Министерство образования и науки Российской Федерации

**Федеральное государственное бюджетное образовательное учреждение науки и высшего образования**

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»

(ВлГУ)

**Институт информационных технологий и радиотехники**

**(ИИТР)**

Кафедра информационных систем и программной инженерии

**Лабораторная работа № 01**

**по дисциплине**

**««Распределенные программные системы»**

Выполнил:

ст. гр. ПРИ-117

Хлызова В.Г.

Принял:

Трифонов Д.А.

Владимир, 2020

ЦЕЛЬ РАБОТЫ

Познакомиться с принципами построения веб-приложений с использованием шаблона MVC и на базе принципов REST, получить практические навыки реализации MVC-приложения на платформе Spring MVC.

ЗАДАНИЕ

1. Разработать простейшее веб-приложение на платформе Spring MVC. Приложение должно состоять из одной JSP-страницы, которая является т.н. index-страницей (возвращается по умолчанию при обращении к приложению).
2. Разработать один (или более) Java-класс, представляющий модель данных приложения.
3. Разработать необходимые Java-классы, реализующие шаблон Data Access Object и обеспечивающие независимость бизнес-логики от механизма хранения данных.
4. Для каждого класса модели данных в объекте доступа к данным должны быть предусмотрены так называемые CRUD-операции: сохранение, изменение, удаление экземпляра, а также извлечение экземпляра по идентификатору и получение всех экземпляров. Настроить подключение источника данных с помощью инъекции зависимостей, реализуемой Spring-фреймворком.
5. Для каждого класса модели данных реализовать Spring-контроллер, обрабатывающий http-запросы к соответствующему ресурсу в стиле REST. Настроить подключение необходимого DAO-класса в класс контроллера с помощью инъекции зависимостей, реализуемой Spring-фреймворком.
6. Для каждого класса модели данных реализовать JSP-страницы представления, соответствующие CRUD-операциям: страница просмотра информации об объекте, форма создания, форма изменения, страница сотображением списка экземпляров класса. На страницах для обработки динамического контента использовать EL выражения и тэги библиотеки JSTL.

ВОПРОСЫ

1. Какие шаги необходимо выполнить для того, чтобы реализовать простейшее веб-приложение на платформе Spring MVC?

Создаем maven проект, затем в нем структуру веб-приложения. Добавляем в pom зависимости, которые нам будут необходимы для полноценного построения Spring MVC приложения – библиотеки spring,  библиотека тегов для jsp страниц – jstl, log4j для логирования и драйвер для mysql. Настраиваем конфигурацию(web.xml и spring-servlet.xml). Далее создаем модель данных и класс, реализующий шаблон dao.  Затем создаём страничку и обработчик событий этой страницы (контроллер) (Описаны шаги создания Spring MVC приложения, которым я следовала при выполнении данной лабораторной работы)

1. Какие классы модели данных будут разработаны, каким реляционным таблицам они соответствуют? Какие поля будут содержаться в каждом из классов модели данных.

В данной лабораторной работе был разработал класс User(соответствует таблице user, название БД - lab07). Поля класса User соответствуют атрибутам в таблице user, а именно: userID, login, password, email.

ВЫПОЛНЕНИЕ РАБОТЫ

### Создаем мавен проект и добавляем зависимости в pom.xml.

*<?***xml version="1.0" encoding="UTF-8"***?>*<**project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"**>  
 <**modelVersion**>4.0.0</**modelVersion**>  
  
 <**groupId**>lab07</**groupId**>  
 <**artifactId**>lab07</**artifactId**>  
 <**version**>1.0-SNAPSHOT</**version**>  
 <**packaging**>war</**packaging**>  
  
 <**name**>lab07</**name**>  
 *<!-- FIXME change it to the project's website -->* <**url**>http://www.example.com</**url**>  
  
 <**properties**>  
 <**project.build.sourceEncoding**>UTF-8</**project.build.sourceEncoding**>  
 <**maven.compiler.source**>1.7</**maven.compiler.source**>  
 <**maven.compiler.target**>1.7</**maven.compiler.target**>  
 </**properties**>  
  
 <**dependencies**>  
 <**dependency**>  
 <**groupId**>junit</**groupId**>  
 <**artifactId**>junit</**artifactId**>  
 <**version**>4.11</**version**>  
 <**scope**>test</**scope**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-jdbc</**artifactId**>  
 <**version**>5.2.3.RELEASE</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-webmvc</**artifactId**>  
 <**version**>5.2.3.RELEASE</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-core</**artifactId**>  
 <**version**>5.2.3.RELEASE</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-context</**artifactId**>  
 <**version**>5.2.3.RELEASE</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>mysql</**groupId**>  
 <**artifactId**>mysql-connector-java</**artifactId**>  
 <**version**>8.0.11</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.springframework</**groupId**>  
 <**artifactId**>spring-web</**artifactId**>  
 <**version**>5.2.3.RELEASE</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>javax.servlet</**groupId**>  
 <**artifactId**>servlet-api</**artifactId**>  
 <**version**>3.0-alpha-1</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>javax.servlet</**groupId**>  
 <**artifactId**>jstl</**artifactId**>  
 <**version**>1.2</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.slf4j</**groupId**>  
 <**artifactId**>slf4j-api</**artifactId**>  
 <**version**>1.7.25</**version**>  
 </**dependency**>  
 <**dependency**>  
 <**groupId**>org.slf4j</**groupId**>  
 <**artifactId**>slf4j-log4j12</**artifactId**>  
 <**version**>1.7.25</**version**>  
 </**dependency**>  
 </**dependencies**>  
</**project**>

## Настраиваем логирование. В папке [src](https://github.com/arranay/LAB-RPS/tree/master/lab07/lab07/src)/[main](https://github.com/arranay/LAB-RPS/tree/master/lab07/lab07/src/main)/**resources**/ создаем файл **log4j.properties** с конфигурацией подсистемы логирования. Логи будут выводится в консоль, по указанному шаблону. Содержимое файла:

**log4j.appender.stdout**=**org.apache.log4j.ConsoleAppender  
log4j.rootLogger**=**INFO, stdout  
log4j.appender.stdout.layout**=**org.apache.log4j.PatternLayout  
log4j.appender.stdout.layout.ConversionPattern**= **[ %-5p] - %c:%L %d{HH:mm:ss.SSS} - %m%n**

**Конфигурирование.** В установочном дескрипторе подключаем DispatcherServlet, чтобы Spring MVC приложение могло обрабатывать входящие запросы. Так же в этом файле прописываем CharacterEncodingFilter, отвечающий за кодировку запросов. Содержание файла web.xml:

*<?***xml version="1.0" encoding="UTF-8"***?>*<**web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns="http://java.sun.com/xml/ns/javaee"  
 xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
 http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"  
 id="WebApp\_ID" version="3.0"**>  
  
 <**servlet**>  
 <**servlet-name**>spring</**servlet-name**>  
 <**servlet-class**>org.springframework.web.servlet.DispatcherServlet</**servlet-class**>  
 <**load-on-startup**>1</**load-on-startup**>  
 </**servlet**>  
 <**servlet-mapping**>  
 <**servlet-name**>spring</**servlet-name**>  
 <**url-pattern**>/</**url-pattern**>  
 </**servlet-mapping**>  
  
 <**filter**>  
 <**filter-name**>CharacterEncodingFilter</**filter-name**>  
 <**filter-class**>org.springframework.web.filter.CharacterEncodingFilter</**filter-class**>  
 <**init-param**>  
 <**param-name**>encoding</**param-name**>  
 <**param-value**>UTF-8</**param-value**>  
 </**init-param**>  
 <**init-param**>  
 <**param-name**>forceEncoding</**param-name**>  
 <**param-value**>true</**param-value**>  
 </**init-param**>  
 </**filter**>  
 <**filter-mapping**>  
 <**filter-name**>CharacterEncodingFilter</**filter-name**>  
 <**url-pattern**>/\*</**url-pattern**>  
 </**filter-mapping**>  
</**web-app**>

**Определяем бины** в конфигурационном файле Spring-приложения (у меня это **spring-servlet.xml**) Сначала указываем чтобы поиск компонентов осуществлялся в пакете lab07.controllers. Далее описываем интерфейс ViewResolver, он будет определять какое представление необходимо использовать на основании полученного имени.  Описываем источник данных (ссылку на БД, логин и пароль) В конце настраиваем JdbcTemplate – класс, который будет управлять обработкой всех событий в БД. Содержимое файла:

*<?***xml version="1.0" encoding="UTF-8"***?>*<**beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xmlns:context="http://www.springframework.org/schema/context"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd"**>  
  
 <**context:component-scan base-package="lab07.controllers"**></**context:component-scan**>  
  
 <**bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"**>  
 <**property name="prefix" value="/WEB-INF/views/"**></**property**>  
 <**property name="suffix" value=".jsp"**></**property**>  
 </**bean**>  
  
 <**bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource"**>  
 <**property name="driverClassName" value="com.mysql.jdbc.Driver"**></**property**>  
 <**property name="url" value="jdbc:mysql://localhost:3306/lab07?useSSL=false&amp;useUnicode=true&amp;useJDBCCompliantTimezoneShift=true&amp;useLegacyDatetimeCode=false&amp;serverTimezone=UTC"**></**property**>  
 <**property name="username" value="root"**></**property**>  
 <**property name="password" value="root"**></**property**>  
 </**bean**>  
  
 <**bean id="jt" class="org.springframework.jdbc.core.JdbcTemplate"**>  
 <**property name="dataSource" ref="ds"**></**property**>  
 </**bean**>  
  
 <**bean id="UserDao" class="lab07.dao.UserDao"**>  
 <**property name="template" ref="jt"**></**property**>  
 </**bean**>  
</**beans**>

### Создаем модель данных.

**package** lab07.models;  
  
**public class** User {  
 **private int userId**;  
 **private** String **login**;  
 **private** String **password**;  
 **private** String **email**;  
  
 **public** User(){  
 }  
  
 **public** User(**int** id, String login, String password, String email){  
 **userId** = id;  
 **this**.**login** = login;  
 **this**.**password** = password;  
 **this**.**email** = email;  
 }  
  
 **public int** getUserId() {  
 **return userId**;  
 }  
  
 **public void** setUserId(**int** userId) {  
 **this**.**userId** = userId;  
 }  
  
 **public** String getLogin() {  
 **return login**;  
 }  
  
 **public void** setLogin(String login) {  
 **this**.**login** = login;  
 }  
  
 **public** String getPassword() {  
 **return password**;  
 }  
  
 **public void** setPassword(String password) {  
 **this**.**password** = password;  
 }  
  
 **public** String getEmail() {  
 **return email**;  
 }  
  
 **public void** setEmail(String email) {  
 **this**.**email** = email;  
 }  
}

### Создаем класс UserDao.

**package** lab07.dao;  
  
**import** lab07.models.User;  
**import** org.slf4j.Logger;  
**import** org.slf4j.LoggerFactory;  
**import** org.springframework.jdbc.core.BeanPropertyRowMapper;  
**import** org.springframework.jdbc.core.JdbcTemplate;  
**import** org.springframework.jdbc.core.RowMapper;  
**import** org.springframework.stereotype.Repository;  
  
**import** javax.xml.bind.SchemaOutputResolver;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.sql.Types;  
**import** java.util.List;  
  
@Repository  
**public class** UserDao {  
 JdbcTemplate template;  
 **private static** Logger logger= LoggerFactory.getLogger(UserDao.**class**);  
  
 **public** List<User> getAllUsers(){  
 logger.info(**"Выполнение метода getAllUsers для вывода всех пользователей"**);  
 **try**{  
 **return** template.query(**"select \* from user"**,**new** RowMapper<User>(){  
  
 **public** User mapRow(ResultSet rs, **int** row) **throws** SQLException {  
 User u = **new** User();  
 u.setUserId(rs.getInt(**"userId"**));  
 u.setLogin(rs.getString(**"login"**));  
 u.setPassword(rs.getString(**"password"**));  
 u.setEmail(rs.getString(**"email"**));  
 **return** u;  
 }  
 });  
  
 }**catch** (Exception e){  
 logger.error(**"Ошибка при выполнении метода listAllTasks: "**, e);  
 **return null**;  
 }  
 }

Пояснение: RowMapper обрабатывает отдельно каждую запись, полученную из БД, и возвращает список этих записей. Каждая строка здесь обрабатывается отдельно при помощи объекта **ResultSet**. Если при получении данных возникают ошибки, то возвращается null и на странице выводится сообщение, что в таблице нет записей.

**public** User getUserById(**int** idUser){  
 logger.info(**"Выполнение метода getUserById - получаем информацию о конкретном пользователе"**);  
 String query=**"select\*from user where userId=?"**;  
 **try**{  
 **return** template.queryForObject(query, **new** Object[]{idUser}, **new** BeanPropertyRowMapper<User>(User.**class**));  
 }**catch** (Exception e) {  
 logger.error(**"Ошибка при выполнении метода getUserById: "**, e);  
 **return null**;  
 }  
 }

Пояснение: в методе queryForObject указываем строку запроса, объект (в нашем случае это id пользователя) который будет подставлен в запрос вместо ? и указываем объект какого класса, который будет возращен.

**public int** update(User user){  
 logger.info(**"Выполнение метода update - изменение данных о пользователе"**);  
 String query=**"update user set login=?, password=?, email=? where userId=?"**;  
 Object[] params = {user.getLogin(), user.getPassword(), user.getEmail(), user.getUserId()};  
 **int**[] types = {Types.VARCHAR, Types.VARCHAR, Types.VARCHAR, Types.INTEGER};  
 **try** {  
 **return template**.update(query,params,types);  
 }**catch** (Exception e) {  
 *logger*.error(**"Ошибка при выполнении метода update: "**, e);  
 **return** -1;  
 }  
 }

Пояснение: Создаем строку запроса, массив параметров и массив типов этих параметров в бд. Вызывается метод template.update, который возвращает id записи, которая была изменена.

**public int** insert(User user){  
 logger.info(**"Выполнение метода insert - добавление нового пользователя"**);  
 String query=**"insert into user(login, password, email) values (?, ?, ?)"**;  
 Object[] params = {user.getLogin(), user.getPassword(), user.getEmail()};  
 **int**[] types = {Types.VARCHAR, Types.VARCHAR, Types.VARCHAR};  
 **try** {  
 **return** template.update(query,params,types);  
 }**catch** (Exception e) {  
 logger.error(**"Ошибка при выполнении метода insert: "**, e);  
 **return** -1;  
 }  
 }  
  
 **public int** delete(**int** idUser){  
 logger.info(**"Выполнение метода delete - удаление пользователя"**);  
 String query=**"delete from user where userId=?"**;  
 Object[] params = {idUser};  
 **int**[] types = {Types.INTEGER};  
 **try** {  
 **return** template.update(query,params,types);  
 }**catch** (Exception e) {  
 logger.error(**"Ошибка при выполнении метода delete: "**, e);  
 **return** -1;  
 }  
 }  
  
 **public void** setTemplate(JdbcTemplate template) {  
 **this**.template = template;  
 }  
}

### Создаем обработчики на все операции.

**package** lab07.controllers;  
  
**import** lab07.dao.UserDao;  
**import** lab07.models.User;  
**import** org.springframework.beans.factory.annotation.Autowired;  
**import** org.springframework.stereotype.Controller;  
**import** org.springframework.ui.Model;  
**import** org.springframework.web.bind.annotation.ModelAttribute;  
**import** org.springframework.web.bind.annotation.PathVariable;  
**import** org.springframework.web.bind.annotation.RequestMapping;  
**import** org.springframework.web.bind.annotation.RequestMethod;  
  
**import** java.util.List;  
  
@Controller  
**public class** UserController {  
 @Autowired  
 UserDao userDao;  
  
 @RequestMapping(**"/viewusers"**)//Пояснение: указание какой url будет обрабатываться  
 **public** String viewUsers(Model m){  
 List<User> list=userDao.getAllUsers();//Пояснение: добавление атрибутов в представление  
 m.addAttribute(**"list"**,list);  
 **return "viewusers"**;  
 }  
  
 @RequestMapping(**"index"**)  
 **public** String back(){  
 **return "../../index"**;  
 }  
  
 @RequestMapping(value=**"/show/{id}"**)//Пояснение: @PathVariable используется для работы с параметрами, передаваемыми через адрес запроса  
 **public** String edit(@PathVariable **int** id, Model m){  
 User user=userDao.getUserById(id);  
 m.addAttribute(**"command"**,user);  
 **return "editUser"**;  
 }  
  
 @RequestMapping(value = **"show/editsave"**)  
 **public** String editSave(@ModelAttribute(**"command"**) User user){  
 **int** id = **userDao**.update(user);  
 **if** (id!=-1) **return "redirect:../viewusers"**;  
 **else return "redirect:/error"**;  
 }  
  
 @RequestMapping(value=**"/saveuser"**)  
 **public** String saveUser(@ModelAttribute(**"command"**) User user){  
 **int** id = userDao.insert(user);  
 **if** (id!=-1) **return "redirect:/viewusers"**;  
 **else return "redirect:/error"**;//Пояснение: возвращаем страницу error если при изменении записи произошли ошибки  
 }  
  
 @RequestMapping(**"/addUser"**)  
 **public** String addUser(Model m){  
 m.addAttribute(**"command"**, **new** User());  
 **return "addUser"**;  
 }  
  
 @RequestMapping(value = **"/delete/{id}"**)  
 **public** String delete(@PathVariable **int** id){  
 **int** i = userDao.delete(id);  
 **if** (i!=-1) **return "redirect:/viewusers"**;  
 **else return "redirect:/error"**;  
 }  
  
 @RequestMapping(**"/error"**)  
 **public** String viewUsers(){  
 **return "error"**;  
 }  
}

### Jsp страницы.

<%@ **page contentType**="**text/html;charset=UTF-8**" **language**="**java**" %>  
<**html**>  
<**head**>  
 <**title**>Менюшечка -\_-</**title**>  
 <**style**>  
 **a**:**link**{**text-decoration**: **none**; **color**: **green**; **font-size**: 15**px**;}  
 **a**:**hover**{**text-decoration**: **none**; **color**: **green**; **font-size**: 20**px**;}  
 **a**:**visited**{**text-decoration**: **none**; **color**: **green**}  
 **h3**{**color**: **green**}  
 </**style**>  
</**head**>  
<**body**>  
<**h3**>Менюшечка</**h3**>  
<**ul**>  
 <**li**><**a href="viewusers"**>Посмотреть список пользователей</**a**>  
 <**li**><**a href="addUser"**>Добавить нового пользователя</**a**>  
</**ul**>  
</**body**>  
</**html**>

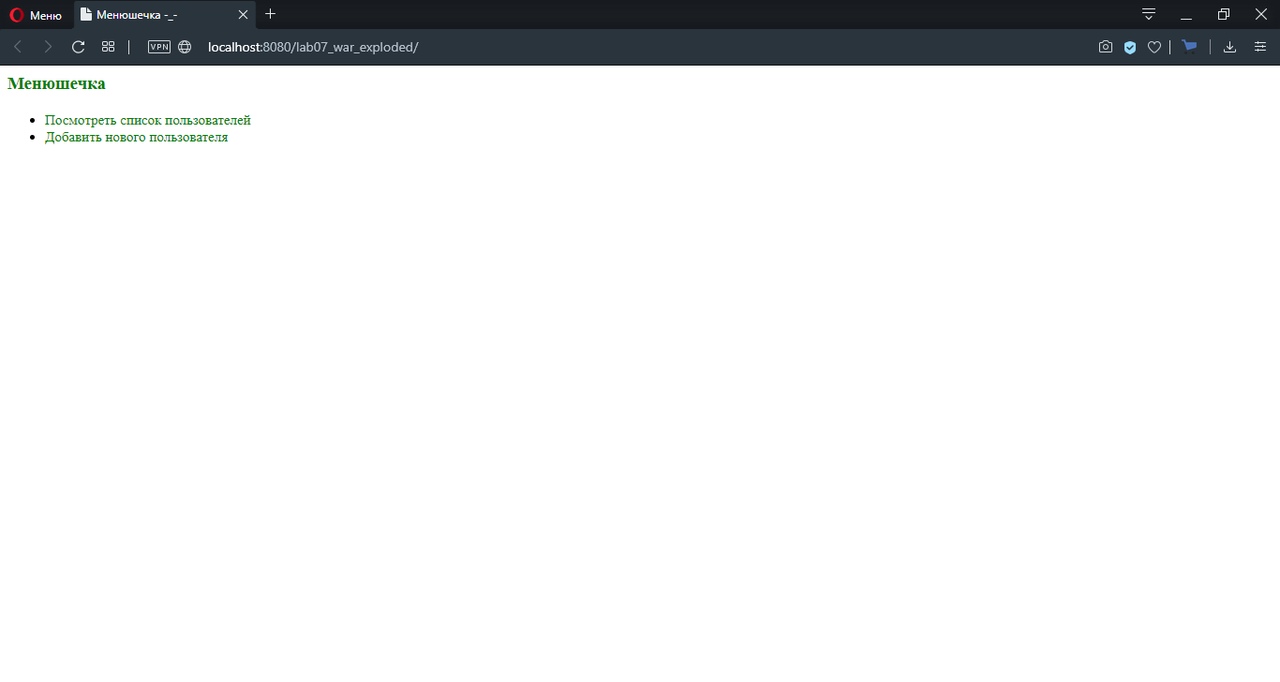


Рисунок 1. Главная страница.

<%@ **taglib uri**="**http://www.springframework.org/tags/form**" **prefix**="**form**"%>  
<%@ **taglib uri**="**http://java.sun.com/jsp/jstl/core**" **prefix**="**c**"%>  
<%@ **taglib uri**="**http://java.sun.com/jsp/jstl/functions**" **prefix**="**fn**" %>  
<%@ **page contentType**="**text/html;charset=UTF-8**" **language**="**java**" %>  
<**html**>  
<**head**>  
 <**title**>Пользователи</**title**>  
 <**style**>  
 **TABLE**{  
 **width**: 50%;  
 **border-collapse**: **collapse**;  
 **border**: 2**px solid white**;  
 **align-self**: **center**;  
 **margin**: **auto**;  
 }  
 **TD**, **TH** {  
 **padding**: 3**px**;  
 **border**: 1**px solid greenyellow**;  
 **text-align**: **center**;  
 }  
 **a**:**link**{**text-decoration**: **none**; **color**: **green**; **font-size**: 15**px**; **margin**: **auto**;}  
 **a**:**hover**{**text-decoration**: **none**; **color**: **green**; **font-size**: 20**px**;}  
 **a**:**visited**{**text-decoration**: **none**; **color**: **green**}  
 **h3**{**color**: **green**}  
 </**style**>  
</**head**>  
<**a href="index"**>Вернутся назад</**a**>  
<**center**><**h3**>Список пользователей</**h3**></**center**>  
<**c:choose**>  
<**c:when test="${**fn:length(list) **gt** 0**}"**>  
<**table border="2" width="70%" cellpadding="2"**>  
 <**tr**><**td**>Id</**td**><**td**>nickname</**td**><**td**>пароль</**td**><**td**>email</**td**><**td colspan="2"**>Действия</**td**></**tr**>  
 <**c:forEach var="user" items="${**list**}"**>  
 <**tr**>  
 <**td**>**${**user.userId**}**</**td**>  
 <**td**>**${**user.login**}**</**td**>  
 <**td**>**${**user.password**}**</**td**>  
 <**td**>**${**user.email**}**</**td**>  
 <**td**><**a href="show/${**user.userId**}"**>Изменить</**a**></**td**>  
 <**td**><**a href="delete/${**user.userId**}"**>Удалить</**a**></**td**>  
 </**tr**>  
 </**c:forEach**>  
</**table**>  
 </**c:when**>  
 <**c:otherwise**>  
 <**center**><**h3**>Список пуст, добавте пользователя</**h3**></**center**>  
 </**c:otherwise**></**c:choose**>  
</**body**>  
</**html**>

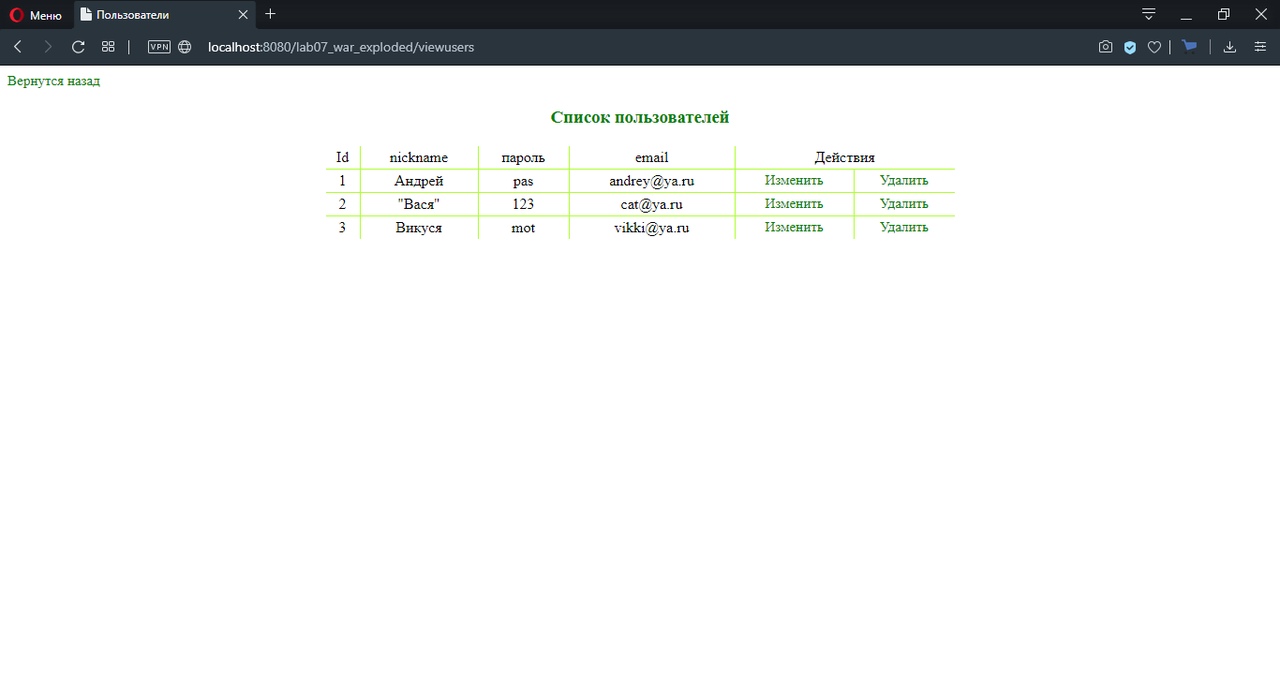


Рисунок 2. Список пользователей.

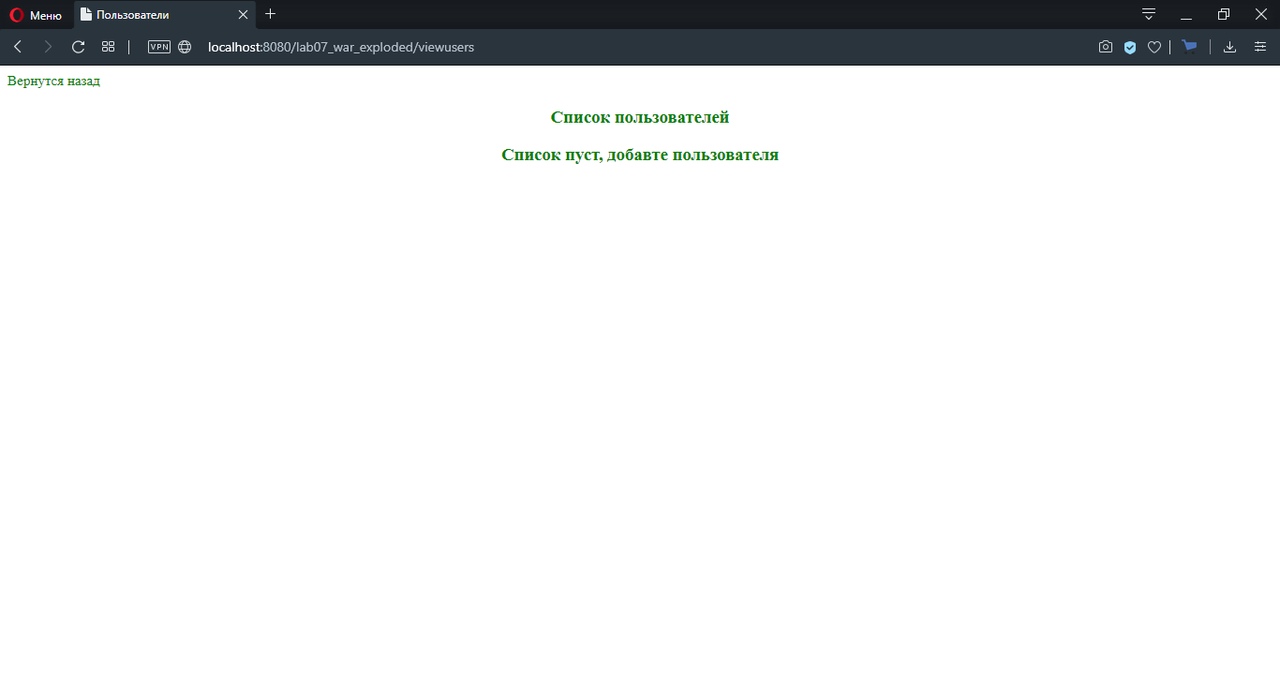


Рисунок 3.Страница со списком пользователей, если список пуст.

<%@ **page contentType**="**text/html;charset=UTF-8**" **language**="**java**" %>  
<%@ **taglib prefix**="**form**" **uri**="**http://www.springframework.org/tags/form**" %>  
<%@ **taglib uri**="**http://java.sun.com/jsp/jstl/core**" **prefix**="**c**"%>  
<**html**>  
<**head**>  
 <**title**>Добавление</**title**>  
 <**style**>  
 **TABLE**{  
 **width**: 50%;  
 **border-collapse**: **collapse**;  
 **border**: 2**px solid white**;  
 **margin**: **auto**;  
 }  
 **TD**, **TH** {  
 **padding**: 3**px**;  
 **border**: 1**px solid greenyellow**;  
 **text-align**: **center**;  
 }  
 **input**{**background**: **greenyellow**}  
 **a**:**link**{**text-decoration**: **none**; **color**: **green**; **font-size**: 15**px**; **margin**: **auto**;}  
 **a**:**hover**{**text-decoration**: **none**; **color**: **green**; **font-size**: 20**px**;}  
 **a**:**visited**{**text-decoration**: **none**; **color**: **green**}  
 **h3**{**color**: **green**}  
 </**style**>  
</**head**>  
<**body**>  
<**a href="index"**>Вернутся назад</**a**>  
<**center**><**h3**>Введите данные о пользователе</**h3**></**center**>  
<**form:form method="post" action="saveuser"**>  
 <**table** >  
 <**tr**>  
 <**td**>NickName</**td**>  
 <**td**><**form:input path="login"** /></**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>пароль</**td**>  
 <**td**><**form:input path="password"** /></**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>email</**td**>  
 <**td**><**form:input path="email"** /></**td**>  
 </**tr**>  
 <**tr**>  
 <**td**> </**td**>  
 <**td**><**input type="submit" value="Добавить"** /></**td**>  
 </**tr**>  
 </**table**>  
</**form:form**>  
</**body**>  
</**html**>

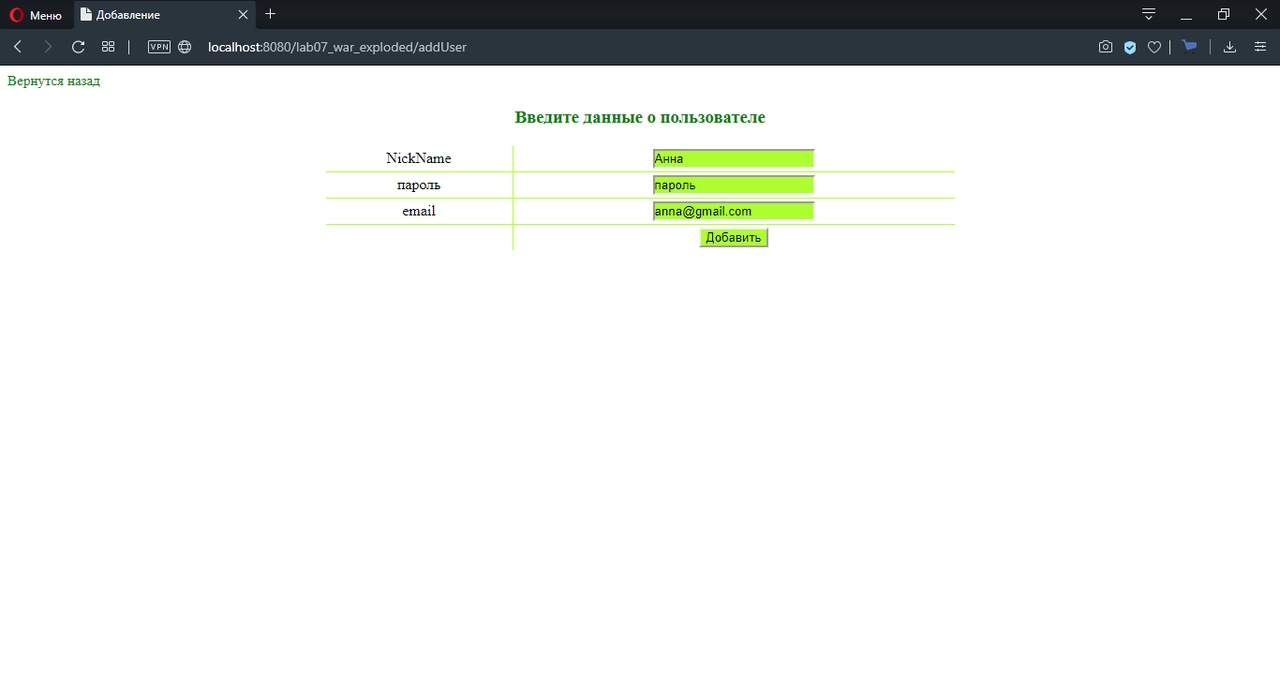


Рисунок 9. Страница добавления нового пользователя.

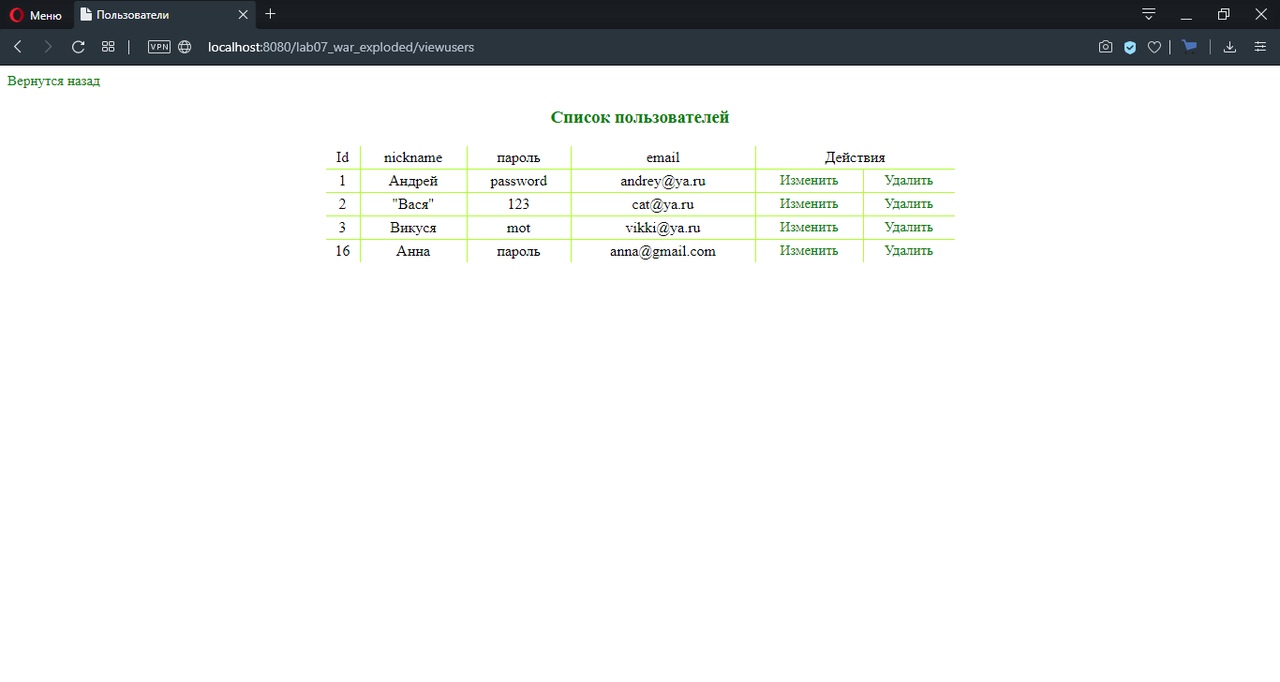


Рисунок 10. Успешное добавление нового пользователя.

<%@ **taglib prefix**="**form**" **uri**="**http://www.springframework.org/tags/form**" %>  
<%@ **taglib uri**="**http://java.sun.com/jsp/jstl/core**" **prefix**="**c**"%>  
<%@ **page contentType**="**text/html;charset=UTF-8**" **language**="**java**" %>  
<**html**>  
<**head**>  
 <**title**>Редактировать</**title**>  
 <**style**>  
 **TABLE**{  
 **width**: 50%;  
 **border-collapse**: **collapse**;  
 **border**: 2**px solid white**;  
 **margin**: **auto**;  
 }  
 **TD**, **TH** {  
 **padding**: 3**px**;  
 **border**: 1**px solid greenyellow**;  
 **text-align**: **center**;  
 }  
 **input**{**background**: **greenyellow**}  
 **a**:**link**{**text-decoration**: **none**; **color**: **green**; **font-size**: 15**px**; **margin**: **auto**;}  
 **a**:**hover**{**text-decoration**: **none**; **color**: **green**; **font-size**: 20**px**;}  
 **a**:**visited**{**text-decoration**: **none**; **color**: **green**}  
 **h3**{**color**: **green**}  
 </**style**>  
</**head**>  
<**body**>  
<**a href="index"**>Вернутся назад</**a**>  
<**center**><**h3**>Измените данные о пользователе</**h3**></**center**>  
<**form:form method="post" action="editsave"**>  
 <**table**>  
 <**form:hidden path="userId"**/>  
 <**tr**><**td**>nickname</**td**>  
 <**td**><**form:input path="login"**/></**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>пароль</**td**>  
 <**td**><**form:input path="password"** /></**td**>  
 </**tr**>  
 <**tr**>  
 <**td**>email</**td**>  
 <**td**><**form:input path="email"** /></**td**>  
 </**tr**>  
 <**tr**>  
 <**td**> </**td**>  
 <**td**><**input type="submit" value="Принять изменения"** /></**td**>  
 </**tr**>  
</**table**>  
</**form:form**>  
</**body**>  
</**html**>

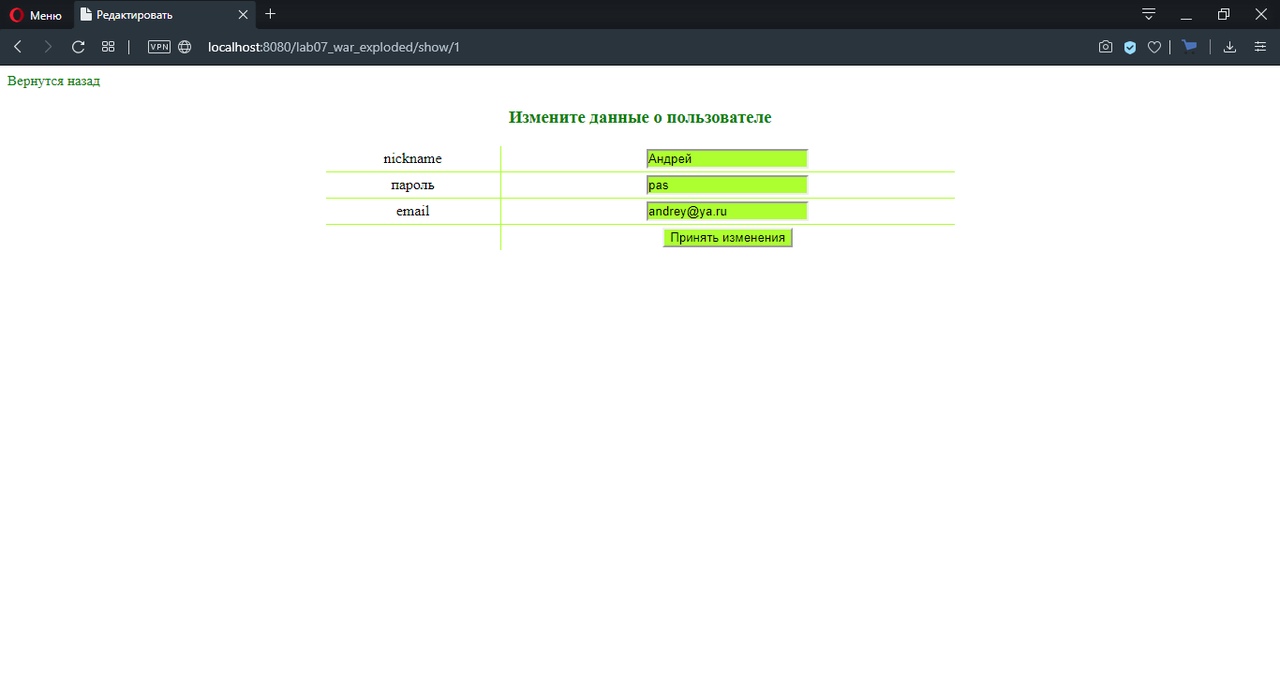


Рисунок 4. Пример редактирования пользователя.

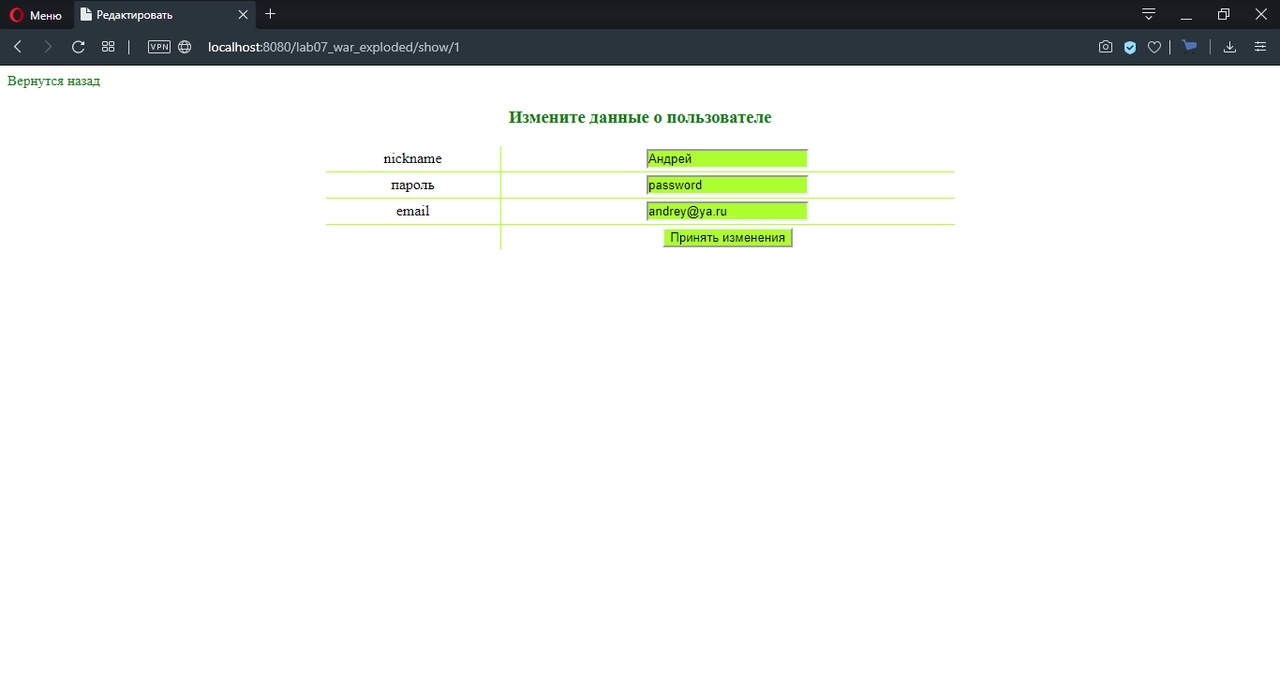


Рисунок 5. Изменение данных.

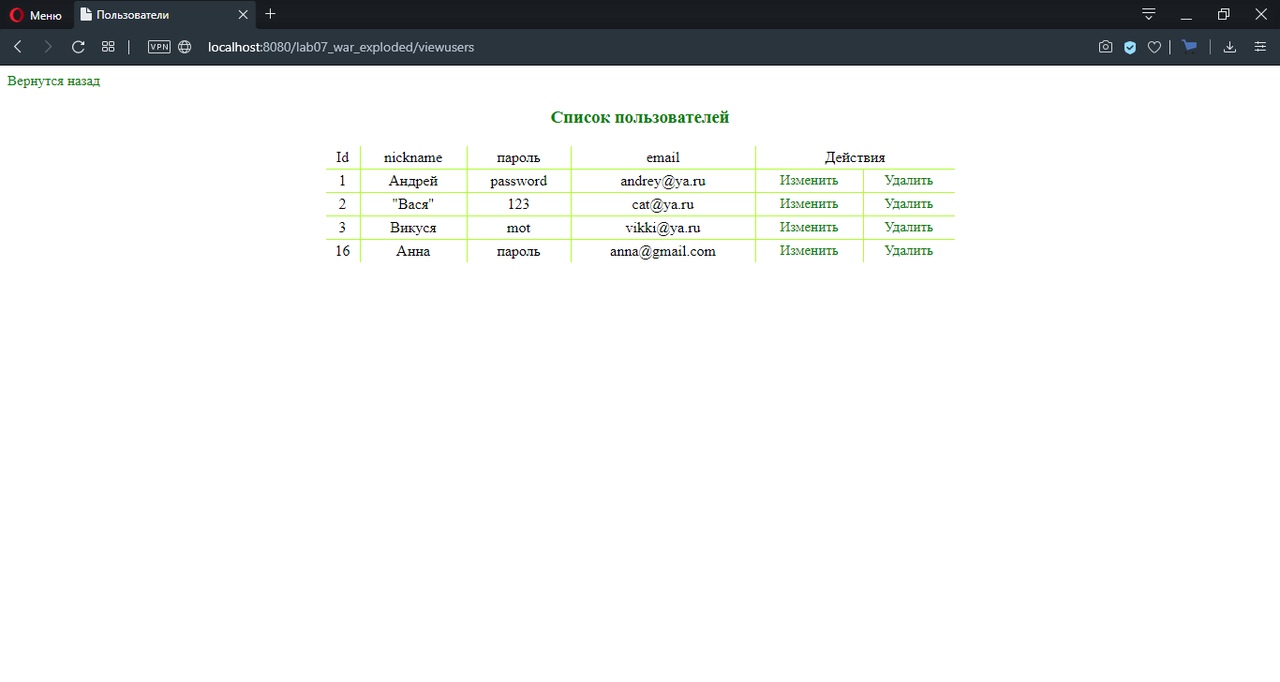


Рисунок 6. Выполнение редактирования.

<%@ **page contentType**="**text/html;charset=UTF-8**" **language**="**java**" %>  
<**html**>  
<**head**>  
 <**title**>Error</**title**>  
 <**style**>  
 **a**:**link**{**text-decoration**: **none**; **color**: **green**; **font-size**: 15**px**;}  
 **a**:**hover**{**text-decoration**: **none**; **color**: **green**; **font-size**: 20**px**;}  
 **a**:**visited**{**text-decoration**: **none**; **color**: **green**}  
 </**style**>  
</**head**>  
<**body**>  
<**center**><**a href="index"**>Ошибка при работе с данными, нажмите чтобы вернутся на страницу меню</**a**></**center**>  
</**body**>  
</**html**>

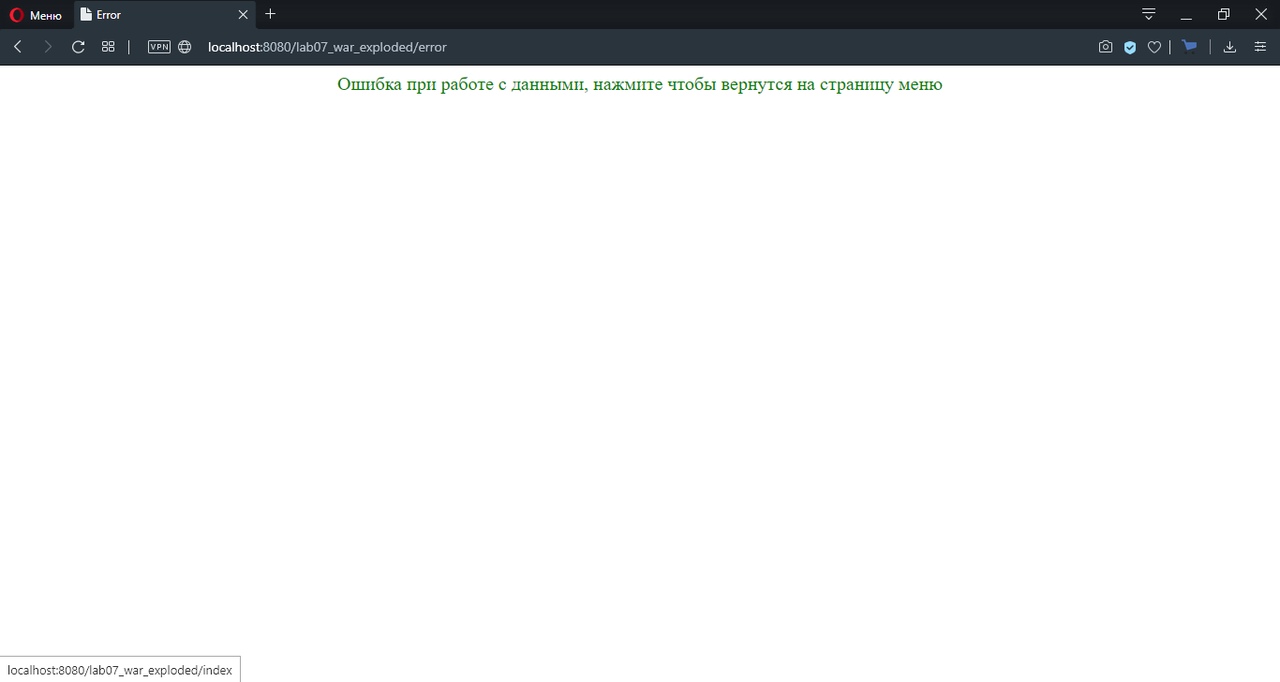


Рисунок 7. Если при работе с данными возникают ошибки отображается данная страница.

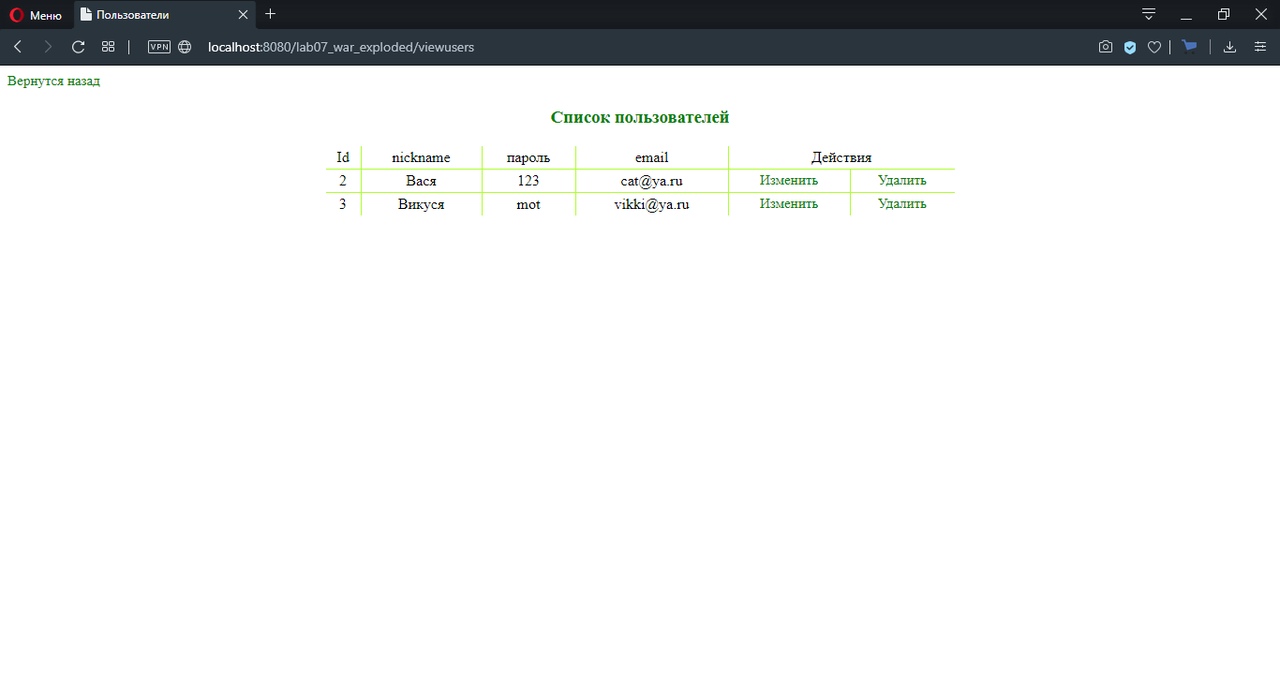


Рисунок 8. Удаление пользователей.

|  |
| --- |
|  |

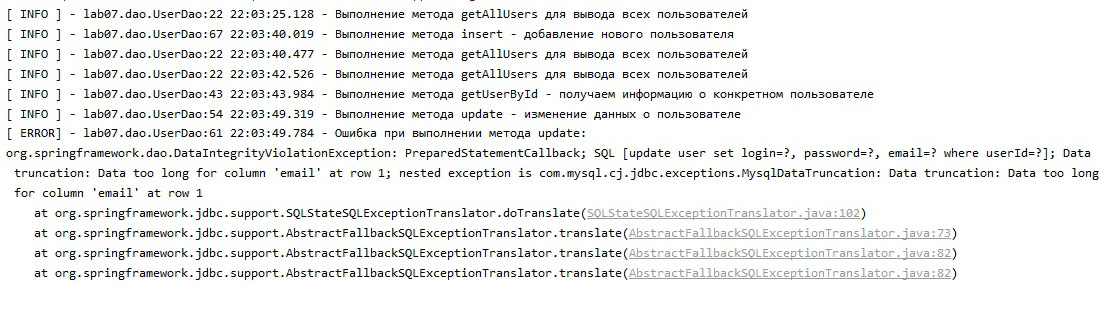


Рисунок 11. Пример логов.

ВЫВОД

В ходе лабораторной работы были изучены принципы построения веб-приложений с использованием шаблона MVC и на базе принципов REST, получены практические навыки реализации MVC-приложения на платформе Spring MVC.