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

Organizare gratar

Proiect la disciplina

Baze de Date

Student:
Spiridon Ioan
Grupa:
1311A
Cadru didactic coordonator:
Cătălin Mironeanu

CUPRINS

1. Scopul

Scopul acestui proiect a fost acela de a implementa o soluție în legătură cu o problema cauzată de organizarea unui grup pentru a merge la gratar. Astfel prin acest proiect vreau sa ajut acei oameni care au nevoie de o buna organizarea si desfasurare a evenimentelor. În momentul de fata, la acest proiect ar mai putea fi aduse îmbunătățiri pentru o mai buna derulare a lucrurilor.

2. Tehnologii folosite

Pentru realizare s-a folosit limbajul de programare Java (versiunea JDK 21) împreuna cu limbajul SQL.
Din limbajul Java s-a folosit AWT împreuna cu SWING pentru modelarea si interfata aplicatiei iar pentru conectarea la baza de date s-a folosit un driver denumit JDBC (versiunea ojdbc11).
Totodata, pentru tabela Events la introducerea datelor, cea calendaristică mai ales, s-a folosit JDatePicker(versiunea jdatepicker-1.3.4) pentru a alege mai ușor data care se dorește a fi introdusă.

3. SWING

SWING este o librărie JFC(Java Foundation Classes) dar și o extensie pentru AWT(Abstract Window Toolkit). Oferă multe functionalitati precum componente noi, caracteristici extinse și o buna manipulare pentru evenimentele de tip “drag-and-drop”.

Ca exemplu folosit în aplicație:

```
      frame = new JFrame("Organizare gratar");

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Set the size of the frame based on the pc's specs

        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
```

```
        frame.pack();

        screenWidth = screenSize.width * 70 / 100;

        screenHeight = screenSize.height * 70 / 100;

        frame.setSize(screenWidth, screenHeight);

        // The location of the frame is set to center

        frame.setLocationRelativeTo(null);
```

Folosim “frame” pentru a crea o nouă fereastra cu denumirea “`Organizare gratar`” si aplicăm o operație implicită de inchidere. Totodată ii setam dimensiunile ferestrei prin parametrii “`screenWidth`“ si “`screenHeight `” iar prin metoda “`setLocationRelativeTo`” setam aplicație sa nu fie dependentă de nicio margina, adică să fie centrată pe mijloc.

4. AWT

AWT sau Abstract Window Toolkit este un API(Application Programming Interface) folosit in dezvoltarea unui GUI(Graphic Uset Interfaces) sau aplicații de tip fereastra in Java. El face parte din JFC care oferă o modalitate de a crea o aplicație independentă de platforma.

Un exemplu de AWT pe care l-am folosit in aplicatie este:

```
initButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                boolean init = conn.init();

                if(init){
                    System.out.println("Connection established");
                } else {
                    System.out.println("Connection not
established");
                }
            }
        });
```

Acest “`ActionListener`” este o metoda care este utilizată pentru a asculta evenimentele de acțiune generate de componentele grafice, cum ar fi butoanele, meniurile, sau orice alt element, în acest exemplu pentru butonul “`initButton`”. La apăsarea acestui buton se seteaza o conexiunea între aplicație și baza de date la care dorim sa ne conectam

“`boolean init = conn.init();`” iar ea returnează o valoare de tip boolean prin care aflăm dacă conexiunea s-a realizat cu succes sau nu.

5. JDBC

JDBC(Java Database Connectivity) reprezinta un Java API prin care sa ne conectăm și sa executam interogari cu o baza de date. Face parte din JavaSE(java Standard Edition). El folosește anumite drivere prin care conexiunea sa se poate realiza cu succes între aplicația în care este folosită și baza de date la care se dorește sa se ajunga.

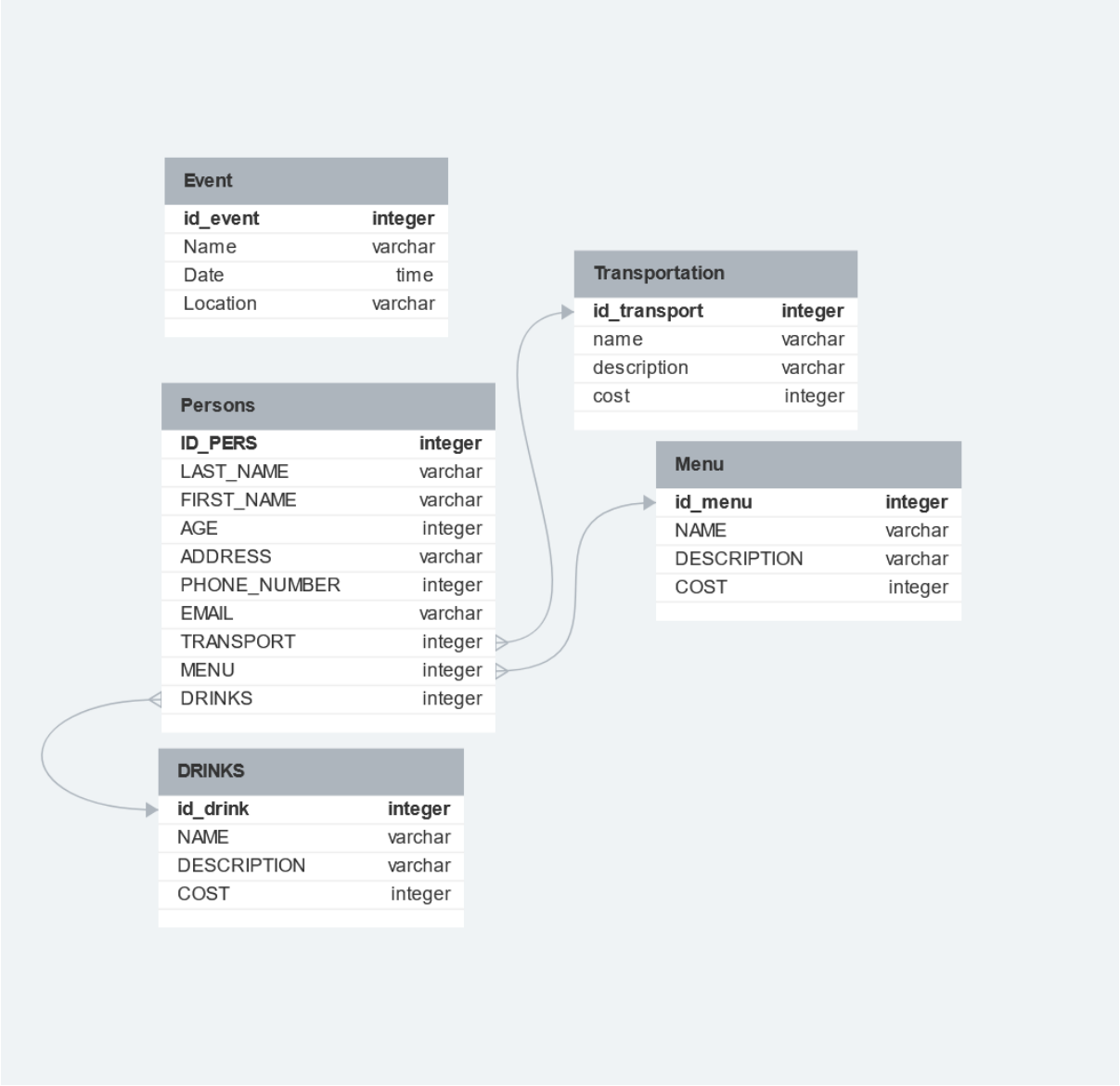
Ca de exemplu:

```
// the connection param
            ods = new OracleDataSource();

ods.setURL("jdbc:oracle:thin:@//bd-dc.cs.tuiasi.ro:1539/orcl");
            ods.setUser("bd161");
            ods.setPassword("bd161");
            // Attempts to establish a connection with the data source
that
            conn = ods.getConnection();
```

“`ods`” este de tipul “`OracleDataSource`” iar cu ajutorul ei ne folosim pentru a crea conexiunea oferindu-i parametri necesari precum, link-ul către baza de date “`jdbc:oracle:thin:@//bd-dc.cs.tuiasi.ro:1539/orcl`”, username-ul “`bd161`” împreuna cu parola “`bd161`” iar mai apoi prin metoda “`getConnection`” se încearcă a se stabili conexiunea.
Pentru a opri conexiunea vom folosi metoda “`conn.close();`”.

6. Tabele folosite



Astfel, sunt folosite 5 tabele. O tabela “Event” care sa descrie evenimentul care va avea loc, impreuna cu locatia si data. O tabela principala “Persons” care va avea date despre fiecare persoana impreuna cu meniul ce il vor avea la eveniment, cu ce vor ajunge acolo, adica transport si ce vor consuma.

La constrangeri, am folosit Unique Key pentru fiecare tabela astfel ID-ul sa fie unic impreuna cu Primary Key. Si totodata, am folosit cateva regex pentru email astfel incat sa aibe un

format adecvat, la numarul de telefon sa fie compus doar in cifre iar First/Last Name sa contina doar litere.

7. Inserarea in tabele

Pentru a insera in tabele informatii oferite de catre utilizator vom folosi metoda "`populateTable`":

```
public boolean populateTable(String query) {
        try {
            conn.setAutoCommit(false);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        try (Statement stmt = conn.createStatement()) {
            int rowsAffected = stmt.executeUpdate(query);
            conn.commit();
            return rowsAffected > 0; // Return true if at least one row
was affected
        } catch (SQLException e) {
            try {
                conn.rollback();
            } catch (SQLException e1) {
                // TODO Auto-generated catch block
                e1.printStackTrace();
            }
            e.printStackTrace();
            return false; // Return false if an exception occurred
        }
    }
```

Astfel aceasta metoda primeste ca parametru un String ce reprezinta secventa SQL care va trebui sa fie executata de catre driver-ul JDBC "`stmt.executeUpdate(query);`". Ea va returna o valoare de tip "`int`" astfel, daca exista randuri afectate, inseamna ca inserarea a avut loc, daca nu, atunci contrariul.

```
String query = "INSERT INTO TRANSPORTATION(NAME, DESCRIPTION, COST)
VALUES('" +
                    inputData1 + "','" + inputData2 + "'," + inputData3
+ ")";
```

“`inputData`” reprezinta datele luate din interfata de la utilizator.

8. Afisarea din tabele

Afisarea datelor din tabele se realizeaza prin metode de felul acesta:

```
public void viewTableTransportation(DefaultTableModel tableModel)
throws SQLException {
        String query = "select ID_TRANSPORT, NAME, DESCRIPTION, COST
from TRANSPORTATION";
        try (Statement stmt = conn.createStatement()) {
        ResultSet rs = stmt.executeQuery(query);
        while (rs.next()) {
            int id = rs.getInt("ID_TRANSPORT");
            String name = rs.getString("NAME");
            String description = rs.getString("DESCRIPTION");
            int price = rs.getInt("COST");
            Object[] row = {id, name, description, price};
            tableModel.addRow(row);
        }
        } catch (SQLException e) {
        e.printStackTrace();
        }
    }
```

Secventa SQL selecteaza toate coloanele care sunt necesare pentru afisarea corespunzatoare a informatiilor. “`tableModel`” reprezinta o tabela pe interfata aplicatiei in care vom adauga informatiile extrase.

9. Stergerea din tabele

Pentru a sterge date din tabele ne vom folosit de metoda “`deleteTable`”:

```
public boolean deleteTable(String query) {
        try {
            conn.setAutoCommit(false);
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();}
        try (Statement stmt = conn.createStatement()) {
            int rowsAffected = stmt.executeUpdate(query);
            conn.commit();
            return rowsAffected > 0;
        } catch (SQLException e) {
```

```
            e.printStackTrace();
            try {
                conn.rollback();
            } catch (SQLException e1) {
                e1.printStackTrace();}
            return false; // Return false if an exception occurred
        }
    }
```

Ca si prin metoda de a insera date in tabele, ne vom folosi de o secventa SQL pe care o vom executa pe tabela pe care dorim.

```
String query = "DELETE FROM TRANSPORTATION WHERE ID_TRANSPORT = ";
```

Mai exact, vom cere utilizatorului sa furnizeze ID-ul acelui rand care doreste a fi sters iar prin acest ID, vom merge la tabela corespunzatoare si il vom sterge din aceasta tabela.

Bibliografie:

https://www.geeksforgeeks.org/java-awt-tutorial/
https://www.geeksforgeeks.org/introduction-to-java-swing/
https://www.javatpoint.com/java-jdbc