Cuprins

1. Introducere

Perioada de desfasurare a proiectului: 01-07-2024 – 16-07-2024

Competente exersate: Java, MySQL, JDBC

Tema proiectului: Firmă de închirieri casete/DVD-uri audio/video/jocuri

Prin acest proiect am dorit sa implementez o aplicatie simpla, de baza pentru o firma ce pune la dispozitia clientilor sai diverse filme, jocuri si altele spre inchiriere, pe o durata limitata de timp, o aplicatie in care clientii se pot inregistra, se pot conecta si pot vedea ce astfel de ’obiecte’ au inchiriat, dar si sa inchirieze altele.

In crearea acestei aplicatii m-am folosit de limbajul de programare Java, si de sistemul de gestiune al bazelor de date MySQL, dar si de driverul ce face conexiunea intre acestea, numit JDBC (Java Data base Connectivity).

Java este un limbaj de programare orientat pe obiecte, robust, sigur și independent de platformă, dezvoltat de Sun Microsystems (acum parte din Oracle Corporation). A fost lansat în 1995 și de atunci a devenit unul dintre cele mai populare limbaje de programare din lume, utilizat în diverse domenii, de la aplicații web și mobile, la software pentru întreprinderi și sisteme integrate.

MySQL este un sistem de gestiune al bazelor de date relaționale (RDBMS) open-source, dezvoltat inițial de compania suedeză MySQL AB și achiziționat ulterior de Sun Microsystems. MySQL este unul dintre cele mai utilizate sisteme de baze de date din lume, fiind cunoscut pentru viteza, fiabilitatea și ușurința de utilizare.

JDBC (Java Database Connectivity) este un API (Application Programming Interface) furnizat de Java, care permite accesul și manipularea bazelor de date relaționale într-un mod standardizat. JDBC face parte din Java Standard Edition și oferă un set de clase și interfețe

pentru conectarea la baze de date, executarea interogărilor SQL și manipularea rezultatelor obținute.

2. Desfasurare

Dezvoltarea proiectului a inceput greu si incet, avand 0 experienta cu MySQL si in jur de 5% cunostinte in cee ace priveste Java. Primele 2 zile au fost zile de informare, in care nu am facut nimic decat sa citesc si sa urmaresc tutorial despre ce este MySQL, cum se lucreaza cu acesta si cum se foloseste JDBC pentru a face conexiunea intre Java si MySQL.

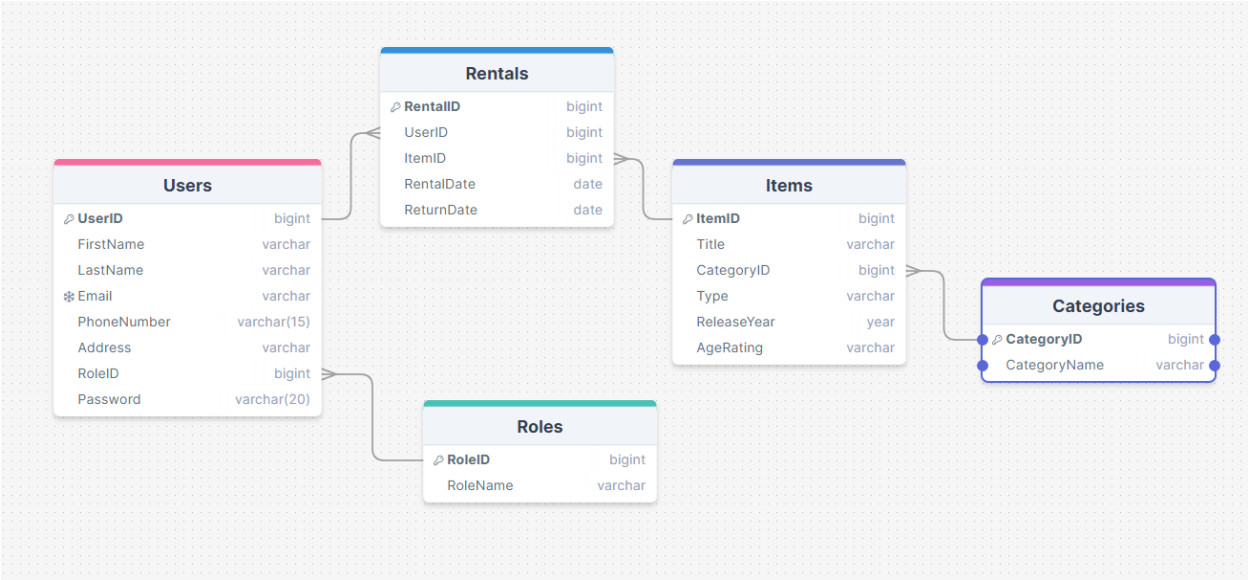
Dupa aceste zile, am inceput prin a descarca programele necesare, si anume Apache NetBeans IDE, programul in care am scris codul, MySQL Workbench, programul in care am create si populat baza de date, si driverul JDBC pentru a le putea adduce impreuna.

Au urmat zile de testare, zile in care am exersat conexiunea la baza de date din Java, am exersat crearea de interfete grafice in Java, lucru care a fost facilitate de utilizarea IDE-ului NetBeans, deoarece acesta are un GUI (Graphical User Interface) designer ce face lucrul cu interfetele grafice mult mai usor.

Abia dupa o saptamana de la alegerea acestei aplicatii am inceput lucrul la proiect in sine. Am inceput prin a crea o interfata grafica simpla, care initial mi-a dat multe batai de cap, pana am descoperit puterea unui mod de aranjare a elementelor in interfetele grafice in Java, si anume BoxLayout. Din momentul in care am descoperit cum functioneaza, designul a venit de la sine si asezarea paginilor a mers perfect.

Dup ace am rezolvat problema partii de design, am inceput implementarea functiilor ce leaga aplicatia de baza de date, si a celor ce fac afisari, cautari, inserari si stergeri, dar si functiile ce se ocupa de inregistrarea si conectarea utilizatorilor.

Intr-un final, dupa doua saptamani de studiere, informare, vizionat tutoriale si invatare de metode noi, am ajuns in punctul in care pot spune ca am o aplicatie terminata.



Aceasta este diagrama bazei de date, unde se pot vedea tabelele impreuna cu atributele fiecareia. Pentru a face aceasta diagrama, am folosit o aplicatie web, numita DrawSQL.

```java
// Functie pentru verificarea corectitudinii datelor de logare
public static char emailCheck(String email, String password){
    try{
        String query = "SELECT roleid FROM users WHERE email = ? AND password = ?";
        PreparedStatement stmt = connection.prepareStatement(query);
        stmt.setString(1, email);
        stmt.setString(2, password);
        ResultSet rs = stmt.executeQuery();
        userEmail = email;
        if(rs.next())
            if(rs.getInt("roleID")==2)
                return 'u';
            else
                return 'a';

    }catch(SQLException e){
        e.printStackTrace();
    }
    return 'i';
}
```

Aceasta este functia care preia din campurile de text datele introduse pentru logare si returneaza, atunci cand gaseste o intrare care se potriveste, un caracter care reprezinta tipul de utilizator care se logheaza, pentru a sti pe ce pagina il vom trimite.

```java
// Functie ce populeaza tabelul cu itemele inchiriate de utilizator
private void userViewPanelComponentShown(java.awt.event.ComponentEvent evt) {
    Connection conn = JDBC.getConnection();
    if (conn != null) {
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try {
            String query =  "SELECT Title, RentalDate AS \"Rented on\", ReturnDate AS \"Due on\" FROM Items, Rentals, Users\n" +
                            "WHERE email = '" + userEmail + "' AND users.userid = rentals.userid AND items.itemid = rentals.itemid;";
            stmt = conn.prepareStatement(query);
            rs = stmt.executeQuery();

            // Obține metadatele pentru a obține numele coloanelor
            ResultSetMetaData rsmd = rs.getMetaData();
            int columnCount = rsmd.getColumnCount();
            String[] columnNames = new String[columnCount];
            for (int i = 1; i <= columnCount; i++) {
                columnNames[i-1] = rsmd.getColumnName(i);
            }

            // Adaugă datele în modelul de tabel
            DefaultTableModel model = new DefaultTableModel(columnNames, 0);
            while (rs.next()) {
                Object[] row = new Object[columnCount];
                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }
                model.addRow(row);
            }

            // Setează modelul în JTable
            userRentalsTable.setModel(model);

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
}
```

Aceasta este functia care cauta in baza de date utilizatorul care este logat si afiseaza intr-un container de tip tabel toate itemele pe care acesta le are inchiriate, impreuna cu titlul lor, data inchirierii si data de returnare.

```java
// Functie pentru conectarea la baza de date
public static Connection getConnection() {
    if (connection == null) {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection(URL, USER, PASSWORD);
            System.out.println("Conexiune reusita la baza de date!");
        } catch (ClassNotFoundException | SQLException e) {
            e.printStackTrace();
        }
    }
    return connection;
}
```

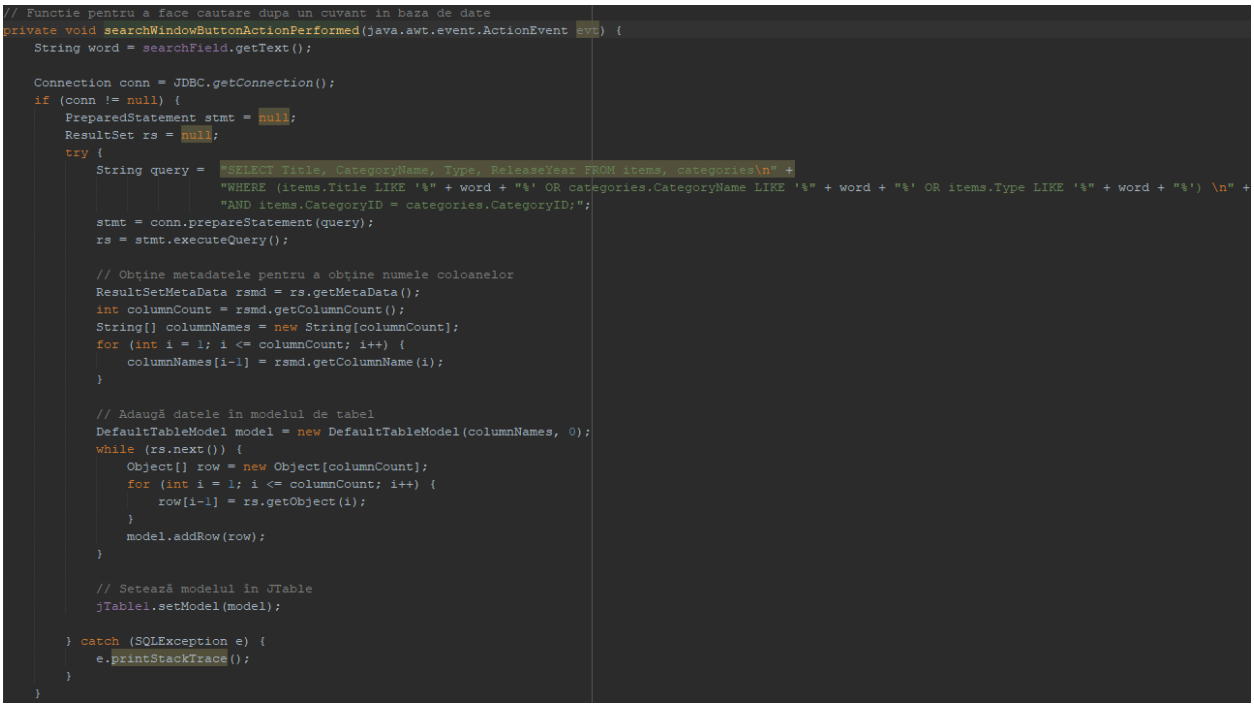
Functie ce face conectarea la baza de date in baza unui URL, a unui user si a unei parole ale acestei baze de date

```java
// Functie pentru stergerea unui item
private void adminDeleteItemsButtonActionPerformed(java.awt.event.ActionEvent evt) {
    int selectedRow = itemsTable.getSelectedRow();
    if (selectedRow != -1) {
        String title = (String)itemsTable.getValueAt(selectedRow, 0);

        Connection conn = JDBC.getConnection();
        if (conn != null) {
            try{
                String query =  "DELETE FROM Items WHERE Title LIKE '" + title + "';";
                Statement stmt = conn.createStatement();
                stmt.executeUpdate(query);
                JOptionPane.showMessageDialog(this, "Item Erased Successfully!");
            }catch(SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Aceasta este functia ce se ocupa de stergerea din baza de date a unui item, atunci cand un administrator face acest lucru. Se preia din table titlul itemului selectat si se sterge randul din baza de date in care apare acesta.

```java
// Functie pentru a adauga o noua inchiriere in baza de date
private void rentButtonActionPerformed(java.awt.event.ActionEvent evt) {
    int selectedRow = jTable1.getSelectedRow();
    if (selectedRow != -1) {
        String email = JDBC.getEmail();
        String title = (String)jTable1.getValueAt(selectedRow, 0);
        int userID = 0;
        int itemID  = 0;


        Connection conn = JDBC.getConnection();
        if (conn != null) {
            PreparedStatement stmt = null;
            ResultSet rs = null;
            try{
                String query =  "SELECT UserID FROM users  WHERE email = '" + email + "';";
                stmt = conn.prepareStatement(query);
                rs = stmt.executeQuery();

                if(rs.next())
                {
                    userID = rs.getInt("UserID");
                }

                query =  "SELECT ItemID FROM items  WHERE Title = '" + title + "';";
                stmt = conn.prepareStatement(query);
                rs = stmt.executeQuery();
                if(rs.next())
                {
                    itemID = rs.getInt("ItemID");
                }

                query = "INSERT INTO  Rentals (UserID, ItemID, RentalDate, ReturnDate)"+
                        "VALUES ('"+userID+"', '"+itemID+"', curdate(), adddate(curdate(), interval 31 day))";
                stmt = conn.prepareStatement(query);
                stmt.executeUpdate(query);
                JOptionPane.showMessageDialog(this, "Rent Successful!");
            }catch(SQLException e) {
                e.printStackTrace();
            }
        }
    }
    else {
        JOptionPane.showMessageDialog(this, "No row selected!");
    }
}
```

Aceasta este functia care introduce in baza de date, in tabela pentru inchirieri, itemul selectat de catre utilizator, cu data curenta, si ii seteaza data de returnare cu 31 de zile in fata.



```java
// Functie pentru a face cautare dupa un cuvant in baza de date
private void searchWindowButtonActionPerformed(java.awt.event.ActionEvent evt) {
    String word = searchField.getText();

    Connection conn = JDBC.getConnection();
    if (conn != null) {
        PreparedStatement stmt = null;
        ResultSet rs = null;
        try {
            String query =  "SELECT Title, CategoryName, Type, ReleaseYear FROM items, categories\n" +
                            "WHERE (items.Title LIKE '%" + word + "%' OR categories.CategoryName LIKE '%" + word + "%' OR items.Type LIKE '%" + word + "%') \n" +
                            "AND items.CategoryID = categories.CategoryID;";
            stmt = conn.prepareStatement(query);
            rs = stmt.executeQuery();

            // Obține metadatele pentru a obține numele coloanelor
            ResultSetMetaData rsmd = rs.getMetaData();
            int columnCount = rsmd.getColumnCount();
            String[] columnNames = new String[columnCount];
            for (int i = 1; i <= columnCount; i++) {
                columnNames[i-1] = rsmd.getColumnName(i);
            }

            // Adaugă datele în modelul de tabel
            DefaultTableModel model = new DefaultTableModel(columnNames, 0);
            while (rs.next()) {
                Object[] row = new Object[columnCount];
                for (int i = 1; i <= columnCount; i++) {
                    row[i-1] = rs.getObject(i);
                }
                model.addRow(row);
            }

            // Setează modelul în JTable
            jTable1.setModel(model);

        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

Aceasta este functia ce se ocupa de cautarea in baza de date a itemelor dupa un cuvant scris de utilizator. Aici se face un query care cauta in titlurile itemelor, in tipul de item sau in ccategoriile din care fac parte orice intrare ce contine cuvantul scris de la tastatura si le afiseaza intr-un table.

3. Concluzii

In urma lucrului la acest proiect pot spune ca am primit o intelegere mai aprofundata a mai multor tehnologii si concept esentiale in dezvoltarea aplicatiilor in limbajul Java, consolidand cunostintele si abilitatile practice. Cateva aspect cheie invatate au fost:

- Dezvoltarea de interfete grafice cu Swing: am invatat cum sa creez interfete grafice si sa gestionez corect componentele intr-o fereastra JFrame cu ajutorul layout-urilor
- Conectarea la baza de date MySQL cu JDBC: am invatat sa folosesc JDBC pentru a conecta aplicatiile Java la o baza de date MySQL, inclusiv sa execut interogari SQL si sa gestionez rezultatele acestora.
- Gestionarea evenimentelor și interacțiunilor utilizatorului: adaugarea si gestionarea ‘ActionListener’-elor pentru butoane si alte componente interactive a fost esentiala pentru a raspunde actiunilor utilizatorului si pentru a actualiza interfata grafica in urma acestor interactiuni.
- Bune practici în dezvoltarea software-ului: am invatat importanta scrierii unui cod clar si modularizat, asistat de comentarii de ofera explicatii, gestionarea eficienta a resurselor, precum conexiunea la baza de date si testarea diferitelor functii implementate pentru a asigura functionarea corecta a aplicatiei.