**Aplicație de Gestionare a Datelor Comerciale cu MySql Workbench si JavaFX**

**Dumitru Ana-Maria   
 444A**

**Cuprins propus:**

1. **Introducere**
   * 1.1 Descrierea proiectului
   * 1.2 Cerința temei
2. **Tehnologii Folosite**
   * 2.1 MySQL Workbench – Baza de date relațională
   * 2.2 JavaFX – Platforma de dezvoltare a aplicației
   * 2.3 JDBC – Conectarea aplicației cu baza de date
   * 2.4 Scene Builder – Dezvoltarea interfeței grafice
3. **Arhitectura și Implementarea Aplicației**
   * 3.1 Structura Generală a Aplicației
   * 3.2 Diagrama logică a bazei de date
   * 3.3 Interfața Grafică a Utilizatorului (GUI)
     + 3.3.1 Pagina de autentificare
     + 3.3.2 Meniul Principal „Admin”
     + 3.3.3 Formular Clienți
     + 3.3.4 Formular Furnizori
     + 3.3.5 Formular Distribuitori
     + 3.3.6 Formular Produse
     + 3.3.7 Formular Comenzi
     + 3.3.8 Funcționalități pentru utilizatorii standard
   * 3.4 Gestionarea conexiunii cu baza de date
   * 3.5 Implementarea operațiunilor CRUD
   * 3.6 Gestionarea securității utilizatorului
4. **Concluzii**
5. **Bibliografie**

**1. Introducere**

**1.1 Descrierea proiectului**

Proiectul dezvoltat reprezintă o aplicație desktop realizată în JavaFX, având ca scop gestionarea relațiilor dintre clienți, furnizori și produse, utilizând o bază de date relațională creată în MySQL. Această aplicație permite utilizatorilor să efectueze operații esențiale de tip CRUD (Create, Read, Update, Delete) asupra tabelelor **Clienti**, **Furnizori**, **Produse** și **Comenzi**, precum și vizualizarea asocierilor dintre acestea.

**1.2 Cerința temei**

Tema solicită crearea unei aplicații care să includă o bază de date în MySQL și o interfață grafică realizată cu JavaFX. Baza de date va conține următoarele tabele:

* **Produse** (idprodus, Denumire,Descriere, Cantitate)
* **Distribuitor**(iddistribuitor,idprodus,idfurnizor,numeDistribuitor,pretUnitar,stocDispobil)
* **Comenzi**(idcomanda,idfurnizor,idclient,dataComanda,status,valoareComanda)
* **Furnizori** (idfurnizor, NumeFurnizor, Adresă)
* **Clienti** (idclient, Nume, Prenume, Adresă, Email)

Asocierile între tabele sunt următoarele:

* în **tabela Produse** și **tabela Furnizori** există un raport de cardinalitate M:N (mulți la mulți), realizat prin creearea tabelei de legatura Distribuitor, care contine pe langa atributele necesare realizarii multiplicitatii numeDistribuitor,pretUnitar,stocDisponibil.
* în **tabela Furnizori** și **tabela Clienti** există un raport de cardinalitate M:N, realizat prin creearea tabelei de legatura Comenzi, care contine pe langa atributele necesare realizarii multiplicitatii dataComanda,status si valoareComanda.

Interfața aplicației trebuie să permită utilizatorilor să efectueze următoarele operații pe toate tabelele:

* Vizualizare
* Adăugare
* Modificare
* Stergere

Vizualizarea tabelelor de legătură va presupune afișarea datelor referite din celelalte tabele, pentru a facilita o gestionare eficientă a relațiilor dintre entitati.

**2. Tehnologii Folosite**

**2.1 MySQL Workbench – Baza de date relațională**

MySQL Workbench este un instrument vizual complet pentru proiectarea, dezvoltarea și administrarea bazelor de date MySQL. Acesta oferă funcționalitċi precum:

* **Modelare vizuală a datelor**: Utilizatorii pot crea diagrame ER (Entity-Relationship) pentru a defini structura bazei de date, relațiile dintre tabele și constrângerile acestora.
* **Gestionarea serverului MySQL**: Permite monitorizarea performanței serverului, backup-uri și restaurări de baze de date, și configurarea utilizatorilor și a permisiunilor.
* **Scriere și executare de interogări SQL**: Un editor SQL integrat facilitează dezvoltarea, testarea și rularea interogărilor direct din aplicație.

Pentru acest proiect, MySQL Workbench a fost utilizat pentru definirea tabelelor **Produse**, **Furnizori**, **Clienti** și **Comenzi**, precum și pentru gestionarea relațiilor de tip M:N între aceste tabele.

**2.2 JavaFX – Platforma de dezvoltare a aplicației**

JavaFX este o platformă modernă pentru dezvoltarea aplicațiilor desktop și mobile, oferind un set extensiv de componente UI (User Interface) și suport pentru efecte grafice avansate. Caracteristicile principale includ:

* **Component-based UI**: Oferă controale standard precum butoane, tabele, liste și ferestre de dialog, ușor de personalizat.
* **Suport CSS și FXML**: Permite stilizarea interfeței utilizator prin CSS și structurarea codului folosind FXML, un limbaj XML pentru definirea layout-ului.
* **Grafică 2D/3D și animații**: JavaFX oferă suport pentru grafică avansată, inclusiv efecte de tranziții, animații și manipulare multimedia.

JavaFX a fost utilizat pentru crearea interfeței grafice a aplicației, facilitând o interacțiune intuitivă a utilizatorului cu baza de date prin formulare de adăugare, modificare și ștergere a datelor.

**2.3 JDBC – Conectarea aplicației cu baza de date**

Java Database Connectivity (JDBC) este o API standard care permite conectarea aplicațiilor Java la baze de date relaționale. Funcționalitċi cheie includ:

* **Conectarea la baza de date**: Prin driverele JDBC, aplicația JavaFX se conectează la MySQL pentru a executa interogări SQL.
* **Executarea de interogări și actualizări**: JDBC permite trimiterea de comenzi SQL pentru inserarea, actualizarea și ștergerea datelor.
* **Gestionarea rezultatelor**: Rezultatele interogărilor sunt gestionate folosind obiecte de tip ResultSet, care permit navigarea și extragerea datelor necesare pentru afișare în interfață.

Pentru acest proiect, JDBC este esențial în asigurarea unei comunicări eficiente între aplicația JavaFX și baza de date MySQL.

**2.4 Scene Builder – Dezvoltarea interfeței grafice**

Scene Builder este un instrument vizual pentru proiectarea interfețelor JavaFX, care simplifică procesul de dezvoltare printr-o interfață intuitivă de tip drag-and-drop. Caracteristicile principale includ:

* **Crearea vizuală a layout-urilor**: Permite adăugarea de componente UI (butoane, tabele, texte) fără a scrie cod manual.
* **Generarea fișierelor FXML**: Scene Builder salvează structura interfeței în fițiere FXML, care sunt ulterior integrate în aplicația Java.
* **Integrare cu IDE-uri**: Compatibil cu IntelliJ IDEA și Eclipse, facilitând trecerea rapidă de la design vizual la codul sursă.

Utilizarea Scene Builder a permis dezvoltarea rapidă a unei interfețe ușor de utilizat, care să permită gestionarea eficientă a datelor din baza de date.

**Arhitectura și Implementarea Aplicației**

3.1 Structura Generală a Aplicației

Aplicația este organizată în mai multe module, fiecare având un rol bine definit:

* **Model:** Conține clasele care reprezintă entitățile din baza de date, cum ar fi Client.java, Furnizor.java, Produs.java și Comanda.java. Fiecare clasă include atributele corespunzătoare coloanelor din tabele și metode pentru accesarea și modificarea acestora.
* **View:** Interfața grafică este dezvoltată în JavaFX folosind Scene Builder. Fiecare fereastră sau panou din aplicație are un fișier FXML asociat, cum ar fi AdaugaClient.fxml, AplicatieAdmin.fxml, Login.fxml, etc.
* **Controller:** Gestionarea logicii aplicației este realizată prin clasele de tip controller, cum ar fi AdaugaClientController.java, ModificaProdusController.java, LoginController.java, etc. Aceste clase se ocupă de manipularea evenimentelor de la utilizator și actualizarea interfeței grafice cu datele din baza de date.
* **DAO (Data Access Object):** Clasa DBOperations.java gestionează interacțiunea directă cu baza de date, implementând operațiunile CRUD pentru fiecare entitate.

3.2 Diagrama logică a bazei de date

A screenshot of a computer

Description automatically generated

Diagrama logică a bazei de date MySQL oferă o vedere de ansamblu asupra structurii utilizate în aplicație, evidențiind tabelele esențiale pentru gestionarea informațiilor comerciale:

* **utilizatori**: Stochează datele de autentificare și rolurile (Admin sau Client), cu atribute precum *idutilizator*, *username*, *parola* și *rol*.
* **clienti**: Conține informații despre clienți, inclusiv *idclient*, *nume*, *prenume*, *adresa*, *email* și *idutilizator* (pentru asocierea cu contul de utilizator).
* **furnizori**: Păstrează detalii despre furnizori, cum ar fi *idfurnizor*, *nume*, *prenume*, *adresa* și *idutilizator*.
* **produse**: Include detalii despre produse, precum *idprodus*, *denumire*, *descriere* și *cantitate*.
* **distribuitor**: Asociază produsele cu furnizorii, având câmpuri precum *iddistribuitor*, *idprodus*, *idfurnizor*, *pretUnitar* și *stocDisponibil*.
* **comenzi**: Înregistrează comenzile plasate de clienți, cu atribute precum *idcomanda*, *idclient*, *idfurnizor*, *dataComanda*, *status* și *valoareComanda*.

Relațiile dintre tabele sunt definite prin chei primare și străine, asigurând integritatea datelor. De exemplu, tabelul **comenzi** conectează clienții cu furnizorii și produsele prin *idclient* și *idfurnizor*, facilitând astfel operațiile CRUD.

3.3 Interfața Grafică a Utilizatorului (GUI)

Interfața grafică a utilizatorului reprezintă componenta principală prin care utilizatorii interacționează cu aplicația. Aceasta a fost proiectată pentru a fi intuitivă și ușor de utilizat, oferind acces rapid la funcționalitățile de gestionare a datelor medicale. În această secțiune vor fi prezentate principalele elemente de interfață, paginile aplicației și modul de navigare între acestea.

3.3.1 Pagina de autentificare

A screenshot of a login screen

Description automatically generated

Pagina de autentificare reprezintă punctul de acces în aplicație și asigură securitatea datelor prin validarea utilizatorilor. Aceasta conține două câmpuri de introducere a datelor: User și Password, unde utilizatorii trebuie să introducă datele lor de acces. Procesul de autentificare este declanșat prin apăsarea butonului Login, care verifică informațiile introduse și permite accesul în funcție de rolul utilizatorului. Aplicația definește două tipuri de utilizatori:

• Admin(Furnizori), care are drepturi complete de gestionare a datelor, inclusiv adăugarea, editarea și ștergerea in toate tabelele.

• Clienti, care are acces limitat, putând vizualiza doar informațiile din tabele, asigurând acces diferențiat în funcție de permisiuni.

3.3.2 Pagina Principal „Admin”

A screenshot of a computer

Description automatically generated

În această secțiune, utilizatorii pot încărca, adăuga, șterge și modifica datele clienților folosind butoanele disponibile. Tabela afișează informații precum ID-ul clientului, numele, prenumele, adresa și email-ul. De asemenea aceste operatii sunt disponibile pentru toate tabelele, navigarea intre ele.

3.3.3 Formular Adauga Clienți

A screenshot of a computer

Description automatically generated  
Formularul contine patru campuri obligatorii: Nume,Prenume, Adresa si E-mail. Butonul **Adaugă** salvează datele în baza de date și închide fereastra, actualizând automat tabelul din pagina principală. Utilizatorul completează câmpurile obligatorii și apasă **Adaugă** pentru a salva. După salvare, este redirecționat înapoi la tabelul respectiv, unde datele sunt actualizate automat.

3.3.4 Formular Adauga Furnizori

A screenshot of a computer

Description automatically generated

Formularul pentru adăugarea furnizorilor include câmpurile **Nume,Prenume**  si **Adresă**  Design-ul este similar cu cel al formularului clienților, cu butoane **Adaugă**. Utilizatorul completează câmpurile obligatorii și apasă **Adaugă** pentru a salva. După salvare, este redirecționat înapoi la tabelul respectiv, unde datele sunt actualizate automat.

3.3.5 Formular Adauga Distribuitori

A screenshot of a computer

Description automatically generated

Acest formular permite asocierea unui produs cu un furnizor prin selecția **ID Produs** și **ID Furnizor** din dropdown-uri. De asemenea, include câmpurile **Nume Distribuitor**, **Preț Unitar** și **Stoc Disponibil**. Butonul **Adaugă** actualizează tabelul de distribuitori și închide fereastra. Tabela de legătura Distribuitor(Magazin) permite introducerea mai multor produse de la diverși Furnizori. După adaugarea datelor si salvarea acestora prin apasarea butonului Adauga, user-ul este redirecționat înapoi la tabelul respectiv, unde datele sunt actualizate automat.

3.3.6 Formular Adauga Produse

A screenshot of a computer

Description automatically generated

Câmpurile **Denumire**, **Descriere** și **Cantitate** sunt obligatorii pentru adăugarea unui produs. Butonul **Adaugă** salvează datele. După adaugarea datelor si salvarea acestora prin apasarea butonului Adauga, user-ul este redirecționat înapoi la tabelul respectiv, unde datele sunt actualizate automat.

3.3.7 Formular Adauga Comenzi

A screenshot of a computer

Description automatically generated

Formularul de comandă necesită selectarea **ID Client** și **ID Furnizor** din liste dropdown, precum și completarea **Datei Comenzii**, **Statusului** (e.g., "În curs", "Finalizată") și **Valoare Comandă**. Butonul **Adaugă** actualizează tabelul de comenzi după salvare. După adaugarea datelor si salvarea acestora prin apasarea butonului Adauga, user-ul este redirecționat înapoi la tabelul respectiv, unde datele sunt actualizate automat.

3.3.8 Formular Modifica Clienți

A screenshot of a computer

Description automatically generated

Formularele de modificare au același layout ca cele de adăugare, dar câmpurile sunt precompletate cu datele entității selectate. Utilizatorul selectează un rând din tabel, apasă **Modifică**, și editează câmpurile în formularul deschis. Utilizatorul poate edita orice câmp și apăsa **Modifică** pentru a salva schimbările. Schimbările se reflectă imediat în tabel după confirmare.

3.3.9 Formular Modifica Furnizori

A screenshot of a computer

Description automatically generated

3.3.10 Formular Modifica Distribuitori

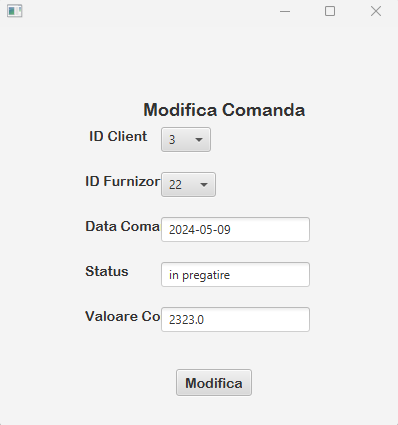


3.3.11 Formular Modifica Produse

A screenshot of a computer

Description automatically generated

3.3.12 Formular Modifica Comenzi



3.3.13 Meniu Principal “Clienti”

A screenshot of a computer

Description automatically generatedAcesta nu contine si butoanele de incarca,adauga,modifica sau stergere, fiind posibila doar vizualizarea tabelelor. Clientul poate naviga între tabele (**Produse**, **Furnizori**, **Comenzi**) folosind aceeași bară laterală ca Adminul, dar fără butoane CRUD.

3.3.14 Formular Inregistrare Client

A screenshot of a computer

AI-generated content may be incorrect.  
Formularul de înregistrare include câmpuri pentru **Nume**, **Prenume**, **Adresă**, **Email**, **Username** și **Parolă**. Parola este hash-uită cu BCrypt înainte de salvare. Butonul **Înregistrare** creează un cont de utilizator cu rolul **Client** și un client în tabelul **clienti**.

3.3.15 Formular Inregistrare Furnizor

A screenshot of a computer

AI-generated content may be incorrect.

**Descriere:**  
Formularul de înregistrare include câmpuri pentru **Nume**, **Prenume**, **Adresă**,  **Username** și **Parolă**. Parola este hash-uită cu BCrypt înainte de salvare. Butonul **Înregistrare** creează un cont de utilizator cu rolul **Furnizor** și un client în tabelul **furnizori**.

3.4 Gestionarea conexiunii cu baza de date

**Cum funcționează conexiunea la baza de date?**

Codul de mai jos gestionează conexiunea la baza de date MySQL folosind JDBC (Java Database Connectivity).

A screenshot of a computer code

AI-generated content may be incorrect.

A white background with blue text

AI-generated content may be incorrect.

**Explicație:**

1. **Class.forName("com.mysql.cj.jdbc.Driver"):**
   * Încarcă driverul JDBC pentru MySQL. Acest pas este necesar pentru a permite Java să comunice cu baza de date MySQL.
2. **DriverManager.getConnection(...):**
   * Creează o conexiune la baza de date folosind URL-ul, username-ul și parola furnizate.
   * URL-ul conține:
     + jdbc:mysql:// – Protocolul pentru conexiunea la MySQL.
     + localhost:3306 – Adresa și portul serverului de baze de date.
     + proiect3 – Numele bazei de date.
     + useSSL=false – Dezactivează SSL pentru conexiune (folosit în medii de dezvoltare, dar nu în producție).
3. **con.close():**
   * Închide conexiunea pentru a elibera resursele sistemului. Este important să închizi conexiunea după ce ai terminat de folosit-o.

**3.5 Implementarea operațiunilor CRUD**

**Pattern-ul CRUD în Aplicație:**

**Create (Adăugare Client):**

public void adaugaClient(String nume, String prenume, String adresa, String email) throws SQLException {

String sql = "INSERT INTO clienti (nume, prenume, adresa, email) VALUES (?, ?, ?, ?)";

try (PreparedStatement pstmt = con.prepareStatement(sql)) {

pstmt.setString(1, nume);

pstmt.setString(2, prenume);

pstmt.setString(3, adresa);

pstmt.setString(4, email);

pstmt.executeUpdate();

}

}

**Read (Încărcare Date):**

public ObservableList<Client> incarcaClienti() throws SQLException {

ObservableList<Client> clienti = FXCollections.observableArrayList();

try (Statement stmt = con.createStatement();

ResultSet rs = stmt.executeQuery("SELECT \* FROM clienti")) {

while (rs.next()) {

clienti.add(new Client(

rs.getInt("idclient"),

rs.getString("nume"),

rs.getString("prenume"),

rs.getString("adresa"),

rs.getString("email")

));

}

}

return clienti;

}

**Update (Modificare Client):**

public void modificaClient(int idclient, String numeNou, String prenumeNou) throws SQLException {

String sql = "UPDATE clienti SET nume = ?, prenume = ? WHERE idclient = ?";

try (PreparedStatement pstmt = con.prepareStatement(sql)) {

pstmt.setString(1, numeNou);

pstmt.setString(2, prenumeNou);

pstmt.setInt(3, idclient);

pstmt.executeUpdate();

}

}

**Delete (Ștergere Client):**

public void stergeClient(int idclient) throws SQLException {

try (PreparedStatement pstmt = con.prepareStatement(

"DELETE FROM clienti WHERE idclient = ?")) {

pstmt.setInt(1, idclient);

pstmt.executeUpdate();

}

}

**Integrare cu JavaFX:**

tabela\_Clienti.setItems(FXCollections.observableArrayList(dbOperations.incarcaClienti()));

atribut\_numeC.setCellValueFactory(new PropertyValueFactory<>("nume"));

3.6 Gestionarea securității utilizatorului

**Sistemul de Autentificare:**

A close-up of a computer screen

AI-generated content may be incorrect.

**Componente Cheie:**

1. **Hashing Parole:** Folosire BCrypt

String hashedPassword = BCrypt.hashpw(rawPassword, BCrypt.gensalt());

1. **Management Roluri:**

CREATE TABLE utilizatori (

role ENUM('client', 'furnizor') NOT NULL

);

**Mecanisme de Protecție:**

* Parametrizare interogări SQL pentru prevenire SQL injection
* Validare input în controalele JavaFX
* Separare vizuală a funcționalităților pe roluri

**Logout Securizat:**

A computer screen shot of a program

AI-generated content may be incorrect.

**Concluzii:**

**Proiectul realizat demonstrează integrarea eficientă a tehnologiilor JavaFX și MySQL pentru dezvoltarea unei aplicații desktop care permite gestionarea datelor despre produse, furnizori, distribuitori, clienți și comenzi. Aplicația include o funcționalitate de autentificare care diferențiază accesul utilizatorilor în funcție de rolul acestora:**

* ***Admini (Furnizori):* Aceștia au acces complet la aplicație, putând efectua toate operațiunile CRUD (creare, citire, actualizare, ștergere) asupra tuturor tabelelor din baza de date. Acest lucru asigură o gestionare completă a informațiilor legate de produse, comenzi și clienți.**
* ***Clienți:* Utilizatorii autentificați ca clienți au acces limitat, putând doar să vizualizeze datele disponibile în aplicație, fără a avea posibilitatea de a modifica sau șterge informații. Aceasta asigură integritatea datelor și o experiență sigură pentru utilizatorii finali.**

**Utilizarea Scene Builder a simplificat procesul de creare a interfeȟei grafice, iar MySQL Connector/J a facilitat comunicarea fiabilă între aplicație și baza de date. Proiectul a atins toate obiectivele propuse, asigurând o experiență de utilizare intuitivă și funcționalități complete pentru gestionarea datelor.**

**Posibile îmbunătățiri:**

1. **Optimizarea performanței bazei de date: Indexarea tabelelor și optimizarea interogărilor SQL ar putea crește viteza de acces la date.**
2. **Extinderea funcționalitċții de autentificare: Implementarea autentificării pe mai multe niveluri, cu roluri suplimentare (de exemplu, operatori de vânzări), ar permite o gestionare mai granulară a accesului.**
3. **Interfață responsive: Adaptarea interfeței pentru dispozitive mobile ar crește accesibilitatea aplicației.**
4. **Export de date: Introducerea unei funcționalități de export al datelor în formate populare (CSV, PDF) pentru rapoarte și analize suplimentare.**

**Bibliografie:**

1. **Oracle. (n.d.). *JavaFX Documentation*. Retrieved from** [**https://openjfx.io/**](https://openjfx.io/)
2. **Oracle. (n.d.). *MySQL Documentation*. Retrieved from** [**https://dev.mysql.com/doc/**](https://dev.mysql.com/doc/)
3. **Gluon. (n.d.). *Scene Builder Documentation*. Retrieved from** [**https://gluonhq.com/products/scene-builder/**](https://gluonhq.com/products/scene-builder/)
4. **Oracle. (n.d.). *MySQL Connector/J Documentation*. Retrieved from** [**https://dev.mysql.com/doc/connector-j/en/**](https://dev.mysql.com/doc/connector-j/en/)