**Задание.**

Реализовать web-приложение, принимающее заявки на подключение интернета.

1. Создать БД с таблицей «Заявки на подключение». Заявка содержит:

* имя абонента
* email абонента
* контактный телефон абонента
* адрес подключения

2. Создать web-приложение Java EE + Hibernate. Создать объект persistence api. Создать модель (EDM – Entity data model на основе БД из п.1.).

3. Разработать интерфейс DAO и имплементацию данного интерфейса, взаимодействующего с созданной ранее БД.

Интерфейс должен позволять:

* создавать заявку
* получить определенную заявку (по Id)
* получение заявок

4. Протестировать имплементацию DAO-объекта через юнит-тесты с демонстрацией работы.

5\*. Реализовать форму добавления заявки клиентом, просмотра ранее созданной заявки клиента.

+ отдельной страницей реализовать просмотре всех заявок (JSP + Servlet).

Servlet

package com.example.order\_conection;  
  
import DAO.OrderDAO;  
import OrderTEntity.OrderTEntity;  
import jakarta.servlet.\*;  
import jakarta.servlet.http.\*;  
import jakarta.servlet.annotation.\*;  
  
import java.io.IOException;  
import java.sql.SQLException;  
import java.util.List;  
  
@WebServlet("/")  
public class ControlServlet extends HttpServlet {  
 private static final long *serialVersionUID* = 1L;  
 private OrderDAO orderDAO;  
  
 @Override  
 public void init() {  
 orderDAO = new OrderDAO();  
  
 }  
  
 @Override  
 protected void doPost(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 doGet(request, response);  
 }  
  
 @Override  
 protected void doGet(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 String action = request.getServletPath();  
  
 try {  
 switch (action) {  
 case "/new":  
 showNewForm(request, response);  
 break;  
 case "/insert":  
 insertOrder(request, response);  
 break;  
 case "/delete":  
 deleteOrder(request, response);  
 break;  
 case "/edit":  
 showEditForm(request, response);  
 break;  
 case "/update":  
 updateOrder(request, response);  
 break;  
 default:  
 listOrders(request, response);  
 break;  
 }  
 } catch (SQLException ex) {  
 throw new ServletException(ex);  
 }  
 }  
  
 private void listOrders(HttpServletRequest request, HttpServletResponse response)  
 throws SQLException, IOException, ServletException {  
 List<OrderTEntity> listOrders = orderDAO.getAllOrders();  
 request.setAttribute("listOrders", listOrders);  
 RequestDispatcher dispatcher = request.getRequestDispatcher("OrderList.jsp");  
 dispatcher.forward(request, response);  
 }  
  
 private void showNewForm(HttpServletRequest request, HttpServletResponse response)  
 throws ServletException, IOException {  
 RequestDispatcher dispatcher = request.getRequestDispatcher("OrderForm.jsp");  
 dispatcher.forward(request, response);  
 }  
  
 private void showEditForm(HttpServletRequest request, HttpServletResponse response)  
 throws SQLException, ServletException, IOException {  
 Long id = Long.*valueOf*(request.getParameter("id"));  
 OrderTEntity existingOrder = orderDAO.getOrderById(id);  
 RequestDispatcher dispatcher = request.getRequestDispatcher("OrderForm.jsp");  
 request.setAttribute("order", existingOrder);  
 dispatcher.forward(request, response);  
 }  
  
 private void insertOrder(HttpServletRequest request, HttpServletResponse response)  
 throws SQLException, IOException {  
 String name = request.getParameter("name");  
 String email = request.getParameter("email");  
 String phone = request.getParameter("phone");  
 String address = request.getParameter("address");  
  
 OrderTEntity newOrder = new OrderTEntity();  
 newOrder.setName(name);  
 newOrder.setEmail(email);  
 newOrder.setPhone(phone);  
 newOrder.setAddress(address);  
 orderDAO.insertOrder(newOrder);  
 response.sendRedirect("list");  
 }  
  
 private void updateOrder(HttpServletRequest request, HttpServletResponse response)  
 throws SQLException, IOException {  
 Long id = Long.*parseLong*(request.getParameter("id"));  
 String name = request.getParameter("name");  
 String email = request.getParameter("email");  
 String phone = request.getParameter("phone");  
 String address = request.getParameter("address");  
  
 OrderTEntity newOrder = orderDAO.getOrderById(id);  
 newOrder.setName(name);  
 newOrder.setEmail(email);  
 newOrder.setPhone(phone);  
 newOrder.setAddress(address);  
 orderDAO.updateOrder(newOrder);  
 response.sendRedirect("list");  
 }  
  
 private void deleteOrder(HttpServletRequest request, HttpServletResponse response)  
 throws SQLException, IOException {  
 Long id = Long.*parseLong*(request.getParameter("id"));  
 orderDAO.deleteOrderById(id);  
 response.sendRedirect("list");  
  
 }  
}

Interface

package DAO;  
  
import OrderTEntity.OrderTEntity;  
  
import java.util.List;  
  
public interface IControlOrder {  
 void insertOrder(OrderTEntity order); // добавить  
 OrderTEntity getOrderById(Long id); // получить по id  
  
 List<OrderTEntity> getAllOrders(); // получить все  
  
 void updateOrder(OrderTEntity car); // обновить данные  
  
 void deleteOrderById(Long id); // удалить по id  
}

DAO

package DAO;  
  
import OrderTEntity.OrderTEntity;  
import jakarta.persistence.EntityManager;  
import jakarta.persistence.EntityManagerFactory;  
import jakarta.persistence.EntityTransaction;  
import jakarta.persistence.Persistence;  
  
import java.util.List;  
  
public class OrderDAO implements IControlOrder {  
  
 //CREATE  
 @Override  
 public void insertOrder(OrderTEntity order) {  
 // 1. создаем фабрику  
 EntityManagerFactory entityManagerFactory = Persistence.*createEntityManagerFactory*("default");  
 // 2. manager  
 EntityManager entityManager = entityManagerFactory.createEntityManager();  
 // 3. объект транзакции  
 EntityTransaction transaction = entityManager.getTransaction();  
 try {  
 // выполнение самой операции создания записи в БД  
 transaction.begin();  
 entityManager.persist(order); // создание новой записи на основе объекта  
 transaction.commit();  
 } finally {  
 // все закрыть и откатить транзакцию, если нужно  
 if (transaction.isActive()) {  
 transaction.rollback();  
 }  
 entityManager.close();  
 entityManagerFactory.close();  
 }  
 }  
  
 @Override  
 public OrderTEntity getOrderById(Long id) {  
 EntityManagerFactory entityManagerFactory = Persistence.*createEntityManagerFactory*("default");  
 EntityManager entityManager = entityManagerFactory.createEntityManager();  
 EntityTransaction transaction = entityManager.getTransaction();  
 // переменная для хранения объекта-результата поиска  
 OrderTEntity result = null;  
 try {  
 transaction.begin();  
 // сама операция  
 result = entityManager.find(OrderTEntity.class, id);  
 transaction.commit();  
 } finally {  
 if (transaction.isActive()) {  
 transaction.rollback();  
 }  
 entityManager.close();  
 entityManagerFactory.close();  
 }  
 return result; // вернуть результат  
 }  
  
 @Override  
 public List<OrderTEntity> getAllOrders() {  
 EntityManagerFactory entityManagerFactory = Persistence.*createEntityManagerFactory*("default");  
 EntityManager entityManager = entityManagerFactory.createEntityManager();  
 EntityTransaction transaction = entityManager.getTransaction();  
 // список результатов  
 List<OrderTEntity> orders = null;  
  
 try {  
 transaction.begin();  
 // сама операция  
 orders = entityManager.createQuery("SELECT e FROM OrderTEntity e").getResultList();  
 transaction.commit();  
 } finally {  
 if (transaction.isActive()) {  
 transaction.rollback();  
 }  
 entityManager.close();  
 entityManagerFactory.close();  
 }  
 return orders;  
 }  
  
 @Override  
 public void updateOrder(OrderTEntity order) {  
 EntityManagerFactory entityManagerFactory = Persistence.*createEntityManagerFactory*("default");  
 EntityManager entityManager = entityManagerFactory.createEntityManager();  
 EntityTransaction transaction = entityManager.getTransaction();  
 try {  
 transaction.begin();  
 // 1. получаю обновляемый объект по Id  
 OrderTEntity updated = entityManager.find(OrderTEntity.class, order.getId());  
 // 2. обновляю поля обновляемого объекта  
 updated.updateFields(order);  
 transaction.commit();  
 } finally {  
 if (transaction.isActive()) {  
 transaction.rollback();  
 }  
 entityManager.close();  
 entityManagerFactory.close();  
 }  
 }  
  
 @Override  
 public void deleteOrderById(Long id) {  
 EntityManagerFactory entityManagerFactory = Persistence.*createEntityManagerFactory*("default");  
 EntityManager entityManager = entityManagerFactory.createEntityManager();  
 EntityTransaction transaction = entityManager.getTransaction();  
  
 try {  
 transaction.begin();  
 // сама операция  
 // 1. получить удаляемый объект  
 OrderTEntity deleted = entityManager.find(OrderTEntity.class, id);  
 // 2. удалить  
 entityManager.remove(deleted);  
 transaction.commit();  
 } finally {  
 if (transaction.isActive()) {  
 transaction.rollback();  
 }  
 entityManager.close();  
 entityManagerFactory.close();  
 }  
 }  
}

Class

package OrderTEntity;  
  
import jakarta.persistence.\*;  
  
@Entity  
@Table(name = "auto", schema = "db\_car", catalog = "")  
public class OrderTEntity {  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 @Id  
 @Column(name = "id")  
 private Long id;  
 @Basic  
 @Column(name = "name")  
 private String name;  
 @Basic  
 @Column(name = "email")  
 private String email;  
 @Basic  
 @Column(name = "phone")  
 private String phone;  
 @Basic  
 @Column(name = "address")  
 private String address;  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getEmail() {  
 return email;  
 }  
  
 public void setEmail(String email) {  
 this.email = email;  
 }  
  
 public String getPhone() {  
 return phone;  
 }  
  
 public void setPhone(String phone) {  
 this.phone = phone;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 //копирование полей объекта  
 public void updateFields(OrderTEntity src) {  
 name = src.name;  
 phone = src.phone;  
 email = src.email;  
 address = src.address;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
  
 OrderTEntity that = (OrderTEntity) o;  
  
 if (id != null ? !id.equals(that.id) : that.id != null) return false;  
 if (name != null ? !name.equals(that.name) : that.name != null) return false;  
 if (email != null ? !email.equals(that.email) : that.email != null) return false;  
 if (phone != null ? !phone.equals(that.phone) : that.phone != null) return false;  
 if (address != null ? !address.equals(that.address) : that.address != null) return false;  
  
 return true;  
 }  
  
 @Override  
 public int hashCode() {  
 int result = id != null ? id.hashCode() : 0;  
 result = 31 \* result + (name != null ? name.hashCode() : 0);  
 result = 31 \* result + (email != null ? email.hashCode() : 0);  
 result = 31 \* result + (phone != null ? phone.hashCode() : 0);  
 result = 31 \* result + (address != null ? address.hashCode() : 0);  
 return result;  
 }  
}

Form JSP

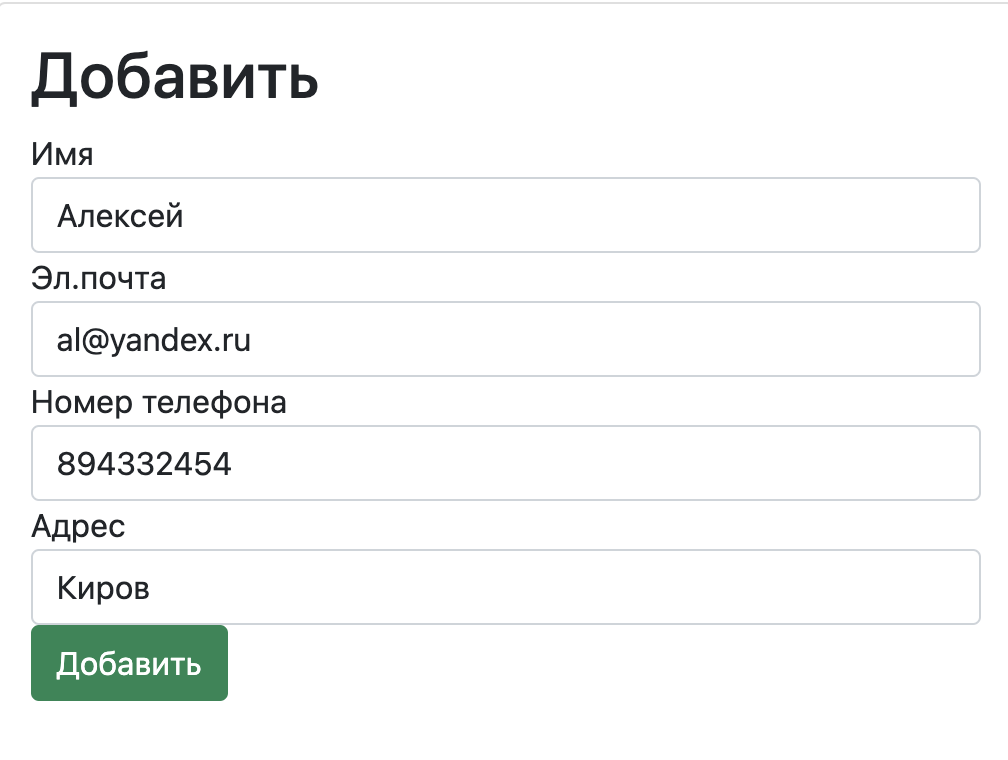
<%--  
 Created by IntelliJ IDEA.  
 User: aleksandr  
 Date: 15.10.2022  
 Time: 23:05  
 To change this template use File | Settings | File Templates.  
--%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<html>  
<head>  
 <title>Страница заказов</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">  
 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"  
 integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"  
 crossorigin="anonymous"></script>  
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"  
 integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"  
 crossorigin="anonymous"></script>  
</head>  
<header>  
<%-- <nav class="navbar navbar-expand-md navbar-dark"--%>  
<%-- style="background-color: gray">--%>  
<%-- <div>--%>  
<%-- <a href="<%=request.getContextPath()%>/list" class="btn btn-primary btn-lg disabled">На главную</a>--%>  
<%-- </div>--%>  
<%-- </nav>--%>  
</header>  
<br>  
<div class="container col-md-5">  
 <div class="card">  
 <div class="card-body">  
 <**c:if** test="${order != null}">  
 <form action="update" method="post">  
 </**c:if**>  
 <**c:if** test="${order == null}">  
  
 <form action="insert" method="post">  
 </**c:if**>  
  
 <caption>  
 <h2>  
 <**c:if** test="${order != null}">  
 Изменить  
 </**c:if**>  
 <**c:if** test="${order == null}">  
 Добавить  
 </**c:if**>  
 </h2>  
 </caption>  
  
 <**c:if** test="${order != null}">  
 <input type="hidden" name="id" value="<**c:out** value='${order.id}' />"/>  
 </**c:if**>  
  
 <fieldset class="form-group">  
 <label>Имя</label> <input type="text"  
 value="<**c:out** value='${order.name}' />"  
 class="form-control"  
 name="name" required="required">  
 </fieldset>  
  
 <fieldset class="form-group">  
 <label>Эл.почта</label> <input type="text"  
 value="<**c:out** value='${order.email}' />" class="form-control"  
 name="email">  
 </fieldset>  
  
 <fieldset class="form-group">  
 <label>Номер телефона</label> <input type="text"  
 value="<**c:out** value='${order.phone}' />" class="form-control"  
 name="phone">  
 </fieldset>  
 <fieldset class="form-group">  
 <label>Адрес</label> <input type="text"  
 value="<**c:out** value='${order.address}' />" class="form-control"  
 name="address">  
 </fieldset>  
  
 <button type="submit" class="btn btn-success">Добавить</button>  
 </form>  
 </div>  
 </div>  
</div>  
</body>  
</html>

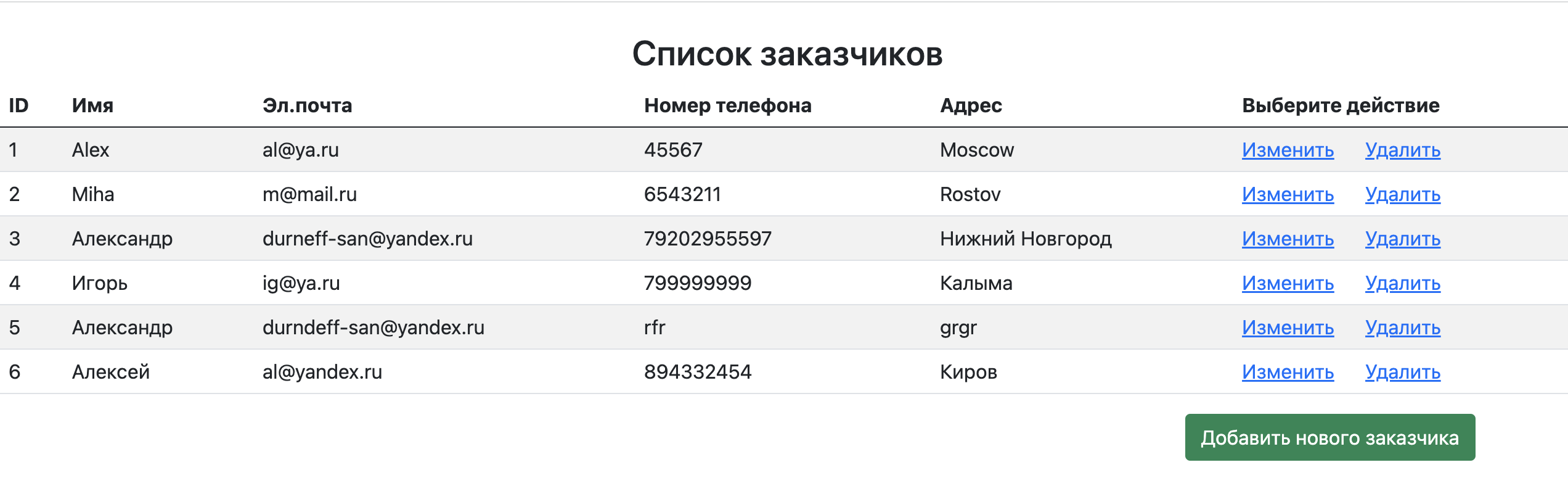
List JSP

<%--  
 Created by IntelliJ IDEA.  
 User: aleksandr  
 Date: 15.10.2022  
 Time: 23:07  
 To change this template use File | Settings | File Templates.  
--%>  
<%@ **page** contentType="text/html;charset=UTF-8" language="java" %>  
<%@ **taglib** prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<html>  
<head>  
 <title>Заказы</title>  
 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css" rel="stylesheet"  
 integrity="sha384-EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC" crossorigin="anonymous">  
 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min.js"  
 integrity="sha384-IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p"  
 crossorigin="anonymous"></script>  
 <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.min.js"  
 integrity="sha384-cVKIPhGWiC2Al4u+LWgxfKTRIcfu0JTxR+EQDz/bgldoEyl4H0zUF0QKbrJ0EcQF"  
 crossorigin="anonymous"></script>  
</head>  
<body>  
  
<header>  
<%-- <nav class="navbar navbar-expand-lg navbar-light bg-light"--%>  
<%-- style="background-color: gray">--%>  
<%-- <div>--%>  
<%-- <a href="<%=request.getContextPath()%>/list" class="btn btn-primary btn-lg disabled">На главную</a>--%>  
<%-- </div>--%>  
<%-- </nav>--%>  
</header>  
<br>  
  
<div class="row">  
  
 <div class="container">  
 <h3 class="text-center">Список заказчиков</h3>  
<%-- <hr>--%>  
<%-- <div class="container text-left" style="text-align: center">--%>  
  
<%-- <a href="<%=request.getContextPath()%>/new" class="btn btn-success">Добавить</a>--%>  
<%-- </div>--%>  
<%-- <br>--%>  
 <table class="table table-striped">  
 <thead>  
 <tr>  
 <th>ID</th>  
 <th>Имя</th>  
 <th>Эл.почта</th>  
 <th>Номер телефона</th>  
 <th>Адрес</th>  
 <th>Выберите действие</th>  
 </tr>  
 </thead>  
 <tbody>  
  
 <**c:forEach** var="order" items="${listOrders}">  
  
 <tr>  
 <td><**c:out** value="${order.id}"/></td>  
 <td><**c:out** value="${order.name}"/></td>  
 <td><**c:out** value="${order.email}"/></td>  
 <td><**c:out** value="${order.phone}"/></td>  
 <td><**c:out** value="${order.address}"/></td>  
 <td><a href="edit?id=<**c:out** value='${order.id}' />">Изменить</a>  
 &nbsp;&nbsp;&nbsp;&nbsp; <a  
 href="delete?idF=<**c:out** value='${order.id}' />">Удалить</a></td>  
 </tr>  
  
 </**c:forEach**>  
  
 </tbody>  
 </table>  
 </div>  
</div>  
  
<div class="container text-left" style="text-align: right">  
 <a href="<%=request.getContextPath()%>/new" class="btn btn-success">Добавить нового заказчика</a>  
</div>  
</body>  
</html>

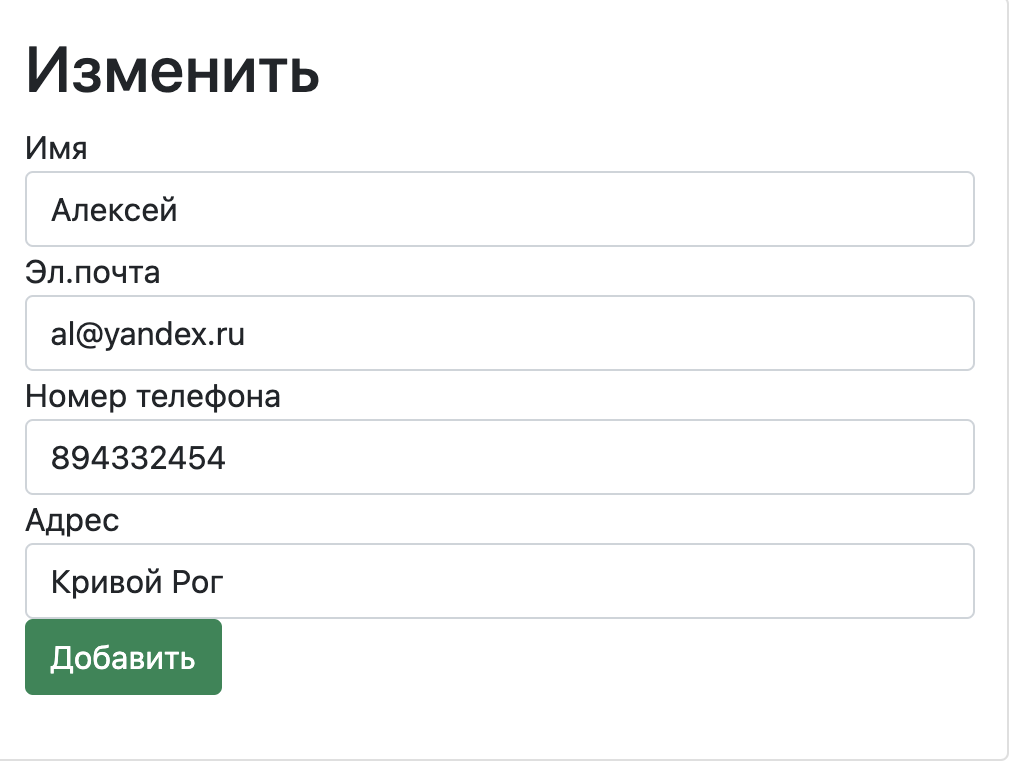


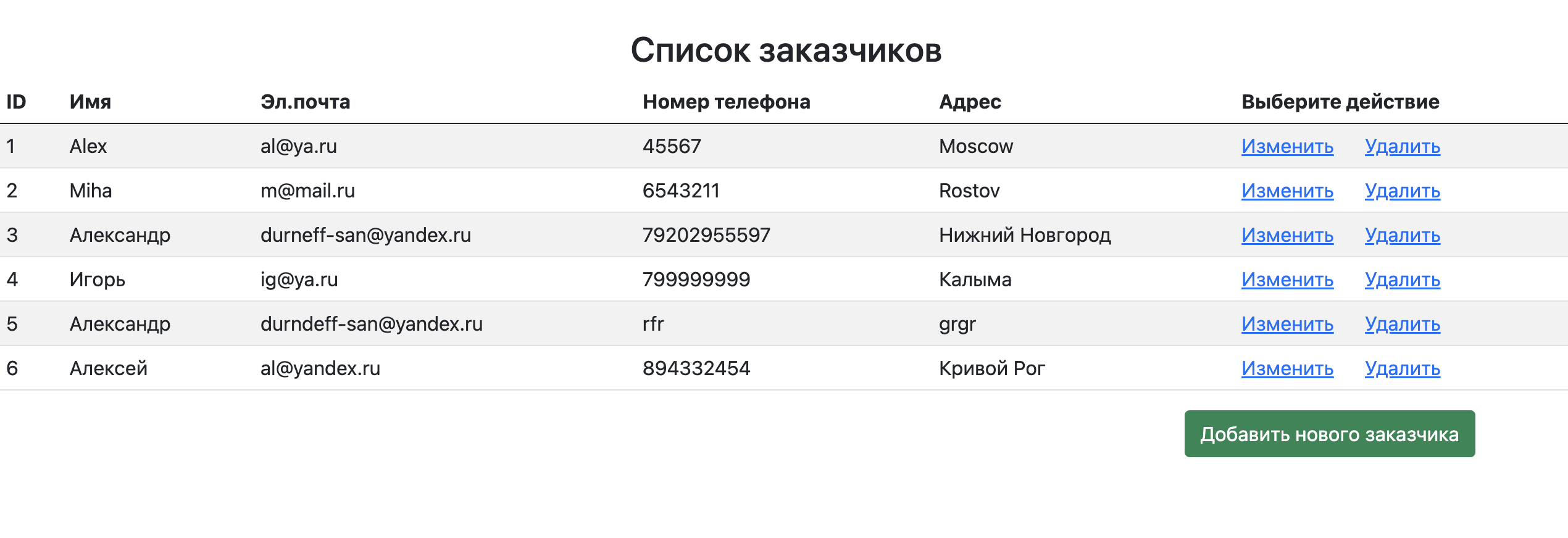
Добавляем





Изменяем





Удаляем

