Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа № 6

По дисциплине ГИИС

Тема: «Разработка оконного приложения для работы с БД»

Выполнил:

Студент 4 курса

Группы ИИ-16

Журавлёв В.А.

Проверил:

Михно Е.В.

Брест, 2021

**Цель работы:** разработать оконное приложение для работы с БД.

**Ход работы:**

**Задание:**

Выполнить проектирование БД согласно варианту, представить результаты в виде физической и логической схема БД. Вариант №3 (Троллейбусный парк).

**Код программы:**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class TrolleybusPark {

public Statement statement = null;

public TrolleybusPark() {

try {

Connection connection = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/giis", "postgres", "postgres");

statement = connection.createStatement();

} catch (Exception e) {

e.printStackTrace();

}

}

public void addNote(String table, Object[] objects) {

try {

switch (table) {

case "drivers": {

statement.execute("INSERT INTO drivers (first\_name, second\_name, experience) VALUES ('" + objects[0] + "', '" + objects[1] + "', '" + objects[2] + "');");

break;

}

case "trolleybuses": {

statement.execute("INSERT INTO trolleybuses (number, mileage, production\_year) VALUES ('" + objects[0] + "', '" + objects[1] + "', '" + objects[2] + "');");

break;

}

case "routes": {

statement.execute("INSERT INTO routes (description, trolleybus\_id, driver\_id, route\_date) VALUES ('" + objects[0] + "', '" + objects[1] + "', '" + objects[2] + "', '" + objects[3] + "');");

break;

}

}

} catch (SQLException e) {

System.out.println("Adding error :" + e.getMessage());

}

}

public void removeNote(String table, Integer index) {

try {

switch (table) {

case "drivers": {

statement.execute("DELETE FROM drivers WHERE id=" + index + ";");

break;

}

case "trolleybus": {

statement.execute("DELETE FROM trolleybuses WHERE id=" + index + ";");

break;

}

case "routes": {

statement.execute("DELETE FROM routes WHERE id=" + index + ";");

break;

}

}

} catch (SQLException e) {

System.out.println("Deleting error :" + e.getMessage());

}

}

public void updateNote(String table, Object[] objects, Integer index) {

try {

switch (table) {

case "drivers": {

statement.execute("UPDATE drivers SET first\_name='" + objects[0] + "', second\_name='" + objects[1] + "', experience='" + objects[2] + "' WHERE id=" + index + ";");

break;

}

case "trolleybuses": {

statement.execute("UPDATE trolleybuses SET number='" + objects[0] + "', mileage='" + objects[1] + "', production\_year='" + objects[2] + "' WHERE id=" + index + ";");

break;

}

case "routes": {

statement.execute("UPDATE routes SET description='" + objects[0] + "', trolleybus\_id='" + objects[1] + "', driver\_id='" + objects[2] + "', route\_date='" + objects[3] + "' WHERE id=" + index + ";");

break;

}

}

} catch (SQLException e) {

System.out.println("Updating error :" + e.getMessage());

}

}

}

import javax.swing.\*;

import java.awt.\*;

import java.sql.ResultSet;

import java.sql.ResultSetMetaData;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.LinkedHashMap;

public class TrolleybusParkForm extends JFrame {

Integer selectedTab = 0;

TrolleybusPark database;

JTabbedPane tabbedPane = new JTabbedPane();

ArrayList<JTable> tables;

String[] tabs = {"Drivers", "Trolleybuses", "Routes"};

JPanel panel = new JPanel(new BorderLayout());

public TrolleybusParkForm() {

tables = new ArrayList<>();

database = new TrolleybusPark();

createWindow();

tables.add(createDriversTable());

tables.add(createTrolleybusesTable());

tables.add(createRoutesTable());

for (int i = 0; i < tables.size(); i++) {

JScrollPane scroll = new JScrollPane(tables.get(i));

tables.get(i).setFillsViewportHeight(true);

tabbedPane.add(tabs[i], scroll);

}

panel.add(bottomPanel(), BorderLayout.SOUTH);

panel.add(tabbedPane, BorderLayout.CENTER);

tabbedPane.addChangeListener(e -> selectedTab = ((JTabbedPane) e.getSource()).getSelectedIndex());

add(panel);

setVisible(true);

}

private void createWindow() {

setTitle("Trolleybus Park");

setSize(600, 600);

setLocationRelativeTo(null);

setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

}

public void update() {

remove(panel);

int selectedI = tabbedPane.getSelectedIndex();

tabbedPane.removeAll();

panel = new JPanel(new BorderLayout());

for (int i = 0; i < tables.size(); i++) {

JScrollPane scroll = new JScrollPane(tables.get(i));

tables.get(i).setFillsViewportHeight(true);

tabbedPane.add(tabs[i], scroll);

}

panel.add(bottomPanel(), BorderLayout.SOUTH);

panel.add(tabbedPane, BorderLayout.CENTER);

tabbedPane.addChangeListener(e -> selectedTab = ((JTabbedPane) e.getSource()).getSelectedIndex());

add(panel);

revalidate();

tabbedPane.setSelectedIndex(selectedI);

repaint();

}

private JTable createDriversTable() {

return createTable("SELECT \* FROM drivers");

}

private JTable createRoutesTable() {

return createTable("SELECT id,trolleybus\_id,driver\_id,route\_date, description FROM routes");

}

private JTable createTrolleybusesTable() {

return createTable("SELECT \* FROM trolleybuses");

}

private JTable createTable(String query) {

JTable table = null;

try {

ResultSet resultSet = database.statement.executeQuery(query);

ResultSetMetaData resultSetMetaData = resultSet.getMetaData();

String[] columnNames = new String[resultSetMetaData.getColumnCount()];

for (int i = 1; i <= resultSetMetaData.getColumnCount(); i++)

columnNames[i - 1] = resultSetMetaData.getColumnLabel(i);

ArrayList<ArrayList<Object>> dataList = new ArrayList<>();

for (int i = 0; resultSet.next(); i++) {

dataList.add(new ArrayList<>());

for (String value : columnNames) {

dataList.get(i).add(resultSet.getObject(value));

}

}

Object[][] data = new Object[dataList.size()][dataList.get(0).size()];

for (int i = 0; i < dataList.size(); i++)

for (int j = 0; j < dataList.get(0).size(); j++)

data[i][j] = dataList.get(i).get(j);

table = new JTable(data, columnNames);

table.setSelectionMode(0);

} catch (SQLException e) {

e.printStackTrace();

}

return table;

}

private JPanel bottomPanel() {

JPanel panel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

JButton addButton = new JButton("Add");

addButton.addActionListener(e -> {

editForm(-1, -1);

});

JButton edit = new JButton("Edit");

edit.addActionListener(e -> {

editForm((Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 0), tables.get(selectedTab).getSelectedRow());

// if(selectedTab.equals(2)){

// editForm((Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 0), tables.get(selectedTab).getSelectedRow());

// } else {

// editForm((Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 3), tables.get(selectedTab).getSelectedRow());

// }

});

JButton remove = new JButton("Remove");

remove.addActionListener(e -> {

switch (selectedTab) {

case 0: {

database.removeNote("drivers", (Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 0));

// database.removeNote("drivers", (Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 3));

tables.set(0, createDriversTable());

update();

break;

}

case 1: {

database.removeNote("trolleybus", (Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 0));

// database.removeNote("trolleybus", (Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 3));

tables.set(1, createTrolleybusesTable());

update();

break;

}

case 2: {

database.removeNote("routes", (Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 0));

// database.removeNote("routes", (Integer) tables.get(selectedTab).getValueAt(tables.get(selectedTab).getSelectedRow(), 0));

tables.set(2, createRoutesTable());

update();

break;

}

}

});

panel.add(addButton);

panel.add(edit);

panel.add(remove);

return panel;

}

public void func(JPanel p, JComponent c, Integer x, Integer y, Integer w, Integer h, boolean b, GridBagConstraints gbc) {

gbc.gridx = x;

gbc.gridy = y;

gbc.gridwidth = w;

gbc.gridheight = h;

gbc.fill = b ? GridBagConstraints.HORIZONTAL : GridBagConstraints.VERTICAL;

p.add(c, gbc);

}

private void editForm(Integer index, Integer row) {

JFrame frame = new JFrame();

GridBagConstraints gbc = new GridBagConstraints();

JPanel panel = new JPanel(new BorderLayout());

JPanel p1 = new JPanel(new GridBagLayout());

gbc.insets = new Insets(5, 5, 5, 5);

gbc.anchor = GridBagConstraints.NORTH;

JPanel p2 = new JPanel(new FlowLayout(FlowLayout.RIGHT));

JButton save = new JButton("Save");

JButton cancel = new JButton("Cancel");

switch (selectedTab) {

// Drivers

case 0: {

JTextField firstName = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 1).toString(), 10);

func(p1, new JLabel("First name"), 0, 0, 1, 1, true, gbc);

func(p1, firstName, 1, 0, 4, 1, true, gbc);

JTextField secondName = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 2).toString(), 10);

func(p1, new JLabel("Second name"), 0, 1, 1, 1, true, gbc);

func(p1, secondName, 1, 1, 4, 1, false, gbc);

JTextField experience = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 3).toString(), 10);

func(p1, new JLabel("Experience"), 0, 2, 1, 1, true, gbc);

func(p1, experience, 1, 2, 4, 1, false, gbc);

save.addActionListener(e -> {

try {

if (firstName.getText().equals("") || secondName.getText().equals("") || experience.getText().equals("")) {

throw new Exception();

}

if (Double.parseDouble(experience.getText()) < 0) {

throw new NumberFormatException();

}

if (index.equals(-1)) {

database.addNote("drivers", new Object[]{

firstName.getText(),

secondName.getText(),

Double.parseDouble(experience.getText())

});

} else {

database.updateNote("drivers", new Object[]{

firstName.getText(),

secondName.getText(),

Double.parseDouble(experience.getText())

}, index);

}

frame.dispose();

frame.setVisible(false);

tables.set(0, createDriversTable());

update();

} catch (NumberFormatException exception) {

JOptionPane.showMessageDialog(this, "Experiences must be positive number", "Error", JOptionPane.OK\_OPTION);

} catch (Exception generalException) {

JOptionPane.showMessageDialog(this, "Empty fields", "Error", JOptionPane.OK\_OPTION);

}

});

cancel.addActionListener(e -> {

frame.dispose();

frame.setVisible(false);

});

break;

}

// Trolleybuses

case 1: {

JTextField number = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 1).toString(), 10);

func(p1, new JLabel("Number"), 0, 0, 1, 1, true, gbc);

func(p1, number, 1, 0, 4, 1, true, gbc);

JTextField mileage = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 2).toString(), 10);

func(p1, new JLabel("Mileage"), 0, 1, 1, 1, true, gbc);

func(p1, mileage, 1, 1, 4, 1, false, gbc);

JTextField productionYear = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 3).toString(), 10);

func(p1, new JLabel("Production year"), 0, 2, 1, 1, true, gbc);

func(p1, productionYear, 1, 2, 4, 1, false, gbc);

save.addActionListener(e -> {

try {

if (number.getText().equals("") || mileage.getText().equals("") || productionYear.getText().equals(""))

throw new Exception();

if (index.equals(-1)) {

database.addNote("trolleybuses", new Object[]{

number.getText(),

Integer.parseInt(mileage.getText()),

Integer.parseInt(productionYear.getText())

});

} else {

database.updateNote("trolleybuses", new Object[]{

number.getText(),

Double.parseDouble(mileage.getText()),

Integer.parseInt(productionYear.getText())}, index);

}

frame.dispose();

frame.setVisible(false);

tables.set(1, createTrolleybusesTable());

update();

} catch (NumberFormatException exception) {

JOptionPane.showMessageDialog(this, "Fields must be number", "Error", JOptionPane.OK\_OPTION);

} catch (Exception generalException) {

JOptionPane.showMessageDialog(this, "Empty fields", "Error", JOptionPane.OK\_OPTION);

}

});

cancel.addActionListener(e -> {

frame.dispose();

frame.setVisible(false);

});

break;

}

// Routes

case 2: {

LinkedHashMap<String, Integer> drivers = new LinkedHashMap<>();

LinkedHashMap<String, Integer> trolleybuses = new LinkedHashMap<>();

JComboBox trolleybus = new JComboBox();

JComboBox driver = new JComboBox();

try {

ResultSet resultSet = database.statement.executeQuery("SELECT \* FROM drivers");

ResultSetMetaData resultSetMetaData = resultSet.getMetaData();

String[] columnNames = new String[resultSetMetaData.getColumnCount()];

for (int i = 1; i <= resultSetMetaData.getColumnCount(); i++) {

columnNames[i - 1] = resultSetMetaData.getColumnLabel(i);

}

columnNames = new String[]{columnNames[0], columnNames[2]};

for (int i = 0; resultSet.next(); i++) {

drivers.put(resultSet.getString(columnNames[1]), resultSet.getInt(columnNames[0]));

driver.addItem(resultSet.getString(columnNames[1]));

}

resultSet = database.statement.executeQuery("SELECT \* FROM trolleybuses");

resultSetMetaData = resultSet.getMetaData();

columnNames = new String[resultSetMetaData.getColumnCount()];

for (int i = 1; i <= resultSetMetaData.getColumnCount(); i++) {

columnNames[i - 1] = resultSetMetaData.getColumnLabel(i);

}

columnNames = new String[]{columnNames[0], columnNames[1]};

for (int i = 0; resultSet.next(); i++) {

trolleybuses.put(resultSet.getString(columnNames[1]), resultSet.getInt(columnNames[0]));

trolleybus.addItem(resultSet.getString(columnNames[1]));

}

} catch (SQLException e) {

System.out.println("Rotes table, getting initial content error : " + e.getMessage());

}

if (!index.equals(-1)) {

trolleybus.setSelectedItem(tables.get(selectedTab).getValueAt(row, 2).toString());

}

func(p1, new JLabel("Trolleybus"), 0, 1, 1, 1, true, gbc);

func(p1, trolleybus, 1, 1, 4, 1, false, gbc);

if (!index.equals(-1)) {

driver.setSelectedItem(tables.get(selectedTab).getValueAt(row, 3).toString());

}

func(p1, new JLabel("Driver"), 0, 2, 1, 1, true, gbc);

func(p1, driver, 1, 2, 4, 1, false, gbc);

JTextField date = index.equals(-1) ? new JTextField("2021-01-01", 10) : new JTextField(tables.get(selectedTab).getValueAt(row, 3).toString(), 10);

func(p1, new JLabel("Date"), 0, 3, 1, 1, true, gbc);

func(p1, date, 1, 3, 4, 1, false, gbc);

JTextField description = index.equals(-1) ? new JTextField(10) : new JTextField(tables.get(selectedTab).getValueAt(row, 4).toString(), 10);

func(p1, new JLabel("Description"), 0, 0, 1, 1, true, gbc);

func(p1, description, 1, 0, 4, 1, true, gbc);

save.addActionListener(e -> {

try {

if (description.getText().equals("") || date.getText().equals("")) {

throw new Exception();

}

if (index.equals(-1)) {

database.addNote("routes", new Object[]{

description.getText(),

trolleybuses.get(trolleybus.getSelectedItem().toString()),

drivers.get(driver.getSelectedItem()),

date.getText()

});

} else {

database.updateNote("routes", new Object[]{

description.getText(),

trolleybuses.get(trolleybus.getSelectedItem().toString()),

drivers.get(driver.getSelectedItem()),

date.getText()

}, index);

}

frame.dispose();

frame.setVisible(false);

tables.set(2, createRoutesTable());

update();

} catch (NumberFormatException exception) {

JOptionPane.showMessageDialog(this, "Fields must be number", "Error", JOptionPane.OK\_OPTION);

} catch (Exception generalException) {

JOptionPane.showMessageDialog(this, "Empty fields", "Error", JOptionPane.OK\_OPTION);

}

});

cancel.addActionListener(e -> {

frame.dispose();

frame.setVisible(false);

});

break;

}

}

p2.add(save);

p2.add(cancel);

p1.setSize(700, 700);

panel.add(p1, BorderLayout.CENTER);

panel.add(p2, BorderLayout.SOUTH);

frame.add(panel);

frame.setSize(300, 300);

frame.setLocationRelativeTo(null);

frame.pack();

frame.setVisible(true);

}

public static void main(String[] args) {

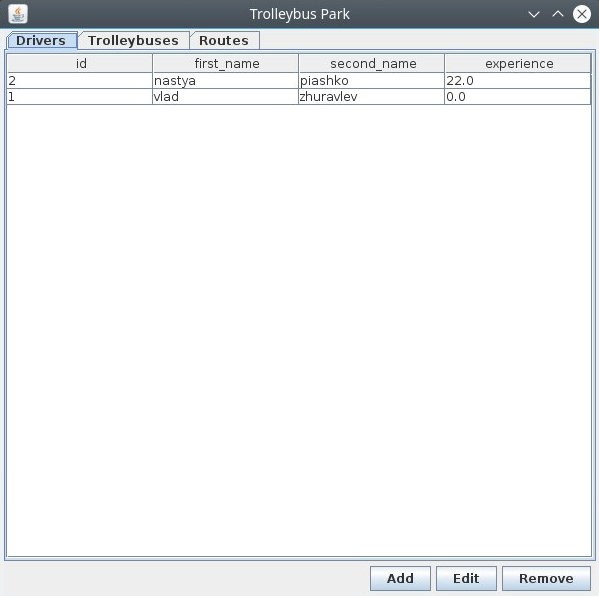
new TrolleybusParkForm();

}

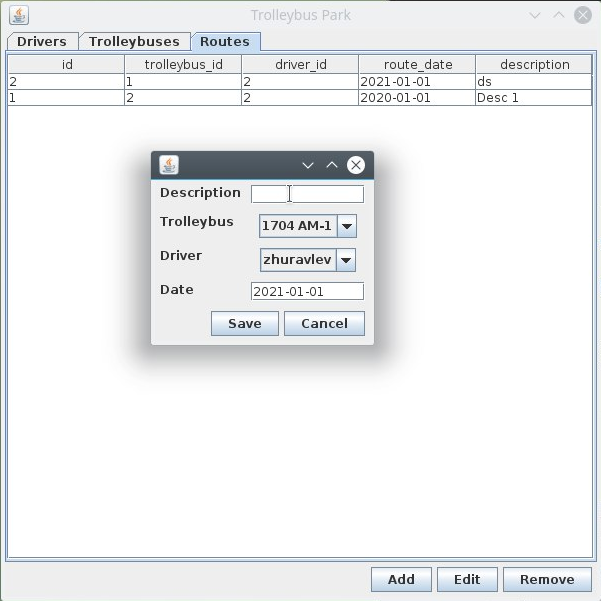
}

**Результат:**

Возможность создания, удаления, редактирования записей:



Пример добавления нового маршрута с выбором водителя и автобуса:



**Вывод:** разработал оконное приложение для работы с БД.