package org.example;

import java.awt.\*;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.sql.\*;

import javax.swing.\*;

import javax.swing.table.DefaultTableModel;

import javax.swing.\*;

import java.awt.\*;

import java.sql.\*;

import java.util.Vector;

public class NotesDatabaseGUI extends JFrame {

private JTable usersTable;

private JTable notesTable;

private JTable datesTable;

private JTable statusTable;

private DefaultTableModel usersTableModel;

private DefaultTableModel notesTableModel;

private DefaultTableModel datesTableModel;

private DefaultTableModel statusTableModel;

private Connection connection;

public NotesDatabaseGUI() {

super("База данных заметок");

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setSize(800, 600);

setLocationRelativeTo(null);

// Создаем таблицы

usersTable = new JTable();

notesTable = new JTable();

datesTable = new JTable();

statusTable = new JTable();

// Создаем модели таблиц

usersTableModel = new DefaultTableModel();

notesTableModel = new DefaultTableModel();

datesTableModel = new DefaultTableModel();

statusTableModel = new DefaultTableModel();

// Устанавливаем модели для таблиц

usersTable.setModel(usersTableModel);

notesTable.setModel(notesTableModel);

datesTable.setModel(datesTableModel);

statusTable.setModel(statusTableModel);

// Создаем вкладки

JTabbedPane tabbedPane = new JTabbedPane();

tabbedPane.addTab("Пользователи", new JScrollPane(usersTable));

tabbedPane.addTab("Заметки", new JScrollPane(notesTable));

tabbedPane.addTab("Дата и время", new JScrollPane(datesTable));

tabbedPane.addTab("Статус заметки", new JScrollPane(statusTable));

// Создаем кнопку "Обновить"

JButton updateButton = new JButton("Обновить");

updateButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

updateTables();

}

});

// Добавляем кнопку в панель инструментов

JToolBar toolBar = new JToolBar();

toolBar.add(updateButton);

// Добавляем вкладки и кнопку в фрейм

getContentPane().add(toolBar, BorderLayout.NORTH);

getContentPane().add(tabbedPane, BorderLayout.CENTER);

// Устанавливаем соединение с базой данных

connectToDatabase();

// Заполняем таблицы

updateTables();

setVisible(true);

}

private void connectToDatabase() {

try {

connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/notes", "root", "122333444455555");

System.out.println("Соединение с базой данных установлено.");

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Ошибка при подключении к базе данных: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

private void updateTables() {

try {

// Обновляем данные в таблице "Пользователи"

updateUsersTable();

// Обновляем данные в таблице "Заметки"

updateNotesTable();

// Обновляем данные в таблице "Дата и время"

updateDatesTable();

// Обновляем данные в таблице "Статус заметки"

updateStatusTable();

} catch (SQLException e) {

JOptionPane.showMessageDialog(this, "Ошибка при обновлении данных: " + e.getMessage(), "Ошибка", JOptionPane.ERROR\_MESSAGE);

}

}

// Методы для обновления таблиц

private void updateUsersTable() throws SQLException {

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT \* FROM users");

usersTableModel.setDataVector(getResultSetData(rs), new String[]{"ID", "Логин", "Пароль", "Имя", "Фамилия"});

rs.close();

stmt.close();

}

private void updateNotesTable() throws SQLException {

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT \* FROM notes");

notesTableModel.setDataVector(getResultSetData(rs), new String[]{"Номер", "Заметка", "Доп. файл", "ID пользователя"});

rs.close();

stmt.close();

}

private void updateS tatusTable() throws SQLException {

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT \* FROM status");

statusTableModel.setDataVector(getResultSetData(rs), new String[]{"Номер", "Статус заметки"});

rs.close();

stmt.close();

}

private void updateDatesTable() throws SQLException {

Statement stmt = connection.createStatement(ResultSet.TYPE\_SCROLL\_INSENSITIVE, ResultSet.CONCUR\_UPDATABLE);

ResultSet rs = stmt.executeQuery("SELECT \* FROM dates");

datesTableModel.setDataVector(getResultSetData(rs), new String[]{"Номер", "Дата создания", "Время создания", "Дата выполнения"});

rs.close();

stmt.close();

}

private Object[][] getResultSetData(ResultSet rs) throws SQLException {

// Получаем количество столбцов

ResultSetMetaData metaData = rs.getMetaData();

int columnCount = metaData.getColumnCount();

// Получаем количество строк

rs.last();

int rowCount = rs.getRow();

rs.beforeFirst();

// Создаем двумерный массив для хранения данных

Object[][] data = new Object[rowCount][columnCount];

// Заполняем массив данными

int row = 0;

while (rs.next()) {

for (int col = 0; col < columnCount; col++) {

data[row][col] = rs.getObject(col + 1);

}

row++;

}

return data;

}

public static void main(String[] args) {

SwingUtilities.invokeLater(new Runnable() {

@Override

public void run() {

new NotesDatabaseGUI();

}

});

}

}