Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №10

По дисциплине «СПП»

Выполнил:
студент 3 курса
группы ПО-5
Брич М.Н.

Проверил:
Крощенко А.А.

Брест, 2022

Цель работы: приобрести практические навыки разработки многооконных приложений на JavaFX для работы с базами данных.

Задание:

На основе БД, разработанной в лабораторной работе №9, реализовать многооконное приложениеклиент, позволяющее выполнять основные операции над таблицей в БД (добавление, удаление, модификацию данных).
Основные требования к приложению:
• Для отображения выбирать таблицу с внешними ключами;
• Осуществлять вывод основных данных в табличном представлении;
• При выводе краткого представления записи в таблице (т.е. если выводятся не все поля), по щелчку мышкой на запись осуществлять вывод всех полей в подготовленные компоненты на форме;
• Для всех полей, представленных внешними ключами, выводить их текстовое представление из связанных таблиц (например, таблица-справочник «Времена года» содержит два поля – идентификатор и название сезона, в связанной таблице «Месяц года» есть внешний ключ на таблицу «Времена года»; в этом случае при выводе таблицы «Месяц года» нужно выводить название сезона, а не его идентификатор);
• При выводе предусмотреть упорядочивание по столбцу;
• Реализовать простейший фильтр данных по одному-двум полям;
• При добавлении новых данных в таблицу использовать дополнительное окно для ввода;
• При модификации данных можно использовать ту же форму, что и для добавления, но с внесенными актуальными значениями полей;
• При добавлении/модификации выводить варианты значений полей с внешним ключом с помощью выпадающего списка;
• При удалении данных осуществлять удаление записи, на которой в данных момент находится фокус.

Код программы:

```
package FacultyCompany.Actions;

import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;

public class InfoDialog {
    public InfoDialog(Stage primaryStage, String groupname, String subjectname,
                      Integer semesterid, Integer weekday, String lessonTime, String
lecturerFullName){
        StackPane secondaryLayout = new StackPane();

        final VBox vbox = new VBox();

        vbox.setSpacing(5);

        Label groupLabel = new Label("Group name: " + groupname);
        Label subjectLabel = new Label("Subject name: " + subjectname);
        Label semestrLabel = new Label("Semester id: " + semesterid);
        Label weekLabel = new Label("Week day: " + weekday);
        Label lessonLabel = new Label("Lesson time: " + lessonTime);
        Label lecturerNameLabel = new Label("Lecturer name: " + lecturerFullName);

        vbox.getChildren().addAll(groupLabel, subjectLabel, semestrLabel, weekLabel,
lessonLabel, lecturerNameLabel);
        secondaryLayout.getChildren().addAll(vbox);

        Scene secondScene = new Scene(secondaryLayout, 250, 150);

        // New window (Stage)
        Stage newWindow = new Stage();
        newWindow.setTitle("Information");
        newWindow.setScene(secondScene);

        // Specifies the modality for new window.
        newWindow.initModality(Modality.WINDOW_MODAL);

        // Specifies the owner Window (parent) for new window
        newWindow.initOwner(primaryStage);

        // Set position of second window, related to primary window.
        newWindow.setX(primaryStage.getX() + 200);
        newWindow.setY(primaryStage.getY() + 100);

        newWindow.setHeight(200);
        newWindow.setWidth(300);

        newWindow.setResizable(false);

        newWindow.show();
    }
}
```

```
package FacultyCompany.Actions;

import FacultyCompany.Core.RepositoryManager;
import FacultyCompany.Entities.*;

import javafx.collections.FXCollections;

import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;

import javafx.stage.Modality;
import javafx.stage.Stage;

import java.sql.SQLException;
import java.util.List;

public class TimeTableAddDialog {
    Label tittleLabel;
    Button addButton;

    Label weekDayLabel;
    TextField weekDayField;

    ComboBox<Subject> subjectComboBox;
    ComboBox<Group> groupComboBox;
    ComboBox<Lecturer> lecturerComboBox;
    ComboBox<Calendar> calendarComboBox;

    public TimeTableAddDialog(Stage primaryStage) {
        initControls();

        List<Subject> subjects;
        List<Group> groups;
        List<Lecturer> lecturers;
        List<Calendar> calendars;

        try {
            RepositoryManager repositoryManager = new RepositoryManager();

            subjects = repositoryManager.subjectRepository.GetAll();
            groups = repositoryManager.groupRepository.GetAll();
            lecturers = repositoryManager.lecturerRepository.GetAll();
            calendars = repositoryManager.calendarRepository.GetAll();

            var subjectObservableList = FXCollections.observableArrayList(subjects);
            var groupObservableList = FXCollections.observableArrayList(groups);
            var lecturerObservableList = FXCollections.observableArrayList(lecturers);
            var calendarObservableList = FXCollections.observableArrayList(calendars);

            subjectComboBox = new ComboBox<>(subjectObservableList);
            groupComboBox = new ComboBox<>(groupObservableList);
            lecturerComboBox = new ComboBox<>(lecturerObservableList);
            calendarComboBox = new ComboBox<>(calendarObservableList);

            subjectComboBox.setValue(subjectObservableList.get(0));
            groupComboBox.setValue(groupObservableList.get(0));
            lecturerComboBox.setValue(lecturerObservableList.get(0));
            calendarComboBox.setValue(calendarObservableList.get(0));
        }
        catch (SQLException throwables) {
            throwables.printStackTrace();
        }

        StackPane secondaryLayout = new StackPane();
        final VBox vbox = new VBox();

        vbox.setSpacing(5);

        vbox.getChildren().addAll(tittleLabel, weekDayLabel, weekDayField,
                new Label("Enter subject"), subjectComboBox,
                new Label("Enter group"), groupComboBox,
                new Label("Enter lecturer"), lecturerComboBox,
                new Label("Enter lesson"), calendarComboBox, addButton);
        secondaryLayout.getChildren().addAll(vbox);

        Scene secondScene = new Scene(secondaryLayout, 300, 240);

        // New window (Stage)
        Stage newWindow = new Stage();
        newWindow.setTitle("Add new information");
        newWindow.setScene(secondScene);

        // Specifies the modality for new window.
        newWindow.initModality(Modality.WINDOW_MODAL);

        // Specifies the owner Window (parent) for new window
        newWindow.initOwner(primaryStage);

        // Set position of second window, related to primary window.
        newWindow.setX(primaryStage.getX() + 200);
        newWindow.setY(primaryStage.getY() + 200);

        newWindow.setHeight(400);
        newWindow.setWidth(250);
        newWindow.setResizable(false);

        newWindow.show();

        addButton.setOnAction(event -> {
            Integer weekDay = Integer.parseInt(weekDayField.getText());

            Integer subjectId = subjectComboBox.getSelectionModel().getSelectedItem().getId();
            Integer groupId = groupComboBox.getSelectionModel().getSelectedItem().getId();
            Integer lecturerId = lecturerComboBox.getSelectionModel().getSelectedItem().getId();
            Integer calendarId = calendarComboBox.getSelectionModel().getSelectedItem().getId();

            if (weekDay > 7 || weekDay <= 0)
                return;

            try {
                RepositoryManager repositoryManager = new RepositoryManager();
                repositoryManager.timeTableRepository.Add(new TimeTable(groupId, subjectId,
lecturerId, weekDay, calendarId));
                newWindow.close();
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });
    }
    public void initControls()
    {
        tittleLabel = new Label("Enter the information:");

        weekDayLabel = new Label("Week day");
        weekDayField = new TextField();

        addButton = new Button("Add the information");

        weekDayField.setMaxSize(50, 50);
    }
}

package FacultyCompany.Actions;

import FacultyCompany.Core.RepositoryManager;
import FacultyCompany.Entities.*;
import javafx.collections.FXCollections;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.StackPane;
import javafx.scene.layout.VBox;
import javafx.stage.Modality;
import javafx.stage.Stage;

import java.sql.SQLException;
import java.util.List;

public class TimeTableUpdateDialog {
    Label tittleLabel;
    Button addButton;

    Label weekDayLabel;
    TextField weekDayField;

    ComboBox<Subject> subjectComboBox;
    ComboBox<Group> groupComboBox;
    ComboBox<Lecturer> lecturerComboBox;
    ComboBox<Calendar> calendarComboBox;

    public TimeTableUpdateDialog(Stage primaryStage, TimeTable table) {
        initControls();

        List<Subject> subjects;
        List<Group> groups;
        List<Lecturer> lecturers;
        List<Calendar> calendars;

        try {
            RepositoryManager repositoryManager = new RepositoryManager();

            subjects = repositoryManager.subjectRepository.GetAll();
            groups = repositoryManager.groupRepository.GetAll();
            lecturers = repositoryManager.lecturerRepository.GetAll();
            calendars = repositoryManager.calendarRepository.GetAll();

            var subjectObservableList = FXCollections.observableArrayList(subjects);
            var groupObservableList = FXCollections.observableArrayList(groups);
            var lecturerObservableList = FXCollections.observableArrayList(lecturers);
            var calendarObservableList = FXCollections.observableArrayList(calendars);

            subjectComboBox = new ComboBox<>(subjectObservableList);
            groupComboBox = new ComboBox<>(groupObservableList);
            lecturerComboBox = new ComboBox<>(lecturerObservableList);
            calendarComboBox = new ComboBox<>(calendarObservableList);

            subjectComboBox.setValue(subjectObservableList.get(table.getSubjectid() - 1));
            groupComboBox.setValue(groupObservableList.get(table.getGroupid() - 1));
            lecturerComboBox.setValue(lecturerObservableList.get(table.getLecturerid()));
            calendarComboBox.setValue(calendarObservableList.get(table.getLessonid() - 1));
            String text = Integer.toString(table.getWeekday()) ;
            weekDayField.setText(text);
        }
        catch (SQLException throwables) {
            throwables.printStackTrace();
        }

        StackPane secondaryLayout = new StackPane();
        final VBox vbox = new VBox();

        vbox.setSpacing(5);

        vbox.getChildren().addAll(tittleLabel, weekDayLabel, weekDayField,
                new Label("Enter subject"), subjectComboBox,
                new Label("Enter group"), groupComboBox,
                new Label("Enter lecturer"), lecturerComboBox,
                new Label("Enter lesson"), calendarComboBox, addButton);
        secondaryLayout.getChildren().addAll(vbox);

        Scene secondScene = new Scene(secondaryLayout, 300, 240);

        // New window (Stage)
        Stage newWindow = new Stage();
        newWindow.setTitle("Update cell");
        newWindow.setScene(secondScene);

        // Specifies the modality for new window.
        newWindow.initModality(Modality.WINDOW_MODAL);

        // Specifies the owner Window (parent) for new window
        newWindow.initOwner(primaryStage);

        // Set position of second window, related to primary window.
        newWindow.setX(primaryStage.getX() + 200);
        newWindow.setY(primaryStage.getY() + 200);

        newWindow.setHeight(400);
        newWindow.setWidth(250);
        newWindow.setResizable(false);

        newWindow.show();

        addButton.setOnAction(event -> {
            Integer id = table.getId();

            Integer weekDay = Integer.parseInt(weekDayField.getText());
            Integer subjectId = subjectComboBox.getSelectionModel().getSelectedItem().getId();
            Integer groupId = groupComboBox.getSelectionModel().getSelectedItem().getId();
            Integer lecturerId = lecturerComboBox.getSelectionModel().getSelectedItem().getId();
            Integer calendarId = calendarComboBox.getSelectionModel().getSelectedItem().getId();

            if (weekDay > 7 || weekDay < 0)
                return;

            try {
                RepositoryManager  repositoryManager = new RepositoryManager();
                repositoryManager.timeTableRepository.Update(new TimeTable(id, groupId,
                        subjectId, lecturerId, weekDay, calendarId));
                newWindow.close();
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });
    }

    public void initControls()
    {
        tittleLabel = new Label("Enter the information:");

        weekDayLabel = new Label("Week day");
        weekDayField = new TextField();

        addButton = new Button("Update cell");
        weekDayField.setMaxSize(50, 50);
    }
}

package FacultyCompany.Constants;

public class ConfigurationConstants
{
    public static final String URL = "jdbc:postgresql://localhost:5433/NordSPP";
    public static final String USER = "postgres";
    public static final String PASSWORD = "nird12";
}

package FacultyCompany.Core;

import FacultyCompany.Constants.ConfigurationConstants;

import java.sql.DriverManager;
import java.sql.SQLException;

public final class Connection {

    public java.sql.Connection GetConnection() throws SQLException {
        var connectionURL = ConfigurationConstants.URL;
        var connectionUser = ConfigurationConstants.USER;
        var connectionPassword = ConfigurationConstants.PASSWORD;

        var connection = DriverManager.getConnection(connectionURL, connectionUser,
connectionPassword);
        return connection;
    }
}

package FacultyCompany.Core;

import FacultyCompany.Entities.*;
import FacultyCompany.Persistence.Interfaces.IBaseRepository;
import FacultyCompany.Persistence.Repositories.*;

import java.sql.SQLException;

public class RepositoryManager {
    public IBaseRepository<Subject> subjectRepository;
    public IBaseRepository<Group> groupRepository;
    public IBaseRepository<Lecturer> lecturerRepository;
    public IBaseRepository<Calendar> calendarRepository;
    public IBaseRepository<TimeTable> timeTableRepository;

    private static Connection connection = new Connection();

    public RepositoryManager() throws SQLException {
        this.subjectRepository = new SubjectRepository(connection.GetConnection());
        this.groupRepository = new GroupRepository(connection.GetConnection());
        this.lecturerRepository = new LecturerRepository(connection.GetConnection());
        this.calendarRepository = new CalendarRepository(connection.GetConnection());
        this.timeTableRepository = new TimeTableRepository(connection.GetConnection());
    }
}

package FacultyCompany.Entities;

public class Calendar {
    private int id;
    private int semesterid;
    private int weekday;
    private int lessonid;
    private String lessontime;

    public Calendar() {}

    public Calendar(int semesterId, int weekDay, int lessonId, String lessonTime) {
        this.semesterid = semesterId;
        this.weekday = weekDay;
        this.lessonid = lessonId;
        this.lessontime = lessonTime;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getSemesterid() {
        return semesterid;
    }

    public void setSemesterid(int semesterId) {
        this.semesterid = semesterId;
    }

    public int getWeekday() {
        return weekday;
    }

    public void setWeekday(int weekDay) {
        this.weekday = weekDay;
    }

    public int getLessonid() {
        return lessonid;
    }

    public void setLessonid(int lessonId) {
        this.lessonid = lessonId;
    }

    public String getLessontime() {
        return lessontime;
    }

    public void setLessontime(String lessonTime) {
        this.lessontime = lessonTime;
    }

    @Override
    public String toString()
    {
        return getLessontime();
    }
}
```

```
package FacultyCompany.Entities;

public class Group {
    private int id;
    private String groupname;

    public Group() {}

    public Group(String groupName) {
        this.groupname = groupName;
    }

    public String getGroupname() {
        return groupname;
    }

    public void setGroupname(String groupName) {
        this.groupname = groupName;
    }

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }

    @Override
    public String toString()
    {
        return getGroupname();
    }
}
```

```
package FacultyCompany.Entities;

public class Lecturer {
    private int id;
    private String firstname;
    private String lastname;
    private String patronymic;

    public Lecturer() {}

    public Lecturer(String firstName, String lastName, String patronymic) {
        this.firstname = firstName;
        this.lastname = lastName;
        this.patronymic = patronymic;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstName) {
        this.firstname = firstName;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastName) {
        this.lastname = lastName;
    }

    public String getPatronymic() {
        return patronymic;
    }

    public void setPatronymic(String patronymic) {
        this.patronymic = patronymic;
    }

    @Override
    public String toString()
    {
        return getFirstname() + " " + getLastname();
    }
}
```

```
package FacultyCompany.Entities;

public class Subject {
    private int id;
    private String subjectName;

    public Subject() {}
    public Subject(String subjectName) {
        this.subjectName = subjectName;
    }

    public String getSubjectName() {
        return subjectName;
    }

    public void setSubjectName(String subjectName) {
        this.subjectName = subjectName;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    @Override
    public String toString()
    {
        return getSubjectName();
    }
}
```

```
package FacultyCompany.Entities;

public class TimeTable {
    private int id;

    private int groupid;
    private Group group;

    private int subjectid;
    private Subject subject;

    private int lecturerid;
    private Lecturer lecturer;

    private int weekday;

    private int lessonid;
    private Calendar calendar;

    public TimeTable() {}

    public TimeTable(int groupId, int subjectId, int lecturerId, int weekDay, int lessonId) {
        this.groupid = groupId;
        this.subjectid = subjectId;
        this.lecturerid = lecturerId;
        this.weekday = weekDay;
        this.lessonid = lessonId;
    }

    public TimeTable(int id) {
        this.id = id;
    }

    public TimeTable(int id, int groupId, int subjectId, int lecturerId, int weekDay, int
lessonId) {
        this.id = id;
        this.groupid = groupId;
        this.subjectid = subjectId;
        this.lecturerid = lecturerId;
        this.weekday = weekDay;
        this.lessonid = lessonId;
    }

    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }

    public int getGroupid() {
        return groupid;
    }

    public void setGroupid(int groupId) {
        this.groupid = groupId;
    }

    public Group getGroup() {
        return group;
    }

    public void setGroup(Group group) {
        this.group = group;
    }

    public int getSubjectid() {
        return subjectid;
    }

    public void setSubjectid(int subjectId) {
        this.subjectid = subjectId;
    }

    public Subject getSubject() {
        return subject;
    }

    public void setSubject(Subject subject) {
        this.subject = subject;
    }

    public int getLecturerid() {
        return lecturerid;
    }

    public void setLecturerid(int lecturerId) {
        this.lecturerid = lecturerId;
    }

    public String getLecturerName() {
        return lecturer.getFirstname() + " " + lecturer.getLastname();
    }

    public Lecturer getLecturer() {
        return lecturer;
    }

    public void setLecturer(Lecturer lecturer) {
        this.lecturer = lecturer;
    }

    public int getWeekday() {
        return weekday;
    }

    public void setWeekday(int weekDay) {
        this.weekday = weekDay;
    }

    public int getLessonid() {
        return lessonid;
    }

    public void setLessonid(int lessonId) {
        this.lessonid = lessonId;
    }

    public Calendar getCalendar() {
        return calendar;
    }

    public void setCalendar(Calendar calendar) {
        this.calendar = calendar;
    }
}
```

```
package FacultyCompany.Persistence.Interfaces;

import java.sql.SQLException;
import java.util.ArrayList;

public interface IBaseRepository<T> {
    T Add(T entity) throws SQLException;
    void Update(T entity) throws SQLException;
    void Delete(T entity) throws SQLException;
    T GetByIdOrNull(int id) throws SQLException;
    ArrayList<T> GetAll() throws SQLException;
}
```

```
package FacultyCompany.Persistence.Repositories;

import FacultyCompany.Entities.Calendar;
import FacultyCompany.Persistence.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class CalendarRepository implements IBaseRepository<Calendar> {

    private final Connection connection;

    public CalendarRepository(Connection connection) {
        this.connection = connection;
    }

    @Override
    public Calendar Add(Calendar entity) throws SQLException {
        var query =
                "INSERT INTO public.calendar( " +
                        " semesterid, weekday, lessonid, lessontime) " +
                        " VALUES (?, ?, ?, ?)";

        var statement = connection.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
        statement.setInt(1, entity.getSemesterid());
        statement.setInt(2, entity.getWeekday());
        statement.setInt(3, entity.getLessonid());
        statement.setString(4, entity.getLessontime());

        statement.execute();

        var generatedKeys = statement.getGeneratedKeys();
        generatedKeys.next();
        entity.setId(generatedKeys.getInt(1));

        return entity;
    }

    @Override
    public void Update(Calendar entity) throws SQLException {
        var query ="UPDATE public.calendar " +
                " SET semesterid=?, weekday=?, lessonid=?, lessontime=?" +
                " WHERE id=?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getSemesterid());
        statement.setInt(2, entity.getWeekday());
        statement.setInt(3, entity.getLessonid());
        statement.setString(4, entity.getLessontime());
        statement.setInt(5, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public void Delete(Calendar entity) throws SQLException {
        var query = "DELETE FROM public.calendar" +
                " WHERE id=?";
        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public Calendar GetByIdOrNull(int id) throws SQLException {
        var query =
                "SELECT * FROM public.calendar" +
                        " WHERE Id = ?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        var reader = statement.executeQuery();
        if(reader.next())
        {
            var result = new Calendar();
            result.setId(reader.getInt("id"));
            result.setLessonid(reader.getInt("lessonid"));
            result.setSemesterid(reader.getInt("semesterid"));
            result.setWeekday(reader.getInt("weekday"));
            result.setLessontime(reader.getString("lessontime"));
            return result;
        }

        return null;
    }

    @Override
    public ArrayList<Calendar> GetAll() throws SQLException {
        var query =
                "SELECT * FROM public.calendar Order by id";

        var statement = connection.prepareStatement(query);

        var reader = statement.executeQuery();
        var result = new ArrayList<Calendar>();
        while (reader.next())
        {
            var calendar = new Calendar();
            calendar.setId(reader.getInt("id"));
            calendar.setLessonid(reader.getInt("lessonid"));
            calendar.setSemesterid(reader.getInt("semesterid"));
            calendar.setWeekday(reader.getInt("weekday"));
            calendar.setLessontime(reader.getString("lessontime"));

            result.add(calendar);
        }

        return result;
    }
}

package FacultyCompany.Persistence.Repositories;

import FacultyCompany.Entities.Group;
import FacultyCompany.Persistence.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class GroupRepository implements IBaseRepository<Group> {
    private final Connection connection;
    public GroupRepository(Connection connection) {
        this.connection = connection;
    }

    @Override
    public Group Add(Group entity) throws SQLException {
        var query =
                "INSERT INTO public.groups(" +
                        " groupname)" +
                        " VALUES (?)";

        var statement = connection.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, entity.getGroupname());

        statement.execute();

        var generatedKeys = statement.getGeneratedKeys();
        generatedKeys.next();
        entity.setId(generatedKeys.getInt(1));

        return entity;
    }

    @Override
    public void Update(Group entity) throws SQLException {
        var query =
                "UPDATE public.groups" +
                        " SET groupname = ?" +
                        " WHERE id = ?";

        var statement = connection.prepareStatement(query);
        statement.setString(1, entity.getGroupname());
        statement.setInt(2, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public void Delete(Group entity) throws SQLException {
        var query = "DELETE FROM public.groups" +
                " WHERE id=?";
        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public Group GetByIdOrNull(int id) throws SQLException {
        var query =
                "SELECT * FROM public.groups" +
                        " WHERE Id = ?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        var reader = statement.executeQuery();
        if(reader.next())
        {
            var result = new Group();
            result.setId(reader.getInt("id"));
            result.setGroupname(reader.getString("groupname"));
            return result;
        }

        return null;
    }

    @Override
    public ArrayList<Group> GetAll() throws SQLException {
        var query =
                "SELECT * FROM public.groups Order by id";

        var statement = connection.prepareStatement(query);
        var reader = statement.executeQuery();
        var result = new ArrayList<Group>();
        while (reader.next())
        {
            var group = new Group();
            group.setId(reader.getInt("id"));
            group.setGroupname(reader.getString("groupname"));

            result.add(group);
        }

        return result;
    }
}
```

```
package FacultyCompany.Persistence.Repositories;

import FacultyCompany.Entities.Lecturer;
import FacultyCompany.Persistence.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class LecturerRepository implements IBaseRepository<Lecturer> {

    private final Connection connection;

    public LecturerRepository(Connection connection) {
        this.connection = connection;
    }

    @Override
    public Lecturer Add(Lecturer entity) throws SQLException {
        var query =
                "INSERT INTO public.lecturers(" +
                        "firstname, lastname, patronymic)" +
                        " VALUES (?, ?, ?)";

        var statement = connection.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
        statement.setString(1, entity.getFirstname());
        statement.setString(2, entity.getLastname());
        statement.setString(3, entity.getPatronymic());

        statement.execute();

        var generatedKeys = statement.getGeneratedKeys();
        generatedKeys.next();
        entity.setId(generatedKeys.getInt(1));

        return entity;
    }

    @Override
    public void Update(Lecturer entity) throws SQLException {
        var query =
                "UPDATE public.lecturers" +
                        " SET firstname=?, lastname=?, patronymic=?" +
                        " WHERE id=?";


        var statement = connection.prepareStatement(query);
        statement.setString(1, entity.getFirstname());
        statement.setString(2, entity.getLastname());
        statement.setString(3, entity.getPatronymic());
        statement.setInt(4, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public void Delete(Lecturer entity) throws SQLException {
```

```java
        var query = "DELETE FROM public.lecturers" +
                " WHERE id=?";
        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public Lecturer GetByIdOrNull(int id) throws SQLException {
        var query =
                "SELECT * FROM public.lecturers" +
                        " WHERE Id = ?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        var reader = statement.executeQuery();
        if(reader.next())
        {
            var result = new Lecturer();
            result.setId(reader.getInt("id"));
            result.setFirstname(reader.getString("firstname"));
            result.setLastname(reader.getString("lastname"));
            result.setPatronymic(reader.getString("patronymic"));
            return result;
        }

        return null;
    }

    @Override
    public ArrayList<Lecturer> GetAll() throws SQLException {
        var query =
                "SELECT * FROM public.lecturers Order by id";

        var statement = connection.prepareStatement(query);

        var reader = statement.executeQuery();
        var result = new ArrayList<Lecturer>();
        while (reader.next())
        {
            var lecturer = new Lecturer();
            lecturer.setId(reader.getInt("id"));
            lecturer.setFirstname(reader.getString("firstname"));
            lecturer.setLastname(reader.getString("lastname"));
            lecturer.setPatronymic(reader.getString("patronymic"));

            result.add(lecturer);
        }

        return result;
    }
}

package FacultyCompany.Persistence.Repositories;

import FacultyCompany.Entities.Subject;
import FacultyCompany.Persistence.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class SubjectRepository implements IBaseRepository<Subject> {

    private final Connection connection;

    public SubjectRepository(Connection connection) {
        this.connection = connection;
    }

    @Override
```

```
public Subject Add(Subject entity) throws SQLException {
    var query =
            "INSERT INTO public.subjects(" +
                    " subjectname)" +
                    " VALUES (?)";

    var statement = connection.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
    statement.setString(1, entity.getSubjectName());


    statement.execute();

    var generatedKeys = statement.getGeneratedKeys();
    generatedKeys.next();
    entity.setId(generatedKeys.getInt(1));

    return entity;
}

@Override
public void Update(Subject entity) throws SQLException {
    var query =
            "UPDATE public.subjects" +
                    " SET subjectname = ?" +
                    " WHERE id = ?";

    var statement = connection.prepareStatement(query);
    statement.setString(1, entity.getSubjectName());
    statement.setInt(2, entity.getId());

    statement.executeUpdate();
}

@Override
public void Delete(Subject entity) throws SQLException {
    var query = "DELETE FROM public.subjects" +
            " WHERE id=?";
    var statement = connection.prepareStatement(query);
    statement.setInt(1, entity.getId());

    statement.executeUpdate();
}

@Override
public Subject GetByIdOrNull(int id) throws SQLException {
    var query =
            "SELECT * FROM public.subjects" +
                    " WHERE Id = ? " +
                    "Order by id";

    var statement = connection.prepareStatement(query);
    statement.setInt(1, id);

    var reader = statement.executeQuery();
    if(reader.next())
    {
        var result = new Subject();
        result.setId(reader.getInt("id"));
        result.setSubjectName(reader.getString("subjectname"));
        return result;
    }

    return null;
}

@Override
public ArrayList<Subject> GetAll() throws SQLException {
    var query =
            "SELECT * FROM public.subjects Order by id";

    var statement = connection.prepareStatement(query);

    var reader = statement.executeQuery();
    var result = new ArrayList<Subject>();
```

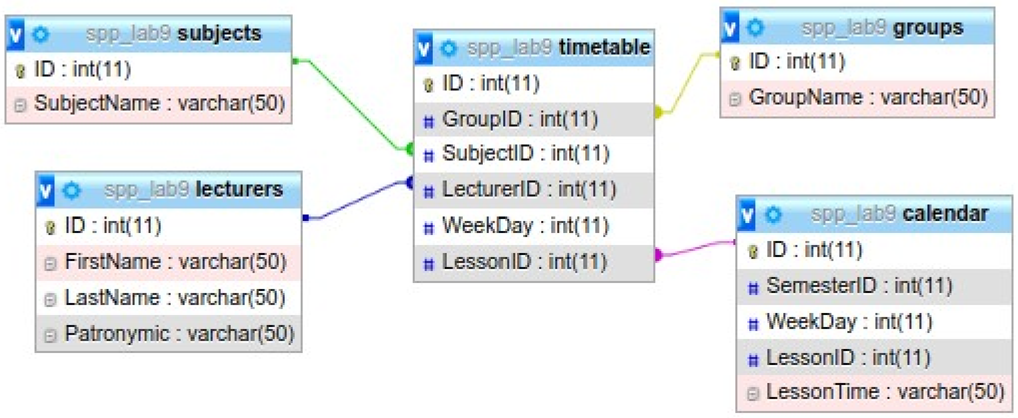
```
        while (reader.next())
        {
            var subject = new Subject();
            subject.setId(reader.getInt("id"));
            subject.setSubjectName(reader.getString("subjectname"));

            result.add(subject);
        }

        return result;
    }
}

package FacultyCompany.Persistence.Repositories;

import FacultyCompany.Entities.TimeTable;
import FacultyCompany.Persistence.Interfaces.IBaseRepository;

import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

public class TimeTableRepository implements IBaseRepository<TimeTable> {

    private final Connection connection;

    public TimeTableRepository(Connection connection) {
        this.connection = connection;
    }

    @Override
    public TimeTable Add(TimeTable entity) throws SQLException {
        var query =
                "INSERT INTO public.timetable(" +
                        "groupid, subjectid, lecturerid, weekday, lessonid)" +
                        " VALUES (?, ?, ?, ?, ?);";

        var statement = connection.prepareStatement(query, Statement.RETURN_GENERATED_KEYS);
        statement.setInt(1, entity.getGroupid());
        statement.setInt(2, entity.getSubjectid());
        statement.setInt(3, entity.getLecturerid());
        statement.setInt(4, entity.getWeekday());
        statement.setInt(5, entity.getLessonid());

        statement.execute();

        var generatedKeys = statement.getGeneratedKeys();
        generatedKeys.next();
        entity.setId(generatedKeys.getInt(1));

        return entity;
    }

    @Override
    public void Update(TimeTable entity) throws SQLException {

        var query = "UPDATE public.timetable " +
                " SET groupid=?, subjectid=?, lecturerid=?, weekday=?, lessonid=? " +
                " WHERE id=?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getGroupid());
        statement.setInt(2, entity.getSubjectid());
        statement.setInt(3, entity.getLecturerid());
        statement.setInt(4, entity.getWeekday());
        statement.setInt(5, entity.getLessonid());
        statement.setInt(6, entity.getId());

        statement.executeUpdate();
        return;
    }

    @Override
    public void Delete(TimeTable entity) throws SQLException {
        var query = "DELETE FROM public.timetable" +
                " WHERE id=?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, entity.getId());

        statement.executeUpdate();
    }

    @Override
    public TimeTable GetByIdOrNull(int id) throws SQLException {
        var query =
                "SELECT * FROM public.timetable" +
                        " WHERE Id = ?";

        var statement = connection.prepareStatement(query);
        statement.setInt(1, id);

        var reader = statement.executeQuery();
        if(reader.next())
        {
            var result = new TimeTable();
            result.setId(reader.getInt("id"));
            result.setLessonid(reader.getInt("lessonid"));
            result.setWeekday(reader.getInt("weekday"));
            result.setGroupid(reader.getInt("groupid"));
            result.setSubjectid(reader.getInt("subjectid"));
            return result;
        }

        return null;
    }

    @Override
    public ArrayList<TimeTable> GetAll() throws SQLException {
        var query =
                "SELECT * FROM public.timetable Order by id";

        var statement = connection.prepareStatement(query);

        var reader = statement.executeQuery();
        var result = new ArrayList<TimeTable>();
        while (reader.next())
        {
            var timeTable = new TimeTable();
            timeTable.setId(reader.getInt("id"));
            timeTable.setGroupid(reader.getInt("groupid"));
            timeTable.setSubjectid(reader.getInt("subjectid"));
            timeTable.setLecturerid(reader.getInt("lecturerid"));
            timeTable.setWeekday(reader.getInt("weekday"));
            timeTable.setLessonid(reader.getInt("lessonid"));

            result.add(timeTable);
        }

        return result;
    }
}
```

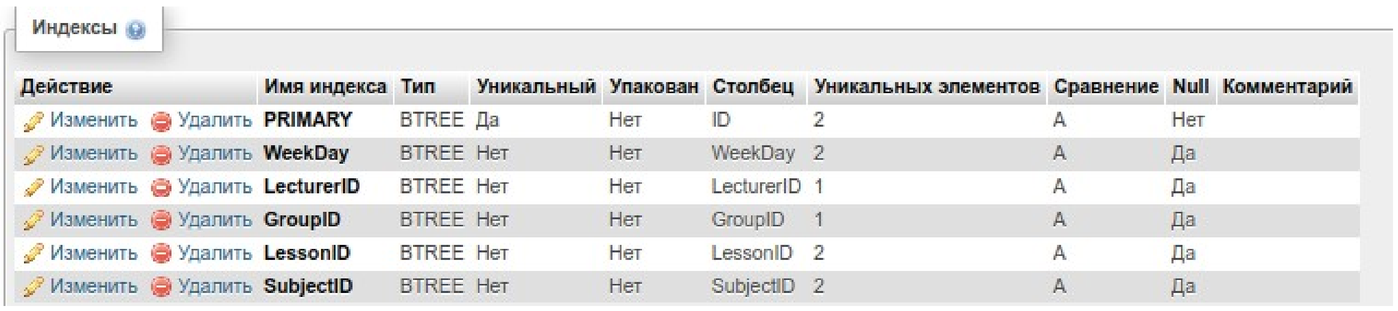
```
package FacultyCompany.ViewModels;

import FacultyCompany.Entities.*;

public class TimeTableViewModel {
    private int id;

    private String groupname;

    private String subjectName;

    private String lecturerFullName;

    private int semesterid;
    private int weekday;
    private String lessontime;

    private int groupid;
    private int subjectid;
    private int lecturerid;
    private int lessonid;

    public TimeTableViewModel(TimeTable table) {
        this.id = table.getId();
        this.groupname = table.getGroup().getGroupname();
        this.subjectName = table.getSubject().getSubjectName();
        this.semesterid = table.getCalendar().getSemesterid();
        this.weekday = table.getWeekday();
        this.lessontime = table.getCalendar().getLessontime();
        this.groupid = table.getGroupid();
        this.subjectid = table.getSubjectid();
        this.lessonid = table.getLessonid();
        this.lecturerFullName = table.getLecturerName();
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getGroupname() {
        return groupname;
    }

    public void setGroupname(String groupname) {
        this.groupname = groupname;
    }

    public String getSubjectName() {
        return subjectName;
    }

    public void setSubjectName(String subjectName) {
        this.subjectName = subjectName;
    }

    public int getSemesterid() {
        return semesterid;
    }

    public void setSemesterid(int semesterid) {
        this.semesterid = semesterid;
    }

    public int getWeekday() {
        return weekday;
    }

    public void setWeekday(int weekday) {
        this.weekday = weekday;
    }

    public int getLessonid() {
        return lessonid;
    }

    public void setLessonid(int lessonid) {
        this.lessonid = lessonid;
    }

    public String getLessontime() {
        return lessontime;
    }

    public void setLessontime(String lessontime) {
        this.lessontime = lessontime;
    }

    public String getLecturerFullName() {
        return lecturerFullName;
    }
}
```

App

```
package FacultyCompany;

import FacultyCompany.Actions.InfoDialog;
import FacultyCompany.Actions.TimeTableAddDialog;
import FacultyCompany.Actions.TimeTableUpdateDialog;
import FacultyCompany.Core.RepositoryManager;
import FacultyCompany.Entities.*;
import FacultyCompany.ViewModels.TimeTableViewModel;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.fxml.FXMLLoader;
import javafx.geometry.Insets;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.cell.PropertyValueFactory;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

import java.sql.SQLException;

import java.util.ArrayList;

public class App extends Application {
    private static RepositoryManager repositoryManager;

    public App() throws SQLException {
        repositoryManager = new RepositoryManager();
    }

    TableView<TimeTableViewModel> tableTimeTables;

    @Override
    public void start(Stage primaryStage) throws Exception {
        var timeTables= repositoryManager.timeTableRepository.GetAll();

        var timeTablesWithData = convertWithData(timeTables);

        ArrayList<TimeTableViewModel> VmTimeTable = new ArrayList<>();

        timeTablesWithData.forEach(t -> {
            VmTimeTable.add(new TimeTableViewModel(t));
        });

        var observableTimeTables = FXCollections.observableArrayList(VmTimeTable);

        tableTimeTables = new TableView<>(observableTimeTables);

        tableTimeTables.setColumnResizePolicy(TableView.CONSTRAINED_RESIZE_POLICY);
        tableTimeTables.setPrefSize(720,200);

        TableColumn<TimeTableViewModel, Integer> idColumn = new TableColumn<>("Id");
        idColumn.setCellValueFactory(new PropertyValueFactory<>("id"));
        tableTimeTables.getColumns().add(idColumn);

        TableColumn<TimeTableViewModel, String> groupnameColumn = new TableColumn<>("Group 
Name");
        groupnameColumn.setCellValueFactory(new PropertyValueFactory<>("groupname"));
        tableTimeTables.getColumns().add(groupnameColumn);
        TableColumn<TimeTableViewModel, String> subjectNameColumn = new TableColumn<>("Subject
Name");
        subjectNameColumn.setCellValueFactory(new PropertyValueFactory<>("subjectName"));
        tableTimeTables.getColumns().add(subjectNameColumn);

        TableColumn<TimeTableViewModel, Integer> semesteridColumn = new TableColumn<>("Semestre
Id");
        semesteridColumn.setCellValueFactory(new PropertyValueFactory<>("semesterid"));
        tableTimeTables.getColumns().add(semesteridColumn);

        TableColumn<TimeTableViewModel, Integer> weekdayColumn = new TableColumn<>("Week Day");
        weekdayColumn.setCellValueFactory(new PropertyValueFactory<>("weekday"));
        tableTimeTables.getColumns().add(weekdayColumn);

        TableColumn<TimeTableViewModel, String> lessontimeColumn = new TableColumn<>("Lesson
Time");
        lessontimeColumn.setCellValueFactory(new PropertyValueFactory<>("lessontime"));
        tableTimeTables.getColumns().add(lessontimeColumn);

        executeRefresh();

        final VBox vbox = new VBox();
        vbox.setSpacing(5);
        vbox.setPadding(new Insets(10, 10, 10, 10));
        vbox.getChildren().addAll(new Label("TimeTable"), tableTimeTables);

        Button addCellButton = new Button();
        addCellButton.setText("Add a new cell");

        addCellButton.setOnAction(event -> {
            new TimeTableAddDialog(primaryStage);
        });

        vbox.getChildren().addAll(addCellButton);

        Button infoButton = new Button();
        infoButton.setText("Cell's details");

        infoButton.setOnAction(event -> {
            TimeTableViewModel infoTimeTable =
tableTimeTables.getSelectionModel().getSelectedItems().get(0);
            new InfoDialog(primaryStage, infoTimeTable.getGroupname(),
infoTimeTable.getSubjectName(), infoTimeTable.getSemesterid(),
                    infoTimeTable.getWeekday(), infoTimeTable.getLessontime(),
infoTimeTable.getLecturerFullName());
        });

        vbox.getChildren().addAll(infoButton);

        Button updateButton = new Button();
        updateButton.setText("Update cell");

        updateButton.setOnAction(event -> {
            TimeTableViewModel updateTimeTable =
tableTimeTables.getSelectionModel().getSelectedItems().get(0);
            TimeTable vmtable = new TimeTable();

            try {
                vmtable =
repositoryManager.timeTableRepository.GetByIdOrNull(updateTimeTable.getId());
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }

            new TimeTableUpdateDialog(primaryStage, vmtable);
        });

        vbox.getChildren().addAll(updateButton);

        Button deleteCellButton = new Button();
        deleteCellButton.setText("Delete cell");

        deleteCellButton.setOnAction(event -> {
            try {
                executeDelete();
                executeRefresh();
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });

        vbox.getChildren().addAll(deleteCellButton);

        Button refreshCellButton = new Button();
        refreshCellButton.setText("Refresh table");

        refreshCellButton.setOnAction(event -> {
            try {
                executeRefresh();
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });
        vbox.getChildren().addAll(refreshCellButton);
        Parent root = FXMLLoader.load(getClass().getResource("/sample.fxml"));
        primaryStage.setTitle("TimeTable");

        Scene scene = new Scene(root, 500, 500);
        ((GridPane) scene.getRoot()).getChildren().addAll(vbox);
        primaryStage.setScene(scene);

        primaryStage.setHeight(425);
        primaryStage.setWidth(550);

        primaryStage.setResizable(false);

        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }

    private void executeRefresh() throws SQLException {
        tableTimeTables.getItems().clear();

        var timeTables= repositoryManager.timeTableRepository.GetAll();

        var timeTablesWithData = convertWithData(timeTables);
        ArrayList<TimeTableViewModel> VmTimeTable = new ArrayList<>();

        timeTablesWithData.forEach(t -> {
            VmTimeTable.add(new TimeTableViewModel(t));
        });

        var observableTimeTables = FXCollections.observableArrayList(VmTimeTable);

        tableTimeTables.getItems().addAll(observableTimeTables);
    }

    private void executeDelete() throws SQLException {
        TimeTableViewModel deletedTimeTable =
tableTimeTables.getSelectionModel().getSelectedItems().get(0);
        repositoryManager.timeTableRepository.Delete(new TimeTable(deletedTimeTable.getId()));
    }

    public static ArrayList<TimeTable> convertWithData(ArrayList<TimeTable> table) {
        table.forEach(t -> {
            try {
                t.setGroup(repositoryManager.groupRepository.GetByIdOrNull(t.getGroupid()));

t.setSubject(repositoryManager.subjectRepository.GetByIdOrNull(t.getSubjectid()));

t.setLecturer(repositoryManager.lecturerRepository.GetByIdOrNull(t.getLecturerid()));
t.setCalendar(repositoryManager.calendarRepository.GetByIdOrNull(t.getLessonid()));
            }
            catch (SQLException throwables) {
                throwables.printStackTrace();
            }
        });

        return table;
    }
}
```

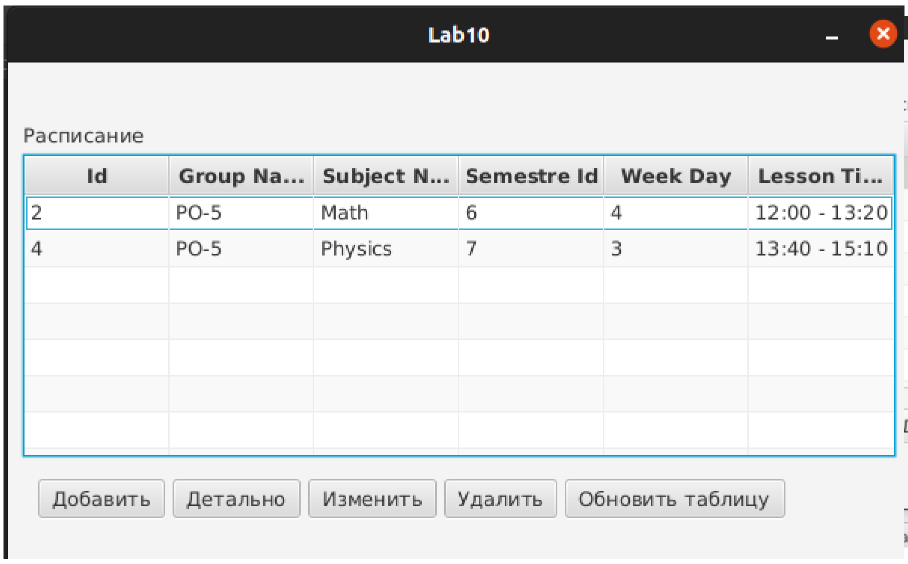
Тестирование:





Индексы

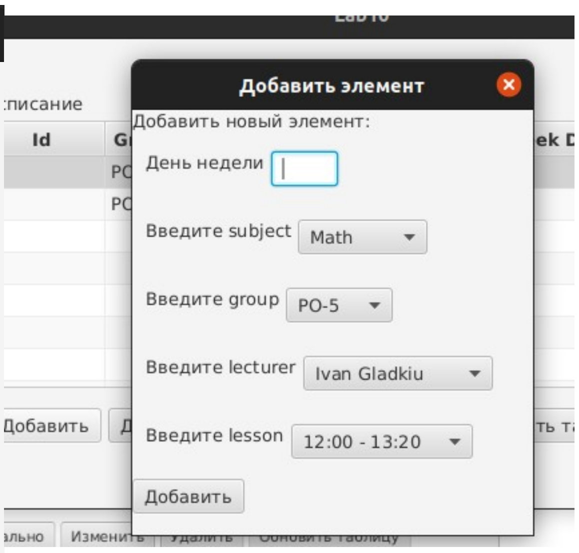
Действие	Имя индекса	Тип	Уникальный	Упакован	Столбец	Уникальных элементов	Сравнение	Null	Комментарий
Изменить Удалить	PRIMARY	BTREE	Да	Нет	ID	2	A	Нет	
Изменить Удалить	WeekDay	BTREE	Нет	Нет	WeekDay	2	A	Да	
Изменить Удалить	LecturerID	BTREE	Нет	Нет	LecturerID	1	A	Да	
Изменить Удалить	GroupID	BTREE	Нет	Нет	GroupID	1	A	Да	
Изменить Удалить	LessonID	BTREE	Нет	Нет	LessonID	2	A	Да	
Изменить Удалить	SubjectID	BTREE	Нет	Нет	SubjectID	2	A	Да	



Lab10

Расписание

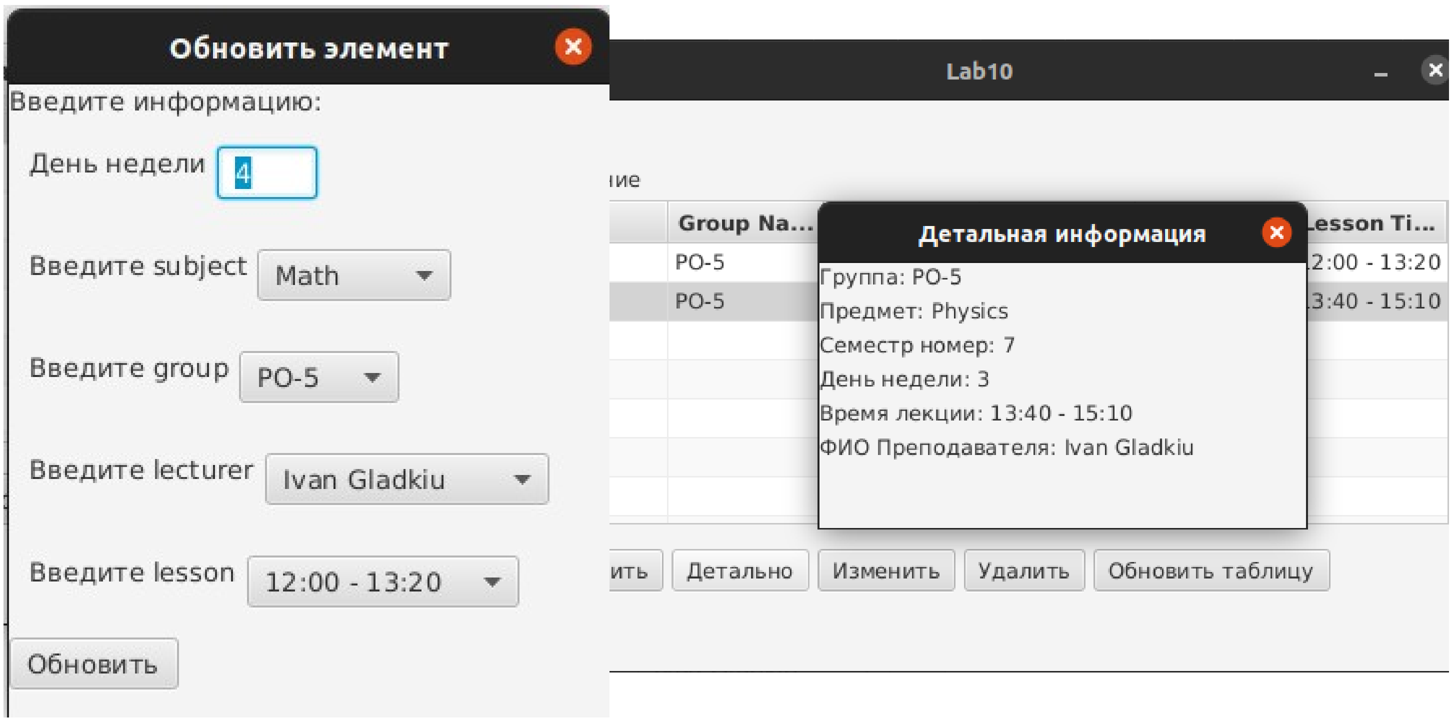
Id	Group Na...	Subject N...	Semestre Id	Week Day	Lesson Ti...
2	PO-5	Math	6	4	12:00 - 13:20
4	PO-5	Physics	7	3	13:40 - 15:10



Добавить элемент

Добавить новый элемент:

День недели	
Введите subject	Math
Введите group	PO-5
Введите lecturer	Ivan Gladkiu
Введите lesson	12:00 - 13:20



Введите информацию:

День недели 4

Введите subject Math

Введите group PO-5

Введите lecturer Ivan Gladkiu

Введите lesson 12:00 - 13:20

Группа: PO-5
Предмет: Physics
Семестр номер: 7
День недели: 3
Время лекции: 13:40 - 15:10
ФИО Преподавателя: Ivan Gladkiu

Выводы: в ходе выполнения лабораторной работы были приобретены практические навыки разработки многооконных приложений на JavaFX для работы с базами данных.