Задача: Разработка JSP веб-приложения для работы с базой данных через JDBC

Описание проекта:

Разработайте веб-приложение на JSP, которое взаимодействует с несколькими таблицами базы данных через JDBC. Программа должна выполнять следующие функции:

– Стартовая страница приложения:
Создайте главную страницу с возможностью переходов на другие разделы, включая страницы навигации, помощи и редактирования объектов.
– Навигация по иерархии объектов:
Реализуйте отображение данных из двух таблиц с необходимыми параметрами объектов.
– Убедитесь в удобстве навигации между различными уровнями иерархии.
– Формы для редактирования и добавления объектов:
Создайте интерфейсы для создания новых записей и редактирования существующих.
– Реализуйте функцию выбора ссылок на объекты из других таблиц.
– Модальное окно для выбора объекта:
При нажатии на ссылку открывается отдельное окно для выбора связанного объекта.
– Поиск объектов:
Реализуйте функцию поиска по одной из таблиц с заданным критерием.
– Отображение объектов по критериям:
Создайте возможность отображения объектов на странице по заданному критерию для каждой из колонок таблицы.

Требования к реализации:

– Используйте JSP для создания веб-страниц и JDBC для работы с базой данных.
– Примените принципы объектно-ориентированного программирования.
– Убедитесь в правильной обработке ошибок и безопасности при работе с данными.
– Реализуйте удобный пользовательский интерфейс для всех функций приложения.

Выполнение:

Для работы Java EE использовались следующие программы и компоненты:

GlassFish, JDK, NetBeans.

Создана база данных со связанными внешним ключом таблицами студентов и групп. Таблицы заполнены данными.

Был создан веб-проект Java EE (содержание которого отбражено на рис.1) с сервлетом для обработки запросов, подготовки и сохранения данных, а также передачи управления в созданные соответствующие jsp-файлы, которые отрисовывают результат. Также созданы дополнительные классы для работы с данными таблиц.

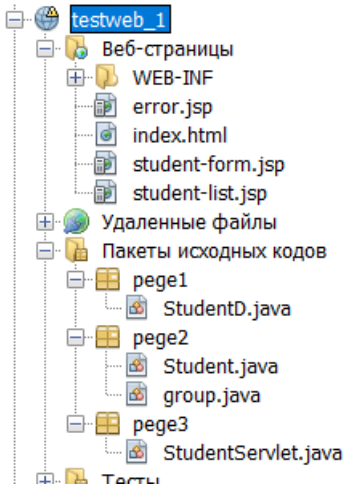

Рисунок 1

Класс Student для работы с данными студентов с соответствующими конструкторами, геттерами и сеттерами представлен на рис.2

Student.java

Источник | История

```
41      public Student(int id, String name, int groups_id, String email, String city) {
42          super();
43          this.id = id;
44          this.name = name;
45          this.groups_id = groups_id;
46          this.email = email;
47          this.city = city;
48      }
49      public int getId() {
50          return id;
51      }
52      public void setId(int id) {
53          this.id = id;
54      }
55      public String getName() {
56          return name;
57      }
58      public void setName(String name) {
59          this.name = name;
60      }
61      public String getEmail() {
62          return email;
63      }
64       public void setEmail(String email) {
65          this.email = email;
66      }
67        public String getGroups_name() {
68          return groups_name;
69      }
70      public int getGroups_id() {
71          return groups_id;
72      }
```

Рисунок 2

Также создан класс уровня доступа к данным StudentD, предоставляеющий операции создания, чтения, обновления, удаления данных таблиц в базе данных. Здесь созданы SQL запросы и методы работы с ними (часть которых на рис.3 и рис.4).

```
public class StudentD {
//данные для подключения

    private String url = "jdbc:mysql://127.0.0.1:3306/mydb?zeroDateTimeBehavior=convertToNull";
    private String username = "root";
    private String password = "1234";
//SQL запросы на получение, изменение, удаление данных из таблицы students
    private static final String INSERT_STUDENTS_SQL = "INSERT INTO students (name, groups_id, email, city) VALUES (?, ?, ?, ?);";
    private static final String SELECT_STUDENTS_BY_ID = "select id,name,groups_id,email,city from students where id =?";
    private static final String SELECT_ALL_STUDENTS = "select students.id, students.name, students.groups_id,groups.name,\n"
            + "students.email,  students.city from students INNER JOIN groups ON students.groups_id = groups.id";
    private static final String DELETE_STUDENTS_SQL = "delete from students where id = ?;";
    private static final String UPDATE_STUDENTS_SQL = "update students set name = ?,groups_id= ?,email= ?, city =? where id = ?;";
    //фильтр по имени и городу
    private static final String SELECT_FILTER_NC_STUDENTS = "select students.id, students.name, students.groups_id,groups.name,\n"
            + "students.email,  students.city from students INNER JOIN groups ON students.groups_id = groups.id where students.name = ? and students.city =?;";

    public StudentD() {
    }
//Подключение к серверу базы данных через JDBC driver

    protected Connection getConnection() {
        Connection connection = null;
        try {
            Class.forName("com.mysql.jdbc.Driver");
            connection = DriverManager.getConnection(url, username, password);
        } catch (SQLException e) {
            e.printStackTrace();
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        return connection;
    }
```

Рисунок 3

```
//вывод списка студентов
public List< Student> selectAllStudents() {
    List< Student> student = new ArrayList<>();
    try (Connection connection = getConnection();
        //использование подготовленного запроса
        PreparedStatement preparedStatement = connection.prepareStatement(SELECT_ALL_STUDENTS);) {
        System.out.println(preparedStatement);
        //получение результат запроса
        ResultSet rs = preparedStatement.executeQuery();
        //создание студента из полученных данных
        while (rs.next()) {
            int id = rs.getInt("id");
            String name = rs.getString("name");
            String groups_name = rs.getString("groups.name");
            String email = rs.getString("email");
            String city = rs.getString("city");
            student.add(new Student(id, name, groups_name, email, city));
        }
    } catch (SQLException e) {
        printSQLException(e);
    }
    return student;
}
//удаление студента
public boolean deleteStudent(int id) throws SQLException {
    boolean rowDeleted;
    //выполнение подготовленного SQL запроса удаления
    try (Connection connection = getConnection(); PreparedStatement statement = connection.prepareStatement(DELETE_STUDENTS_SQL);) {
        statement.setInt(1, id);
        rowDeleted = statement.executeUpdate() > 0;
    }
    return rowDeleted;
}
```

Рисунок 4

Для обработки клиентских запросов и генерации ответов на сервере создан класс сервлет (часть которого на рис.5 и рис.6), который работает как промежуточное звено между клиентскими запросами и базой данных. Здесь реализованы методы открытия, изменения, добавления, удаления студентов посредством получения данных с форм и передачи в вспомогательные классы.

StudentServlet.java

Источник | История

```
36      //методы оработки форм и получения параметров с помощью запросов post и get
37      @Override
        protected void doPost(HttpServletRequest request, HttpServletResponse response)
39              throws ServletException, IOException {
40          doGet(request, response);
41      }
42      @Override
        protected void doGet(HttpServletRequest request, HttpServletResponse response)
44              throws ServletException, IOException {
45          String action = request.getServletPath();
46      //вызов метода, соответсвущего вызванной части URL, относящейся я к текущему сервлету
47          try {
48              switch (action) {
49                  case "/new": //открытие формы создания
50                      showNewForm(request, response);
51                      break;
52                  case "/insert": //создание студента
53                      insertStudent(request, response);
54                      break;
55                  case "/delete": //удаление студента
56                      deleteStudent(request, response);
57                      break;
58                  case "/edit": //открытие формы изменения
59                      showEditForm(request, response);
60                      break;
61                  case "/update": //изменение студента
62                      updateStudent(request, response);
63                      break;
64                  case "/filter": //поиск студента
65                      filterStudent(request, response);
66                      break;
67                  default: //вывод списка студентов
68                      listStudent(request, response);
69                      break;
70              }
```

Рисунок 5

```
//открытие формы изменение студента с передачей параметров
private void showEditForm(HttpServletRequest request, HttpServletResponse response)
        throws SQLException, ServletException, IOException {
    int id = Integer.parseInt(request.getParameter("id"));
    Student existingStudent = studentD.selectStudent(id);
    RequestDispatcher dispatcher = request.getRequestDispatcher("student-form.jsp");
    request.setAttribute("student", existingStudent);
    dispatcher.forward(request, response);
}

//передача списка студентов, полученных в результате выполнения запроса вывода студентов
private void listStudent(HttpServletRequest request, HttpServletResponse response)
        throws SQLException, IOException, ServletException {
    List< Student> listStudent = studentD.selectAllStudents();
    request.setAttribute("listStudent", listStudent);
    RequestDispatcher dispatcher = request.getRequestDispatcher("student-list.jsp");
    dispatcher.forward(request, response);
}
//получение с формы и передача параметров для выполнение запроса создания студента
private void insertStudent(HttpServletRequest request, HttpServletResponse response)
        throws SQLException, IOException {
    String name = request.getParameter("name");
    int groups_id = Integer.parseInt(request.getParameter("groups_id"));
    String email = request.getParameter("email");
    String city = request.getParameter("city");
    Student newStudent = new Student(name,groups_id, email, city);
    studentD.insertStudent(newStudent);
    response.sendRedirect("list");
}
```

Рисунок 6

Также созданы динамические веб-страницы JSP, которые компилируются в сервлете и выполняются на сервере Java, генерируя содержимое страницы на основе полученных данных или выполнения логики. Страница student-list.jsp (часть на рис.7 и рис.8) отвечает за вывод списка студентов и отображение кнопок редактирования, удаления, добавления студентов, а также содержит поиск. Стараница student-form.jsp (часть на рис.9) отвечает за форму ввода данных при создании нового студента и изменении существующего.

```
<body>
    <!-- Панель быстрого переключения страниц-->
<header>
    <nav class="navbar navbar-expand-md navbar-dark" style="background-color: green">
        <div>
            <a class="navbar-brand"> Список студентов </a>
        </div>

        <ul class="navbar-nav">
            <li><a href="<%=request.getContextPath()%>/list" class="nav-link">Студенты</a></li>
        </ul>
    </nav>
</header>
<br>
 <!--название таблицы, кнопка добавления и форма поиска-->
<div class="row">
    <div class="container">
        <h3 class="text-center">Данные студентов</h3>
        <hr>
        <div class="container text-left">

            <a href="<%=request.getContextPath()%>/new" class="btn btn-success">Добавить студента</a>
        </div>
        <br>
            <form action="<%=request.getContextPath()%>/filter" method="post">
            <input  type="text" name="name" >
            <input type="text"  name="city" id='input'>
            <button class="btn btn-success" id='filtr_button' disabled>фильтр</button>
        <!--если данные не введены - кнопка поиска недоступна-->
            <script type="text/javascript">
                let inputElt = document.getElementById('input');
                let btn = document.getElementById('filtr_button');
                inputElt.addEventListener("input", function () {
                    btn.disabled = (this.value === '');
                })
            </script>
```

Рисунок 7

```
<table class="table table-bordered">
    <thead>
        <tr>
            <th>ID</th>
            <th>Имя</th>
            <th>Направление</th>
            <th>Email</th>
            <th>Город</th>
            <th>Редактировать</th>
        </tr>
    </thead>
    <tbody>

        <c:forEach var="student" items="${listStudent}">

            <tr>
                <td>
                    <c:out value="${student.id}" />
                </td>
                <td>
                    <c:out value="${student.name}" />
                </td>
                  <td>
                    <c:out value="${student.groups_name}" />
                </td>
                  <td>
                    <c:out value="${student.email}" />
                </td>
                <td>
                    <c:out value="${student.city}" />
                </td>
                <td><a href="edit?id=<c:out value='${student.id}' />">Изменить</a>
            </tr>
        </c:forEach>
```

Рисунок 8

```
<div class="card-body">
    <c:if test="${student != null}">
        <form action="update" method="post">
        </c:if>
        <c:if test="${student == null}">
            <form action="insert" method="post">
            </c:if>

                <caption>
                    <h2>
                        <c:if test="${student != null}">
                            Изменить студента
                        </c:if>
                        <c:if test="${student == null}">
                            Добавить студента
                        </c:if>

                    </h2>
                </caption>
                <c:if test="${student != null}">
                    <input type="hidden" name="id" value="<c:out value='${student.id}' />" />
                </c:if>

                <fieldset class="form-group">
                    <label>Имя</label> <input type="text" value="<c:out value='${student.name}' />" class="form-control" name="name" required="required">
                </fieldset>
                <fieldset class="form-group">
                    <label>Номер группы</label> <input type="text" value="<c:out value='${student.groups_id}' />" class="form-control" name="groups_id" re
                </fieldset>

                <fieldset class="form-group">
                    <label>Email</label> <input type="text" value="<c:out value='${student.email}' />" class="form-control" name="email">
                </fieldset>

                <fieldset class="form-group">
                    <label>Город</label> <input type="text" value="<c:out value='${student.city}' />" class="form-control" name="city">
                </fieldset>

                <button type="submit" class="btn btn-success">Сохранить</button>
            </form>
```

Рисунок 9

На рис. 10 представлена страница с таблицей студентов, содержащейся в базе данных. Здесь название направления подготовки получено из таблицы групп, связанной с таблицей студентов по внешнему ключу - номеру группы.

На странице кроме таблице представлены ссылки перехода между станицами на панеле быстрого доступа. Также размещены: кнопка добавления нового студента, поиск по таблице. На каждой строчке таблицы содержатся кнопки изменения и удаления соответствующие каждому студену.
Нажатие кнопок добавления и изменения студента приводят к открытию новой страницы (рис.11) с формой для ввода данных, из которой можно быстро вернуться назад путем нажатия ссылки на панеле быстрого доступа.

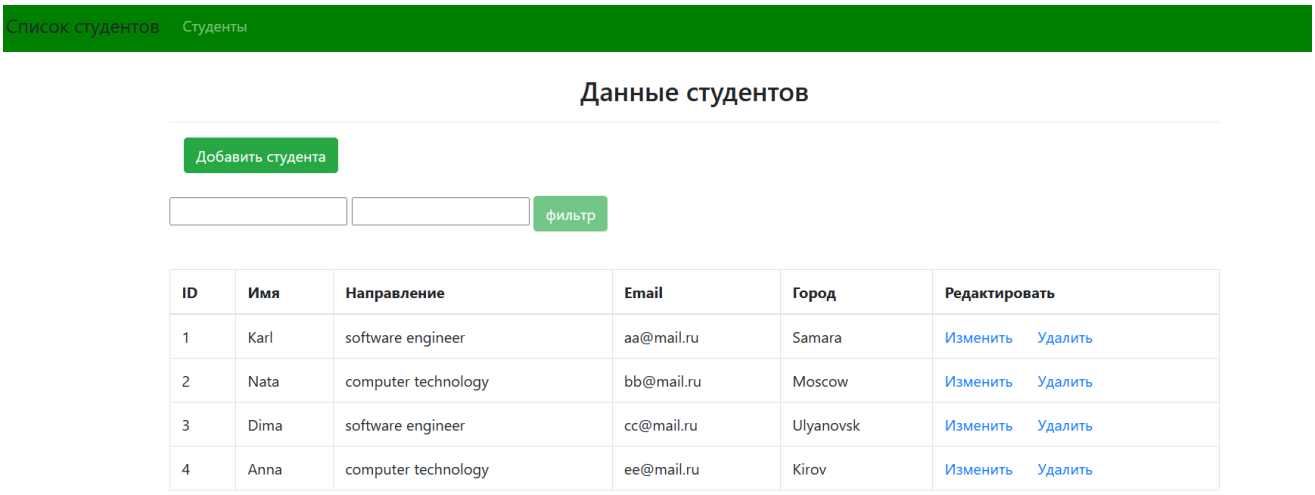

Рисунок 10

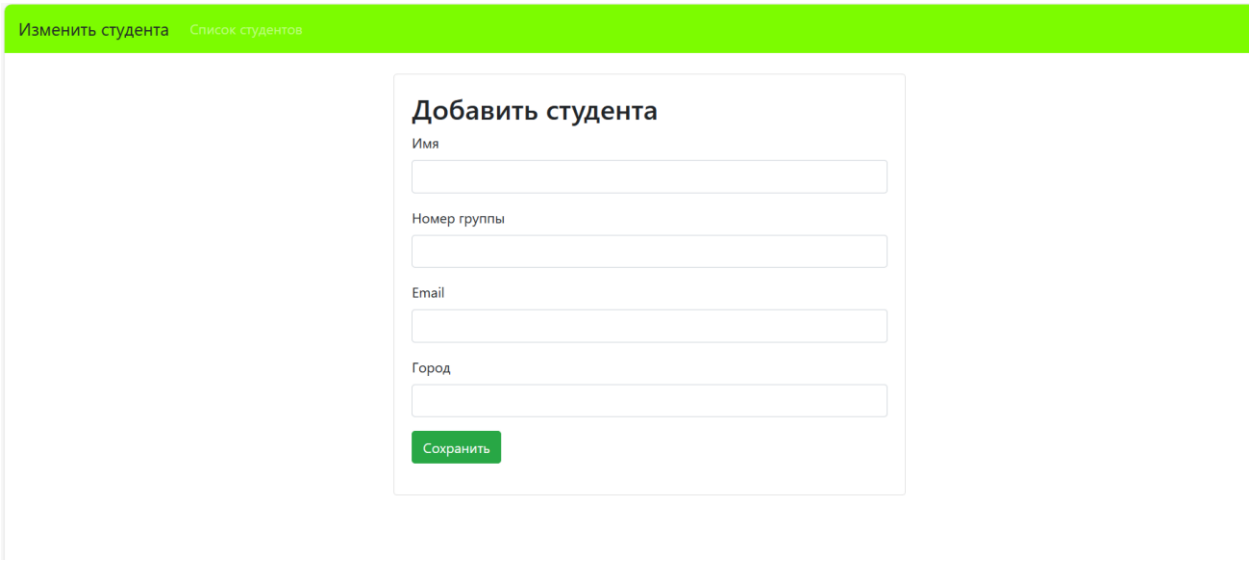

Рисунок 11

На рис 12. представлена форма добавления нового студента, куда были введены данные. Новый студент отобразился в списке студентов (рис.14) после нажатия на кнопку «Сохранить». На рис. 13 предсталена форма изменение студента, где была изменено поле «Email» выбранного студента, новые данные также отображены на рис.14. Также был удален студент из второй строчки.

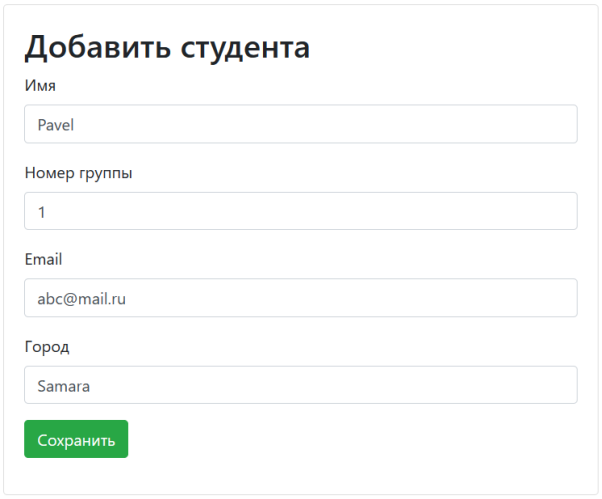

Рисунок 12

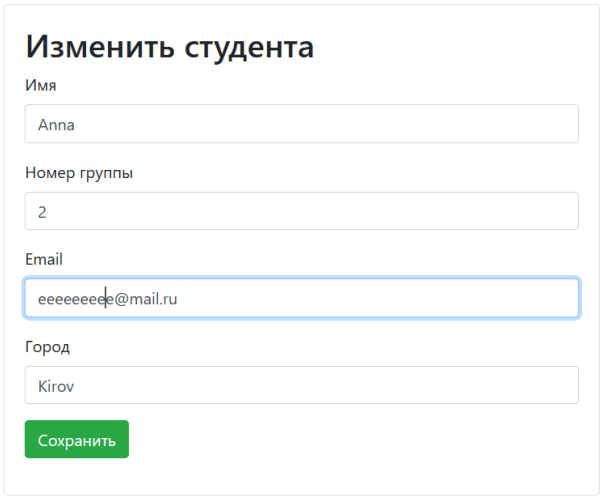

Рисунок 13

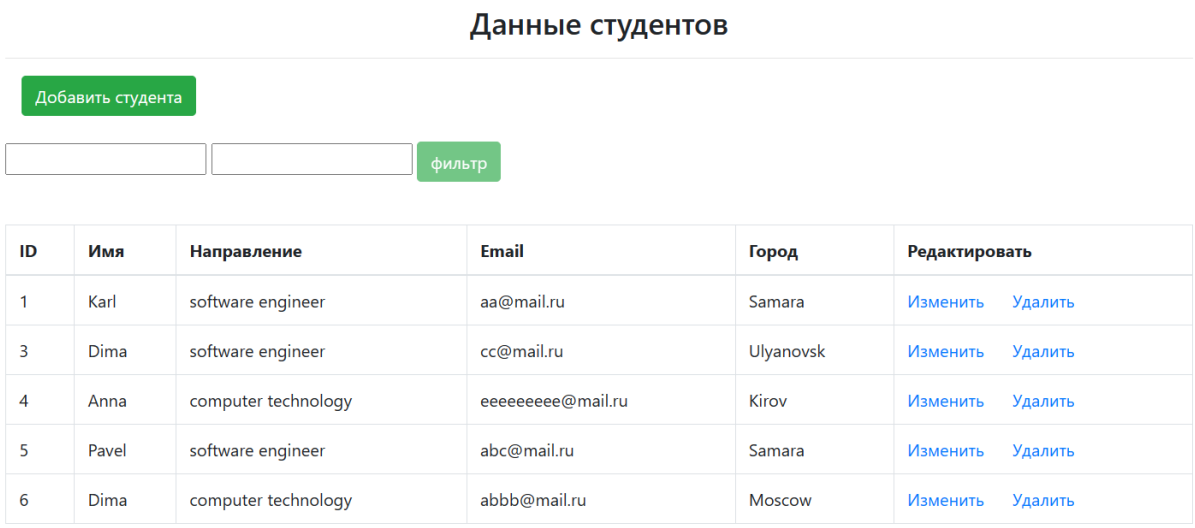

Рисунок 14

На рис. 15 представлен результат поиска студента по имени и городу.