Din perspectiva firmei, este necesară prezența unui sistem rapid și eficient pentru administrarea produselor deja existente, cât și adăugarea de noi produse. În vederea acestui scop aplicația prezintă o suită de unelte destinate administratorilor care permite ștergerea, modificarea și adăugarea de noi produse într-un mod foarte simplu și intuitiv, care nu necesită cunoștințe tehnice.

## **Tehnologii folosite și prezentarea interfeței**

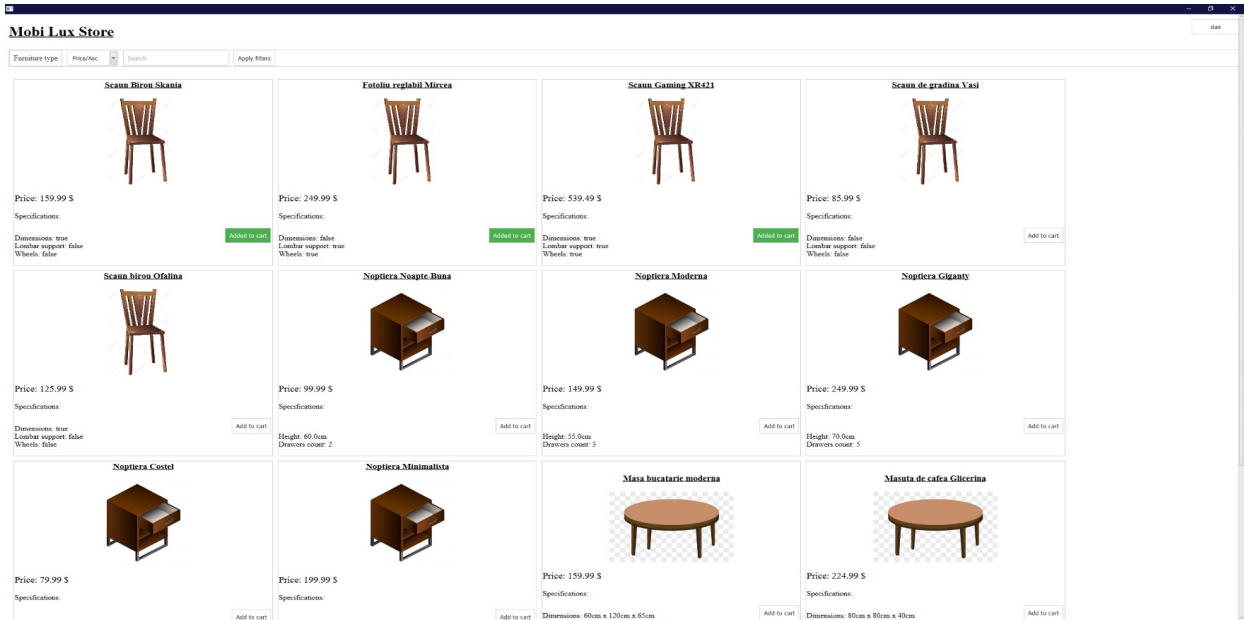
Partea de back-end este formată din 2 componente: conexiunea client-server, facilitată de biblioteca java.net, care permite clientului de a da request-uri la server sub forma unor mesaje text și conexiunea server-database, facilitată de driver-ul ojdbc, necesar pentru trimiterea de query-uri de la aplicația java la baza de date.

La deschiderea serverului, este realizată conexiunea la baza de date, printr-un obiect de tip Connection și utilizarea metodei DriverManager.getConnection

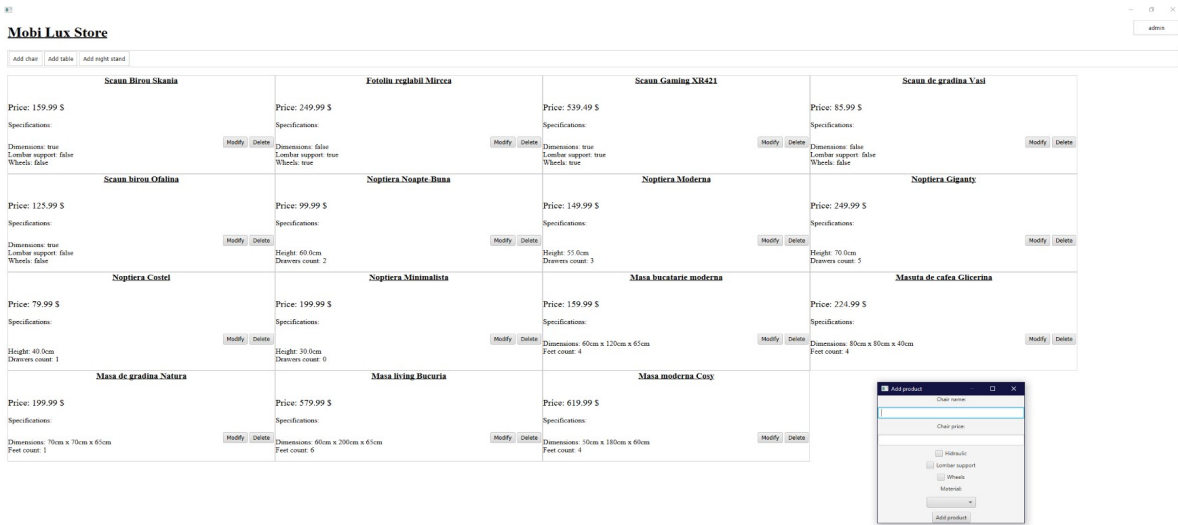
Atunci când un utilizator rulează clientul, se realizează o conexiune la server, care crează un nou thread pentru acesta. Threadul are ca rol ascultarea în permanență a requesturilor trimise de utilizator, requesturi care sunt rescrise ca și query-uri Sql de către thread și apoi trimise la query-runner, care returnează răspunsul la thread, și apoi la client.

Principalul framework folosit în crearea proiectului pe partea de front-end este Javafx, folosit cu scopul de a genera interfața grafică din fișiere html/css. Principalele pagini ale aplicației sunt Home și Admin Tools.

Home este pagina unde clienții navighează lista de produse alături de specificațiile lor, și au opțiunea de a sorta / filtra diferite produse și opțiunea de a adăuga produsul în coșul de cumpărături (comanda poate fi ulterior finalizată în pagina ‘Cart’).

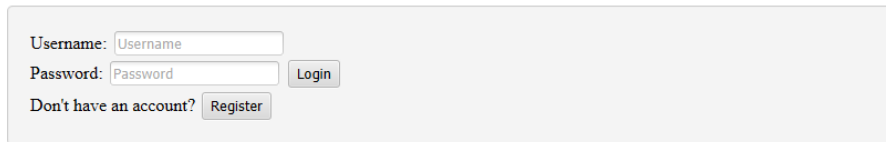


Pagina Admin Tools permite unui administrator de sistem, logat cu un cont special, sa ștergă, să modifice și să adauge produse noi, prin intermediul unor form-uri cu câmpurile necesare. Odată făcută o modificare, clienții vor vedea schimbarile imediat ce pagină nouă este încărcată.



Un exemplu de interacțiune obișnuită a sistemului, este logarea în aplicație:

### Mobi Lux Store

A login form with a light gray background. It contains two input fields: 'Username' and 'Password'. To the right of the 'Password' field is a 'Login' button. Below the 'Password' field is a link 'Don't have an account?' followed by a 'Register' button.

Utilizatorul completează câmpurile de nume și parolă, iar la apăsarea butonului ‘Login’, un request este trimis la server de forma : “login|numeUtilizator|parolaEncriptata”.

Serverul apoi procesează acest request într-un query SQL de forma:

```
SELECT count(*)
```

```
FROM customers
```

```
WHERE name = {nume extras din request}
```

```
AND password = {parola encriptată extrasă din request}
```

Dacă acest query returnează 1, atunci se extrag restul datelor utilizatorului din baza de date pentru a putea fi prezentate ulterior la pagina Account Information și se returnează logarea cu succes (sau eșuarea acesteia) la client, care în caz de succes este trimis în Home Page.

Exemplul în cod:

Client:

```
if(!username.isEmpty() && !password.isEmpty()){
    Client.sendRequest("login|" + username + "|" + password);

    String response = readResponse();
    System.out.println(response);
    if (response.contains("User found: 1")) {
        Client.sessionVariables.put("username", username);
        Client.sessionVariables.put("email", response.split(regex: "\\|")[4]);
        Client.sessionVariables.put("phone", response.split(regex: "\\|")[6]);
        Client.sessionVariables.put("admin", response.split(regex: "\\|")[7]);
        Client.sessionVariables.put("city", response.split(regex: "\\|")[10]);
        Client.sessionVariables.put("street", response.split(regex: "\\|")[11]);
        Client.sessionVariables.put("zipcode", response.split(regex: "\\|")[12]);
        setPage(new HomePage(webView));
    }
    else if(response.contains("User found: 0")){
        displayElement(getElement( by: "id", value: "LoginError"), show: true);
    }
}
```

Server:

```
switch (requestType){
    case "login": {
        String userExist = queryRunner.runSQLQuery("SELECT COUNT(*)" +
            "FROM customers" +
            " WHERE name = '" + requestElements.get(1) +
            "' AND password = '" + requestElements.get(2) + "'");

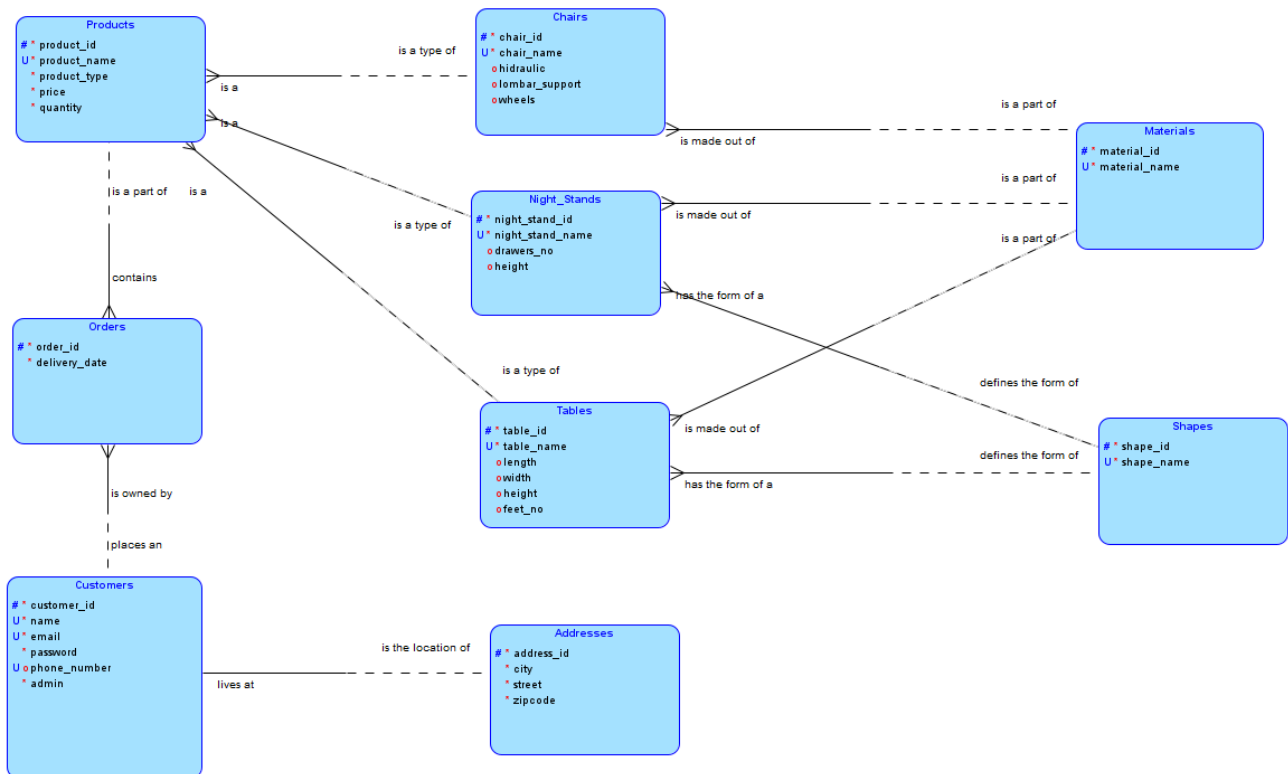
        String userData = queryRunner.runSQLQuery("SELECT * FROM customers WHERE name = '" + requestElements.get(1) + "'" +
            " AND password = '" + requestElements.get(2) + "'");

        if(userExist.contains("1")) {
            String address_id = userData.split(regex: "\\|")[6];
            userData += queryRunner.runSQLQuery("SELECT * FROM addresses WHERE address_id = " + address_id);
        }

        System.out.println("Sending response: " + userExist.replace(target: "\n", replacement: "") + userData);

        response = "User found: " + userExist;
        packetSender.println(response + userExist.replace(target: "\n", replacement: "") + userData);
        break;
    }
}
```

## Modelul logic al bazei de date



## Descrierea relațiilor dintre entități

Entitățile prezente în această aplicație sunt:

- Clienți (Customers)
- Adrese (Addresses)
- Comenzi (Orders)
- Produse (Products)
- Scaune (Chairs)
- Noptiere (Night\_Stands)
- Mese (Tables)
- Materiale (Materials)
- Forme (Shapes)

În proiectarea acestei baze de date se regăsesc atât relații de tipul 1:1 (one-to-one) cât și relații de tipul 1:n (one-to-many).

Între entitatea Customers și entitatea Addresses se stabilește o relație de 1:1. Fiecare client are asociată o adresă de livrare. Cele două entități sunt legate prin atributul `address_id`.

Între Customers și Orders există o relație de 1:n. Un client are posibilitatea de a plasa mai multe comenzi, și fiecare comandă în parte aparține unui singur client. Entitățile sunt legate între ele prin atributul `customer_id`.

Între entitățile Products și Orders există o relație de 1:n. Fiecare produs poate aparține mai multor comenzi, deoarece Products face referire la o anumită marcă, și nu la un produsul individual. Entitățile sunt legate între ele prin atributul `product_id`.

Între entitățile Chairs, Night\_Stands, Tables și Products este prezentă o relație de tipul 1:n. Explicația este că cele trei tipuri de produse se pot găsi de la diferite mărci, în diferite forme, dar toate sunt considerate produse. Legătura dintre Products și cele 3 tipuri de produse se face prin attributele `chair_id`, `night_stand_id` și `table_id`.

Între entitatea Materials și entitățile Chairs, Night\_Stands și Tables există relații de 1:n, deoarece diferite produse pot fi făcute din aceleași materiale. Legătura dintre cele trei tipuri de produse și Materials este realizată de atributul

material\_id.

Similar cu Materials, între Shapes și entitățile Night\_Stands și Tables există relații de 1:n, deoarece mai multe produse pot avea aceeași formă. Legătura dintre tipurile de produse și Shapes este realizată de atributul shape\_id.

## **Descrierea constrângerilor**

Constrângerile de tip check sunt folosite în diverse entități cu scopul de a valida datele introduse.

Spre exemplu, la introducerea codului poștal se face verificarea ca numărul să fie în intervalul 100000, 999999, sau la introducerea dimensiunilor produselor, acestea nu pot fi numere negative.

Constrângerile de tip check sunt folosite și pentru asigurarea ca o valoare se află într-o mulțime finită, spre exemplu, pentru a simula prezența sau absența unei caracteristici (sistem hidraulic, roți sau suport lombar), prin utilizarea valorilor de 0 și 1.

O constrângere similară este folosită pentru a atributul product\_type, care poate lua doar una din trei valori predefinite.

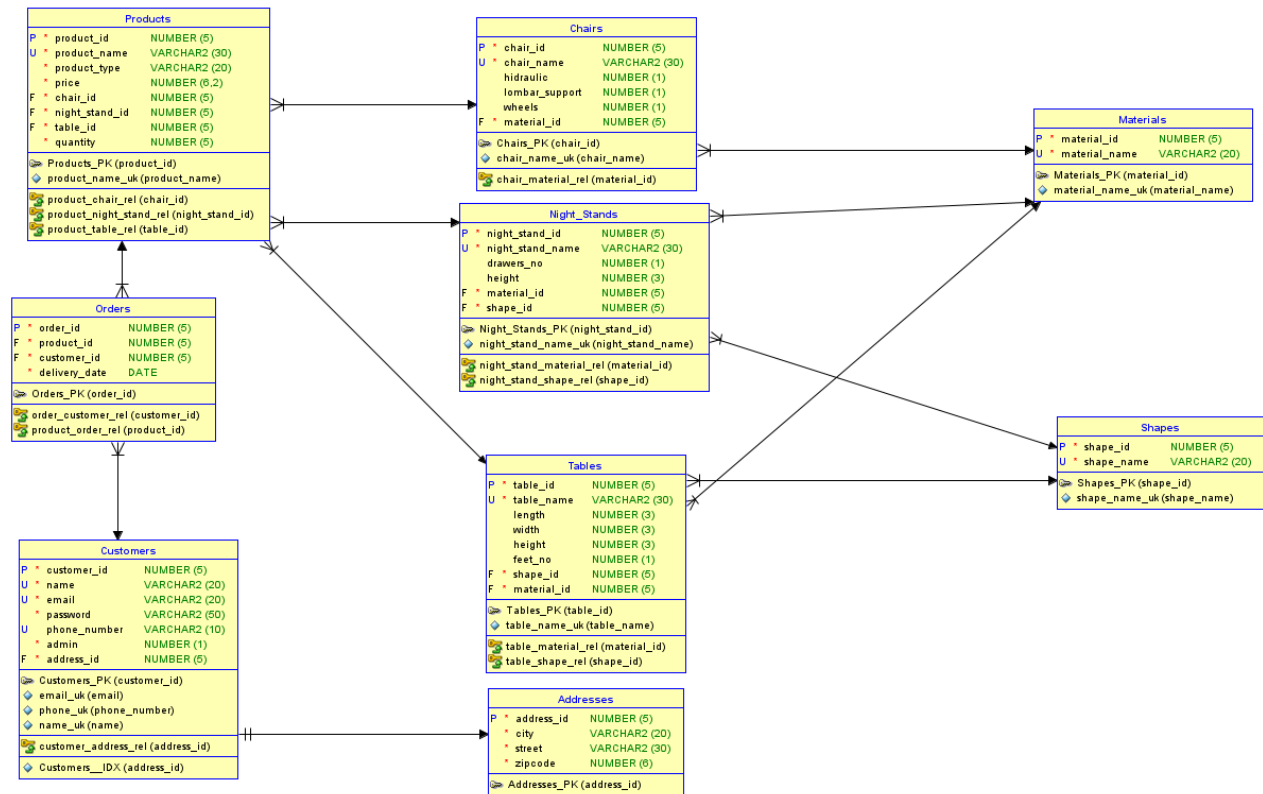
O ultimă utilizare a constrângerilor check este verificare formatului email-ului și al numărului de telefon. Email-ul trebuie să respecte condiții standard, precum caracterul @, și terminația cu o secvență de forma .ro / .com, iar numărul de telefon trebuie să conțină strict 10 caractere de tip număr.

Sunt folosite și constrângeri de tip unique, pentru atributele e-mail, număr de telefon, cât și numele produselor, al materialelor și al formelor.

În cele din urmă, sunt folosite constrângeri not-null, care se găsesc pe majoritatea atributelor, excepție fiind caracteristici ale produselor care pot fi cunoscute/aplicabile, sau nu.

Cheile primare sunt generate prin mecanismul de auto-increment, asigurând unicitatea acestora.

# Modelul Relațional



## Aspecte legate de normalizare

Baza de date a fost normalizată, deoarece îndeplinește următoarele condiții:

1. Toate tabelele respectă condițiile primei forme normale:
  - Fiecare atribut conține o singură valoare
  - Nu conține grupuri care se repetă
2. Toate tabelele respectă condițiile celei de a doua forme normale:
  - Sunt în prima formă normală
  - Toate tabelele au o singură cheie primară
3. Toate tabelele respectă a treia formă normală:
  - Sunt în a doua formă normală
  - Toate atributele non-cheie sunt direct dependente de cheile candidat

Prima formă normală este îndeplinită deoarece pentru fiecare tabelă, se poate insera doar o valoare pe fiecare câmp și dacă se dorește inserarea de mai



multe ori a aceluiași câmp, se inserează noi intrări în tabelă. De exemplu, dacă se dorește inserarea unui scaun care vine în mai multe modele (spre exemplu poate avea, sau nu sistem hidraulic), în acest caz se introduce acest scaun de două ori, având câmpul respectiv diferit.

A doua formă este îndeplinită de toate tabelele, deoarece toate tabelele conțin o singură cheie primară, denumită după numele tabelului, urmată de \_id.

Un exemplu de formă a treia se regăsește între tabela Chairs și tabela Materials, deoarece pentru a determina materialul din care este făcut scaunul în cauză, este folosită cheia primară din tabela Materials.