ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ "САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ ИМ. ПРОФ. М. А. БОНЧ-БРУЕВИЧА"

Факультет инфокоммуникационных сетей и систем

Кафедра программной инженерии и вычислительной техники

ОТЧЕТ ПО

КУРСОВОЙ РАБОТЕ

по дисциплине

«Разработка Java-приложений управления телекоммуникациями»

Вариант №7

Выполнил:

студент 3 курса

группы ИКПИ-11

Дунаев Владислав Евгеньевич

Приняла:  
доцент кафедры ПИВТ

Белая Татьяна Иоанновна

\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2023 год

**РЕФЕРАТ**

Отчет 89с., 27 рис., 4 табл., 3 источн., 3 прил.

JAVA, СЕРВЛЕТ, ВЕБ-ПРИЛОЖЕНИЕ, ВЕБ-СТРАНИЦА, JAKARTA SERVLET API, MAVEN, , JDBC, ТЕСТИРОВАНИЕ, БЕЗОПАСНОСТЬ ПРЕДПРИЯТИ, HTTP

Объектом исследования являются веб-приложения, разрабатываемые с помощью языка программирования Java и программного интерфейса Jakarta Servlet Api.

Цель работы – разработать информационную систему для управления клиентской базой данных и автоматизации продаж.

В ходе выполнение курсовой работы были получены следующие результаты:

1. Определен современный программный инструментарий для разработки веб-приложений на языке программирования Java.
2. Исследованы различные методы предоставления информации на веб-страницах, не препятствующие их доступности.
3. Изучены различные способы работы с базами данных с помощью пользовательских программных интерфейсов, предоставляемых Java и другими ее компонентами
4. Разработан полноценный веб сервис для управления и автоматизации продаж

**ТЕРМИНЫ, ОПРЕДЕЛЕНИЯ И СОКРАЩЕНИЯ**

В настоящем отчете о курсовой работе применяют следующие термины, определения и сокращения:

*Веб-приложение* – программное приложение, которое пользователи запускают через веб-браузер на своем устройстве и взаимодействуют с ним посредством интернета.

*Веб-страница* – документ или информационный ресурс Всемирной паутины, доступ к которому осуществляется с помощью веб-браузера.

*База данных* – совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, независимая от прикладных программ.

*Веб-интерфейс* – веб-страница или совокупность веб-страниц, предоставляющая пользовательский интерфейс для взаимодействия с сервисом или устройством посредством веб-браузера.

*URL* (Uniform Resource Locator c англ. – «единообразный указатель местонахождения ресурса») - адрес ресурса в сети Интернет.

*ER-диаграмма* (от англ. Entity-Relationship diagram, модель «сущность-связь» – модель данных, позволяющая описывать отношения и взаимосвязи между объектами в данной предметной области.

*SQL* (от англ. Structured Query Language - «язык структурированных запросов») – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных

*API* (от англ. application programming interface – «интерфейс программирования приложения») - программный интерфейс, то есть описание способов взаимодействия одной компьютерной программы с другими.

*JDBC* (англ. Java DataBase Connectivity – «соединение с базами данных на Java») – платформенно независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД.

Содержание

[1 Постановка задачи 5](#__RefHeading___Toc15608_3816603122)

[1.1 Требования к функционалу программы 5](#__RefHeading___Toc15610_3816603122)

[1.2 Требования к веб-интерфейсу программы 6](#__RefHeading___Toc15612_3816603122)

[1.3 Среда разработки и используемые технологии 7](#__RefHeading___Toc15614_3816603122)

[2 Структура программы 8](#__RefHeading___Toc15616_3816603122)

[3 Отношения между таблицами в БД 9](#__RefHeading___Toc15618_3816603122)

[4 Разработка собственных классов 10](#__RefHeading___Toc15620_3816603122)

[4.1 Классы-сущности 10](#__RefHeading___Toc15622_3816603122)

[4.2 Классы-сервлеты 10](#__RefHeading___Toc15624_3816603122)

[4.3 Классы-сервисы 11](#__RefHeading___Toc15626_3816603122)

[4.4 Классы-мапперы 13](#__RefHeading___Toc15628_3816603122)

[4.5 Классы-репозитории 13](#__RefHeading___Toc15630_3816603122)

[5 Веб-страницы 15](#__RefHeading___Toc15632_3816603122)

[6 Блок-схема 16](#__RefHeading___Toc15634_3816603122)

[7 Скриншоты веб-страниц 18](#__RefHeading___Toc15636_3816603122)

[8 Тестирование 27](#__RefHeading___Toc15638_3816603122)

[ЗАКЛЮЧЕНИЕ 32](#__RefHeading___Toc15640_3816603122)

[СПИСОК ЛИТЕРАТУРЫ 33](#__RefHeading___Toc15642_3816603122)

[ПРИЛОЖЕНИЕ 34](#__RefHeading___Toc15644_3816603122)

[ПРИЛОЖЕНИЕ A. Файлы .java 35](#__RefHeading___Toc15646_3816603122)

[ПРИЛОЖЕНИЕ Б. Файлы .jsp и .css 70](#__RefHeading___Toc15648_3816603122)

[ПРИЛОЖЕНИЕ В. Файлы конфигурации 88](#__RefHeading___Toc15650_3816603122)

**1 Постановка задачи**

*Вариант 7 (исходные данные): создание информационной системы для управления клиентской базой данных и автоматизации процесса продаж*

Постановка задачи: разработать веб-приложение с использованием Jakarta EE. Приложение должно располагать возможностью создания, изменения, обновления и удаления данных. В процессе достижения цели необходимо решить ряд следующих задач:

* ознакомиться с современными технологиями и по возможности использовать их в своей разработке;
* определить программный инструментарий, применяемый для разработки веб-приложения;
* выявить и учесть методы и способы представления на веб-страницах различных видов информации, не препятствующие их доступности;
* ознакомиться с основными правилами и рекомендациями по разработке и созданию веб-приложений и принципиально следовать им в своей практике;
* определиться со структурой веб-страниц;
  1. **Требования к функционалу программы**

Разрабатываемая программа должна иметь широкий и удобный функционал по работе с товарами и пользователями. В качестве ключевых функций можно выделить следующие:

1. Создание новых товаров
2. Редактирование существующий наименований
3. Просмотр всех доступных товаров
4. Добавление в базу данных новых пользователей
5. Редактирование профиля пользователя
6. Простмотр всех существующих пользователей
7. Автоматический учет совершенных покупок
8. От лица пользователя просмотр как всех сделанных покупок, так и одной конкретной
9. Авторизация и аутентификация на сайте
10. Разделение на 3 роли: обычный пользователь, продавец и администратор
11. Возможнось добавлять в корзину товары по штрих-коду
12. Привязка покупки и пользователя по номеру телефона

## **Требования к веб-интерфейсу программы**

Все веб-страницы должны быть оформлены в одном стиле: использование не более двух различных шрифтов, использование не более трех цветов, допустимо использовать различные оттенки этих цветов. Интерфейс веб-страницы следовать стандартам человеко-машинного взаимодействия.

Помимо вышеперечисленных требований сайт также должен быть адаптивным: расположение элементов должно оставаться удобным и понятным.

Интерфейс должен быть удобным и простом, чтобы можно было легко обучить нового пользователя пользоваться программой.

* 1. **Среда разработки и используемые технологии**

Программа разработана на языке Java (OpenJDK 20) в среде IntelliJ IDEA Ultimate 2023.2.5. в операционной системе Windows 10. Также использовались следующие технологии и библиотеки:

* Jakarta Servlet API - фреймворк, предоставляющий средства для создания веб-сервлетов и JSP-страниц;
* JDBC – программный интерфейс java для взаимодействия с базами данных;
* MariaDB Driver - драйвер JDBC, предоставляющий средства подключения к базе данных MariaDB;
* Apache Tomcat - контейнер сервлетов, предназначенный для развертывания и выполнения веб-приложения;
* Maven – инструмент для сборки проектов, подключения зависимостей и плагинов;

1. **Структура программы**

Структура программы разбита на несколько слоёв. Это упрощает процесс разработки и поддержки приложения. Для веб-приложений обычно выделяют следующие слои:

* слой представления;
* слой контроллера;
* сервисный слой;
* слой доступа к базе данных.

В качестве представления в разрабатываемом приложении выступают jsp-страницы. Они динамически генерируются на сервере и отображают пользователю все необходимые данные.

Веб-сервлеты представляют собой слой контроллера. Они обрабатывают пользовательские запросы и генерируют ответы.

Сервисный слой содержит всю логику работы приложения. В нем производятся различные вычисления, данные запросов проходят валидацию и обработку, отлавливаются и обрабатываются ошибки.

Слой доступа к данным представляет собой репозитории. В них осуществляется подключение к базе данных, выполняются CRUD-операции с сущностями.



*Рисунок 1. Структура приложения*

1. **Отношения между таблицами в БД**

Используемая база данных должна быть нормализована.

В базах данных, нормализация — это процесс организации данных с целью устранения избыточности и обеспечения целостности информации. Нормальные формы определяют правила для структурирования таблиц в базах данных, чтобы минимизировать дублирование данных и обеспечить эффективность и надежность операций. В подавляющем большинстве современных приложений базы данных находятся в третьей нормальной форме.

Основным принципом третьей нормальной формы является то, что неключевые столбцы должны зависеть только от первичного ключа, а не от других неключевых столбцов. То есть, если у нас есть неключевой столбец, который функционально зависит от другого неключевого столбца, то эти зависимости должны быть устранены. Преимуществом третьей нормальной формы является более логичное и четкое представление данных, однако, увеличивается сложность запросов.

Разработанная база данных удовлетворяет требованиям нормальной формы. ER-диаграмма базы данных, используемой в приложении, представлена на рис. 2.

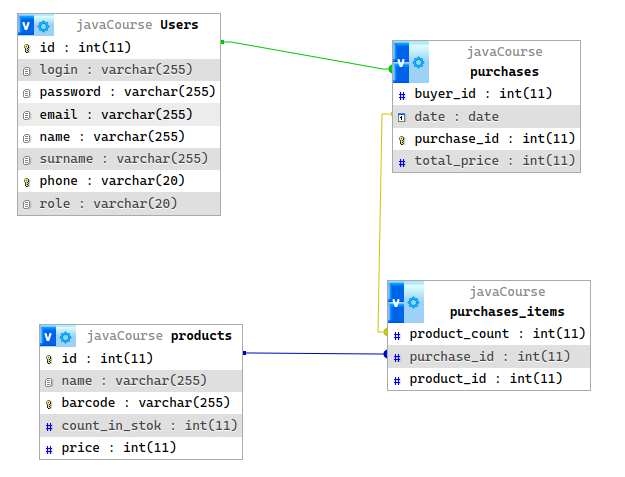


Рисунок 2 – ER-диаграмма

1. **Разработка собственных классов**

В проекте разработанные классы разбиты по соответствующим пакетам:

* servlet – сервлеты;
* model – сущности, описывающие устройства и сотрудников;
* service – сервисы;
* mapper – мапперы;
* exception – собственные исключения;
* util – вспомогательные классы;
  1. **Классы-сущности**

*Таблица 1 – Классы, описывающие сущности*

|  |  |
| --- | --- |
| **Класс** | **Поля класса** |
| User.java | Integer id – идентификатор пользователя  String login – логин пользователя  String password – хэшированный пароль пользователя  String email – почта пользователя  String name – имя  String surname – фамилия  String phone – номер телефона  String role – уровень доступа |
| Product.java | Integer id – идентификатор товара  Integer price – цена товара  Integer count – кол-во купленного товара  Integer count\_in\_stock – кол-во товара в наличии  String barcode – штрих-код товара  String name – наименование товара |
| Order.java | Integer purchase\_id – идентификатор покупки  Integer buyer\_id – идентификатор покупателя  Integer total\_price – итоговая сумма покупки  Integer count – кол-во товара в наличии  Date date – дата совершения покупки |

* 1. **Классы-сервлеты**

В приложении разработано 20 сервлетов обрабатывающий HTTP-запросы, связанные с операциями над пользователями и товарами. Каждый из сервлетов наследуется от HttpServlet и переопределяет два метода: doGet() и doPost(). Эти методы, в свою очередь, обрабатывают конкретные типы запросов, переадресуют пользователя на jsp-страницы или вызывают методы сервисных классов в зависимости от переданных в запросе параметров.

Аннотация @WebServlet указывает контейнеру сервлетов, что этот класс является сервлетом и должен обрабатывать запросы, соответствующие указанным URL-шаблонам. Фрагмент исходного кода одного из сервлетов:

|  |
| --- |
| **CheckBarcode.java** |
| @WebServlet(urlPatterns = "/checkBarcode")  public class CheckBarcode extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // получаем штрих код  String barcode = request.getParameter("barcode");  // создаем/получаем корзину  ListProducts listProducts = (ListProducts) request.getSession().getAttribute("list");  if(listProducts == null) listProducts = new ListProducts();  // добавляем товар по штрих коду + возврааем корзину или выводим ошибку  if (listProducts.addByBarcode(barcode)) {  request.getSession().removeAttribute("errorCheck");  request.getSession().removeAttribute("errorRead");  request.getSession().setAttribute("list", listProducts);  } else {  request.getSession().setAttribute("errorRead", "Товар не найден, либо штрих код считан неверно");  }  // редирект на себя, для очередного товара  response.sendRedirect("/checkBarcode");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  request.getRequestDispatcher("salesman/").forward(request, response);  }  } |

* 1. **Классы-сервисы**

Для операций над сущностями предусмотрены сервисные классы ListProducts, ListUsers и Orders. Сервисные классы имеют следующие функции:

* маппинг параметров запроса в объекты java;
* выбрасывание и обработка исключений;
* проверка на существование того или иного объекта перед произведением операций над ним;
* другая логика, например, сбор несколько позиций в контейнер для дальнейшей отрисовки.

Фрагмент исходного кода одного из сервисных классов:

|  |
| --- |
| **Orders.java** |
| public class Orders {  public Orders(Integer id) {  this.user\_id = id;  }   public ArrayList<Order> getOrders() {  ArrayList<Order> orders = new ArrayList<>();  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from purchases where buyer\_id = ?";  ResultSet result = null;  // подключаем базу данных  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setInt(1, user\_id);  result = preparedStatement.executeQuery();  Order temp = null;  while(result.next()) {  temp = new Order(result.getInt("purchase\_id"),  user\_id,  result.getDate("date"),  result.getInt("total\_price"));  orders.add(temp);  }  connection.close();  } catch (Exception e) {  e.printStackTrace();  return null;  }  return orders;  } |

* 1. **Классы-мапперы**

Мапперы занимаются преобразованием различных данных в объекты java. Все методы разработанных мапперов статические, поэтому в добавлении экземпляра класса в контекст приложения нет необходимости. В разрабатываемом приложении классы-мапперы выполняют следующие функции:

* Преобразование результирующих наборов баз данных (ResultSet’ов) в объекты java или списки этих объектов;
* Преобразование параметров HTTP-запроса в объекты java.

Фрагмент исходного кода одного из мапперов:

|  |
| --- |
| **Login.java** |
| ResultSet resultSet = preparedStatement.executeQuery();  if (resultSet.next()) {  session.setAttribute("user\_id", resultSet.getInt("id"));  session.setAttribute("login", resultSet.getString("login"));  session.setAttribute("password", resultSet.getString("password"));  session.setAttribute("role", resultSet.getString("role"));  session.setAttribute("name", resultSet.getString("name"));  session.setAttribute("surname", resultSet.getString("surname"));  session.setAttribute("email", resultSet.getString("email"));  session.setAttribute("phone", resultSet.getString("phone"));  } |

* 1. **Классы-репозитории**

Репозитории отвечают за взаимодействие с базой данных. Для каждой сущности в приложении предусмотрен свой репозиторий: ListProducts и ListUsers. Репозитории в разрабатываемом приложении выполняют следующие функции:

* Установление соединения с базой данных. Репозитории управляют установлением и разрывом соединения с базой данных. Для этого используются такие параметры БД, как URL, имя пользователя, пароль и так далее;
* Выполнение запросов. Репозитории отвечают за создание и выполнение SQL-запросов к базе данных;
* Обработка результатов запросов. После выполнения запроса репозиторий получает результирующие наборы и преобразует их в формат, понятный для приложения.

|  |
| --- |
| **ListProducts.java** |
| public class ListProducts {  private static String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static String USERNAME = "user";  private static String PASSWORD = "qwer";  private ArrayList<Product> list;  public ListProducts() {  this.list = new ArrayList<Product>();  }  public boolean addByBarcode(String barcode) {  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from products where barcode = ?";  ResultSet result = null;  // подключаем базу данных  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, barcode);  result = preparedStatement.executeQuery();  Product temp;  // если шрих код уже существует в корзине, то кол-во +1 и возвращаем true  if (addOneToProduct(barcode)) { return true; }  // если шрих код сущесвтует в бд, то добавляем в корзину  if (result.next()) {  temp = new Product( result.getInt("id"),  result.getString("name"),  barcode,  result.getInt("count\_in\_stok"),  result.getInt("price"),  1);  } else {  connection.close();  return false;  }  insertProduct(temp);  connection.close();  } catch (Exception e) {  e.printStackTrace();  return false;  }  return true;  } |

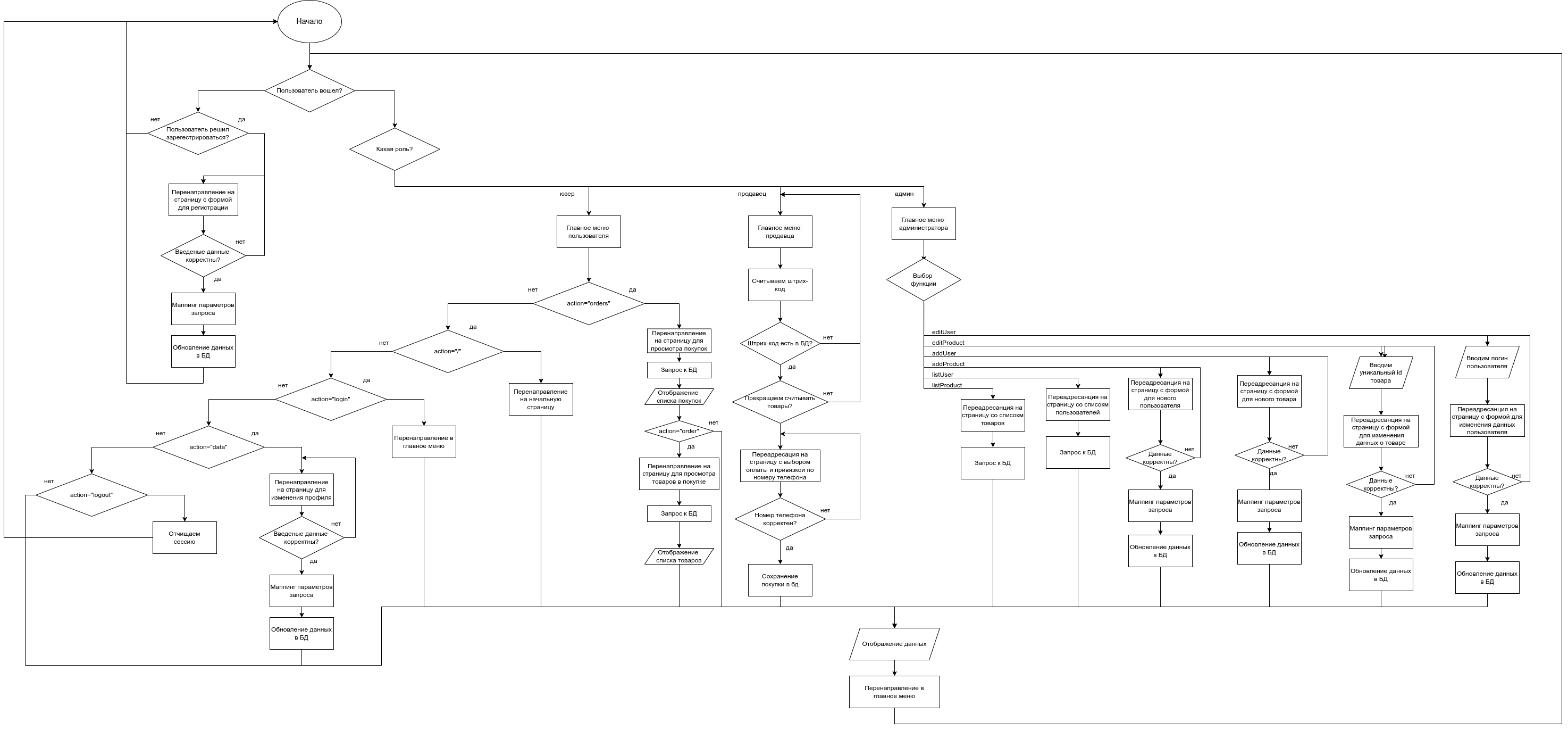
1. **Веб-страницы**

В приложении используются динамически генерируемые JSP страницы. Все страницы имеют схожее оформление: используется одинаковый текст, схожие цвета, одно и то же навигационное меню. Все используемые стили доступны в приложении Б.

*Таблица 2 – Используемые веб-страницы*

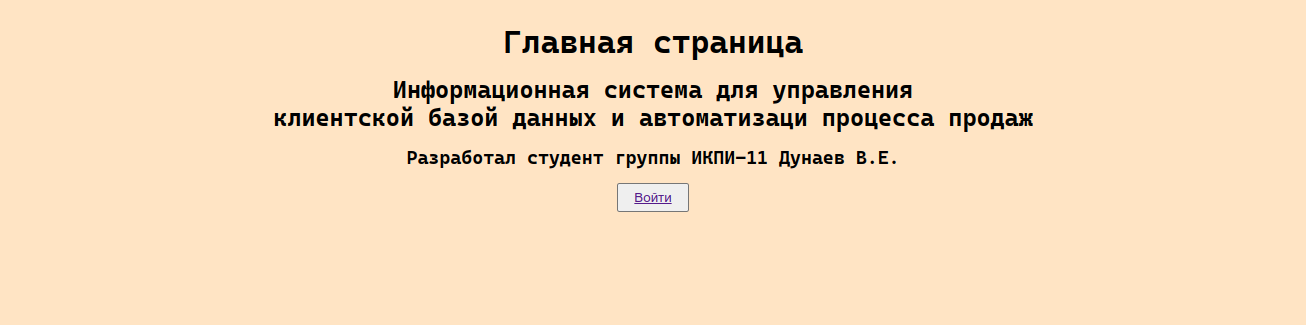
|  |  |
| --- | --- |
| **Веб-страница** | **Функционал** |
| Header.jsp | Добавляет заголовок с кнопками управления |
| Index.jsp | Начальная страница с заданием |
| Login.jsp | Страница входа в аккаунт |
| Register.jsp | Страница регистрации нового пользователя |
| user\_profile.jsp | Редактирует данные аккаунта |
| user/index.jsp | Стартовая страцица обычного юзера |
| user/orders.jsp | Выводит список совершенных покупок |
| user/order.jsp | Отображает информацию по покупке |
| salesman/index.jsp | Страница для продавца со сканером штрих-кодов для оформления покупок |
| salesman/handleBuyer.jsp | Страница с подтверждением корзины, выбором оплаты и привязкой пользователя по номеру телефона |
| salesman/payment.jsp | Иммитация оплаты |
| admin/index.jsp | Стартовая страцица администратора, отображает функции |
| admin/addProduct.jsp | Добавляет товар в бд |
| admin/addUser.jsp | Добавляет пользователя в бд |
| admin/editProduct.jsp | Редактирует товар по уникальному id |
| admin/editUser.jsp | Редактирует информацию о пользователе |
| admin/listProduct.jsp | Выводит список всех товаров |
| admin/listUser.jsp | Выводит список всех зарегестрированных пользователей |

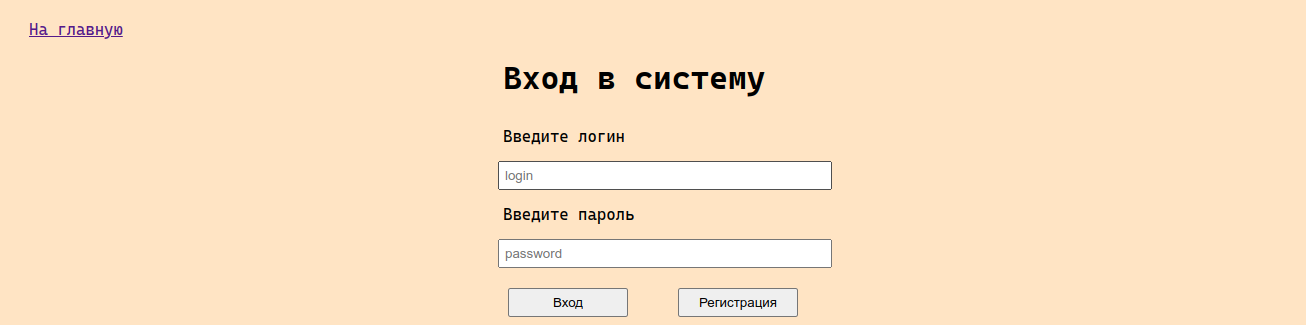
1. **Блок-схема**



*Рисунок 3. Блок-схема приложения*

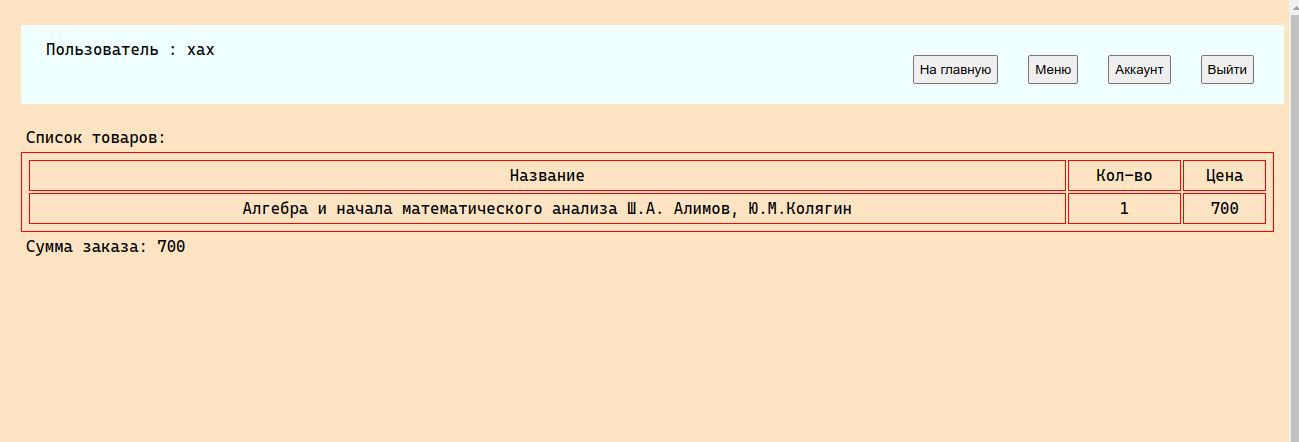
# **Скриншоты веб-страниц**

*Рисунок 4. Главное страница программы*

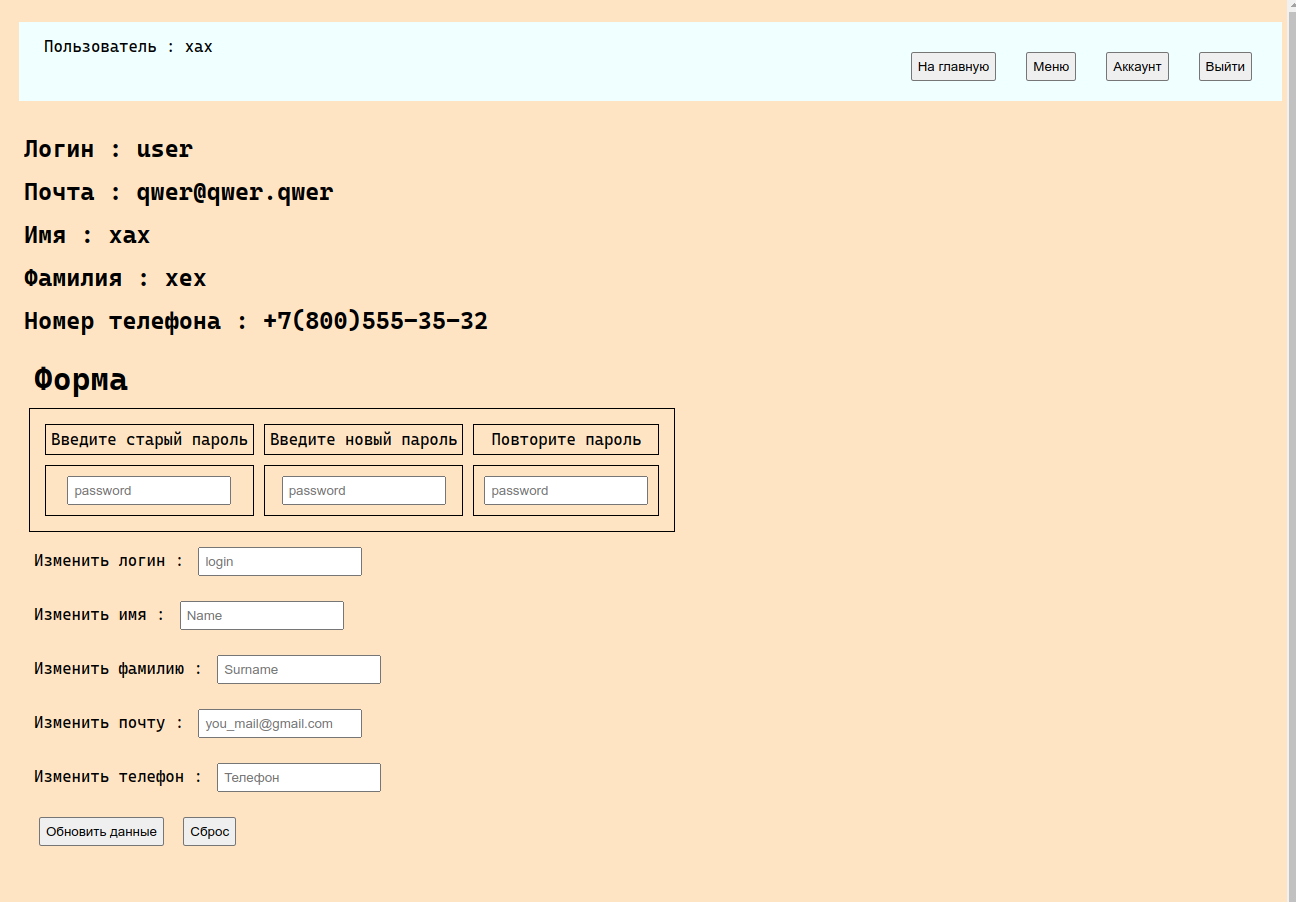
*Рисунок 5. Вход в аккаунт*

*Рисунок 6. Меню обычного пользователя*

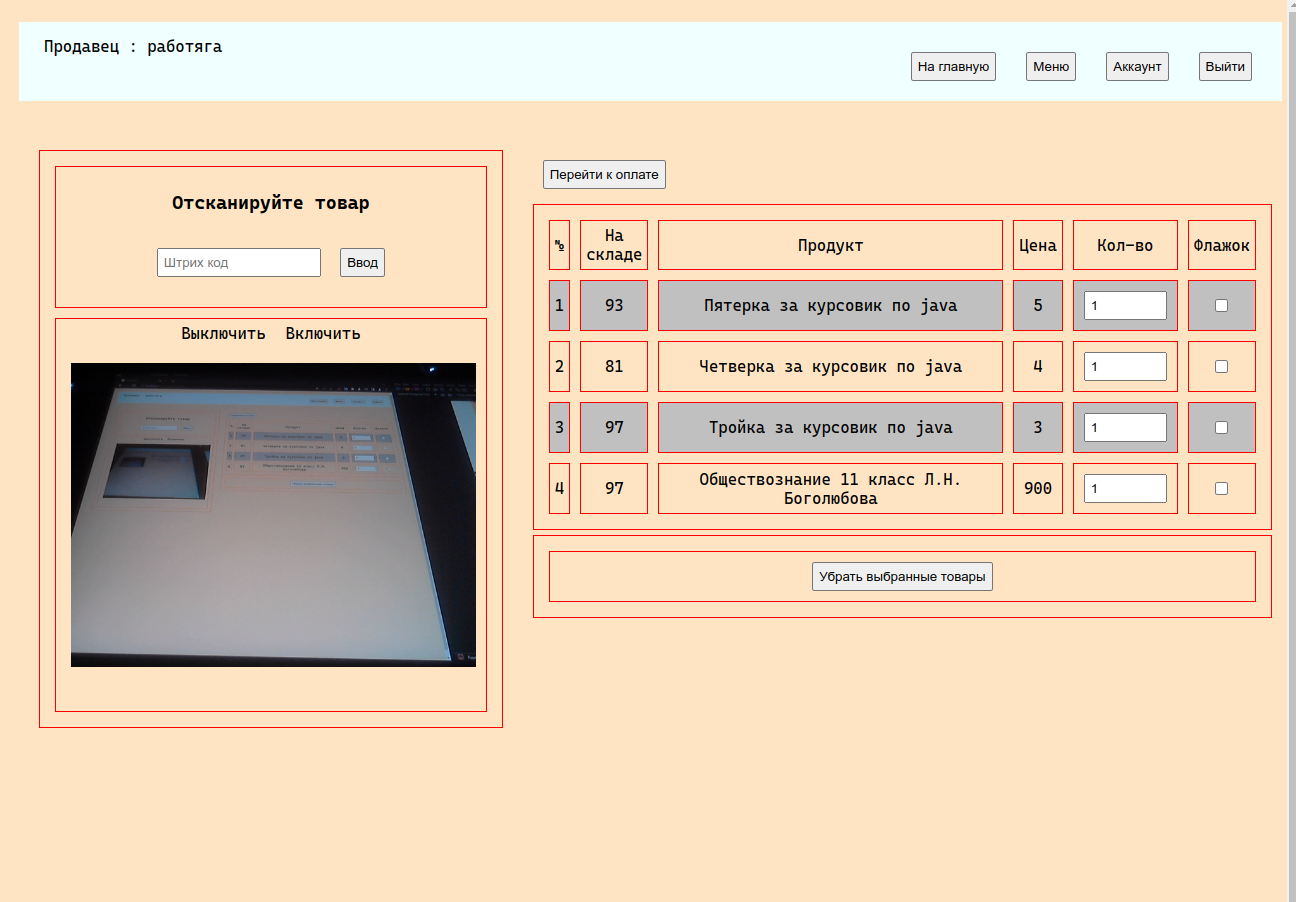
*Рисунок 7. Список покупок*

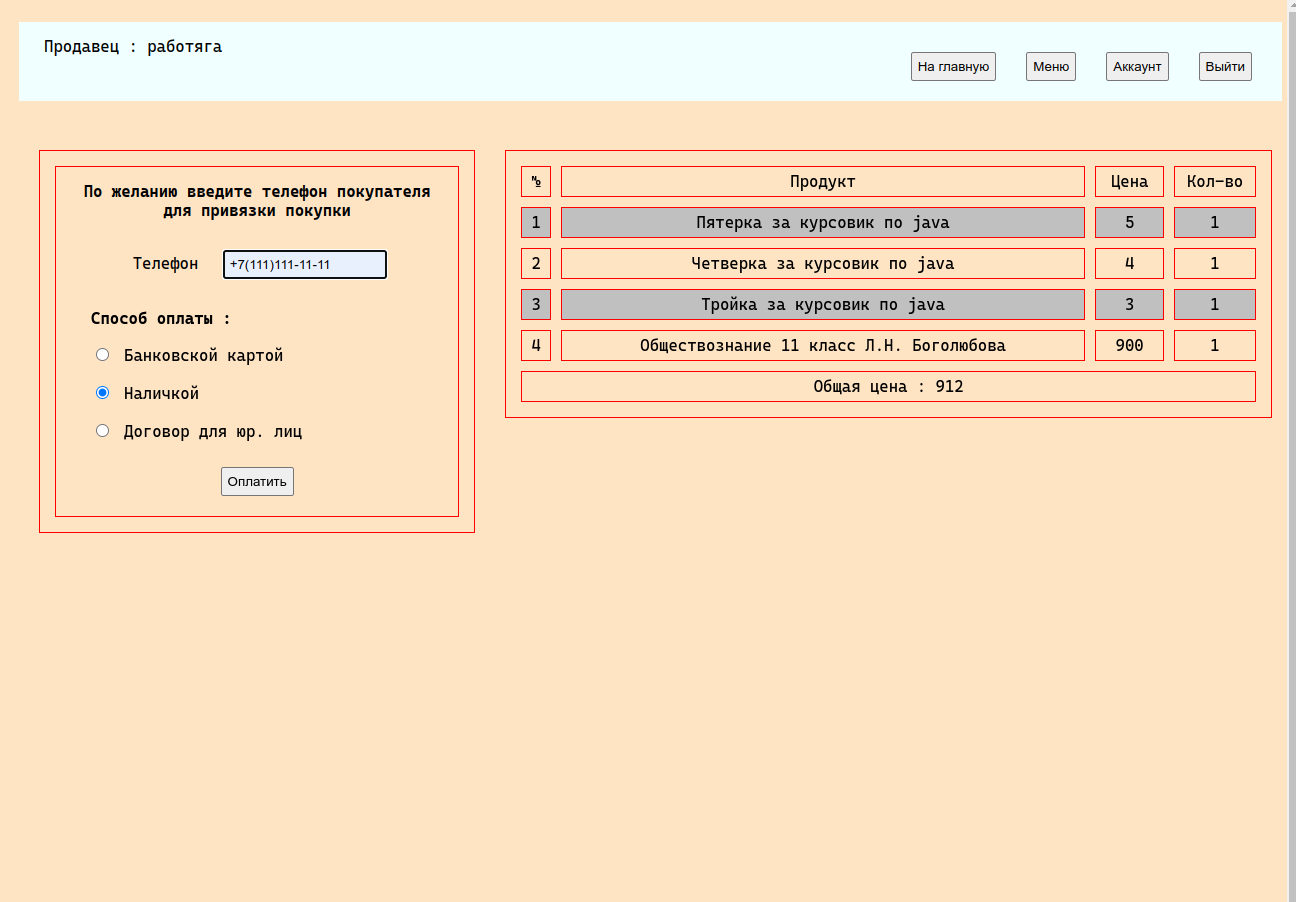
**

*Рисунок 8. Детали покупки*

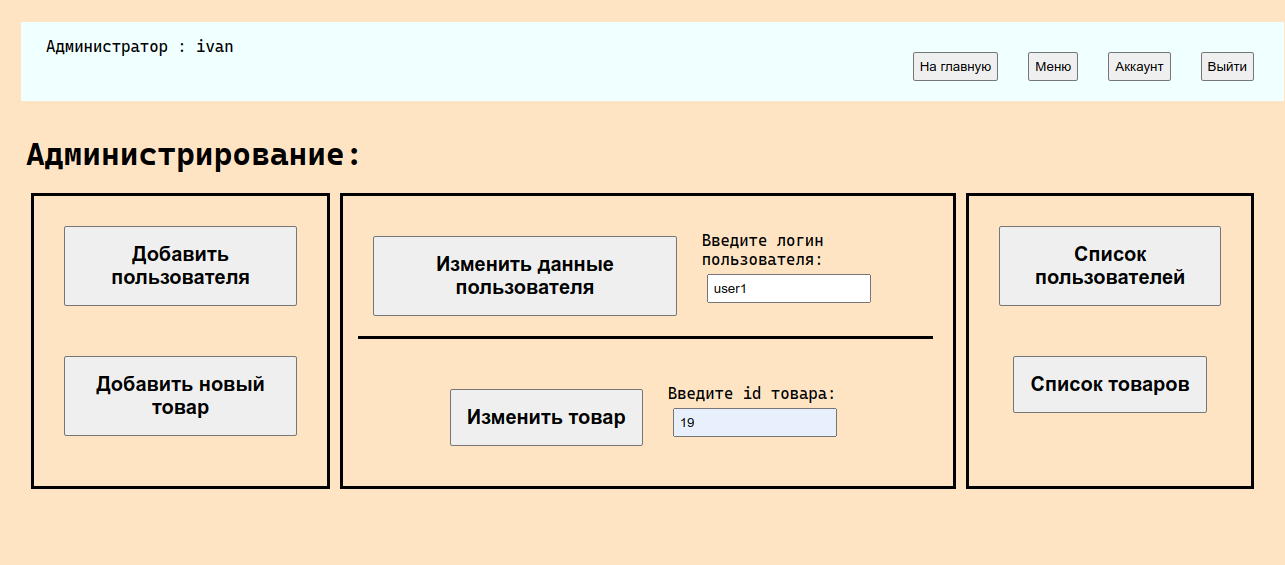
**

*Рисунок 9. Изменение информации в аккаунте*

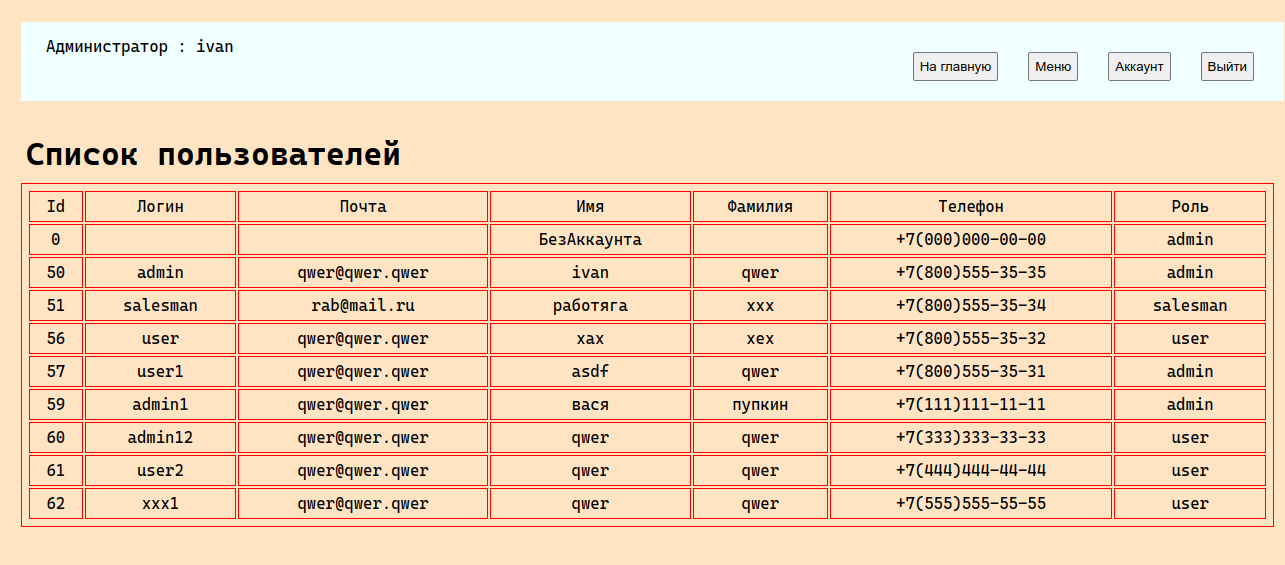
*Рисунок 10. Основная страница продаца*

**

*Рисунок 11. Выбор оплаты и привязка пользователя*

****

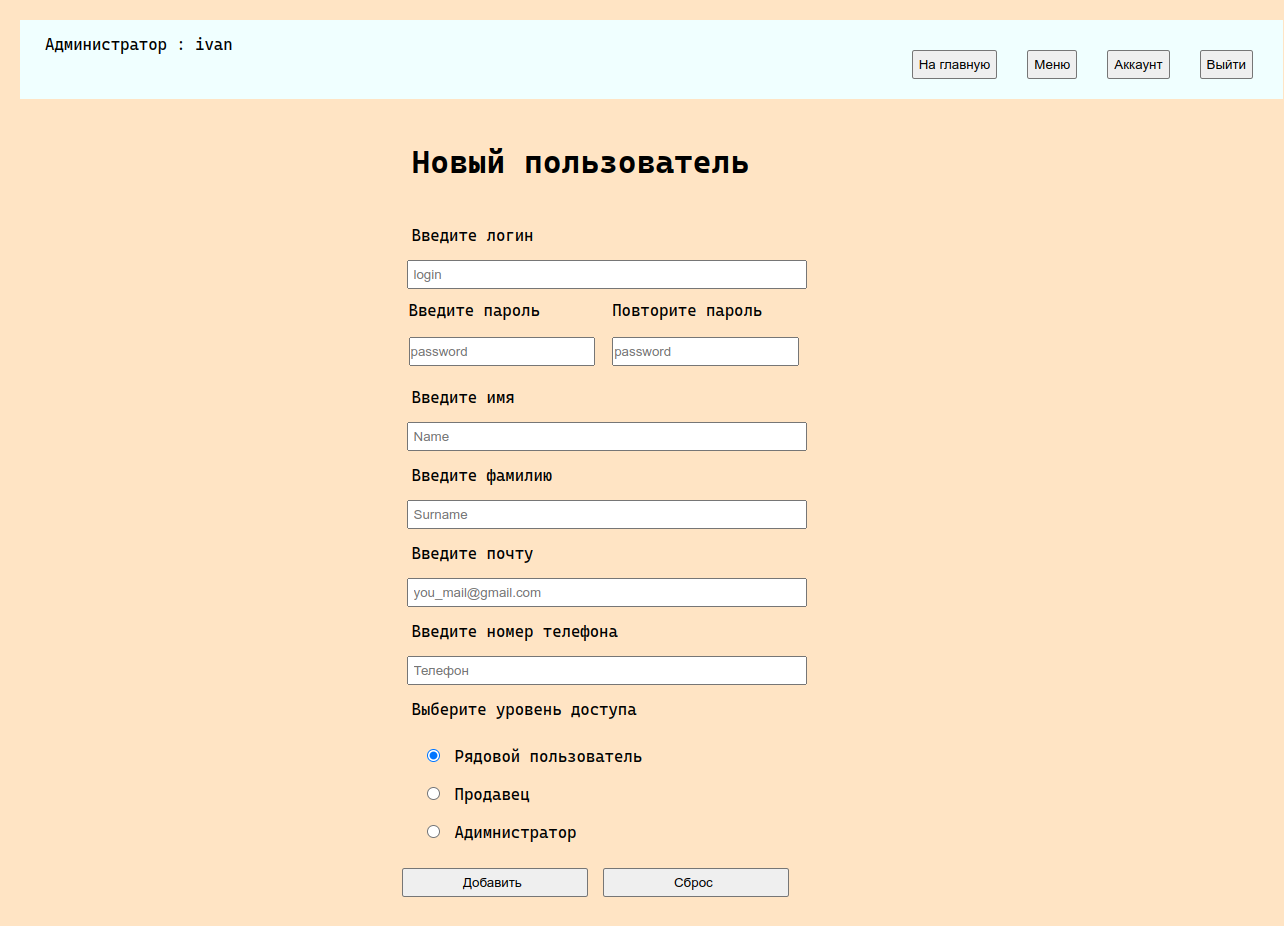
*Рисунок 12. Функции администратора*

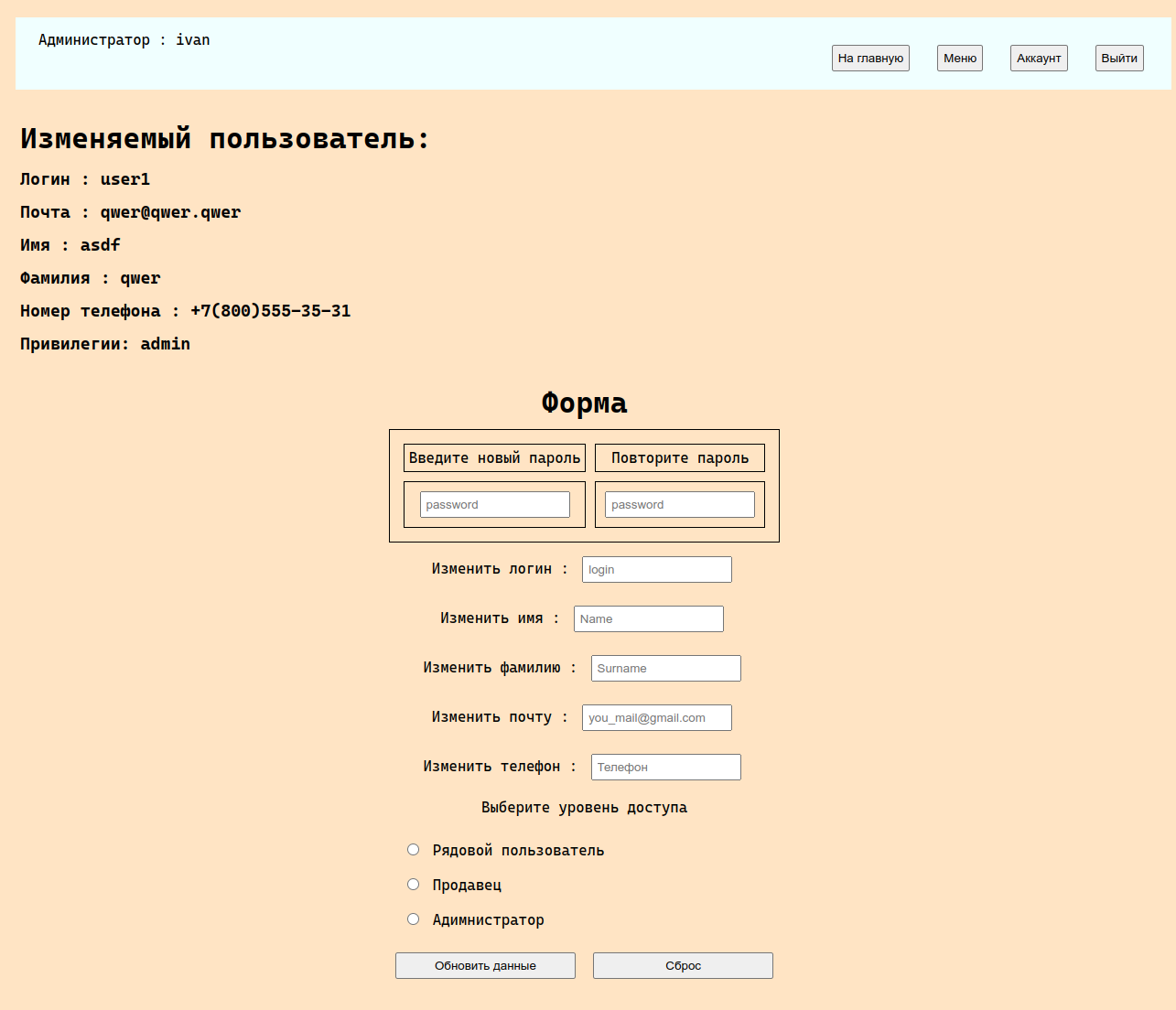
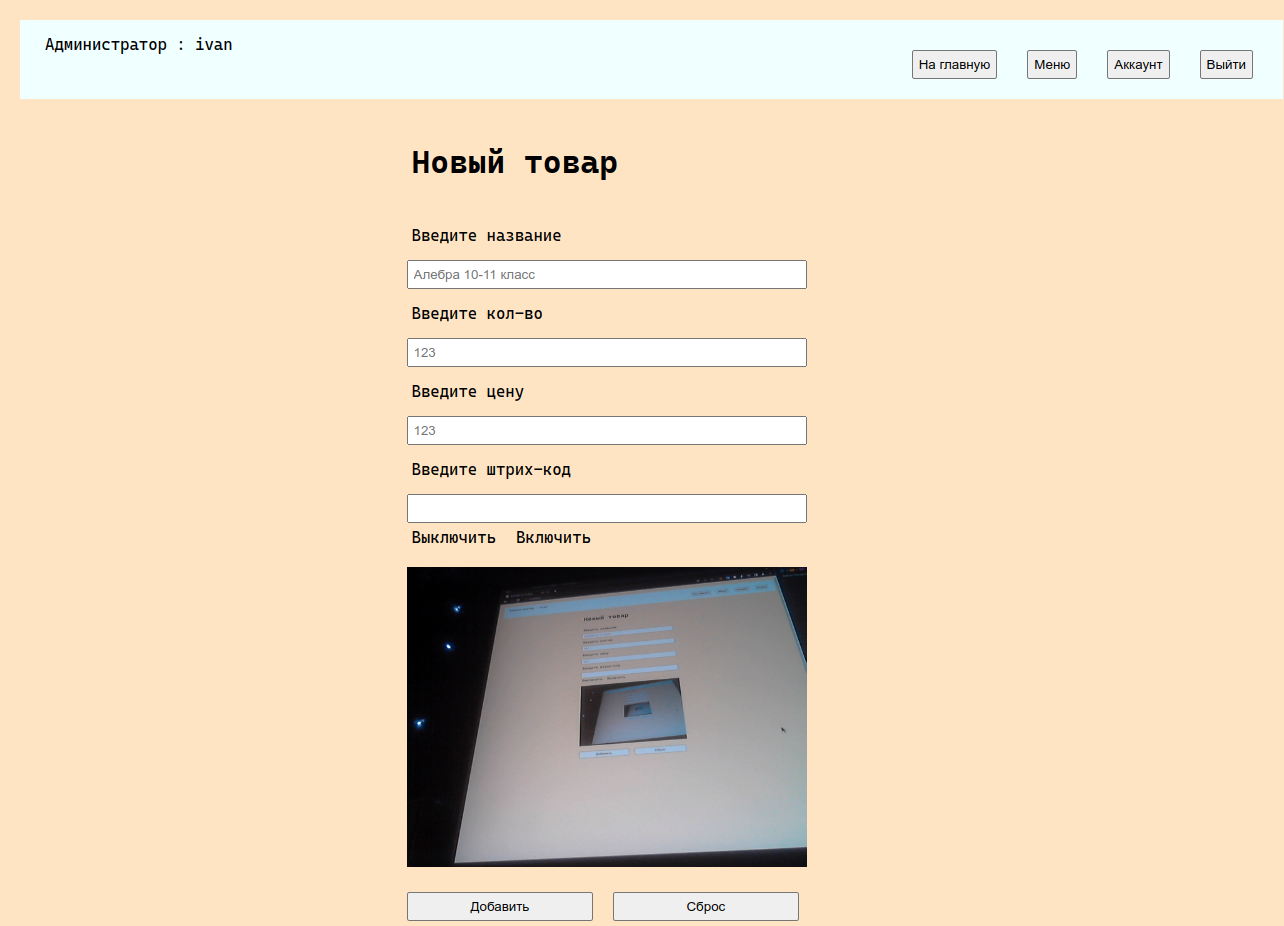
****

*Рисунок 13. Список пользователей*

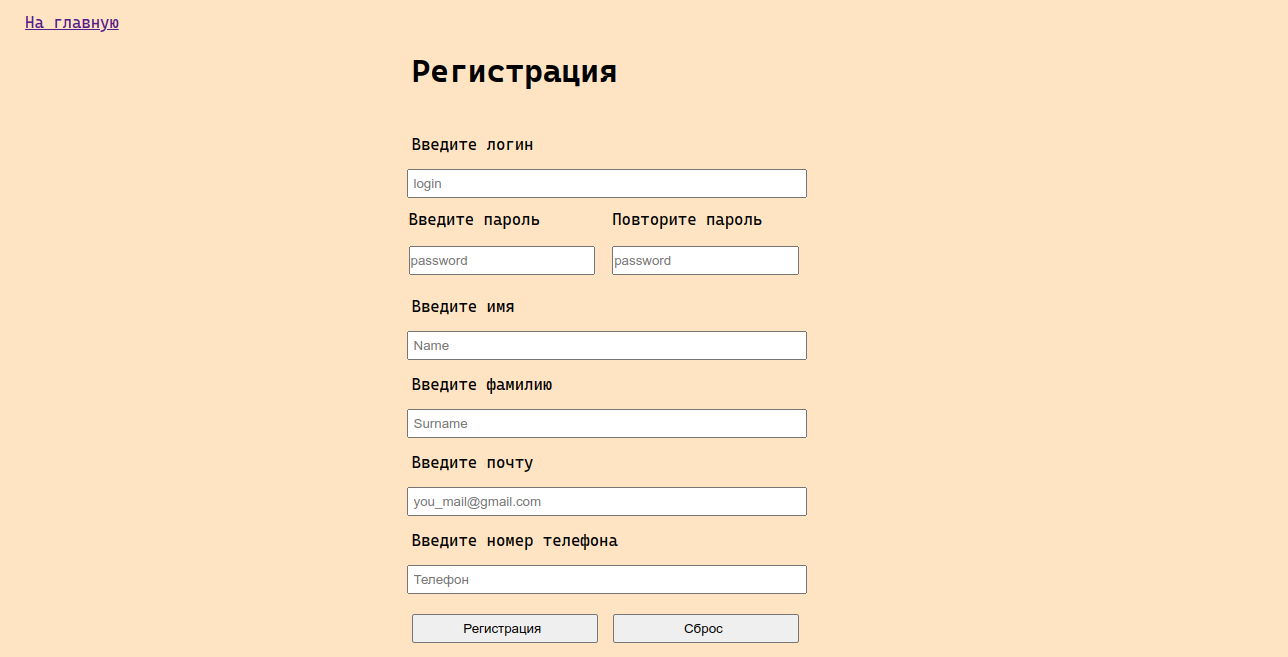
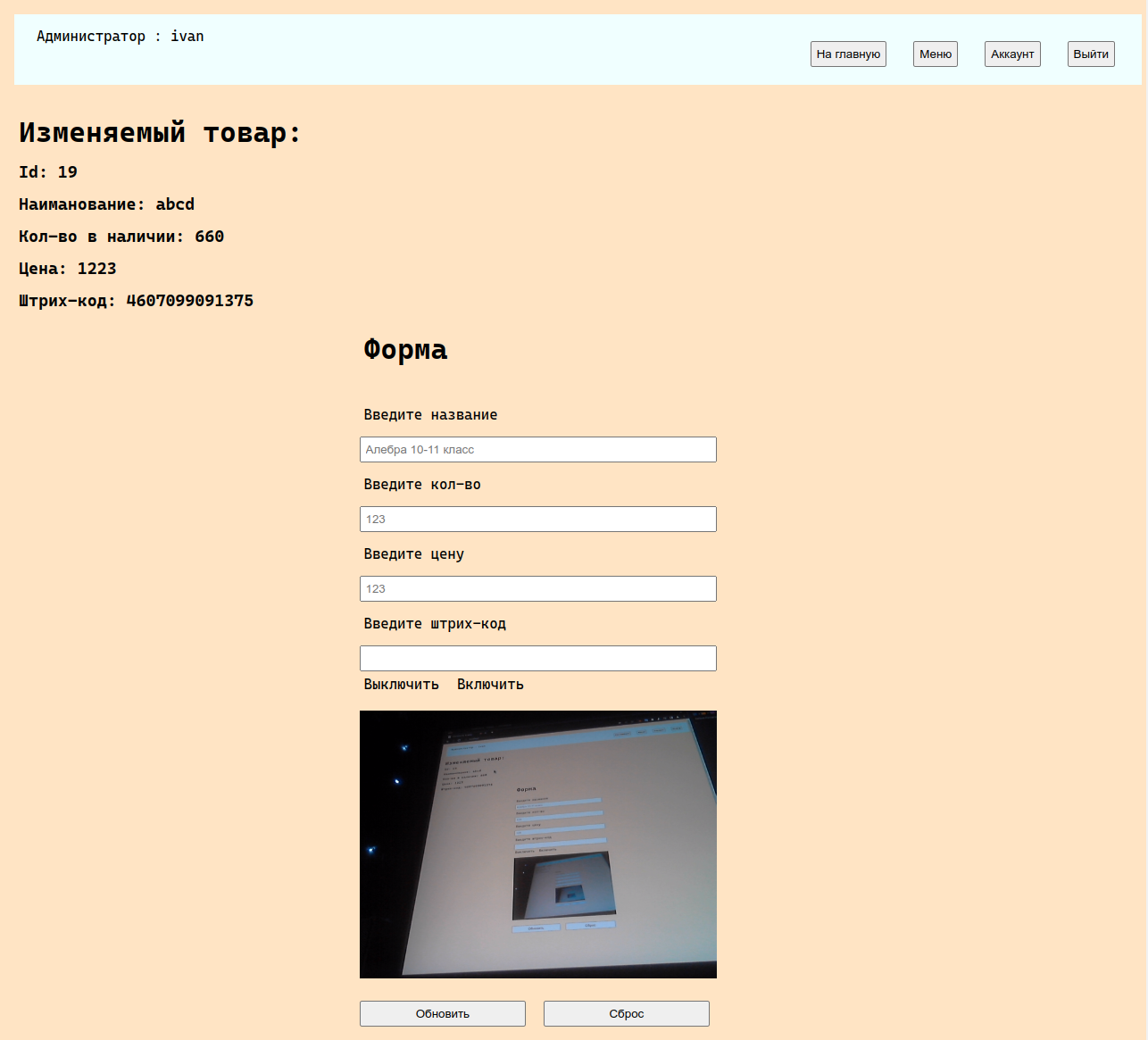
****

*Рисунок 14. Список товаров*

*Рисунок 15. Добавление нового пользователя*

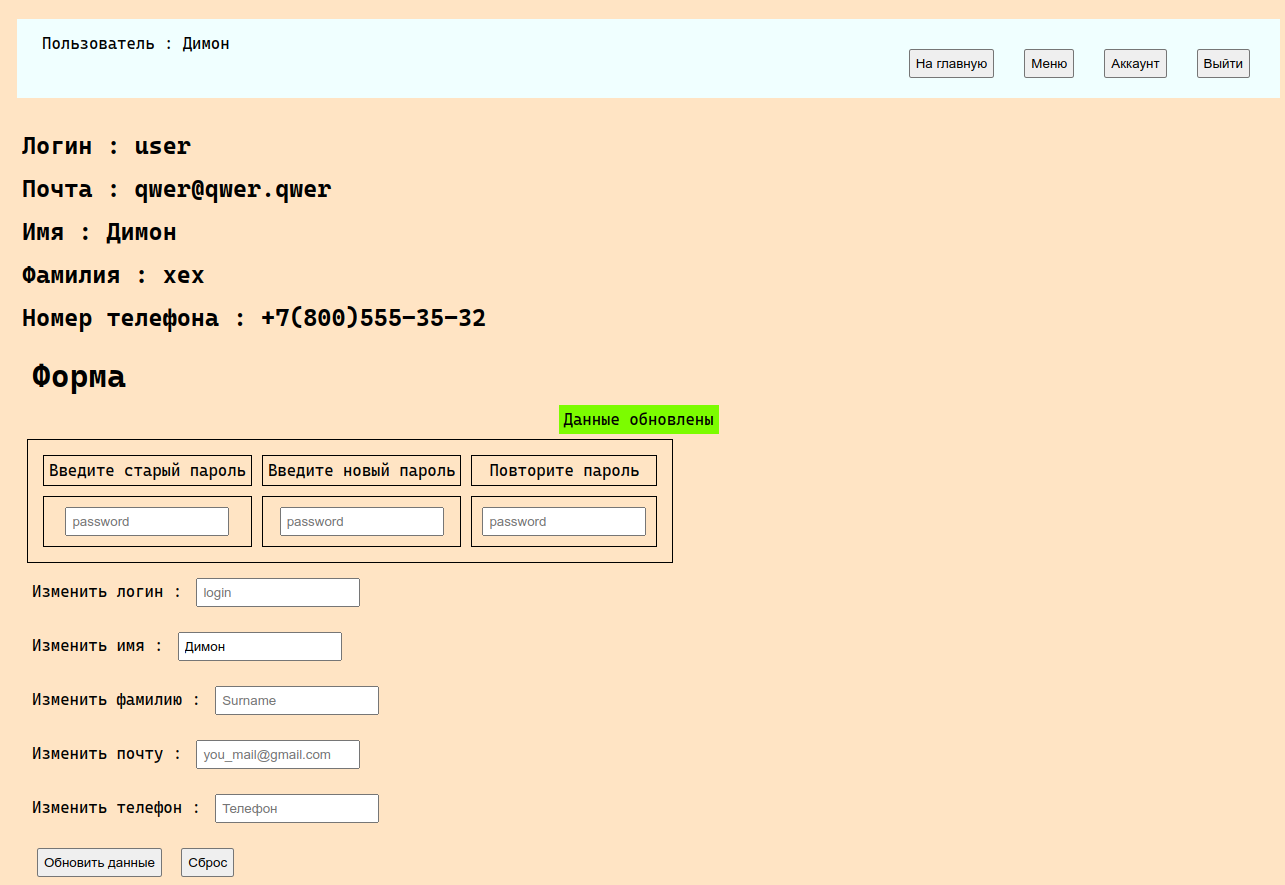
*Рисунок 16. Добавление нового товара*

*Рисунок 17. Изменение данных пользователя user1*

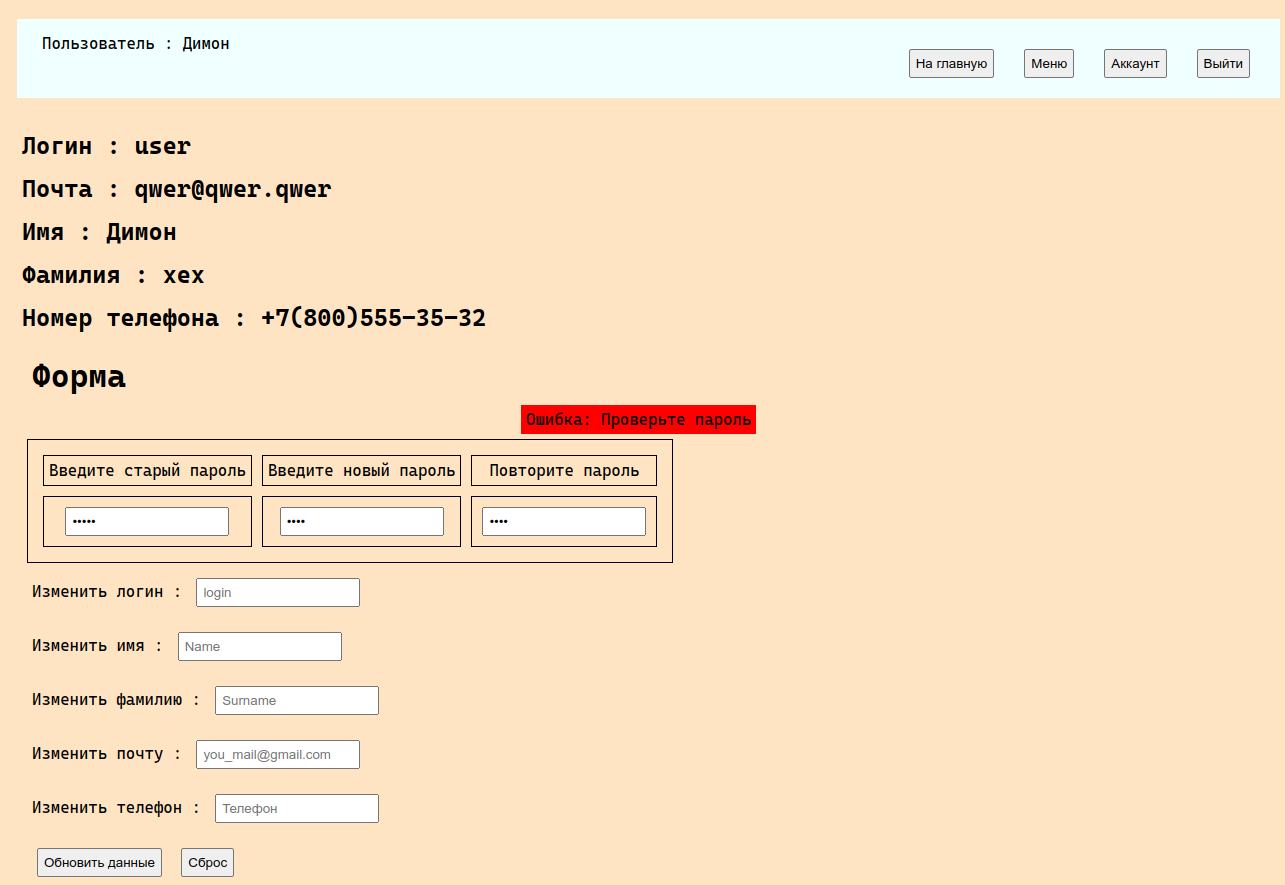
*Рисунок 18. Изменение данных товара под 19 номером*

*Рисунок 19. Форма регистрации*

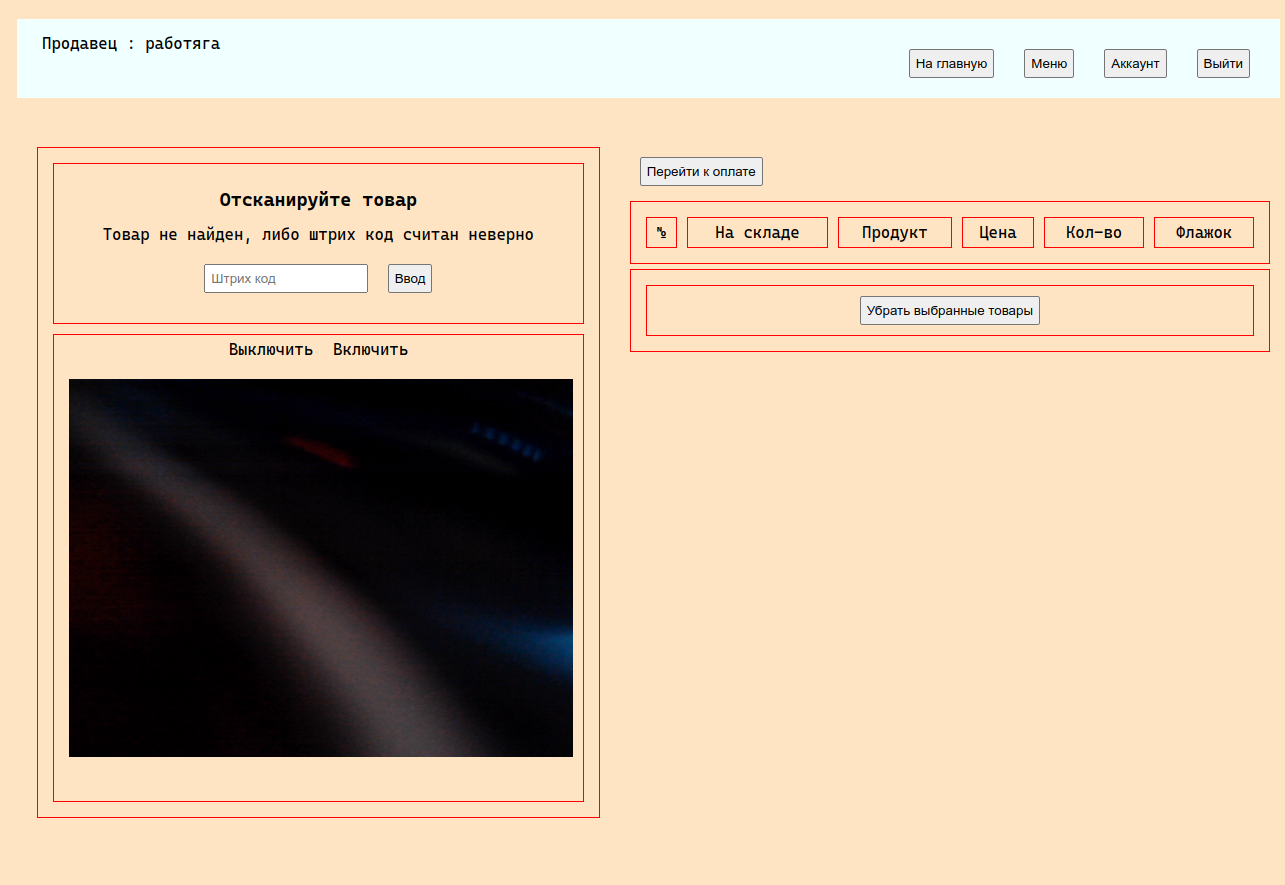
# **Тестирование**



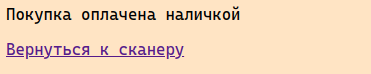
*Рисунок 20. Ответ программы на обновление имени*



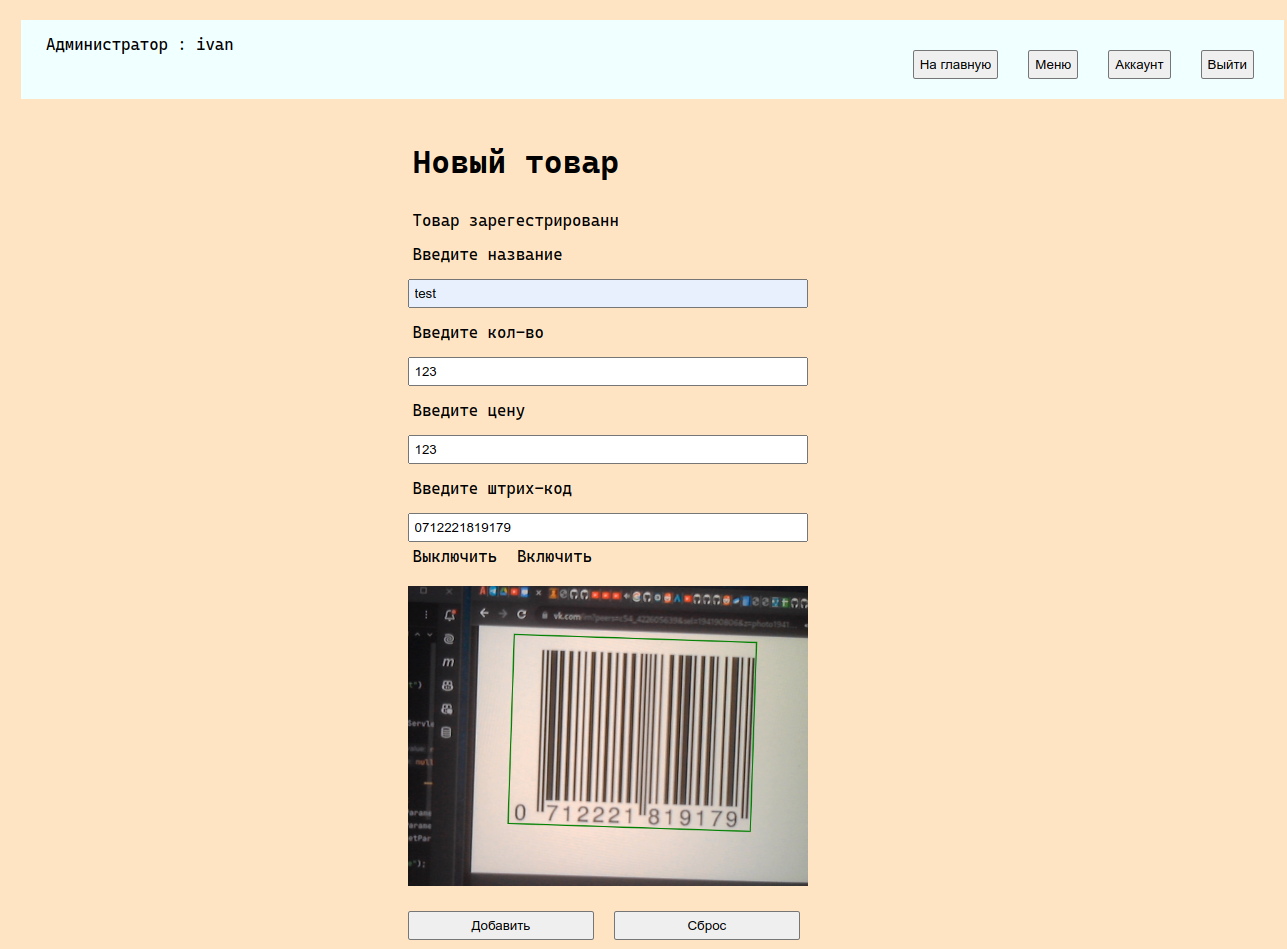
*Рисунок 21. Ответ программы на неправильный ввод пароля*



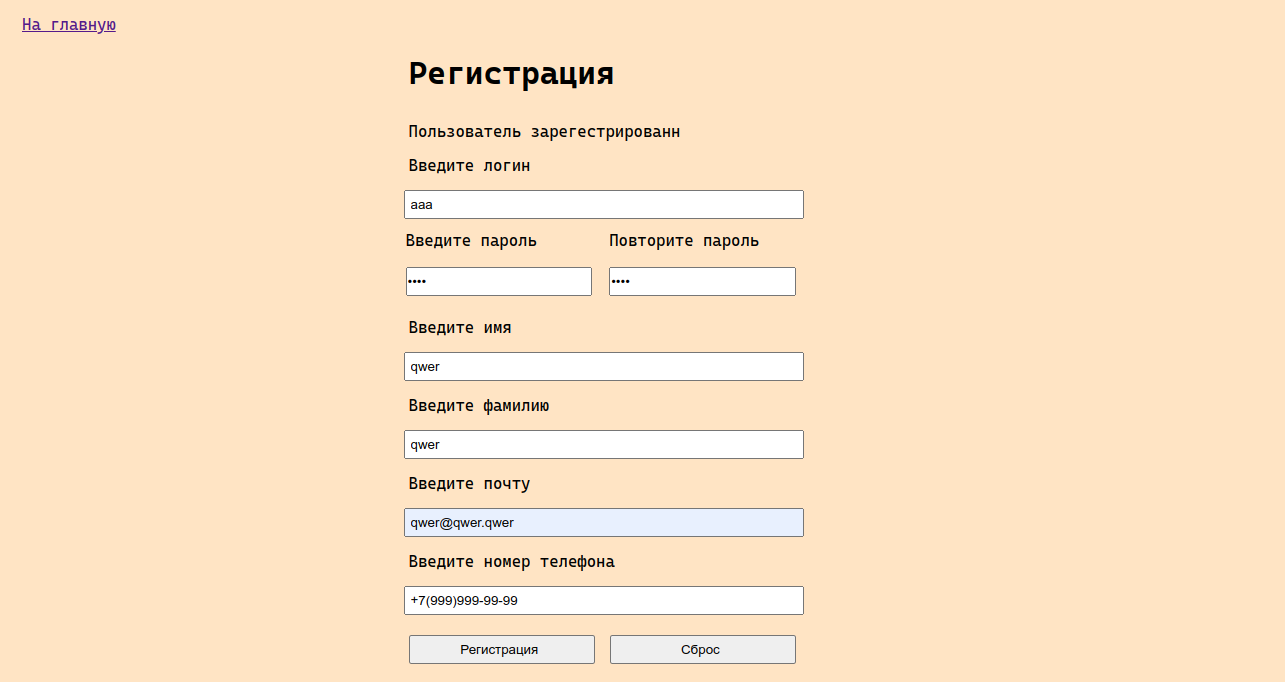
*Рисунок 22. Ответ программы на плохо отсканированный штрих-код*

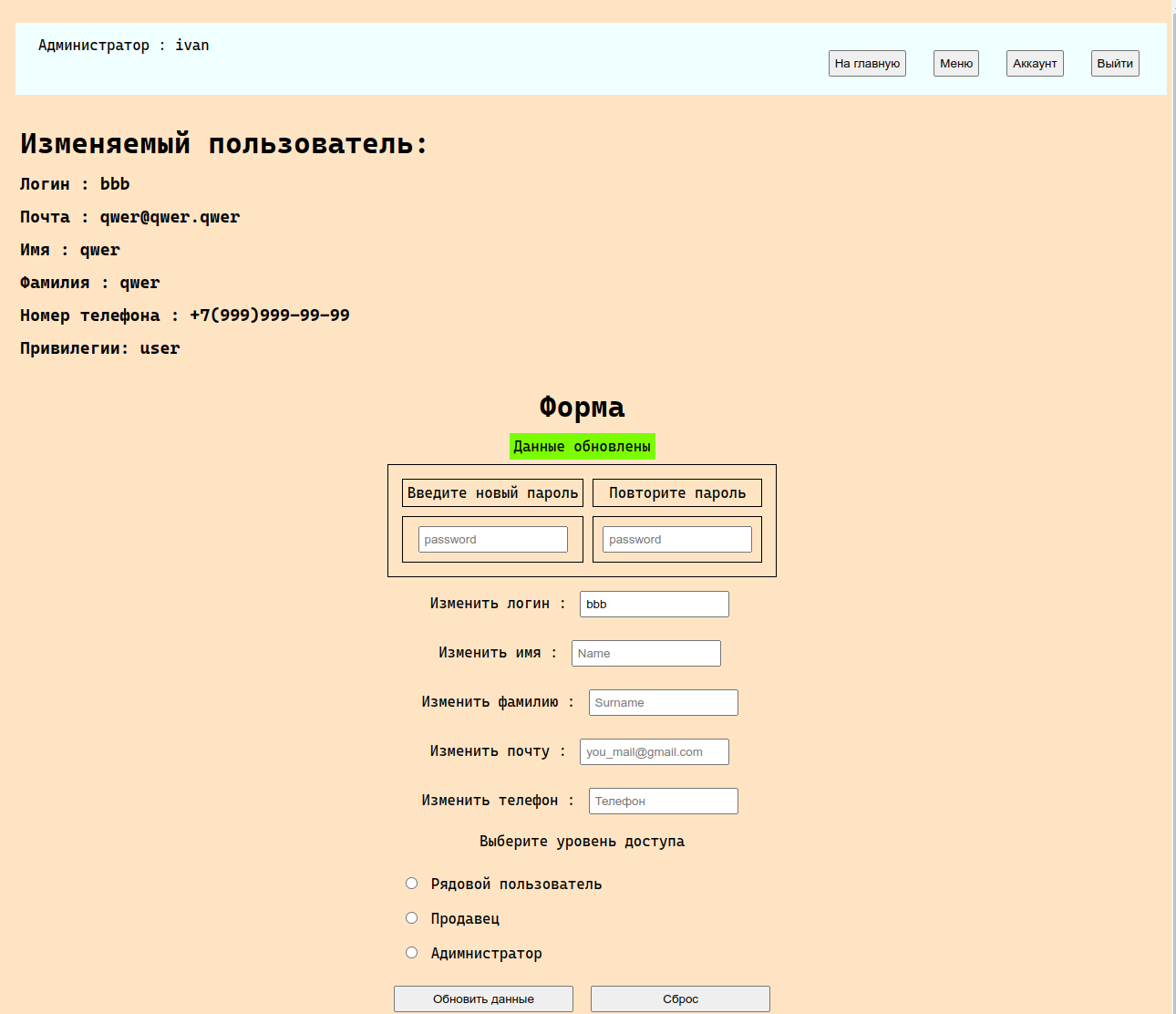


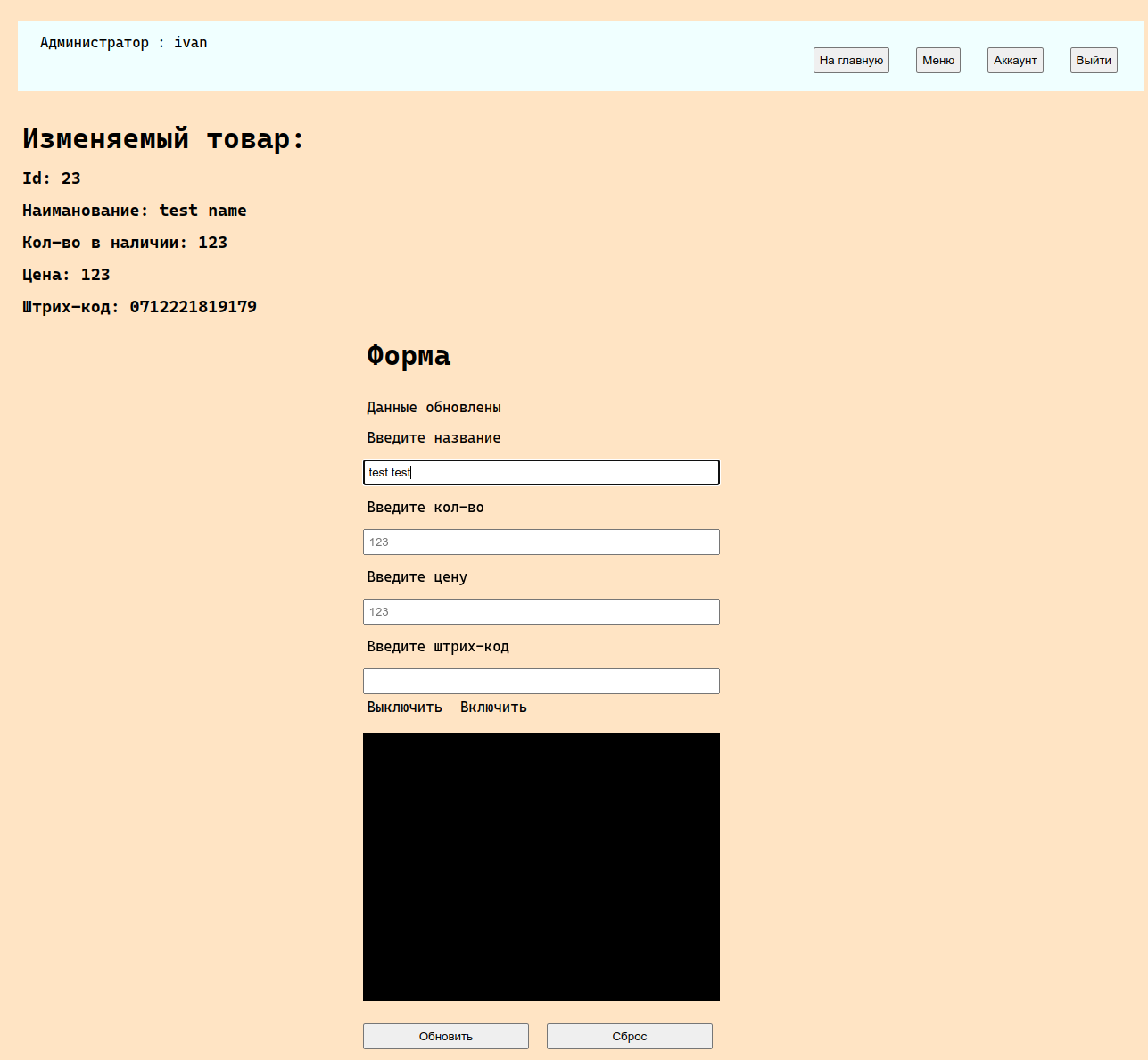
*Рисунок 23. Оплата прошла*



*Рисунок 24. Ответ программы на регистрацию нового товара*

*Рисунок 25. Ответ программы на успешную регистрацию пользователя*

*Рисунок 26. Ответ программы на изменение логина пользователя*



*Рисунок 27. Внесение изменения в товар*

# **ЗАКЛЮЧЕНИЕ**

В ходе курсовой работы:

* Определен программный инструментарий, применяемый для разработки веб-приложения.
* Разработано приложение, позволяющее управлять системой безопасности предприятия и контроля доступа пользователей.
* Проведено ручное тестирование программы.

**СПИСОК ЛИТЕРАТУРЫ**

1. Белая Т. И. Лекции по дисциплине «Разработка Java-приложений управления телекоммуникациями»
2. Jakarta Serlvet Api Javadoc [Электронный ресурс] URL: <https://javadoc.io/doc/jakarta.servlet/jakarta.servlet-api/6.0.0/jakarta.servlet/module-summary.html>
3. Maven Documentation [Электронный ресурс] URL: <https://maven.apache.org/>

**ПРИЛОЖЕНИЕ**

В приложениях приведен исходный код всей программы.

* Код всех java-файлов приведен в приложении А;
* JSP и CSS файлы представлены в приложении Б;
* Файлы конфигурации, например pom.xml, и другие представлены в приложении В.

**ПРИЛОЖЕНИЕ A. Файлы .java**

|  |
| --- |
| **AddProduct.java** |
| package lab.course.controller.admin;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.Product;  import java.io.IOException;  import static lab.course.model.Product.newProduct;  @WebServlet(name = "AddProduct", urlPatterns = "/addProduct")  public class AddProduct extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {  // получаем параметры из формы  String name = request.getParameter("productName");  String barcode = request.getParameter("barcode");  String count = request.getParameter("count");  String price = request.getParameter("price");  if (name == null || name.isEmpty()) {  request.getSession().setAttribute("errorAddProduct", "Наименоавние не может быть пустым");  response.sendRedirect("addProduct");  return;  } else if (barcode == null || barcode.isEmpty()) {  request.getSession().setAttribute("errorAddProduct", "Штрих-код не может быть пустым");  response.sendRedirect("addProduct");  return;  } else if (!Product.checkUniqueField("barcode", barcode)) {  request.getSession().setAttribute("errorAddProduct", "Штрих-код уже занят");  response.sendRedirect("addProduct");  return;  } else if (count == null || count.isEmpty() || Integer.parseInt(count) < 0) {  request.getSession().setAttribute("errorAddProduct", "Неправильно задано кол-во");  response.sendRedirect("addProduct");  return;  } else if (price == null || price.isEmpty() || Integer.parseInt(price) < 0) {  request.getSession().setAttribute("errorAddProduct", "Неправильно задана цена");  response.sendRedirect("addProduct");  return;  }  // если insert прошел успешно, то выводим уведомление  if (newProduct(name, barcode, Integer.valueOf(count), Integer.valueOf(price))) {  request.getSession().removeAttribute("errorAddProduct");  request.getSession().setAttribute("successAddProduct", "Товар зарегестрированн");  } else {  request.getSession().setAttribute("errorAddProduct", "Что то пошло не так");  }  response.sendRedirect("addProduct");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  request.getRequestDispatcher("admin/addProduct.jsp").forward(request, response);  }  } |

|  |
| --- |
| **AddUser.java** |
| package lab.course.controller.admin;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.User;  import java.io.IOException;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  @WebServlet(name = "AddUser", urlPatterns = "/addUser")  public class AddUser extends HttpServlet {  private static final String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static final String USERNAME = "user";  private static final String PASSWORD = "qwer";  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {  // получаем параметры из формы  String login = request.getParameter("login");  // хэшируем пароль  String password = User.getHashedPassword(request.getParameter("password").trim());  String passwordSecond = User.getHashedPassword(request.getParameter("passwordSecond").trim());  String name = request.getParameter("name");  String surname = request.getParameter("surname");  String email = request.getParameter("email");  String phone = request.getParameter("phone");  String role = request.getParameter("roleInput");  // подготавливаем запрос  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "INSERT INTO `Users`(`login`, `password`, `email`, `name`, `surname`, `phone`, `role`) VALUES (?,?,?,?,?,?,?)";  int resultExecute = 0;  if (!passwordSecond.equals(password)) {  request.getSession().setAttribute("errorAddUser", "Пароли не совпадают");  response.sendRedirect("addUser");  return;  } else if (!User.checkEmail(email)) {  request.getSession().setAttribute("errorAddUser", "Неправильно введена почта");  response.sendRedirect("addUser");  return;  } else if (!User.checkUniqueField("phone",phone)) {  request.getSession().setAttribute("errorAddUser", "Аккаунт с этим номером уже существует");  response.sendRedirect("addUser");  return;  } else if (!User.checkUniqueField("login", login)) {  request.getSession().setAttribute("errorAddUser", "Логин занят");  response.sendRedirect("addUser");  return;  }  // подключаем базу данных  // если прошли проверки, то заносим пользователя в бд  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, login);  preparedStatement.setString(2, password);  preparedStatement.setString(3, email);  preparedStatement.setString(4, name);  preparedStatement.setString(5, surname);  preparedStatement.setString(6, phone);  preparedStatement.setString(7, role);  resultExecute = preparedStatement.executeUpdate();  // preparedStatement.execute();  connection.close();  } catch (Exception e) {  e.printStackTrace();  }  // если insert прошел успешно, то выводим уведомление  if (resultExecute != 0) {  request.getSession().removeAttribute("errorAddUser");  request.getSession().setAttribute("successAddUser", "Пользователь зарегестрированн");  } else {  request.getSession().setAttribute("errorAddUser", "Что то пошло не так");  }  response.sendRedirect("addUser");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  request.getRequestDispatcher("admin/addUser.jsp").forward(request, response);  }  } |

|  |
| --- |
| **EditProduct.java** |
| package lab.course.controller.admin;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.Product;  import java.io.IOException;  import java.util.HashMap;  import static lab.course.model.Product.\*;  @WebServlet(name = "EditProduct", urlPatterns = "/editProduct")  public class EditProduct extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  // получаем параметры  int id= Integer.parseInt((String) request.getSession().getAttribute("selected\_product"));  String productName = request.getParameter("productName");  String barcode = request.getParameter("barcode");  String count\_string = request.getParameter("count");  String price\_string = request.getParameter("price");  // инициализируем ассощиативный массив для последующей отправки его в бд  HashMap<String,String> map = new HashMap<>();  // проверка на штрих-код  if (barcode != null && !barcode.isEmpty()) {  if (!Product.checkUniqueField("barcode",barcode)) {  request.getSession().setAttribute("errorEditProduct", "Ошибка, штрих-код уже занят");  response.sendRedirect("/editProduct");  return;  } else {  map.put("barcode", barcode);  request.getSession().setAttribute("product\_barcode",barcode);  }  }  // проверка на имя  if (productName!= null && !productName.isEmpty()) {  map.put("name", productName);  request.getSession().setAttribute("product\_name",productName);  }  // проверка на кол-во  if (count\_string!= null && !count\_string.isEmpty()) {  Integer count = Integer.parseInt(count\_string);  if (count < 0) {  request.getSession().setAttribute("errorEditProduct", "Ошибка, не может быть отрицательного кол-ва");  response.sendRedirect("/editProduct");  return;  } else {  map.put("count\_in\_stok", count\_string);  request.getSession().setAttribute("product\_count", count);  }  }  // проверка на цену  if (price\_string!= null && !price\_string.isEmpty()) {  Integer price = Integer.parseInt(price\_string);  if (price< 0) {  request.getSession().setAttribute("errorEditProduct", "Ошибка, не может быть отрицательной цены");  response.sendRedirect("/editProduct");  return;  } else {  map.put("price", price\_string);  request.getSession().setAttribute("product\_price",price);  }  }  // загружаем изменения в бд + обработка ошибки  boolean check = updateProductParam(id, map);  if (!check) {  request.getSession().setAttribute("errorEditProduct", "Произошла ошибка во время обновления данных");  response.sendRedirect("/editProduct");  return;  }  // если все хорошо, то говорим что все хорошо  request.getSession().setAttribute("successEditProduct", "Данные обновлены");  response.sendRedirect("/editProduct");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.getRequestDispatcher("admin/editProduct.jsp").forward(request, response);  }  } |

|  |
| --- |
| **EditUser.java** |
| package lab.course.controller.admin;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.User;  import java.io.IOException;  import java.util.HashMap;  import java.util.Objects;  import static lab.course.model.User.\*;  @WebServlet(name = "EditUser", urlPatterns = "/editUser")  public class EditUser extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.getSession().setAttribute("errorEdit", null);  request.getSession().setAttribute("sucEdit", null);  // получаем параметры  String user\_login = (String) request.getSession().getAttribute("user\_login");  String login = request.getParameter("login");  String passwordNew = getHashedPassword(request.getParameter("passwordNew"));  String passwordSecond = getHashedPassword(request.getParameter("passwordNewSecond"));  String name = request.getParameter("name");  String surname = request.getParameter("surname");  String email = request.getParameter("email");  String phone = request.getParameter("phone");  String role = request.getParameter("roleInput");  // инициализируем ассощиативный массив для последующей отправки его в бд  HashMap<String,String> map = new HashMap<>();  // правильность пароля  if ( passwordNew != null && !passwordNew.isEmpty() || passwordSecond != null && !passwordSecond.isEmpty()) {  if (!passwordNew.equals(passwordSecond)) {  request.getSession().setAttribute("errorEdit", "Ошибка: Проверьте пароль");  response.sendRedirect("/editUser");  return;  } else {  map.put("password", passwordNew);  }  }  // проверка на правилный email  if (email != null && !email.isEmpty()) {  if (!checkEmail(email)) {  request.getSession().setAttribute("errorEdit", "Некорректный адрес почты");  response.sendRedirect("/editUser");  return;  } else {  map.put("email", email);  request.getSession().setAttribute("user\_email", email);  }  }  // проверка на логин  if (login != null && !login.isEmpty()) {  if (!User.checkUniqueField("login", login)) {  request.getSession().setAttribute("errorEdit", "Ошибка, логин уже занят");  response.sendRedirect("/editUser");  return;  } else {  map.put("login", login);  request.getSession().setAttribute("user\_login", login);  }  }  // проверка на имя  if (name != null && !name.isEmpty()) {  map.put("name", name);  request.getSession().setAttribute("user\_name",name);  }  // проверка на фамилию  if (surname != null && !surname.isEmpty()) {  map.put("surname", surname);  request.getSession().setAttribute("user\_surname", surname);  }  // проверка на телефон  if (phone != null && !phone.isEmpty()) {  if (!User.checkUniqueField("phone",phone)) {  request.getSession().setAttribute("errorEdit", "Ошибка, телефон уже используется");  response.sendRedirect("/editUser");  return;  } else {  map.put("phone", phone);  request.getSession().setAttribute("user\_phone", phone);  }  }  // проверка на роль  if (role != null && !role.isEmpty()) {  map.put("role", role);  request.getSession().setAttribute("user\_role", role);  }  // получаем id пользователя по логину для загрузки в бд  int id = Objects.requireNonNull(getUserByLogin(user\_login)).getId();  // загружаем изменения в бд + обработка ошибки  boolean check = updateUserParam(id, map);  if (!check) {  request.getSession().setAttribute("errorEdit", "Произошла ошибка во время обновления данных");  response.sendRedirect("/editUser");  return;  }  // если все хорошо, то говорим что все хорошо  request.getSession().setAttribute("sucEdit", "Данные обновлены");  response.sendRedirect("/editUser");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.getRequestDispatcher("admin/editUser.jsp").forward(request, response);  }  } |

|  |
| --- |
| **ListAllProducts.java** |
| package lab.course.controller.admin;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import java.io.IOException;  @WebServlet(name = "ListAllProducts", urlPatterns = "/listProduct")  public class ListAllProducts extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.getRequestDispatcher("admin/listProduct.jsp").forward(request, response);  }  } |

|  |
| --- |
| **ListAllUsers.java** |
| package lab.course.controller.admin;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.ListUsers;  import java.io.IOException;  @WebServlet(name = "ListAllUsers", urlPatterns = "/listUser")  public class ListAllUsers extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.getRequestDispatcher("admin/listUser.jsp").forward(request, response);  }  } |

|  |
| --- |
| **RedirectToEditProduct.java** |
| package lab.course.controller.admin;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.Product;  import lab.course.model.User;  import java.io.IOException;  @WebServlet(name = "RedirectToEditProduct ", urlPatterns = "/redirectToEditProduct")  public class RedirectToEditProduct extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.getSession().setAttribute("errorEditProduct", null);  request.getSession().setAttribute("successEditProduct", null);  Integer id = Integer.valueOf( (String) request.getSession().getAttribute("selected\_product"));  Product product = Product.getProductById(id);  if (product != null) {  System.out.println(id);  request.getSession().setAttribute("product\_name", product.getProduct\_name());  request.getSession().setAttribute("product\_barcode", product.getBarcode());  request.getSession().setAttribute("product\_count", product.getCount\_in\_stok());  request.getSession().setAttribute("product\_price", product.getPrice());  }  request.getRequestDispatcher("/editProduct").forward(request, response);  }  } |

|  |
| --- |
| **RedirectToEditUser.java** |
| package lab.course.controller.admin;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.User;  import java.io.IOException;  @WebServlet(name = "RedirectToEditUser ", urlPatterns = "/redirectToEditUser")  public class RedirectToEditUser extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.getSession().setAttribute("errorEdit", null);  request.getSession().setAttribute("sucEdit", null);  String login = (String) request.getSession().getAttribute("user\_login");  User user = User.getUserByLogin(login);  if (user != null) {  request.getSession().setAttribute("user\_email", user.getEmail());  request.getSession().setAttribute("user\_name", user.getName());  request.getSession().setAttribute("user\_surname", user.getSurname());  request.getSession().setAttribute("user\_phone", user.getPhone());  request.getSession().setAttribute("user\_role", user.getRole());  } else {  request.getSession().setAttribute("noUser", true);  request.getRequestDispatcher("/login").forward(request, response);  }  request.getRequestDispatcher("/editUser").forward(request, response);  }  } |

|  |
| --- |
| **RedirectToFunctions.java** |
| package lab.course.controller.admin;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import jakarta.servlet.http.HttpSession;  import java.io.IOException;  @WebServlet(name = "RedirectToFunctions", urlPatterns = "/func")  public class RedirectToFunctions extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String choose = request.getParameter("choose");  request.getSession().setAttribute("choose", choose);  if (choose.equals("editUser")) {  request.getSession().setAttribute("user\_login", request.getParameter("login"));  }  if (choose.equals("editProduct")) {  request.getSession().setAttribute("selected\_product", request.getParameter("product"));  }  response.sendRedirect("/func");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  HttpSession session = request.getSession();  boolean check = (session.getAttribute("role") != null);  String choose = (String) session.getAttribute("choose");  if (check) {  if (choose.equals("addUser")) {  request.getRequestDispatcher("admin/addUser.jsp").forward(request, response);  } else if (choose.equals("editUser")) {  request.getRequestDispatcher("/redirectToEditUser").forward(request, response);  } else if (choose.equals("listUser")) {  request.getRequestDispatcher("/listUser").forward(request, response);  } else if (choose.equals("deleteUser")){  request.getRequestDispatcher("admin/deleteUser.jsp").forward(request, response);  } else if (choose.equals("addProduct")) {  request.getRequestDispatcher("admin/addProduct.jsp").forward(request, response);  } else if (choose.equals("editProduct")) {  request.getRequestDispatcher("/redirectToEditProduct").forward(request, response);  } else if (choose.equals("listProduct")) {  request.getRequestDispatcher("/listProduct").forward(request, response);  } else if (choose.equals("deleteProduct")){  request.getRequestDispatcher("admin/deleteProduct.jsp").forward(request, response);  }  } else {  request.getRequestDispatcher("login.jsp").forward(request, response);  }  }  } |

|  |
| --- |
| **AccountChange.java** |
| package lab.course.controller.auth;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.User;  import java.io.IOException;  import java.util.HashMap;  import static lab.course.model.User.\*;  @WebServlet(name = "AccountChange", urlPatterns = "/account")  public class AccountChange extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  // получаем параметры  String login = request.getParameter("login");  String passwordNew = getHashedPassword(request.getParameter("passwordNew"));  String passwordOld = getHashedPassword(request.getParameter("passwordOld"));  String passwordSecond = getHashedPassword(request.getParameter("passwordNewSecond").trim());  String name = request.getParameter("name");  String surname = request.getParameter("surname");  String email = request.getParameter("email");  String phone = request.getParameter("phone");  int id = (int) request.getSession().getAttribute("user\_id");  // инициализируем ассощиативный массив для последующей отправки его в бд  HashMap<String,String> map = new HashMap<>();  // правильность пароля  if (passwordOld != null && !passwordOld.isEmpty() ||  passwordNew != null && !passwordNew.isEmpty() ||  passwordSecond != null && !passwordSecond.isEmpty()) {  if (!request.getSession().getAttribute("password").equals(passwordOld)  || !passwordNew.equals(passwordSecond)) {  request.getSession().setAttribute("error", "Ошибка: Проверьте пароль");  response.sendRedirect("/account");  return;  } else {  map.put("password", passwordNew);  request.getSession().setAttribute("password", passwordNew);  }  }  // проверка на правилный email  if (email != null && !email.isEmpty()) {  if (!checkEmail(email)) {  request.getSession().setAttribute("error", "Некорректный адрес почты");  response.sendRedirect("/account");  return;  } else {  map.put("email", email);  request.getSession().setAttribute("email", email);  }  }  // проверка на логин  if (login != null && !login.isEmpty()) {  if (!User.checkUniqueField("login", login)) {  request.getSession().setAttribute("error", "Ошибка, логин уже занят");  response.sendRedirect("/account");  return;  } else {  map.put("login", login);  request.getSession().setAttribute("login", login);  }  }  // проверка на имя  if (name != null && !name.isEmpty()) {  map.put("name", name);  request.getSession().setAttribute("name",name);  }  // проверка на фамилию  if (surname != null && !surname.isEmpty()) {  map.put("surname", surname);  request.getSession().setAttribute("surname", surname);  }  // проверка на телефон  if (phone != null && !phone.isEmpty()) {  if (!User.checkUniqueField("phone",phone)) {  request.getSession().setAttribute("error", "Ошибка, телефон уже используется");  response.sendRedirect("/account");  return;  } else {  map.put("phone", phone);  request.getSession().setAttribute("phone", phone);  }  }  // загружаем изменения в бд + обработка ошибки  boolean check = updateUserParam(id, map);  if (!check) {  request.getSession().setAttribute("error", "Произошла ошибка во время обновления данных");  response.sendRedirect("/account");  return;  }  // если все хорошо, то говорим что все хорошо  request.getSession().setAttribute("error", null);  request.getSession().setAttribute("suc", "Данные обновлены");  response.sendRedirect("/account");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  request.getRequestDispatcher("user\_profile.jsp").forward(request, response);  }  } |

|  |
| --- |
| **AccountFillter.java** |
| package lab.course.controller.auth;  import java.io.IOException;  import jakarta.servlet.Filter;  import jakarta.servlet.FilterChain;  import jakarta.servlet.FilterConfig;  import jakarta.servlet.ServletException;  import jakarta.servlet.ServletRequest;  import jakarta.servlet.ServletResponse;  import jakarta.servlet.annotation.WebFilter;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import jakarta.servlet.http.HttpSession;  //  //Защищаемая часть  @WebFilter(urlPatterns = {"/user\_profile.jsp"})  public class AccountFilter implements Filter {  @Override  public void init(FilterConfig filterConfig) throws ServletException { }  @Override  public void destroy() { }  @Override  public void doFilter(ServletRequest servletRequest,  ServletResponse servletResponse,  FilterChain filterChain) throws IOException, ServletException {  HttpServletRequest request = (HttpServletRequest) servletRequest;  HttpServletResponse response = (HttpServletResponse) servletResponse;  HttpSession session = request.getSession(false);  //Существует ли сессиия  boolean loggedIn = session != null &&  session.getAttribute("login") != null &&  session.getAttribute("role") != null;  if (loggedIn) {  //Если существует то получаем роль  response.sendRedirect(request.getContextPath() + "/data");  //Если нет то на страницу входа.  } else response.sendRedirect(request.getContextPath() + "/login");  }  } |

|  |
| --- |
| **AccountRedirect.java** |
| package lab.course.controller.auth;  import java.io.IOException;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import jakarta.servlet.http.HttpSession;  @WebServlet(name = "AccountRedirect", urlPatterns = "/data")  public class AccountRedirect extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  response.sendRedirect("/data");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  HttpSession session = request.getSession();  boolean check = (session.getAttribute("role") != null);  if (check ) {  request.getRequestDispatcher("user\_profile.jsp").forward(request, response);  } else {  request.getRequestDispatcher("login.jsp").forward(request, response);  }  }  } |

|  |
| --- |
| **AuthFilter.java** |
| package lab.course.controller.auth;  import java.io.IOException;  import jakarta.servlet.Filter;  import jakarta.servlet.FilterChain;  import jakarta.servlet.FilterConfig;  import jakarta.servlet.ServletException;  import jakarta.servlet.ServletRequest;  import jakarta.servlet.ServletResponse;  import jakarta.servlet.annotation.WebFilter;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import jakarta.servlet.http.HttpSession;  //  //Защищаемая часть  @WebFilter(urlPatterns = {"/admin/\*", "/salesman/\*", "/user/\*"})  public class AuthFilter implements Filter {  @Override  public void init(FilterConfig filterConfig) throws ServletException { }  @Override  public void destroy() { }  @Override  public void doFilter(ServletRequest servletRequest,  ServletResponse servletResponse,  FilterChain filterChain) throws IOException, ServletException {  HttpServletRequest request = (HttpServletRequest) servletRequest;  HttpServletResponse response = (HttpServletResponse) servletResponse;  HttpSession session = request.getSession(false);  //Существует ли сессиия  boolean loggedIn = session != null &&  session.getAttribute("login") != null &&  session.getAttribute("role") != null;  if (loggedIn) {  //Если существует то получаем роль  String userRole = session.getAttribute("login").toString();  if (userRole.equals("user")) {  response.sendRedirect(request.getContextPath() + "/user/");  } else if (userRole.equals("salesman")) {  response.sendRedirect(request.getContextPath() + "/salesman/");  } else if (userRole.equals("admin")) {  response.sendRedirect(request.getContextPath() + "/admin/");  }  //Если нет то на страницу входа.  } else response.sendRedirect(request.getContextPath() + "/login");  }  } |

|  |
| --- |
| **Login.java** |
| package lab.course.controller.auth;  import java.nio.charset.StandardCharsets;  import java.security.\*;  import java.io.IOException;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import jakarta.servlet.ServletContext;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import jakarta.servlet.http.HttpSession;  @WebServlet(name = "Login", urlPatterns = "/login")  public class Login extends HttpServlet {  private static final String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static final String USERNAME = "user";  private static final String PASSWORD = "qwer";  String getHashedPassword(final String text) {  if (text == null) return null;  MessageDigest md = null;  try {  md = MessageDigest.getInstance("SHA-256");  md.update(text.getBytes("UTF-8"));  } catch (Exception e) {  e.printStackTrace();  }  if (md != null)  return new String(md.digest(), StandardCharsets.UTF\_8);  else return null;  }  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // считываем логин и хэш пароля  String login = request.getParameter("login");  String password = getHashedPassword(request.getParameter("password").trim());  HttpSession session = request.getSession(true);  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from Users where login = ?";  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, login);  ResultSet resultSet = preparedStatement.executeQuery();  if (resultSet.next()) {  session.setAttribute("user\_id", resultSet.getInt("id"));  session.setAttribute("login", resultSet.getString("login"));  session.setAttribute("password", resultSet.getString("password"));  session.setAttribute("role", resultSet.getString("role"));  session.setAttribute("name", resultSet.getString("name"));  session.setAttribute("surname", resultSet.getString("surname"));  session.setAttribute("email", resultSet.getString("email"));  session.setAttribute("phone", resultSet.getString("phone"));  }  connection.close();  } catch (Exception e) {  e.printStackTrace();  }  // проверяем логин и пароль, после перенаправляем на нужную страницу  if (login.equals(session.getAttribute("login")) &&  password.equals(session.getAttribute("password")) &&  "admin".equals(session.getAttribute("role")) ) {  response.sendRedirect("login");  } else if (login.equals(session.getAttribute("login")) &&  password.equals(session.getAttribute("password")) &&  "salesman".equals(session.getAttribute("role")) ) {  response.sendRedirect("login");  } else if (login.equals(session.getAttribute("login")) &&  password.equals(session.getAttribute("password")) &&  "user".equals(session.getAttribute("role")) ) {  response.sendRedirect("login");  } else {  request.getSession().invalidate();  request.getSession().setAttribute("hash", getHashedPassword(password));  request.getSession().setAttribute("errorMessage", "Логин или пароль введены неправильно");  response.sendRedirect("login");  }  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  HttpSession session = request.getSession();  boolean check = (session.getAttribute("role") != null);  ServletContext servletContext = getServletContext();  // ServletRequest servletRequest = request.  if (check && session.getAttribute("role").equals("admin")) {  request.getRequestDispatcher("admin/").forward(request, response);  // servletContext.getRequestDispatcher("/admin/").forward(request, response);  } else if (check && session.getAttribute("role").equals("salesman")) {  request.getRequestDispatcher("salesman/").forward(request, response);  // servletContext.getRequestDispatcher("/salesman/").forward(request, response);  } else if (check && session.getAttribute("role").equals("user")) {  request.getRequestDispatcher("user/").forward(request, response);  // servletContext.getRequestDispatcher("/user/").forward(request, response);  } else {  request.getRequestDispatcher("login.jsp").forward(request, response);  // servletContext.getRequestDispatcher("/login.jsp").forward(request, response);  }  }  } |

|  |
| --- |
| **LoginFilter.java** |
| package lab.course.controller.auth;  import jakarta.servlet.\*;  import jakarta.servlet.annotation.WebFilter;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import jakarta.servlet.http.HttpSession;  import java.io.IOException;  //Все страницы сайта обрабатывает данный фильтр  @WebFilter(urlPatterns = {"/admin/\*", "/salesman/\*", "/user/\*", "/buy/\*",  "/checkBarcode/\*", "/purchase/\*", "/pay/\*", "/remove/\*", "/account/\*", "/data/\*",  "/orders/\*", "/order/\*"})  public class LoginFilter implements Filter {  @Override  public void init(FilterConfig filterConfig) throws ServletException {  }  @Override  public void destroy() {  }  @Override  public void doFilter(ServletRequest servletRequest,  ServletResponse servletResponse,  FilterChain filterChain) throws IOException, ServletException {  //получение данных сессии  HttpServletRequest request = (HttpServletRequest) servletRequest;  HttpServletResponse response = (HttpServletResponse) servletResponse;  HttpSession session = request.getSession(false);  //URL Запроса/переадресации на Servlet входа  String loginURI = request.getContextPath() + "/login";  //Если сессия ранее создана  boolean loggedIn = session != null &&  session.getAttribute("login") != null &&  session.getAttribute("role") != null;  boolean loginRequest = request.getRequestURI().equals(loginURI);  //Если запрос пришел со страницы с входом или сессия не пуста даем добро следовать дальше  //Если нет ридерект на страницу входа  if (loggedIn || loginRequest) {  filterChain.doFilter(request, response);  } else {  response.sendRedirect(loginURI);  }  }  } |

|  |
| --- |
| **Logout.java** |
| package lab.course.controller.auth;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import jakarta.servlet.http.HttpSession;  import java.io.IOException;  @WebServlet(urlPatterns = "/logout")  public class Logout extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws IOException {  HttpSession session = request.getSession(false);  if (session != null) session.invalidate();  response.sendRedirect("/login");  }  } |

|  |
| --- |
| **Register.java** |
| package lab.course.controller.auth;  import java.io.IOException;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.User;  @WebServlet(name = "Register", urlPatterns = "/register")  public class Register extends HttpServlet {  private static final String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static final String USERNAME = "user";  private static final String PASSWORD = "qwer";  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws IOException {  // получаем параметры из формы  String login = request.getParameter("login");  // хэшируем пароль  String password = User.getHashedPassword(request.getParameter("password").trim());  String passwordSecond = User.getHashedPassword(request.getParameter("passwordSecond").trim());  String name = request.getParameter("name");  String surname = request.getParameter("surname");  String email = request.getParameter("email");  String phone = request.getParameter("phone");  // новые пользователи получают роль user  String role = "user";  // подготавливаем запрос  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "INSERT INTO `Users`(`login`, `password`, `email`, `name`, `surname`, `phone`, `role`) VALUES (?,?,?,?,?,?,?)";  int resultExecute = 0;  if (!passwordSecond.equals(password)) {  request.getSession().setAttribute("errorInput", "Пароли не совпадают");  response.sendRedirect("register");  return;  } else if (!User.checkEmail(email)) {  request.getSession().setAttribute("errorInput", "Неправильно введена почта");  response.sendRedirect("register");  return;  } else if (!User.checkUniqueField("phone",phone)) {  request.getSession().setAttribute("errorInput", "Аккаунт с этим номером уже существует");  response.sendRedirect("register");  return;  } else if (!User.checkUniqueField("login", login)) {  request.getSession().setAttribute("errorInput", "Логин занят");  response.sendRedirect("register");  return;  }  // подключаем базу данных  // если прошли проверки, то заносим пользователя в бд  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, login);  preparedStatement.setString(2, password);  preparedStatement.setString(3, email);  preparedStatement.setString(4, name);  preparedStatement.setString(5, surname);  preparedStatement.setString(6, phone);  preparedStatement.setString(7, role);  resultExecute = preparedStatement.executeUpdate();  // preparedStatement.execute();  connection.close();  } catch (Exception e) {  e.printStackTrace();  }  // если insert прошел успешно, то выводим уведомление  request.getSession().invalidate();  if (resultExecute != 0) {  request.getSession().setAttribute("success", "Пользователь зарегестрированн");  } else {  request.getSession().setAttribute("errorInput", "Что то пошло не так");  }  response.sendRedirect("register");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  request.getRequestDispatcher("register.jsp").forward(request, response);  }  } |

|  |
| --- |
| **UserAuthFilter.java** |
| package lab.course.controller.auth;  import jakarta.servlet.\*;  import jakarta.servlet.annotation.WebFilter;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import jakarta.servlet.http.HttpSession;  import java.io.IOException;  //Все страницы сайта обрабатывает данный фильтр  @WebFilter(urlPatterns = {"/\*"})  public class UserAuthFilter implements Filter {  String[] path = {"/register.jsp", "/user/index.jsp","/user/order.jsp",  "/user/orders.jsp", "/login", "/orders", "/order", "/data", "/logout",  "/account","/register"};  @Override  public void init(FilterConfig filterConfig) throws ServletException { }  @Override  public void destroy() { }  @Override  public void doFilter(ServletRequest servletRequest,  ServletResponse servletResponse,  FilterChain filterChain) throws IOException, ServletException {  //получение данных сессии  HttpServletRequest request = (HttpServletRequest) servletRequest;  HttpServletResponse response = (HttpServletResponse) servletResponse;  HttpSession session = request.getSession(false);  //URL Запроса/переадресации на Servlet входа  String loginURI = "/login.jsp";  //Если сессия ранее создана  boolean loggedIn = session != null &&  session.getAttribute("login") != null &&  session.getAttribute("role").equals("user");  int loginRequest = 0;  for (var a: path) {  if (request.getRequestURI().contains(a)) loginRequest = 1;  }  //Если запрос пришел со страницы с входом или сессия не пуста даем добро следовать дальше  //Если нет ридерект на страницу входа  if (loggedIn) {  System.out.println(loginRequest);  if (loginRequest == 1) {  filterChain.doFilter(request, response);  } else {  response.sendRedirect(loginURI);  }  } else{  filterChain.doFilter(request, response);  }  }  } |

|  |
| --- |
| **CheckBarcode.java** |
| package lab.course.controller.sales;  import java.io.IOException;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.ListProducts;  @WebServlet(name = "CheckBarcode", urlPatterns = "/checkBarcode")  public class CheckBarcode extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // получаем штрих код  String barcode = request.getParameter("barcode");  // создаем/получаем корзину  ListProducts listProducts = (ListProducts) request.getSession().getAttribute("list");  if(listProducts == null) listProducts = new ListProducts();  // добавляем товар по штрих коду + возврааем корзину или выводим ошибку  if (listProducts.addByBarcode(barcode)) {  request.getSession().removeAttribute("errorCheck");  request.getSession().removeAttribute("errorRead");  request.getSession().setAttribute("list", listProducts);  } else {  request.getSession().setAttribute("errorRead", "Товар не найден, либо штрих код считан неверно");  }  // редирект на себя, для очередного товара  response.sendRedirect("/checkBarcode");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  request.getRequestDispatcher("salesman/").forward(request, response);  }  } |

|  |
| --- |
| **HandleBuyer.java** |
| package lab.course.controller.sales;  import java.io.IOException;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.ListProducts;  @WebServlet(name = "HandleBuyer", urlPatterns = "/buy")  public class HandleBuyer extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // получаем корзину, если не получили выводим ошибку  ListProducts listProducts = (ListProducts) request.getSession().getAttribute("list");  if(listProducts == null || listProducts.getListProducts().isEmpty()) {  request.getSession().setAttribute("errorCheck", "Не выбран ни один товар");  response.sendRedirect("/checkBarcode");  return;  }  // считываем кол-во товаров в корзине из формы  for (var a :listProducts.getListProducts()){  String basket = request.getParameter("basket" + a.getId());  a.setCount(Integer.parseInt(basket));  }  // проверяем, хватает ли товара на складе  if (!listProducts.checkCountInStok()) {  request.getSession().removeAttribute("errorCheck");  response.sendRedirect("/checkBarcode");  return;  }  request.getSession().setAttribute("errorCheck",null);  response.sendRedirect("/buy");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  if (request.getSession().getAttribute("errorCheck") != null) {  request.getRequestDispatcher("/checkBarcode").forward(request, response);  } else {  request.getRequestDispatcher("salesman/handleBuyer.jsp").forward(request, response);  }  }  } |

|  |
| --- |
| **Purchase.java** |
| package lab.course.controller.sales;  import lab.course.model.User;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.ListProducts;  import java.io.IOException;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  @WebServlet(name = "Purchase", urlPatterns = "/purchase")  public class Purchase extends HttpServlet {  private static String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static String USERNAME = "user";  private static String PASSWORD = "qwer";  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  String phone = request.getParameter("buyer\_phone");  // если покупатель без номера телефона, то записываем такие покупки  // на пустого пользователя  if(phone != null && phone.isEmpty()) { phone = "+7(000)000-00-00"; }  // ищем пользователя с подходящим телефоном  User user = User.getUserByPhone(phone);  // если не нашли телефон, выводим ошибку  if (phone == null || user == null && !phone.isEmpty()) {  request.getSession().setAttribute("err", "Пользователя с таким номером телефона не сушествует");  response.sendRedirect("/purchase");  return;  }  // получаем корзину, если не получили выводим ошибку  ListProducts listProducts = (ListProducts) request.getSession().getAttribute("list");  if(listProducts == null || listProducts.getListProducts().isEmpty()) {  request.getSession().setAttribute("err", "Пустая корзина");  response.sendRedirect("/purchase");  return;  }  // читаем способ оплаты и передаем его в сессию, для дальнейших действий  String payment = request.getParameter("payment");  request.getSession().setAttribute("payment", payment);  // оформляем покупку в бд  listProducts.makePurchase(user);  // отчищаем сессию от совершенной покупки  request.getSession().removeAttribute("list");  request.getSession().removeAttribute("err");  request.getSession().removeAttribute("errorCheck");  response.sendRedirect("/purchase");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  if (request.getSession().getAttribute("err") != null) {  request.getRequestDispatcher("/buy").forward(request, response);  } else {  request.getRequestDispatcher("salesman/payment.jsp").forward(request, response);  }  }  } |

|  |
| --- |
| **RemoveBasket.java** |
| package lab.course.controller.sales;  import java.io.IOException;  import java.util.ArrayList;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.ListProducts;  import lab.course.model.Product;  @WebServlet(name = "RemoveBasket", urlPatterns = "/remove")  public class RemoveBasket extends HttpServlet {  protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  // получаем список товаров, которые удаляем из корзины  String[] checkboxes = request.getParameterValues("delete");  // получаем корзину, если не получили выводим ошибку  ListProducts listProducts = (ListProducts) request.getSession().getAttribute("list");  if(listProducts == null || listProducts.getListProducts().isEmpty() || checkboxes == null) {  request.getSession().setAttribute("errorCheck", "Нечего удалять");  response.sendRedirect("/checkBarcode");  return;  }  // находим все товары, которые будем удаялть из корзины  ArrayList<Product> temp = new ArrayList<>();  for (var check : checkboxes) {  for (var a : listProducts.getListProducts()) {  if (check.equals(a.getId().toString())) {  temp.add(a);  // listProducts.deleteProduct(a);  }  }  }  // удаяем все найденные  for (var a : temp) {  listProducts.deleteProduct(a);  }  // возвращаемся назад  request.getSession().removeAttribute("errorCheck");  response.sendRedirect("/checkBarcode");  }  protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  request.getRequestDispatcher("/checkBarcode").forward(request, response);  }  } |

|  |
| --- |
| **OrderServlet.java** |
| package lab.course.controller.show;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.Order;  import lab.course.model.Orders;  import java.io.IOException;  import java.util.ArrayList;  @WebServlet(name = "OrderServlet", urlPatterns = "/order")  public class OrderServlet extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  String purchase\_id = request.getParameter("order\_id");  Integer user\_id = (Integer) request.getSession().getAttribute("user\_id");  try {  if (purchase\_id == null ) throw new NumberFormatException();  Integer p\_id = Integer.valueOf(purchase\_id);  ArrayList<Order> info = Order.getOrderInfo(p\_id, user\_id);  if (info == null) throw new NumberFormatException();  request.getSession().removeAttribute("errorOrder");  request.getSession().setAttribute("orderInfo", info);  } catch (NumberFormatException e) {  request.getSession().setAttribute("errorOrder", "Неправильный номер заказа");  }  request.getRequestDispatcher("user/order.jsp").forward(request, response);  }  } |

|  |
| --- |
| **OrdersSevlet.java** |
| package lab.course.controller.show;  import jakarta.servlet.ServletException;  import jakarta.servlet.annotation.WebServlet;  import jakarta.servlet.http.HttpServlet;  import jakarta.servlet.http.HttpServletRequest;  import jakarta.servlet.http.HttpServletResponse;  import lab.course.model.Orders;  import java.io.IOException;  @WebServlet(name = "OrdersServlet", urlPatterns = "/orders")  public class OrdersServlet extends HttpServlet {  protected void doGet(HttpServletRequest request, HttpServletResponse response)  throws ServletException, IOException {  Orders orders = new Orders((Integer) request.getSession().getAttribute("user\_id"));  request.getSession().setAttribute("orders", orders);  request.getRequestDispatcher("user/orders.jsp").forward(request, response);  }  } |

|  |
| --- |
| **ListProducts.java** |
| package lab.course.model;  import java.sql.\*;  import java.util.ArrayList;  public class ListProducts {  private static String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static String USERNAME = "user";  private static String PASSWORD = "qwer";  private ArrayList<Product> list;  public ListProducts() {  this.list = new ArrayList<Product>();  }  public void insertProduct(Product a) {  list.add(a);  }  public void deleteProduct(Product a) {  list.remove(a);  }  public ArrayList<Product> getListProducts() {  return list;  }  public Integer getAllPrice() {  Integer sum = 0;  for (var a : list) {  sum += a.getPrice() \* a.getCount();  }  return sum;  }  public boolean checkCountInStok() {  for(var a : this.list) {  if (a.getCount() > a.getCount\_in\_stok()){  return false;  }  }  return true;  }  public boolean addByBarcode(String barcode) {  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from products where barcode = ?";  ResultSet result = null;  // подключаем базу данных  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, barcode);  result = preparedStatement.executeQuery();  Product temp;  // если шрих код уже существует в корзине, то кол-во +1 и возвращаем true  if (addOneToProduct(barcode)) { return true; }  // если шрих код сущесвтует в бд, то добавляем в корзину  if (result.next()) {  temp = new Product( result.getInt("id"),  result.getString("name"),  barcode,  result.getInt("count\_in\_stok"),  result.getInt("price"),  1);  } else {  connection.close();  return false;  }  insertProduct(temp);  connection.close();  } catch (Exception e) {  e.printStackTrace();  return false;  }  return true;  }  // прибавляет 1 продукт в корзину по шрих коду  public boolean addOneToProduct(String barcode) {  for(var a: this.list) {  if (a.getBarcode().equals(barcode)) {  a.setCount(a.getCount() + 1);  return true;  }  }  return false;  }  public Integer makePurchase(User user) {  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "INSERT INTO `purchases`(`buyer\_id`, `date`, `total\_price`) VALUES (?,?,?)";  // подключаем базу данных  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL, Statement.RETURN\_GENERATED\_KEYS);  preparedStatement.setInt(1, user.getId());  preparedStatement.setDate(2, new Date(System.currentTimeMillis()));  preparedStatement.setInt(3, getAllPrice());  // заполняем информацию о покупке  int insert = preparedStatement.executeUpdate();  if (insert == 0) {  return -1;  }  // получаем id покупки  Integer purchase\_id = 0;  try(ResultSet key = preparedStatement.getGeneratedKeys()) {  if(key.next()) {  purchase\_id = key.getInt(1);  } else {  return -2;  }  }  // заполняем информацию о товарах в покупке  SQL = "INSERT INTO `purchases\_items`(`product\_count`, `purchase\_id`, `product\_id`) VALUES (?,?,?)";  preparedStatement = connection.prepareStatement(SQL);  for(var a : this.list) {  preparedStatement.setInt(1, a.getCount());  preparedStatement.setInt(2, purchase\_id);  preparedStatement.setInt(3, a.getId());  if (preparedStatement.executeUpdate() == 0) {  return -3;  }  }  // обновляем наличие товара  SQL = "UPDATE `products` SET `count\_in\_stok`= ? WHERE `products`.`id` = ?";  preparedStatement = connection.prepareStatement(SQL);  for(var a : this.list) {  int sub = a.getCount\_in\_stok() - a.getCount();  preparedStatement.setInt(1, sub);  preparedStatement.setInt(2, a.getId());  if (preparedStatement.executeUpdate() == 0) {  return -4;  }  }  connection.close();  return 0;  } catch (Exception e) {  e.printStackTrace();  return -1;  }  }  public static ArrayList<Product> getProducts() {  ArrayList<Product> products = new ArrayList<>();  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from products";  ResultSet result = null;  // подключаем базу данных  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  result = preparedStatement.executeQuery();  Product temp = null;  while(result.next()) {  temp = new Product(result.getInt("id"),  result.getString("name"),  result.getString("barcode"),  result.getInt("count\_in\_stok"),  result.getInt("price"));  products.add(temp);  }  connection.close();  } catch (Exception e) {  e.printStackTrace();  return null;  }  return products;  }  } |

|  |
| --- |
| **ListUsers.java** |
| package lab.course.model;  import java.sql.\*;  import java.util.ArrayList;  import java.sql.\*;  import java.util.ArrayList;  public class ListUsers{  private static String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static String USERNAME = "user";  private static String PASSWORD = "qwer";  private ArrayList<User> list;  public ListUsers() {  this.list = new ArrayList<User>();  }  public void insertUser(User a) {  list.add(a);  }  public void deleteUser(User a) {  list.remove(a);  }  public ArrayList<User> getListUsers() {  return list;  }  public static ArrayList<User> getUsers() {  ArrayList<User> users = new ArrayList<>();  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from Users";  ResultSet result = null;  // подключаем базу данных  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  result = preparedStatement.executeQuery();  User temp = null;  while(result.next()) {  temp = new User(result.getInt("id"),  result.getString("login"),  result.getString("email"),  result.getString("name"),  result.getString("surname"),  result.getString("phone"),  result.getString("role"));  users.add(temp);  }  connection.close();  } catch (Exception e) {  e.printStackTrace();  return null;  }  return users;  }  } |

|  |
| --- |
| **Order.java** |
| package lab.course.model;  import java.sql.\*;  import java.util.ArrayList;  public class Order {  private static String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static String USERNAME = "user";  private static String PASSWORD = "qwer";  private int purchase\_id;  private int buyer\_id;  private Date date;  private int total\_price;  private String name;  private int price;  private int count;  public Order(int purchase, int buyer\_id, Date date, int price) {  this.purchase\_id = purchase;  this.buyer\_id = buyer\_id;  this.date = date;  this.total\_price = price;  }  public Order(int purchase, int buyer\_id, Date date, int t\_price, String name, int price, int count) {  this.purchase\_id = purchase;  this.buyer\_id = buyer\_id;  this.date = date;  this.total\_price = t\_price;  this.name = name;  this.price = price;  this.count = count;  }  public static ArrayList<Order> getOrderInfo(Integer order\_id, Integer user\_id) {  ArrayList<Order> orders = new ArrayList<>();  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select p.purchase\_id, p.date, p.total\_price, pi.product\_count, pr.name, pr.price " +  "from purchases as p cross join purchases\_items as pi " +  "on p.purchase\_id = pi.purchase\_id cross join products as pr " +  "on pr.id = pi.product\_id where p.purchase\_id = ? and p.buyer\_id = ?";  ResultSet result = null;  // подключаем базу данных  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setInt(1, order\_id);  preparedStatement.setInt(2, user\_id);  result = preparedStatement.executeQuery();  Order temp = null;  while(result.next()) {  temp = new Order(result.getInt("purchase\_id"),  user\_id,  result.getDate("date"),  result.getInt("total\_price"),  result.getString("name"),  result.getInt("price"),  result.getInt("product\_count"));  orders.add(temp);  }  connection.close();  } catch (Exception e) {  e.printStackTrace();  return null;  }  return orders;  }  public Integer getPurchase\_id() {  return purchase\_id;  }  public void setPurchase\_id(Integer purchase\_id) {  this.purchase\_id = purchase\_id;  }  public Integer getBuyer\_id() {  return buyer\_id;  }  public void setBuyer\_id(Integer buyer\_id) {  this.buyer\_id = buyer\_id;  }  public Date getDate() {  return date;  }  public void setDate(Date date) {  this.date = date;  }  public Integer getTotal\_price() {  return total\_price;  }  public void setTotal\_price(Integer total\_price) {  this.total\_price = total\_price;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public Integer getPrice() {  return price;  }  public void setPrice(Integer price) {  this.price = price;  }  public int getCount() {  return count;  }  public void setCount(int count) {  this.count = count;  }  } |

|  |
| --- |
| **Orders.java** |
| package lab.course.model;  import java.sql.Connection;  import java.sql.DriverManager;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.ArrayList;  public class Orders {  private static String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static String USERNAME = "user";  private static String PASSWORD = "qwer";  private Integer user\_id;  public Orders(Integer id) {  this.user\_id = id;  }  public ArrayList<Order> getOrders() {  ArrayList<Order> orders = new ArrayList<>();  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from purchases where buyer\_id = ?";  ResultSet result = null;  // подключаем базу данных  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setInt(1, user\_id);  result = preparedStatement.executeQuery();  Order temp = null;  while(result.next()) {  temp = new Order(result.getInt("purchase\_id"),  user\_id,  result.getDate("date"),  result.getInt("total\_price"));  orders.add(temp);  }  connection.close();  } catch (Exception e) {  e.printStackTrace();  return null;  }  return orders;  }  } |

|  |
| --- |
| **Product.java** |
| package lab.course.model;  import jakarta.persistence.criteria.CriteriaBuilder;  import java.sql.\*;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.Map;  public class Product {  private Integer id;  private String product\_name;  private String barcode;  private Integer count\_in\_stok;  private Integer price;  private Integer count;  private static final String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static final String USERNAME = "user";  private static final String PASSWORD = "qwer";  public Product(Integer id, String name, String barcode,  Integer count\_in\_stok, Integer price, Integer count) {  this.id = id;  this.product\_name = name;  this.barcode = barcode;  this.count\_in\_stok = count\_in\_stok;  this.price = price;  this.count = count;  }  public Product(Integer id, String name, String barcode,  Integer count\_in\_stok, Integer price ) {  this.id = id;  this.product\_name = name;  this.barcode = barcode;  this.count\_in\_stok = count\_in\_stok;  this.price = price;  }  public static boolean updateProductParam(int id, HashMap<String, String> map) {  if (map == null || map.isEmpty() || id < 0) return false;  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = null;  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  connection.setAutoCommit(false);  for(Map.Entry<String, String> entry : map.entrySet()) {  SQL = "UPDATE `products` SET `"+ entry.getKey() +"` = ? where `products`.`id` = "  + String.valueOf(id);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, entry.getValue());  int res = preparedStatement.executeUpdate();  if (res == 0) {  connection.rollback();  connection.close();  return false;  }  }  connection.commit();  connection.close();  } catch (Exception e) {  if (connection != null) {  try {  connection.rollback();  } catch (SQLException ex) {  throw new RuntimeException(ex);  }  }  e.printStackTrace();  }  return true;  }  public static Product getProductById(Integer id) {  if (id == null || id < 0) return null;  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from products where id = ?";  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setInt(1, id);  ResultSet resultSet = preparedStatement.executeQuery();  Product temp = null;  if (resultSet.next()) {  temp = new Product(  id,  resultSet.getString("name"),  resultSet.getString("barcode"),  resultSet.getInt("count\_in\_stok"),  resultSet.getInt("price")  );  }  connection.close();  return temp;  } catch (Exception e) {  e.printStackTrace();  }  return null;  }  public static boolean checkUniqueField(String field, String value) {  String SQL = "select \* from products where " + field + " = ?";  Connection connection = null;  PreparedStatement preparedStatement = null;  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, value);  ResultSet resultSet = preparedStatement.executeQuery();  connection.close();  if (!resultSet.next()) {  connection.close();  return true;  }  connection.close();  } catch (Exception e) {  e.printStackTrace();  }  return false;  }  public static boolean newProduct(String name, String barcode, Integer count\_in\_stok, Integer price) {  // подготавливаем запрос  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "INSERT INTO `products`(`name`, `barcode`, `count\_in\_stok`, `price`) VALUES (?,?,?,?)";  // подключаем базу данных  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, name);  preparedStatement.setString(2, barcode);  preparedStatement.setInt(3, count\_in\_stok);  preparedStatement.setInt(4, price);  // заполняем информацию о покупке  int insert = preparedStatement.executeUpdate();  if (insert == 0) {  return false;  }  connection.close();  return true;  } catch (Exception e) {  e.printStackTrace();  return false;  }  }  public Integer getCount\_in\_stok() {  return count\_in\_stok;  }  public void setCount\_in\_stok(Integer count\_in\_stok) {  this.count\_in\_stok = count\_in\_stok;  }  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  public String getProduct\_name() {  return product\_name;  }  public void setProduct\_name(String product\_name) {  this.product\_name = product\_name;  }  public String getBarcode() {  return barcode;  }  public void setBarcode(String barcode) {  this.barcode = barcode;  }  public Integer getCount() {  return count;  }  public void setCount(Integer count) {  this.count = count;  }  public Integer getPrice() {  return price;  }  public void setPrice(Integer price) {  this.price = price;  }  } |

|  |
| --- |
| **User.java** |
| package lab.course.model;  import java.nio.charset.StandardCharsets;  import java.security.MessageDigest;  import java.sql.\*;  import java.util.HashMap;  import java.util.Map;  import java.util.regex.Matcher;  import java.util.regex.Pattern;  public class User {  private Integer id;  private String login;  private String password;  private String email;  private String name;  private String surname;  private String phone;  private String role;  private static String URL = "jdbc:mariadb://localhost:3306/javaCourse";  private static String USERNAME = "user";  private static String PASSWORD = "qwer";  public User(Integer id, String login, String password, String email,  String name, String surname, String phone, String role) {  this.id = id;  this.login = login;  this.password = password;  this.email = email;  this.name = name;  this.surname = surname;  this.phone = phone;  this.role = role;  }  public User(Integer id, String login, String email,  String name, String surname, String phone, String role) {  this.id = id;  this.login = login;  this.email = email;  this.name = name;  this.surname = surname;  this.phone = phone;  this.role = role;  }  // проверяем на уникальность логин  public static boolean checkUniqueField(String field, String login) {  String SQL = "select \* from Users where " + field + " = ?";  Connection connection = null;  PreparedStatement preparedStatement = null;  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, login);  ResultSet resultSet = preparedStatement.executeQuery();  connection.close();  if (!resultSet.next()) {  connection.close();  return true;  }  connection.close();  } catch (Exception e) {  e.printStackTrace();  }  return false;  }  public static String getHashedPassword(String text) {  if (text == null) return null;  if (text.isEmpty()) return "";  text = text.trim();  MessageDigest md = null;  try {  md = MessageDigest.getInstance("SHA-256");  md.update(text.trim().getBytes("UTF-8"));  } catch (Exception e) {  e.printStackTrace();  }  if (md != null)  return new String(md.digest(), StandardCharsets.UTF\_8);  else return null;  }  public static User getUserByPhone(String phone) {  if (phone == null) return null;  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from Users where phone = ?";  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, phone);  ResultSet resultSet = preparedStatement.executeQuery();  User temp = null;  if (resultSet.next()) {  temp = new User(  resultSet.getInt("id"),  resultSet.getString("login"),  resultSet.getString("password"),  resultSet.getString("email"),  resultSet.getString("name"),  resultSet.getString("surname"),  resultSet.getString("phone"),  resultSet.getString("role"));  }  connection.close();  return temp;  } catch (Exception e) {  e.printStackTrace();  }  return null;  }  public static User getUserByLogin(String login) {  if (login == null) return null;  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = "select \* from Users where login = ?";  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, login);  ResultSet resultSet = preparedStatement.executeQuery();  User temp = null;  if (resultSet.next()) {  temp = new User(  resultSet.getInt("id"),  resultSet.getString("login"),  resultSet.getString("password"),  resultSet.getString("email"),  resultSet.getString("name"),  resultSet.getString("surname"),  resultSet.getString("phone"),  resultSet.getString("role"));  }  connection.close();  return temp;  } catch (Exception e) {  e.printStackTrace();  }  return null;  }  public static boolean updateUserParam(int id, HashMap<String, String> map) {  if (map == null || map.isEmpty() || id < 0) return false;  Connection connection = null;  PreparedStatement preparedStatement = null;  String SQL = null;  try {  Class.forName("org.mariadb.jdbc.Driver");  connection = DriverManager.getConnection(URL, USERNAME, PASSWORD);  connection.setAutoCommit(false);  for(Map.Entry<String, String> entry : map.entrySet()) {  SQL = "UPDATE `Users` SET `"+ entry.getKey() +"` = ? where `Users`.`id` = "  + String.valueOf(id);  preparedStatement = connection.prepareStatement(SQL);  preparedStatement.setString(1, entry.getValue());  int res = preparedStatement.executeUpdate();  if (res == 0) {  connection.rollback();  connection.close();  return false;  }  }  connection.commit();  connection.close();  } catch (Exception e) {  if (connection != null) {  try {  connection.rollback();  } catch (SQLException ex) {  throw new RuntimeException(ex);  }  }  e.printStackTrace();  }  return true;  }  // почта проверяется по регулярному выражению  public static boolean checkEmail(String str) {  if (str == null) return false;  String patterns =  "^[\_A-Za-z0-9-\\+]+(\\.[\_A-Za-z0-9-]+)\*@" +  "[A-Za-z0-9-]+(\\.[A-Za-z0-9]+)\*(\\.[A-Za-z]{2,})$";  Pattern pattern = Pattern.compile(patterns);  Matcher matcher = pattern.matcher(str);  return matcher.matches();  }  public Integer getId() {  return id;  }  public void setId(Integer id) {  this.id = id;  }  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;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  public String getSurname() {  return surname;  }  public void setSurname(String surname) {  this.surname = surname;  }  public String getPhone() {  return phone;  }  public void setPhone(String phone) {  this.phone = phone;  }  public String getRole() {  return role;  }  public void setRole(String role) {  this.role = role;  }  } |

**ПРИЛОЖЕНИЕ Б. Файлы .jsp и .css**

|  |
| --- |
| **AddProduct.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Добавить товар</title>  <script src="quagga.js"></script>  <script src="barcode.js"></script>  <style>  .block {  margin-left: auto;  margin-right: auto;  width: 30em;  }  input {  width: 30em;  }  body {  background: bisque;  }  .scanner {  position: relative;  }  /\* In order to place the tracking correctly \*/  canvas.drawing, canvas.drawingBuffer {  position: absolute;  padding-left: 0;  left: 0;  top: 0;  }  #scanner-container > video {  padding-left: 0;  width: 85%;  }  #scanner-container > canvas {  width: 85%;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <form action="/addProduct" method="post" class="block" >  <h1>Новый товар</h1>  <div> ${errorAddProduct} </div>  <div> ${successAddProduct}</div>  <p>Введите название</p>  <input type="text" name="productName" placeholder="Алебра 10-11 класс" required>  <p>Введите кол-во</p>  <input type="number" name="count" placeholder="123" required>  <p>Введите цену</p>  <input type="number" name="price" placeholder="123" required>  <p>Введите штрих-код</p>  <input type="text" id="barcode" name="barcode" placeholder="" required>  <br>  <label style="cursor:pointer" onclick="Quagga.stop()">Выключить</label><label style="cursor:pointer" onclick="startScanner();">Включить</label>  <div id="scanner-container" class="scanner" style="padding-left: 0;margin-left: 0;"></div>  <button type="submit" style="width: 14em;">Добавить</button>  <button type="reset" style="width: 14em;">Сброс</button>  </form>  <script>  startScanner();  </script>  </body>  </html> |

|  |
| --- |
| **AddUser.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Добавить пользователя</title>  <style>  .block {  margin-left: auto;  margin-right: auto;  width: 30em;  }  input {  width: 30em;  }  body {  background: bisque;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <form action="addUser" method="post" class="block">  <h1>Новый пользователь</h1>  <div> ${errorAddUser} </div>  <div> ${successAddUser}</div>  <p>Введите логин</p>  <input type="text" name="login" placeholder="login" required>  <table style="margin: 0;padding: 0;">  <tr>  <td>Введите пароль</td>  <td>Повторите пароль</td>  </tr>  <tr>  <td><input type="password" name="password" style="padding-left: 0;margin-left: 0; width:14em;"  placeholder="password" required></td>  <td><input type="password" name="passwordSecond" style="padding-left: 0;margin-left: 0;width:14em;"  placeholder="password" required></td>  </tr>  </table>  <p>Введите имя</p>  <input type="text" name="name" placeholder="Name" required>  <p>Введите фамилию</p>  <input type="text" name="surname" placeholder="Surname" required>  <p>Введите почту</p>  <input type="email" name="email" placeholder="you\_mail@gmail.com" required>  <p>Введите номер телефона</p>  <script src="https://cdn.jsdelivr.net/npm/jquery@3.2.1/dist/jquery.min.js" type="text/javascript"></script>  <script src="https://cdn.jsdelivr.net/npm/jquery.maskedinput@1.4.1/src/jquery.maskedinput.js"  type="text/javascript"></script>  <script src="js/jquery.maskedinput.min.js"></script>  <input type="text" placeholder="Телефон" name="phone" class="phone\_mask" required>  <script> $(".phone\_mask").mask("+7(999)999-99-99"); </script>  <p>Выберите уровень доступа</p>  <div style="text-align:left;">  <p><input style="width:auto" type="radio" name="roleInput" value="user" id="user" checked><label for="user">Рядовой  пользователь</label></p>  <p><input style="width:auto" type="radio" name="roleInput" value="salesman" id="salesman"><label for="salesman">Продавец</label>  </p>  <p><input style="width:auto" type="radio" name="roleInput" value="admin" id="admin"><label for="admin">Адимнистратор</label>  </p>  </div>  <button type="submit" style="padding-left: 0;margin-left: 0;width: 14em;">Добавить</button>  <button type="reset" style="padding-left: 0;margin-left: 0;width: 14em;">Сброс</button>  </form>  </body>  </html> |

|  |
| --- |
| **EditProduct.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Изменить товар</title>  <script src="quagga.js"></script>  <script src="barcode.js"></script>  <style>  .block {  margin-left: auto;  margin-right: auto;  width: 30em;  }  input {  width: 30em;  }  body {  background: bisque;  }  .scanner {  position: relative;  }  canvas.drawing, canvas.drawingBuffer {  position: absolute;  padding-left: 0;  left: 0;  top: 0;  }  #scanner-container > video {  padding-left: 0;  width: 85%;  }  #scanner-container > canvas {  width: 85%;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <h1>Изменяемый товар:</h1>  <h3>Id: ${selected\_product}</h3>  <h3>Наиманование: ${product\_name}</h3>  <h3>Кол-во в наличии: ${product\_count}</h3>  <h3>Цена: ${product\_price}</h3>  <h3>Штрих-код: ${product\_barcode}</h3>  <form action="/editProduct" method="post" class="block" >  <h1>Форма</h1>  <div> ${errorEditProduct} </div>  <div> ${successEditProduct}</div>  <p>Введите название</p>  <input type="text" name="productName" placeholder="Алебра 10-11 класс" >  <p>Введите кол-во</p>  <input type="number" name="count" placeholder="123" >  <p>Введите цену</p>  <input type="number" name="price" placeholder="123" >  <p>Введите штрих-код</p>  <input type="text" id="barcode" name="barcode" placeholder="" >  <br>  <label style="cursor:pointer" onclick="Quagga.stop()">Выключить</label><label style="cursor:pointer" onclick="startScanner();">Включить</label>  <div id="scanner-container" class="scanner" style="padding-left: 0;margin-left: 0;"></div>  <button type="submit" style="width: 14em;">Обновить</button>  <button type="reset" style="width: 14em;">Сброс</button>  </form>  <script>  startScanner();  </script>  </body>  </html> |

|  |
| --- |
| **EditUser.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Изменить пользователя</title>  <style>  body {  background: bisque;  }  table {  border-spacing: 10px;  border-collapse: separate;  border: 1px solid black;  }  td {  border: 1px solid black;  text-align: center;  }  .cntr {  margin-left: auto;  margin-right: auto;  width: auto;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <h1>Изменяемый пользователь:</h1>  <h3>Логин : ${user\_login}</h3>  <h3>Почта : ${user\_email}</h3>  <h3>Имя : ${user\_name}</h3>  <h3>Фамилия : ${user\_surname}</h3>  <h3>Номер телефона : ${user\_phone}</h3>  <h3>Привилегии: ${user\_role}</h3>  <div class="cntr" style="width: fit-content; text-align: center">  <form action="/editUser" method="post" >  <h1>Форма</h1>  <% if (request.getSession().getAttribute("errorEdit") != null) { %>  <div class="cntr" style="width: fit-content; background-color: red; text-align: center"> ${errorEdit}</div>  <%} %>  <% if (request.getSession().getAttribute("sucEdit") != null) { %>  <div class="cntr" style="width: fit-content; background-color: lawngreen; text-align: center"> ${sucEdit}</div>  <%} %>  <table>  <tr>  <td>Введите новый пароль</td>  <td>Повторите пароль</td>  </tr>  <tr>  <td><input type="password" name="passwordNew" placeholder="password" ></td>  <td><input type="password" name="passwordNewSecond" placeholder="password" ></td>  </tr>  </table>  <p>Изменить логин : <input type="text" name="login" placeholder="login" ></p>  <p>Изменить имя : <input type="text" name="name" placeholder="Name" ></p>  <p>Изменить фамилию : <input type="text" name="surname" placeholder="Surname" ></p>  <p>Изменить почту : <input type="email" name="email" placeholder="you\_mail@gmail.com" ></p>  <p>Изменить телефон :  <script src="https://cdn.jsdelivr.net/npm/jquery@3.2.1/dist/jquery.min.js" type="text/javascript"></script>  <script src="https://cdn.jsdelivr.net/npm/jquery.maskedinput@1.4.1/src/jquery.maskedinput.js" type="text/javascript"></script>  <script src="js/jquery.maskedinput.min.js"></script>  <input type="text" placeholder="Телефон" name="phone" class="phone\_mask">  <script> $(".phone\_mask").mask("+7(999)999-99-99"); </script></p>  <p>Выберите уровень доступа</p>  <div style="text-align:left;">  <p><input style="width:auto" type="radio" name="roleInput" value="user" id="user"><label for="user">Рядовой  пользователь</label></p>  <p><input style="width:auto" type="radio" name="roleInput" value="salesman" id="salesman"><label for="salesman">Продавец</label>  </p>  <p><input style="width:auto" type="radio" name="roleInput" value="admin" id="admin"><label for="admin">Адимнистратор</label>  </p>  </div>  <button type="submit" style="width:45%">Обновить данные</button>  <button type="reset" style="width:45%">Сброс</button>  </form>  </div>  </body>  </html> |

|  |
| --- |
| **ListProduct.jsp** |
| <%@ page import="lab.course.model.ListProducts" %>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Список товаров</title>  <style>  body {  background: bisque;  }  table {  border: 1px solid red;  }  td {  border: 1px solid red;  text-align: center;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <h1>Список Товаров</h1>  <table style="width: 100%; align-content: center;">  <tr>  <td>Id</td>  <td>Наименование</td>  <td>Штрих-код</td>  <td>Кол-во в наличии</td>  <td>Цена</td>  </tr>  <% if (ListProducts.getProducts() != null) {  for (var a : ListProducts.getProducts()) {  out.print("<tr >");  out.print("<td>" + a.getId() + "</td>");  out.print("<td>" + a.getProduct\_name() + "</td>");  out.print("<td>" + a.getBarcode() + "</td>");  out.print("<td>" + a.getCount\_in\_stok() + "</td>");  out.print("<td>" + a.getPrice() + "</td>");  out.print("</tr>");  }  } else {  out.print("Товары не найдены");  }  %>  </body>  </html>  </body>  </html> |

|  |
| --- |
| **ListUser.jsp** |
| <%@ page import="lab.course.model.ListUsers" %>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Список пользователей</title>  <style>  body {  background: bisque;  }  table {  border: 1px solid red;  }  td {  border: 1px solid red;  text-align: center;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <h1>Список пользователей</h1>  <table style="width: 100%; align-content: center;">  <tr>  <td>Id</td>  <td>Логин</td>  <td>Почта</td>  <td>Имя</td>  <td>Фамилия</td>  <td>Телефон</td>  <td>Роль</td>  </tr>  <% if (ListUsers.getUsers() != null) {  for (var a : ListUsers.getUsers()) {  out.print("<tr >");  out.print("<td>" + a.getId() + "</td>");  out.print("<td>" + a.getLogin() + "</td>");  out.print("<td>" + a.getEmail() + "</td>");  out.print("<td>" + a.getName() + "</td>");  out.print("<td>" + a.getSurname() + "</td>");  out.print("<td>" + a.getPhone() + "</td>");  out.print("<td>" + a.getRole() + "</td>");  out.print("</tr>");  }  } else {  out.print("Пользователи не найдены");  }  %>  </body>  </html> |

|  |
| --- |
| **Index.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Меню</title>  <style>  body {  background: bisque;  }  </style>  </head>  <body>  <% if (request.getSession().getAttribute("noUser") != null && (boolean)request.getSession().getAttribute("noUser")) { %>  <script>alert("Пользователь не найден")</script>  <% } %>  <jsp:include page="/header.jsp"/>  <h1>Администрирование:</h1>  <div style="display: flex; justify-content: center">  <div style="border: 3px solid black; display: flex; flex-direction: column; align-items: center;">  <div><form action="func" method="post">  <button name="choose" value="addUser"><h2>Добавить пользователя</h2></button>  </form></div>  <div><form action="func" method="post">  <button name="choose" value="addProduct"><h2>Добавить новый товар</h2></button>  </form></div>  </div>  <div style="border: 3px solid black; display: flex; flex-direction: column; align-items: center; ">  <div><form action="func" method="post" style="margin-left:0; padding-left: 0; display: flex; border-bottom: 3px solid black;">  <div style=""><button name="choose" value="editUser"><h2>Изменить данные пользователя</h2></button></div>  <div>Введите логин пользователя:<br><input type="text" name="login" placeholder="login" required></div>  </form></div>  <div><form action="func" method="post" style="margin-left:0; padding-left: 0; display: flex">  <div style=""><button name="choose" value="editProduct"><h2>Изменить товар</h2></button></div>  <div>Введите id товара:<br><input type="text" name="product" placeholder=" 0" required></div>  </form></div>  </div>  <div style="border: 3px solid black; display: flex; flex-direction: column; align-items: center;">  <div><form action="func" method="post">  <button name="choose" value="listUser"><h2>Список пользователей</h2></button>  </form></div>  <div><form action="func" method="post">  <button name="choose" value="listProduct"><h2>Список товаров</h2></button>  </form></div>  </div>  </div>  </body>  </html> |

|  |
| --- |
| **HandleBuyer.jsp** |
| <%@ page import="lab.course.model.ListProducts" %>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Данные покупателя</title>  <style>  \* {margin: 5px; padding: 5px;}  table {  border-spacing: 10px;  border-collapse: separate;  border: 1px solid red;  }  td {  border: 1px solid red;  text-align: center;  }  .flex-container {  width: 100%;  height: 300px;  display: flex;  flex-flow: row nowrap;  }  .left {  width: 30%;  flex-grow: 3;  }  .right {  flex-grow: 7;  }  body {  background: bisque;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <% if (request.getSession().getAttribute("err") != null) { %>  <script>alert("${err}")</script>  <%} %>  <div class="flex-container">  <div class="left">  <table>  <td>  <h4>По желанию введите телефон покупателя для привязки покупки </h4>  <form action="purchase" method="post">  <p><label>Телефон</label>  <script src="https://cdn.jsdelivr.net/npm/jquery@3.2.1/dist/jquery.min.js" type="text/javascript"></script>  <script src="https://cdn.jsdelivr.net/npm/jquery.maskedinput@1.4.1/src/jquery.maskedinput.js" type="text/javascript"></script>  <script src="js/jquery.maskedinput.min.js"></script>  <input type="text" placeholder="Телефон" name="buyer\_phone" class="phone\_mask">  <script> $(".phone\_mask").mask("+7(999)999-99-99"); </script></p>  <div style="text-align:left;">  <h4> Способ оплаты :</h4>  <p><input type="radio" name="payment" value="card" id="card"><label for="card">Банковской картой </label></p>  <p><input type="radio" name="payment" value="money" checked id="money"><label for="money">Наличкой </label></p>  <p><input type="radio" name="payment" value="agreement" id="agreement"><label for="agreement">Договор для юр. лиц </label></p>  </div>  <button type="submit">Оплатить</button>  </form>  </td>  </table>  </div>  <div class="right">  <table style="width: 100%;align-content: center;">  <tr>  <td>№</td>  <td>Продукт</td>  <td>Цена</td>  <td>Кол-во</td>  </tr>  <% var list = (ListProducts)request.getSession().getAttribute("list");  if (list != null) {  int i = 1; int sum = 0;  for (var a : list.getListProducts()) {  if (i % 2 == 0){ out.print("<tr><td>" + i++ + "</td>");}  else {out.print("<tr style=\"background: silver\"><td>" + i++ + "</td>");}  out.print("<td>" + a.getProduct\_name() + "</td>");  out.print("<td>" + a.getPrice() + "</td>");  out.print("<td style=\"width:70px;\">" + a.getCount() + "</td></tr>");  sum += a.getPrice() \* a.getCount();  }  out.print("<tr><td colspan=\"4\">Общая цена : " + sum + "</td></tr");  } %>  </table>  </div>  </div>  </body>  </html> |

|  |
| --- |
| **Payment.jps** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Оплата</title>  <style>  body { background: bisque; }  </style>  </head>  <body>  <p>  <% if(request.getSession().getAttribute("payment") != null) {  var pay = request.getSession().getAttribute("payment").toString();  if (pay.equals("card")) {  out.print("<label>Оплата по карте прошла</label>");  } else if (pay.equals("money")) {  out.print("<label>Покупка оплачена наличкой</label>");  } else if (pay.equals("agreement")) {  out.print("<label>Договор на покупку оформлен, и будет рассмотрен в скором времени</label>");  } else {  out.print("как-то оплатил");  }  } %>  </p>  <p><a href="checkBarcode">Вернуться к сканеру</a></p>  </body>  </html> |

|  |
| --- |
| **Index.jsp** |
| <%@ page import="lab.course.model.ListProducts" %>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <script src="quagga.js"></script>  <script src="barcode.js"></script>  <title>Сканер</title>  <style>  body {  background: bisque;  }  .scanner {  position: relative;  }  /\* In order to place the tracking correctly \*/  canvas.drawing, canvas.drawingBuffer {  padding-left: 0;  position: absolute;  left: 0;  top: 0;  }  \* {  margin: 5px;  padding: 5px;  }  table {  border-spacing: 10px;  border-collapse: separate;  border: 1px solid red;  }  td {  border: 1px solid red;  text-align: center;  }  .left {  width: 40%;  flex-grow: 3;  }  .right {  flex-grow: 7;  }  #scanner-container > video {  width: 100%;  height: 100%;  padding-left: 0;  }  #scanner-container > canvas {  width: 100%;  height: 100%;  }  .flex-container {  width: 100%;  height: 300px;  display: flex;  flex-flow: row nowrap;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <% if (request.getSession().getAttribute("errorCheck") != null) { %>  <script>alert("${errorCheck}")</script>  <%} %>  <div class="flex-container">  <div class="left">  <table>  <tr>  <td>  <form action="checkBarcode" method="post">  <h3>Отсканируйте товар</h3>  <div>  ${errorRead}  </div>  <p>  <input type="text" name="barcode" id="barcode" placeholder="Штрих код" required>  <button id="barcode-submit" type="submit">Ввод</button>  </p>  </form>  </td>  </tr>  <tr>  <td>  <label style="cursor:pointer" onclick="Quagga.stop()">Выключить</label><label style="cursor:pointer" onclick="startScanner();">Включить</label>  <div id="scanner-container" class="scanner" style="padding-left: 0"></div>  </td>  </tr>  </table>  </div>  <div class="right">  <form action="buy" method="post" id="basket\_post">  <button type="submit">Перейти к оплате</button>  </form>  <table style="width: 100%; align-content: center;">  <tr>  <td>№</td>  <td>На складе</td>  <td>Продукт</td>  <td>Цена</td>  <td>Кол-во</td>  <td>Флажок</td>  </tr>  <%  var list = (ListProducts) request.getSession().getAttribute("list");  if (list != null) {  int i = 1;  for (var a : list.getListProducts()) {  if (i % 2 == 0){ out.print("<tr><td>" + i++ + "</td>");}  else {out.print("<tr style=\"background: silver\"><td>" + i++ + "</td>");}  out.print("<td style=\"width:50px;\">" + a.getCount\_in\_stok() + "</td>");  out.print("<td>" + a.getProduct\_name() + "</td>");  out.print("<td>" + a.getPrice() + "</td>");  out.print("<td style=\"width:50px;\"><input type=number form=\"basket\_post\" required value=" + a.getCount() + " name=\"basket" + a.getId() + "\" min=\"1\" max=\"1000\" " +  "onkeyup=\"if(this.value>999){this.value='999';}else if(this.value<1){this.value='1';}\"></td>");  out.print("<td style=\"width:50px;\"><input type=\"checkbox\" form=\"check\_del\" name=\"delete\" value=\"" + a.getId() + "\"></td></tr>");  }  }  %>  </table>  <table style="width: 100%; align-content: center">  <form action="remove" id="check\_del" method="post">  <td>  <button type="submit">Убрать выбранные товары</button>  </td>  </form>  </table>  </div>  </div>  <script>  startScanner();  </script>  </body>  </html> |

|  |
| --- |
| **Orders.jsp** |
| <%@ page import="lab.course.model.Orders" %>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Заказы</title>  <style>  body {  background: bisque;  }  table {  border: 1px solid red;  }  td {  border: 1px solid red;  text-align: center;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <h1>Список заказов</h1>  <table style="width: 100%; align-content: center;">  <tr>  <td>Дата заказа</td>  <td>Номер заказа</td>  <td>Сумма заказа</td>  </tr>  <% Orders orders = (Orders) request.getSession().getAttribute("orders");  if (orders != null) {  if (orders.getOrders() != null) {  for (var a : orders.getOrders()) {  out.print("<tr style=\"cursor:pointer;\" method=\"post\" data-href=\"/order?order\_id="+ a.getPurchase\_id() + "\">");  out.print("<td>" + a.getDate() + "</td>");  out.print("<td>" + a.getPurchase\_id() + "</td>");  out.print("<td>" + a.getTotal\_price() + "</td>");  out.print("</tr>");  }  } else {  out.print("Вы пока не делали заказов<br>");  }  } else {  }  %>  <script src="https://code.jquery.com/jquery-3.3.1.min.js"></script>  <script>  $(document).ready(function($) {  $('\*[data-href]').on('click', function() {  window.location = $(this).data("href");  });  });  </script>  </table>  </body>  </html> |

|  |
| --- |
| **Order.jsp** |
| <%@ page import="java.util.ArrayList" %>  <%@ page import="lab.course.model.Order" %>  <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Заказ</title>  <style>  body {  background: bisque;  }  table {  border: 1px solid red;  }  td {  border: 1px solid red;  text-align: center;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <% @SuppressWarnings("unchecked")  ArrayList<Order> orderInfo = (ArrayList<Order>) request.getSession().getAttribute("orderInfo");  if (orderInfo != null && !orderInfo.isEmpty()) { %>  <label>Список товаров:</label>  <table style="width: 100%; align-content: center;">  <tr>  <td>Название</td>  <td>Кол-во</td>  <td>Цена</td>  </tr>  <%  for (var a : orderInfo) {  out.print("<tr>");  out.print("<td>"+a.getName()+"</td>");  out.print("<td>"+a.getCount()+"</td>");  out.print("<td>"+a.getPrice()+"</td>");  out.print("</tr>");  }  out.print("</table>");  out.print("<label>Сумма заказа: " + orderInfo.get(0).getTotal\_price()+ "</label>");  } else { %>  <label>${errorOrder}</label><br>  <label>Нет данных по заказу</label>  <%} %>  </body>  </html> |

|  |
| --- |
| **Index.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Title</title>  <style>  body {  background: bisque;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <form action="/orders" >  <button><h2>Просмотр завершенных покупок</h2></button>  </form>  </body>  </html> |

|  |
| --- |
| **Header.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <style>  \* {margin: 5px; padding: 5px;}  </style>  <header style="width: 100%; position:sticky; top: 10px; overflow: hidden; background-color: azure">  <nav>  <% var role = request.getSession().getAttribute("role").toString();  var name = request.getSession().getAttribute("name").toString();  if (role.equals("admin")) { out.print("<label>Администратор : " + name + "</label>"); }  else if (role.equals("salesman")) { out.print("<label>Продавец : " + name + "</label>"); }  else if (role.equals("user")) { out.print("<label>Пользователь : " + name + "</label>"); }%>  <form style="right: auto; float: right" action="logout" ><button>Выйти</button></form>  <form style="right: auto; float: right" action="/data" method="post"><button>Аккаунт</button></form>  <form style="right: auto; float: right" action="/login" method="get"><button>Меню</button></form>  <form style="right: auto; float: right" action="/" method="post"><button>На главную</button></form>  </nav>  </header>  <br> |

|  |
| --- |
| **Login.jsp** |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" isELIgnored="false" %>  <!DOCTYPE html>  <html>  <head>  <title>Вход</title>  <style>  .block {  margin-left: auto;  margin-right: auto;  width: 20em;  }  \* {  margin: 5px;  padding: 5px;  }  body {  background: bisque;  }  </style>  </head>  <body>  <% String id = (String) request.getSession().getAttribute("role");  if (id != null) { %>  <jsp:include page="header.jsp"/>  <%}%>  <a href="/">На главную</a>  <form action="login" method="post" class="block">  <h1>Вход в систему</h1>  <div>  ${errorMessage}  </div>  <p>Введите логин</p>  <input style="width:100%;" type="text" name="login" placeholder="login" required>  <p>Введите пароль</p>  <input style="width:100%;" type="password" name="password" placeholder="password" required>  <p><button style="width:40%;" type="submit">Вход</button><input form="reg" style="width:40%; float:right;" type="submit" value="Регистрация"/> </p>  </form>  <form id="reg" action="register.jsp" class="block"> </form>  </body>  </html> |

|  |
| --- |
| **Register.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Register</title>  <style>  .block {  margin-left: auto;  margin-right: auto;  width: 30em;  }  \* {margin: 5px; padding: 5px;}  input {  width:30em;  }  body {  background: bisque;  }  </style>  </head>  <body>  <a href="index.jsp">На главную</a>  <form action="register" method="post" class="block">  <h1>Регистрация</h1>  <div> ${errorInput} </div>  <div> ${success}</div>  <p>Введите логин</p>  <input type="text" name="login" placeholder="login" required>  <table style="margin: 0;padding: 0;">  <tr>  <td>Введите пароль</td>  <td>Повторите пароль</td>  </tr>  <tr>  <td><input type="password" name="password" style="padding-left: 0;margin-left: 0; width:14em;" placeholder="password" required></td>  <td><input type="password" name="passwordSecond" style="padding-left: 0;margin-left: 0; width:14em;" placeholder="password" required></td>  </tr>  </table>  <p>Введите имя</p>  <input type="text" name="name" placeholder="Name" required>  <p>Введите фамилию</p>  <input type="text" name="surname" placeholder="Surname" required>  <p>Введите почту</p>  <input type="email" name="email" placeholder="you\_mail@gmail.com" required>  <p>Введите номер телефона</p>  <script src="https://cdn.jsdelivr.net/npm/jquery@3.2.1/dist/jquery.min.js" type="text/javascript"></script>  <script src="https://cdn.jsdelivr.net/npm/jquery.maskedinput@1.4.1/src/jquery.maskedinput.js" type="text/javascript"></script>  <script src="js/jquery.maskedinput.min.js"></script>  <input type="text" placeholder="Телефон" name="phone" class="phone\_mask" required>  <script> $(".phone\_mask").mask("+7(999)999-99-99"); </script>  <p><button type="submit" style="padding-left: 0;margin-left: 0; width: 14em;">Регистрация</button>  <button type="reset" style="padding-left: 0;margin-left: 0;width: 14em;">Сброс</button> </p>  </form>  </body>  </html> |

|  |
| --- |
| **User\_profile.jsp** |
| <%@ page contentType="text/html;charset=UTF-8" language="java" %>  <html>  <head>  <title>Смена данных</title>  <style>  table {  border-spacing: 10px;  border-collapse: separate;  border: 1px solid black;  }  td {  border: 1px solid black;  text-align: center;  }  .cntr {  margin-left: auto;  margin-right: auto;  width: auto;  }  body {  background: bisque;  }  </style>  </head>  <body>  <jsp:include page="/header.jsp"/>  <h2>Логин : ${login}</h2>  <h2>Почта : ${email}</h2>  <h2>Имя : ${name}</h2>  <h2>Фамилия : ${surname}</h2>  <h2>Номер телефона : ${phone}</h2>  <form action="/account" method="post">  <h1>Форма</h1>  <% if (request.getSession().getAttribute("error") != null) { %>  <div class="cntr" style="width: fit-content; background-color: red; text-align: center"> ${error}</div>  <%} %>  <% if (request.getSession().getAttribute("suc") != null) { %>  <div class="cntr" style="width: fit-content; background-color: lawngreen; text-align: center"> ${suc}</div>  <%} %>  <table>  <tr>  <td>Введите старый пароль</td>  <td>Введите новый пароль</td>  <td>Повторите пароль</td>  </tr>  <tr>  <td><input type="password" name="passwordOld" placeholder="password" ></td>  <td><input type="password" name="passwordNew" placeholder="password" ></td>  <td><input type="password" name="passwordNewSecond" placeholder="password" ></td>  </tr>  </table>  <p>Изменить логин : <input type="text" name="login" placeholder="login" ></p>  <p>Изменить имя : <input type="text" name="name" placeholder="Name" ></p>  <p>Изменить фамилию : <input type="text" name="surname" placeholder="Surname" ></p>  <p>Изменить почту : <input type="email" name="email" placeholder="you\_mail@gmail.com" ></p>  <p>Изменить телефон :  <script src="https://cdn.jsdelivr.net/npm/jquery@3.2.1/dist/jquery.min.js" type="text/javascript"></script>  <script src="https://cdn.jsdelivr.net/npm/jquery.maskedinput@1.4.1/src/jquery.maskedinput.js" type="text/javascript"></script>  <script src="js/jquery.maskedinput.min.js"></script>  <input type="text" placeholder="Телефон" name="phone" class="phone\_mask">  <script> $(".phone\_mask").mask("+7(999)999-99-99"); </script></p>  <p><button type="submit">Обновить данные</button>  <button type="reset">Сброс</button> </p>  </form>  </body>  </html> |

|  |
| --- |
| **Barcode.js** |
| function order\_by\_occurrence(arr) {  var counts = {};  arr.forEach(function(value){  if(!counts[value]) {  counts[value] = 0;  }  counts[value]++;  });  return Object.keys(counts).sort(function(curKey,nextKey) {  return counts[curKey] < counts[nextKey];  });  }  function startScanner() {  Quagga.init({  inputStream: {  name: "Live",  type: "LiveStream",  target: document.querySelector('#scanner-container'),  constraints: {  aspectRatio: 4/3,  facingMode: "environment"  }  },  numOfWorkers: 8,  frequency: 'full',  locator: {  patchSize: 'medium',  halfSample: true,  },  decoder: {  readers: [  //"code\_128\_reader",  "ean\_reader",  "ean\_8\_reader"  //"code\_39\_reader",  //"code\_39\_vin\_reader",  //"codabar\_reader",  //"upc\_reader",  //"upc\_e\_reader",  //"i2of5\_reader"  ],  debug: {  showCanvas: true,  showPatches: true,  showFoundPatches: true,  showSkeleton: true,  showLabels: true,  showPatchLabels: true,  showRemainingPatchLabels: true,  boxFromPatches: {  showTransformed: true,  showTransformedBox: true,  showBB: true  }  }  },  }, function (err) {  if (err) { console.log(err); return }  Quagga.initialized = true;  Quagga.start();  });  Quagga.onProcessed(function (result) {  var drawingCtx = Quagga.canvas.ctx.overlay,  drawingCanvas = Quagga.canvas.dom.overlay;  if (result) {  if (result.boxes) {  drawingCtx.clearRect(0, 0, parseInt(drawingCanvas.getAttribute("width")), parseInt(drawingCanvas.getAttribute("height")));  result.boxes.filter(function (box) {  return box !== result.box;  }).forEach(function (box) {  Quagga.ImageDebug.drawPath(box, {x: 0, y: 1}, drawingCtx, {color: "green", lineWidth: 2});  });  }  if (result.box) {  Quagga.ImageDebug.drawPath(result.box, {x: 0, y: 1}, drawingCtx, {color: "#00F", lineWidth: 2});  }  if (result.codeResult && result.codeResult.code) {  Quagga.ImageDebug.drawPath(result.line, {x: 'x', y: 'y'}, drawingCtx, {color: 'red', lineWidth: 3});  }  }  });  var last\_result = [];  if (Quagga.initialized == undefined) {  Quagga.onDetected(function(result) {  var last\_code = result.codeResult.code;  last\_result.push(last\_code);  if (last\_result.length > 30) {  code = order\_by\_occurrence(last\_result)[0];  last\_result = [];  // Quagga.stop();  document.getElementById("barcode").value = code;  document.getElementById("barcode-submit").click();  }  });  }  } |

|  |
| --- |
| **Index.js** |
| <%@ page contentType="text/html; charset=UTF-8" pageEncoding="UTF-8" isELIgnored="false" %>  <!DOCTYPE html>  <html>  <head>  <title>Главная страница</title>  <style>  body {  \* {margin: 5px; padding: 5px;}  background: bisque;:w  }  </style>  </head>  <body>  <% String id = (String) request.getSession().getAttribute("role");  if (id != null) { %>  <jsp:include page="header.jsp"/>  <%}%>  <center>  <h1>Главная страница</h1>  <h2>Информационная система для управления<br>  клиентской базой данных и автоматизаци процесса продаж</h2>  <h3>Разработал студент группы ИКПИ-11 Дунаев В.Е.</h3>  <button><a href="/login">Войти</a></button>  </center>  </body>  </html> |

## **ПРИЛОЖЕНИЕ В. Файлы конфигурации**

|  |
| --- |
| **web.xml** |
| <?xml version="1.0" encoding="UTF-8"?>  <web-app xmlns="https://jakarta.ee/xml/ns/jakartaee"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee https://jakarta.ee/xml/ns/jakartaee/web-app\_5\_0.xsd"  version="5.0">  <filter>  <filter-name>Login Filter</filter-name>  <filter-class>lab.course.controller.auth.LoginFilter</filter-class>  </filter>  <filter-mapping>  <filter-name>Login Filter</filter-name>  <servlet-name>Login</servlet-name>  </filter-mapping>  </web-app> |

|  |
| --- |
| **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 https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>lab</groupId>  <artifactId>course</artifactId>  <version>1.0-SNAPSHOT</version>  <name>course</name>  <packaging>war</packaging>  <properties>  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  <maven.compiler.target>11</maven.compiler.target>  <maven.compiler.source>11</maven.compiler.source>  <junit.version>5.9.2</junit.version>  </properties>  <dependencies>  <dependency>  <groupId>org.mariadb.jdbc</groupId>  <artifactId>mariadb-java-client</artifactId>  <version>3.0.7</version>  </dependency>  <dependency>  <groupId>jakarta.platform</groupId>  <artifactId>jakarta.jakartaee-web-api</artifactId>  <version>9.1.0</version>  <scope>provided</scope>  </dependency>  <dependency>  <groupId>org.junit.jupiter</groupId>  <artifactId>junit-jupiter-api</artifactId>  <version>${junit.version}</version>  <scope>test</scope>  </dependency>  <dependency>  <groupId>org.junit.jupiter</groupId>  <artifactId>junit-jupiter-engine</artifactId>  <version>${junit.version}</version>  <scope>test</scope>  </dependency>  </dependencies>  <build>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-war-plugin</artifactId>  <version>3.3.2</version>  </plugin>  </plugins>  </build>  </project> |