

# СОДЕРЖАНИЕ

Введение.....	3
1. Анализ и моделирование предметной области программного средства.....	4
1.1. Описание предметной области .....	4
1.2. Разработка функциональной модели предметной области .....	4
1.3. Анализ требований к разрабатываемому программному средству .....	6
1.4. Спецификация функциональных требований .....	7
1.4. Разработка информационной модели предметной области .....	8
2. Проектирование и конструирование программного средства .....	10
2.1. Постановка задачи.....	10
2.2. Архитектурные решения .....	10
2.3. Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства.....	13
2.4. Проектирование пользовательского интерфейса.....	17
2.5. Обоснование выбора компонентов и технологий для реализации программного средства.....	23
3. Тестирование и проверка работоспособности программного средства.....	26
4. Руководство по развертыванию и использованию программного средства .....	28
Заключение .....	43
Список использованных источников .....	44

## ВВЕДЕНИЕ

Важнейшая задача компьютерных систем – хранение и обработка данных. Для её решения были предприняты усилия, которые привели к появлению в конце 60-х – начале 70-х годов специализированного программного обеспечения – систем управления базами данных (database management systems). СУБД позволяют структурировать, систематизировать и организовать данные для их компьютерного хранения и обработки. Невозможно представить себе деятельность современного предприятия или учреждения без использования профессиональных СУБД. Несомненно, они составляют фундамент информационной деятельности во всех сферах – начиная с производства и заканчивая финансами и телекоммуникациями [1].

Курсовой проект посвящен изучению теории и практики разработки и проектирования информационных систем с использованием баз данных. В данном курсовом проекте объектом исследования является среда WEB, как платформа приложений информационных систем.

Предметом исследования являются методы разработки интернет-магазина. Целью курсового проекта является разработка интернет-магазина косметики.

Курсовой проект предполагает выполнение следующих задач:

- проанализировать и описать предметную область;
- спроектировать и сконструировать систему интернет-магазина косметики;
- проверить работоспособность полученного приложения;
- описать руководство по развертыванию и использованию программного средства.

В системе проверки на уникальность «Антиплагиат» работа показала результат 77.87%.

# **1. АНАЛИЗ И МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ ПРОГРАММНОГО СРЕДСТВА**

## **1.1. Описание предметной области**

Интернет-магазин (англ. online shop или e-shop) — сайт, торгующий товарами посредством сети Интернет. Позволяет пользователям онлайн, в своём браузере или через мобильное приложение, сформировать заказ на покупку, выбрать способ оплаты и доставки заказа, оплатить заказ. При этом продажа товаров осуществляется дистанционным способом и она накладывает ограничения на продаваемые товары. Так, в некоторых странах имеется запрет на интернет-торговлю алкоголем, оружием, ювелирными изделиями и другими товарами (к примеру, в России запрещена дистанционная продажа алкоголя и других товаров, свободная реализация которых запрещена или ограничена).

Когда онлайн-магазин настроен на то, чтобы позволить компаниям покупать у других компаний, этот процесс называется онлайн-магазинами бизнес для бизнеса (B2B). Типичный интернет-магазин позволяет клиенту просматривать ассортимент продуктов и услуг фирмы, просматривать фотографии или изображения продуктов, а также информацию о технических характеристиках продукта и ценах.

## **1.2. Разработка функциональной модели предметной области**

Отобразим функциональную модель предметной области с использованием методологии SADT. SADT (Structured Analysis and Design Technique) представляет собой методологию анализа и проектирования систем.

Данную методологию целесообразно применять на ранних этапах жизненного цикла, для того чтобы можно было рассмотреть систему до ее воплощения. Методология SADT дает возможность сократить дорогостоящие ошибки на ранних этапах разработки системы, улучшить контакт между пользователями и разработчиками, сгладить переход от анализа к проектированию. Методология SADT – нотация IDEF0 [2].

Разработанная модель представлена на рисунке 1.1.

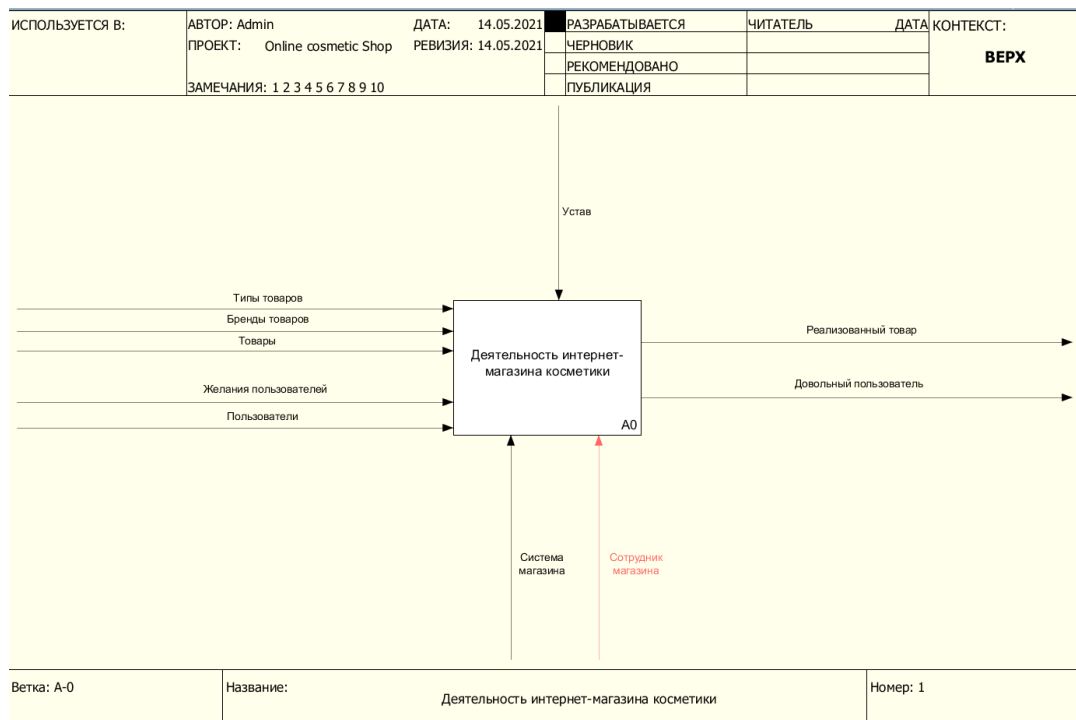


Рисунок 1.1 – Функциональная модель предметной области

Декомпозиция модели состоит из пяти процессов: заполнение справочников магазина, заполнение каталога товаров, регистрация пользователей, принятие заказов от пользователей, реализация товара.

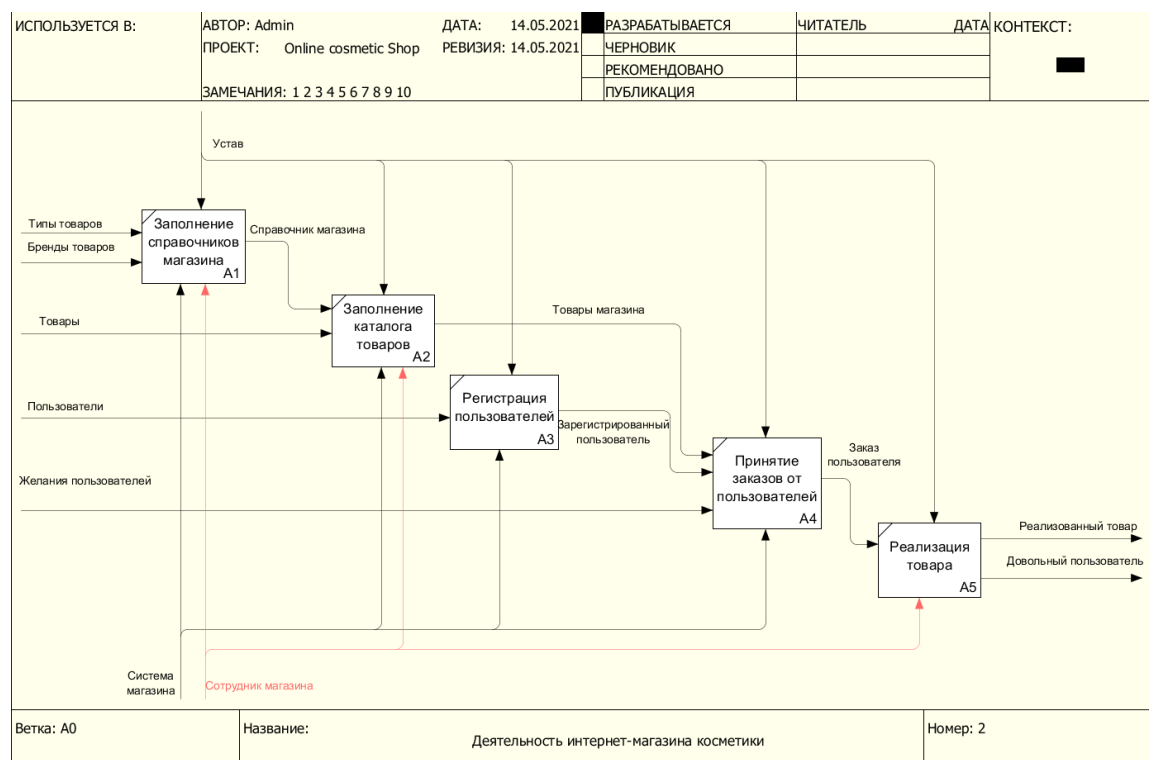


Рисунок 1.2 – Декомпозиция функциональной модели

### 1.3. Анализ требований к разрабатываемому программному средству

Анализ требований подразумевает их очистку, гарантирующую, что требования понимают все заинтересованные лица, а также тщательное исследование требований на предмет ошибок, пробелов и других недостатков. Кроме того, анализ включает разбиение высокоуровневых требований на более детальные, создание прототипов, анализ осуществимости и согласование приоритетов. Цель анализа — достаточно качественно и подробно описать требования, позволяющие менеджерам реалистично оценить все затраты на проект, а техническому персоналу — начать проектирование, разработку и тестирование [3].

Аналитик совместно с разработчиками должен определить, насколько возможно реализовать каждое требование при разумных затратах и с приемлемой производительностью в предполагаемой среде. Это позволяет заинтересованным лицам понять риски, связанные с реализацией каждого требования, включая конфликты с другими требованиями, зависимость от внешних факторов и препятствия технического характера. Технически нереализуемые или очень дорогие в реализации требования можно попытаться упростить и в таком виде включить в проект для достижения бизнес-целей.

Полученные общие требования на курсовой проект. Программное средство следует разработать в архитектуре web-приложение с базой данных с использованием объектно-ориентированного языка программирования Java, современных технологий и фреймворков. В рамках работы должны быть представлены: разработка и использование собственной иерархии классов, реализация не менее 2-х паттернов проектирования, сокрытие данных (инкапсуляция), перегрузка методов, переопределение методов, параметризованные классы (шаблоны), абстрактные типы данных (интерфейсы, абстрактные классы), передача параметров по ссылке и по значению, статические методы, обработка исключительных ситуаций.

**Уровни архитектуры:** приложение должно быть распределено по 2-м отдельным серверам:

**Сервер СУБД** – СУБД для размещения *базы данных* курсового проекта выбирается студентом самостоятельно.

**Сервер Приложений** – используется для размещения “серверной” части приложения, представленной *Моделью* на основе ORM технологии (Hibernate/JPA), *Бизнес-логикой* приложения на основе технологий jsp+servlet или иного MVC фреймворка для разработки веб-приложений. По согласованию с руководителем разрешается использовать фреймворки для других платформ: Ruby, .Net, Python, JavaScript.

В качестве языка программирования будет использован язык java в связке с SpringFramework.

## 1.4. Спецификация функциональных требований

Хорошим вариантом для предоставления пользователю визуального представления функциональных требований будет предоставление диаграммы вариантов использования.

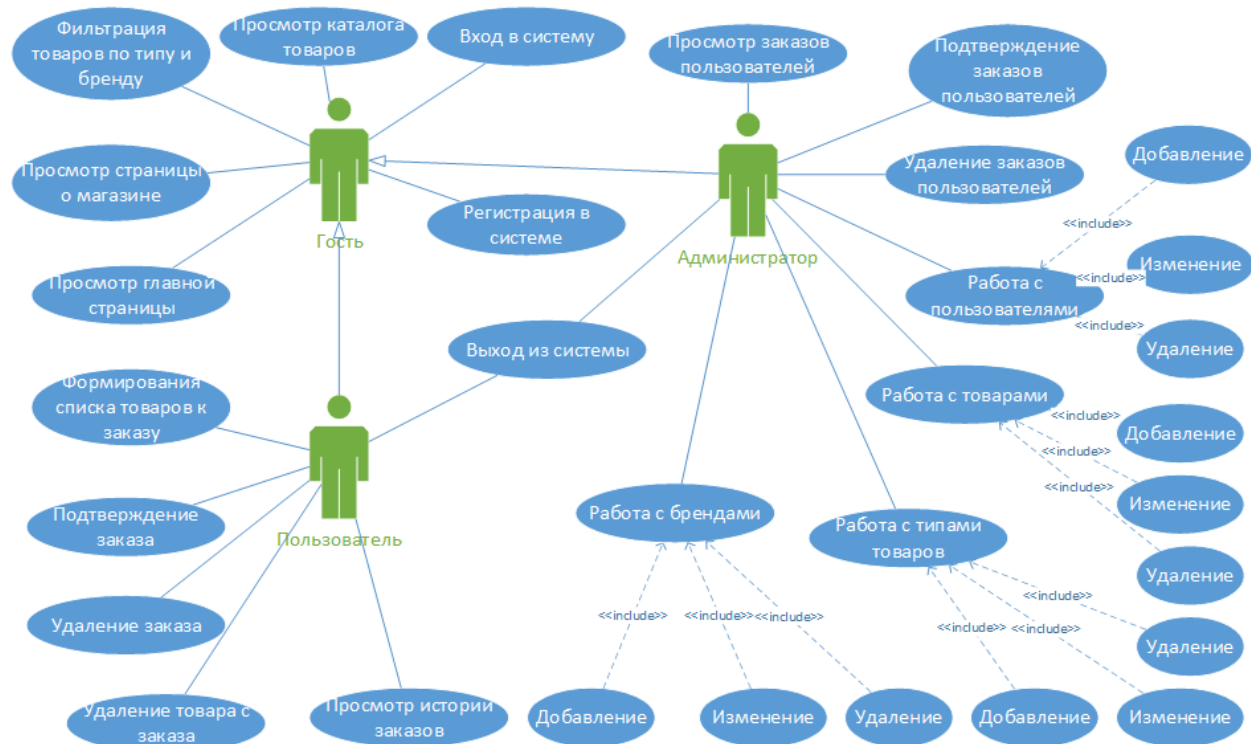


Рисунок 1.3 – Диаграмма вариантов использования

Как видно из диаграммы вариантов использования, пользоваться системой будут пользователи трех ролей, а точнее гость, пользователь (зарегистрированный гость) и администратор. Гость может только просматривать информацию на сервисе (главную страницу, страницу о нас, каталог товаров с возможностью фильтрации по типу и бренду), а также войти или зарегистрироваться в системе.

Когда гость входит в систему, он становится пользователем. Теперь он может не только делать все то же, что и раньше, но и добавлять товары себе в корзину и удалять их. После окончания заполнения корзины пользователь может подтвердить заказ или удалить его, после чего ждут подтверждение администратора и будет просматривать этот заказ через историю.

Администратор же имеет возможность просматривать все заказы пользователей с выводом списка товаров заказов, подтверждать или удалять эти заказы. Также, поскольку он администратор, то имеет полный доступ к данным таблиц, и может добавлять, изменять и удалять данные таких таблиц как типы товаров, бренды, товары, пользователи.

#### 1.4. Разработка информационной модели предметной области

Построение информационной модели предметной области предполагает выделение сущностей, их атрибутов и первичных ключей, идентификацию связей между сущностями. Общепринятым видом графического изображения реляционной модели данных является ER-диаграмма, на которой сущности изображаются прямоугольниками, соединенные между собой связями. Такое графическое представление облегчает восприятие структуры базы данных по сравнению с текстовым описанием [4].

Ниже показаны все сущности системы с их полями.

Статус

Код

Название

Элемент заказа

Код

Товар

Количество

Заказ

Тип товара

Код

Название

Заказ

Код

Дата

Время

Пользователь

Статус

Товар

Код

Описание

Ссылка на изображение

Название

Цена

Бренд

Тип

Пользователь

Код

Почта

Логин

Пароль

Фино

Дата рождения

Телефон

Роль

Роль

Код  
 Название  
 Бренд  
 Код  
 Название

При создании базы данных по текущим сущностям, будем иметь следующие отношения таблиц с указанием размера полей

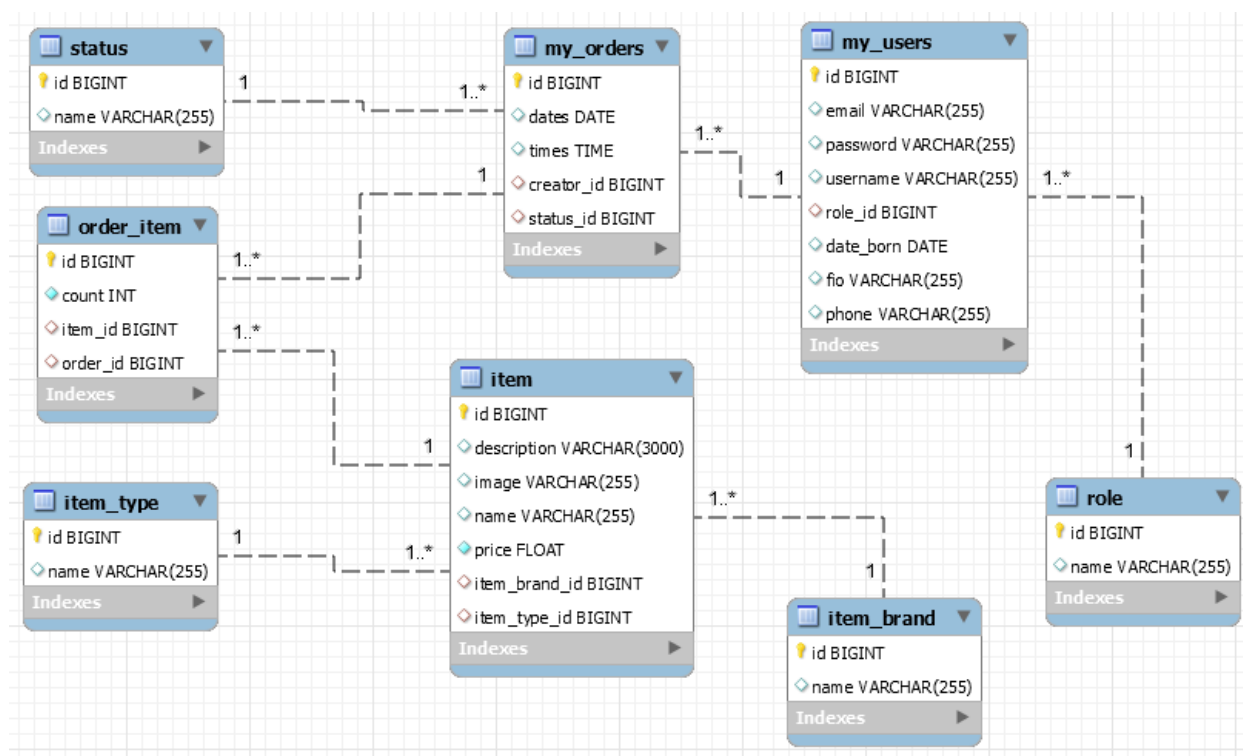


Рисунок 1.4 – Диаграмма базы данных



## 2. ПРОЕКТИРОВАНИЕ И КОНСТРУИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

### 2.1. Постановка задачи

Для достижения цели данного курсового проекта, следует выделить следующие задачи, которые необходимо решить:

- Выбрать архитектуру будущего приложения
- Создать базу данных по информационной модели предыдущего раздела
- Описать алгоритмы, реализующие бизнес-логику приложения
- Спроектировать пользовательский интерфейс.
- Провести тестирование полученного приложения.
- Описать руководство по развертыванию и использованию программного средства.

Кроме того, разрабатываемое приложение должно следующим требованиям

- проектируемая база данных должна отвечать требованиям надёжности, минимальной избыточности, целостности данных
- содержать не менее 5 сущностей и должна быть приведена к третьей нормальной форме
- реализовать приложение необходимо при помощи языка программирования Java и Базы данных SQL.

### 2.2. Архитектурные решения

Поскольку разработка ведется на основе фреймворка спринг, разумным будет использовать архитектуру Spring MVC.

Фреймворк **Spring MVC** обеспечивает архитектуру паттерна Model — View — Controller (Модель — Отображение (далее — Вид) — Контроллер) при помощи слабо связанных готовых компонентов. Паттерн MVC разделяет аспекты приложения (логику ввода, бизнес-логику и логику UI), обеспечивая при этом свободную связь между ними [5].

**Model** (Модель) инкапсулирует (объединяет) данные приложения, в целом они будут состоять из POJO («Старых добрых Java-объектов», или бинов).

**View** (Отображение, Вид) отвечает за отображение данных Модели, — как правило, генерируя HTML, которые мы видим в своём браузере.

**Controller** (Контроллер) обрабатывает запрос пользователя, создаёт соответствующую Модель и передаёт её для отображения в Вид.

Вся логика работы Spring MVC построена вокруг **DispatcherServlet**, который принимает и обрабатывает все HTTP-запросы (из UI) и ответы на них. Рабочий процесс обработки запроса DispatcherServlet'ом проиллюстрирован на следующем изображении:

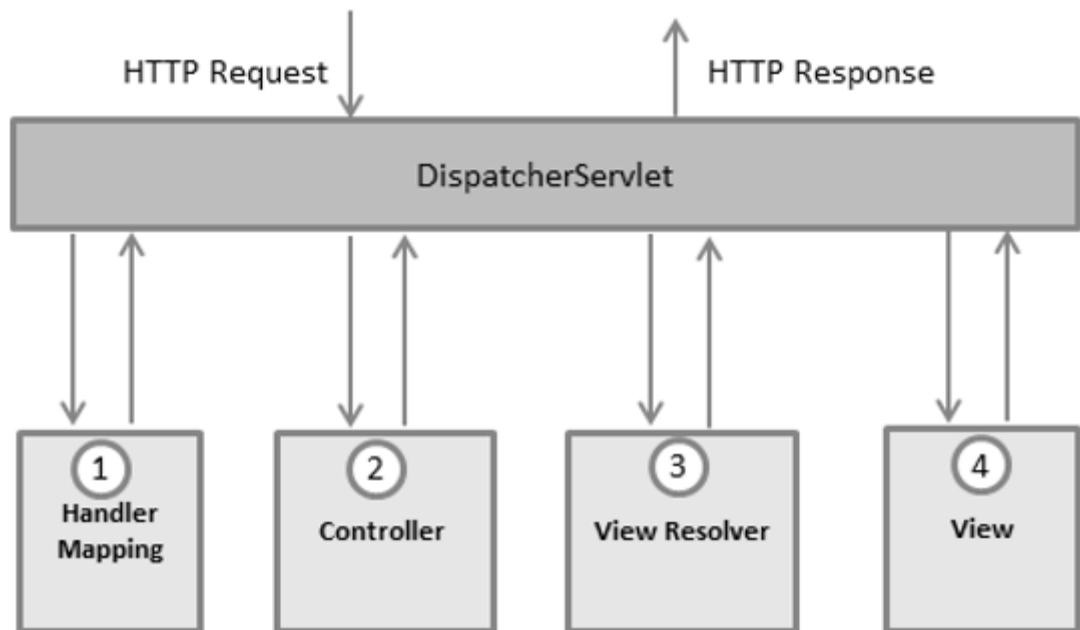


Рисунок 2.1 – DispatcherServlet

Ниже приведена последовательность событий, соответствующая входящему HTTP-запросу:

- После получения HTTP-запроса DispatcherServlet обращается к интерфейсу **HandlerMapping**, который определяет, какой Контроллер должен быть вызван, после чего, отправляет запрос в нужный Контроллер.
- Контроллер принимает запрос и вызывает соответствующий служебный метод, основанный на GET или POST. Вызванный метод определяет данные Модели, основанные на определённой бизнес-логике и возвращает в DispatcherServlet имя Вида (View).
- При помощи интерфейса **ViewResolver** DispatcherServlet определяет, какой Вид нужно использовать на основании полученного имени.
- После того, как Вид (View) создан, DispatcherServlet отправляет данные Модели в виде атрибутов в Вид, который в конечном итоге отображается в браузере.

Все вышеупомянутые компоненты, а именно, **HandlerMapping**, **Controller** и **ViewResolver**, являются частями интерфейса **WebApplicationContext** extends **ApplicationContext**, с некоторыми дополнительными особенностями, необходимыми для создания web-приложений.

DispatcherServlet отправляет запрос контроллерам для выполнения определённых функций. Аннотация **@Controller** указывает, что конкретный класс является контроллером. Аннотация **@RequestMapping** используется для мапинга (связывания) с URL для всего класса или для конкретного метода обработчика.

**@Controller**

```

@RequestMapping("/hello")
public class HelloController {
    @RequestMapping(method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        return "hello";
    }
}

```

Аннотация `Controller` определяет класс как Контроллер Spring MVC. В первом случае, `@RequestMapping` указывает, что все методы в данном Контроллере относятся к URL-адресу `"/hello"`. Следующая аннотация `@RequestMapping(method = RequestMethod.GET)` используется для объявления метода **`printHello()`** как дефолтного метода для обработки HTTP-запросов GET (в данном Контроллере). Вы можете определить любой другой метод как обработчик всех POST-запросов по данному URL-адресу.

Вы можете написать вышеуказанный Контроллер по-другому, указав дополнительные атрибуты для аннотации `@RequestMapping` следующим образом:

```

@Controller
public class HelloController {
    @RequestMapping(value = "/hello", method = RequestMethod.GET)
    public String printHello(ModelMap model) {
        model.addAttribute("message", "Hello Spring MVC Framework!");
        return "hello";
    }
}

```

Атрибут `«value»` указывает URL, с которым мы связываем данный метод (`value = "/hello"`), далее указывается, что этот метод будет обрабатывать GET-запросы (`method = RequestMethod.GET`). Также, нужно отметить важные моменты в отношении приведённого выше контроллера:

- Вы определяете бизнес-логику внутри связанного таким образом служебного метода. Из него Вы можете вызывать любые другие методы.
- Основываясь на заданной бизнес-логике, в рамках этого метода Вы создаёте Модель (Model). Вы можете добавлять атрибуты Модели, которые будут добавлены в Вид (View). В примере выше мы создаём Модель с атрибутом `«message»`.

Данный служебный метод возвращает имя Вода в виде строки `String`. В данном случае, запрашиваемый Вид имеет имя `«hello»`.

### 2.3. Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства

Опишем некоторые алгоритмы работы, использующихся в реализации бизнес-логики.

#### Взаимодействие с БД.

Для взаимодействия с базой данных, необходимо следовать следующему алгоритму:

- 1) Создать сущность. Сущность представляет собой класс с названием, которому будет соответствовать название таблицы в базе данных и полями, соответствующими полями из базы данных
- 2) Для созданной сущности создать репозиторий. Репозиторий – это интерфейс для спринга, в котором мы описываем свои методы (если это необходимо) для работы с данными в базе данных текущей сущности.
- 3) Используя репозиторий создать класс сервиса сущности. Класс сервиса – это связующий класс, который будет получать данные из базы данных и предоставлять его контроллерам и наоборот.

Пример, как это выглядит, отображен в листинге 1.

Листинг 1. Пример классов для взаимодействия с БД.

```
@Entity
@Data
public class Item extends AbstractEntity {
    private String name;
    private float price;
    @Column(length = 3000)
    private String description;
    private String image;
    @ManyToOne
    @JoinColumn(name = "item_type_id")
    private ItemType type;
    @ManyToOne
    @JoinColumn(name = "item_brand_id")
    private ItemBrand brand;
    @OneToMany(fetch = FetchType.LAZY, mappedBy = "item")
    private List<OrderItem> orderItems = new ArrayList<OrderItem>();
    @Override
    public String toString() {
        return name + ", " + price + ", " + type;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
```

```

    if (o == null || getClass() != o.getClass()) return false;
    if (!super.equals(o)) return false;
    Item item = (Item) o;
    if (Float.compare(item.price, price) != 0) return false;
    if (name != null ? !name.equals(item.name) : item.name != null)
        return false;
    if (description != null ? !description.equals(item.description) :
        item.description != null)
        return false;
    if (image != null ? !image.equals(item.image) : item.image != null)
        return false;
    if (type != null ? !type.equals(item.type) : item.type != null)
        return false;
    return brand != null ? brand.equals(item.brand) : item.brand == null;
}
@Override
public int hashCode() {
    int result = super.hashCode();
    result = 31 * result + (name != null ? name.hashCode() : 0);
    result = 31 * result + (price != +0.0f ? Float.floatToIntBits(price)
        : 0);
    result = 31 * result + (description != null ? description.hashCode()
        : 0);
    result = 31 * result + (image != null ? image.hashCode() : 0);
    result = 31 * result + (type != null ? type.hashCode() : 0);
    result = 31 * result + (brand != null ? brand.hashCode() : 0);
    return result;
}
}

```

### **Взаимодействие пользователя с данными**

Для реализации взаимодействия используются контроллеры, со следующим алгоритмом

- 1) Ждем действие пользователя. В данном случае, действием называется переход пользователя на любую страницу сервиса или отправка запроса.
- 2) Получаем адрес, на который пользователь посылает запрос. К примеру, это будет адрес «/index» или «/item/edit»
- 3) В зависимости от адреса, вызываем метод контроллера. Для первого случая, вызывается метод контроллера IndexControler, для второго – вызывается метод edit из контроллера ItemController.
- 4) Возвращаем результат метода. Результатом любого метода является страница или пересылка на страницу. Также, в теле метода могут быть прикреплены данные для страницы.

Затем алгоритм повторяется для каждого запроса.

Ниже отобразим пример метода edit из контроллера ItemController. На вход он принимает адрес пользователя и в зависимости от того, передавал ли пользователь id сущности, включает данные из базы данных или создает новую сущность, которую включает в модель ответа. После чего данные с моделью высылаются пользователю и использованием указанного шаблона.

Листинг 2. Пример метода контроллера

```
@RequestMapping(path = {"/edit", "/edit/{id}"})
public String edit(Model model, @PathVariable(name = "id", required =
    false) Long id, Principal principal) {

    if (id != null) {
        Item entity = service.read(id);
        model.addAttribute("entity", entity);
    } else {
        model.addAttribute("entity", new Item());
    }
    model.addAttribute("types", itemTypeService.repository.findAll());
    model.addAttribute("brands", itemBrandService.repository.findAll());
    return "item-add-edit";
}
```

### **Вывод шаблонных данных**

Вывод множественных данных осуществляется по шаблонам, которые находятся в папке templates. При отправке пользователем запроса на получения данных, и после завершения метода контроллера происходит следующее

- 1) Из папки берется шаблон с тем именем, результат которого прислал метод контроллера
- 2) Данные контроллера, который он записал в модель передаются выбранному шаблону
- 3) Шаблон через внутренние конструкторы и при необходимости циклы, на стороне сервера создает страницу по данным от конструктора
- 4) Эта созданная страница отправляется пользователю.

Для примера вывода множественных данных можно показать код вывода списка товаров для администратора. Шаблон использует внутренний цикл для вывода всех элементов списка, получаемого от контроллера.

Листинг 3. Генерация страниц на стороне сервера по шаблону

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<title>Товары</title>
```

```

    <div th:replace="~{commons :: head}"></div>
<body>

<div th:replace="~{commons :: nav}"></div>

<section class="listItem">
    <div class="container my-2">
        <div class="card">
            <div class="row">
                <div class="card-body col">
                    <h2 class="text-center"> Товары</h2>
                    <div th:switch="{list}" class="container my-5">
                        <div class="col-md-12">
                            <h2 th:case="null">Записей не найдено</h2>
                            <div th:case="*" class="overScroll">
                                <table class="table table-striped table-responsive-md">
                                    <thead>
                                        <tr>
                                            <th>Название</th>
                                            <th>Бренд</th>
                                            <th>Тип</th>
                                            <th>Цена</th>
                                            <th>Описание</th>
                                            <th>Изображение</th>
                                        </tr>
                                    </thead>
                                    <tbody>
                                        <tr th:each="entity : {list}">
                                            <td th:text="{entity.name}"></td>
                                            <td th:text="{entity.brand}"></td>
                                            <td th:text="{entity.type}"></td>
                                            <td th:text="{entity.price}"></td>
                                            <td th:text="{entity.description}"></td>
                                            <td></td>
                                            <td class="widthLastCol ">
                                                <a
                                                    th:href="@{/item/edit/{id}(id={entity.id})}"
class="myButton ">
                                                    Редактировать
                                                </a>
                                            </td>
                                            <td class="widthLastCol ">
                                                <a
                                                    th:href="@{/item/delete/{id}(id={entity.id})}"
class="myButton">
                                                    Удалить

```





## Лучшие крема

Да-да, вы не ослышались

В нашем каталоге можно найти крема под любую кожу и требования, и мы гарантируем вам качество выбранного товара!



## Товары в каталоге доступны по категориями

При переходе на страницу каталога, вам будет доступен выбор категорий для удобства навигации. Ниже представлены возможные категории (конкретно этих может и не быть), и их явно больше чем 3!



Крема



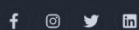
Косметички



Мази

## Shop

📍 Адрес компании  
☎ 010-020-0340  
✉ info@company.com



Copyright © 2021 by Hecha

## Информация

Главная  
О нас  
Каталог

Рисунок 2.2 – Вид главной страницы

На странице о нас будет отображено немного информации о самом интернет-магазине.

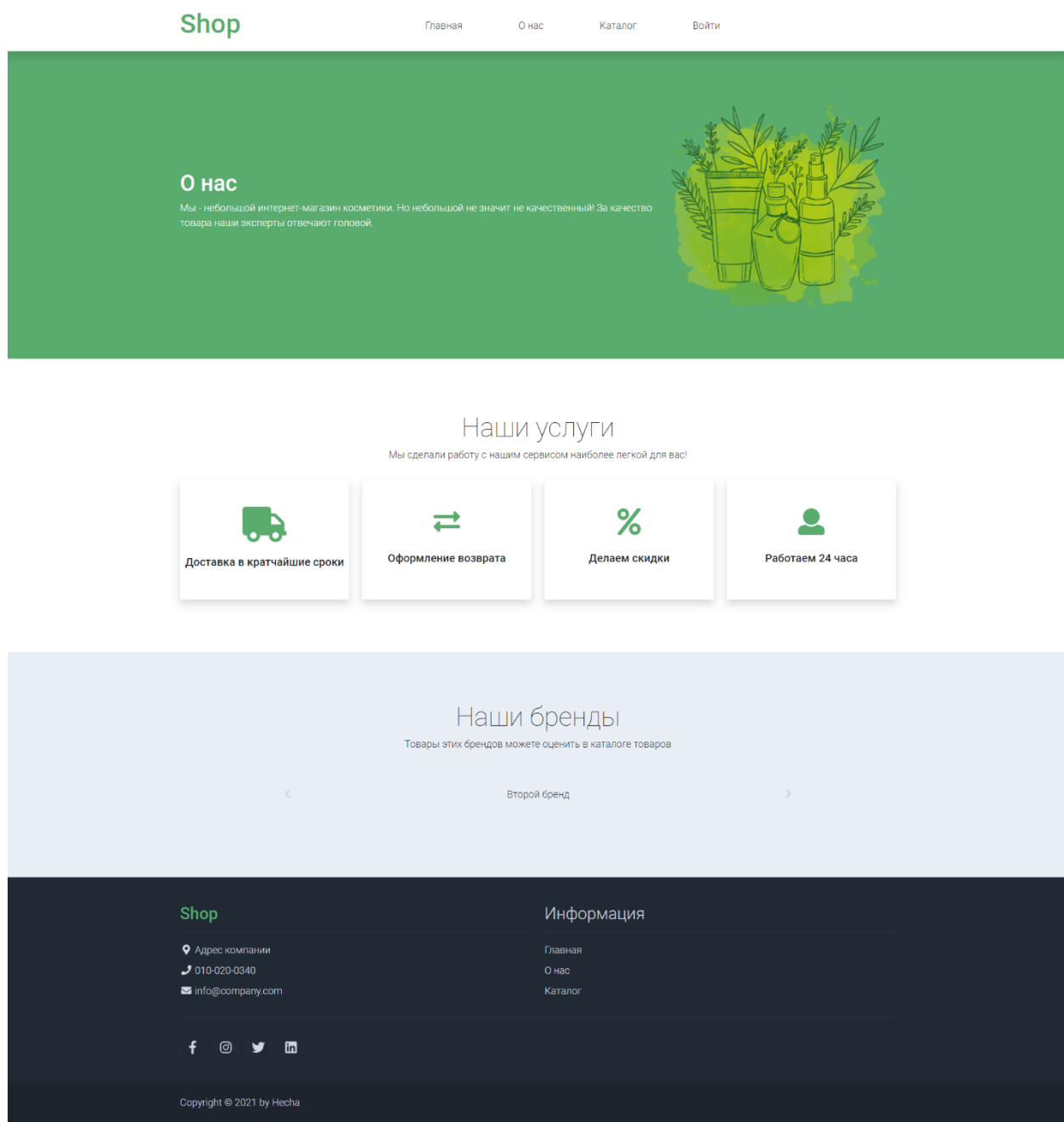


Рисунок 2.3 – Вид страницы о нас

На странице каталога будут отображены все товары магазина. При необходимости, можно сразу же отфильтровать товары по бренду или типу, выбрав соответствующий тип или бренд в левом меню.

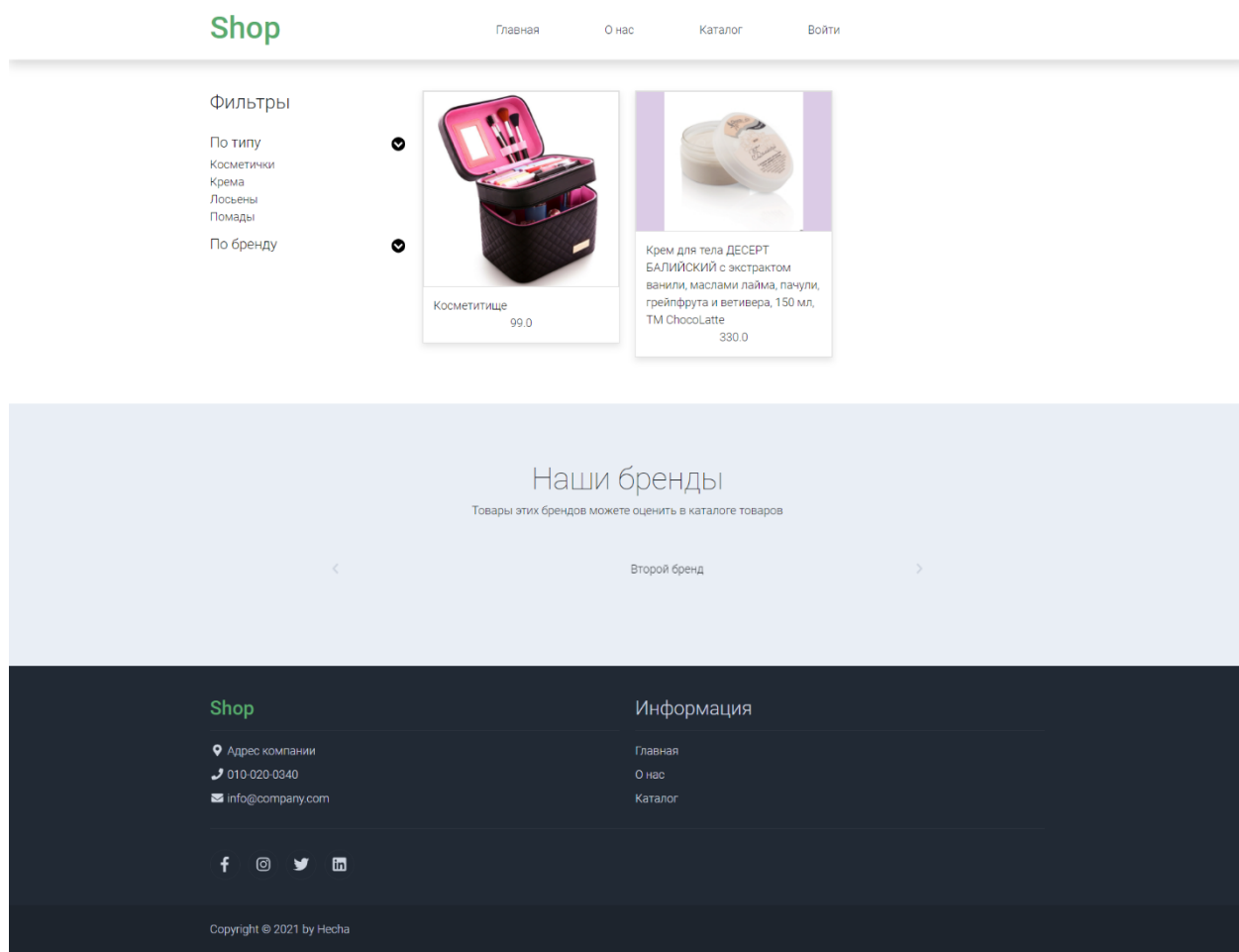


Рисунок 2.4 – Вид каталога товаров

Для входа или регистрации будет использоваться одна страница

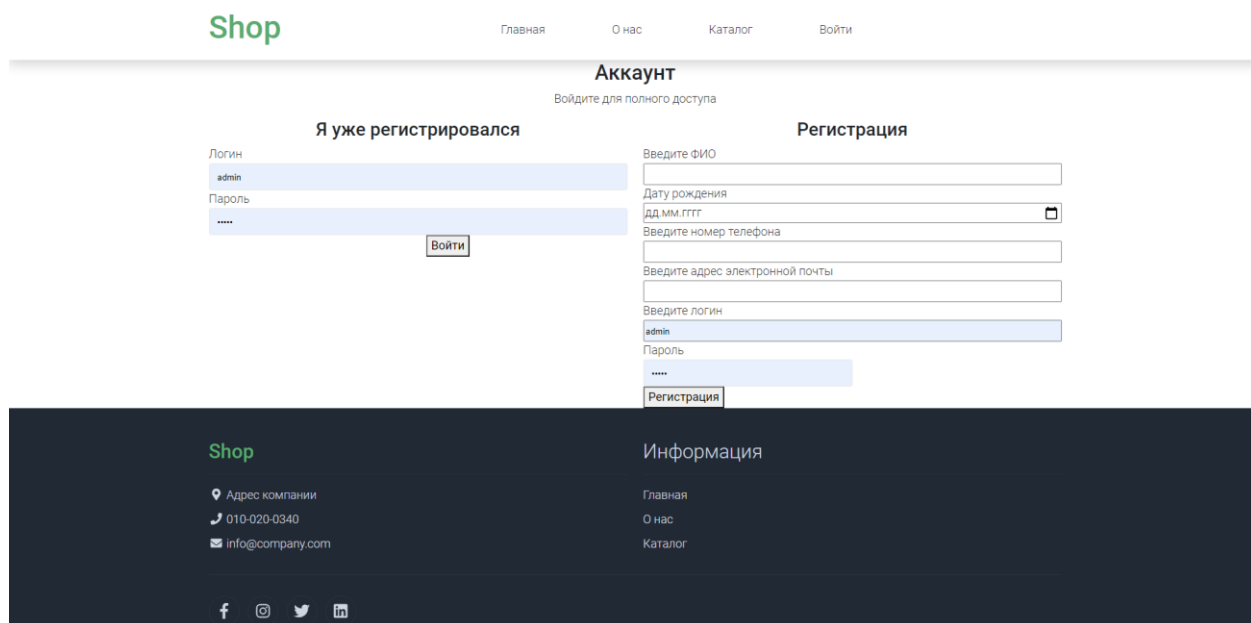


Рисунок 2.5 – Вид страницы входа и регистрации

При входе в систему пользователю будет доступно новое меню.

Рисунок 2.6 – Меню вошедшего пользователя

Теперь вошедшему пользователю также доступен выбор количества элементов для добавления в корзину.

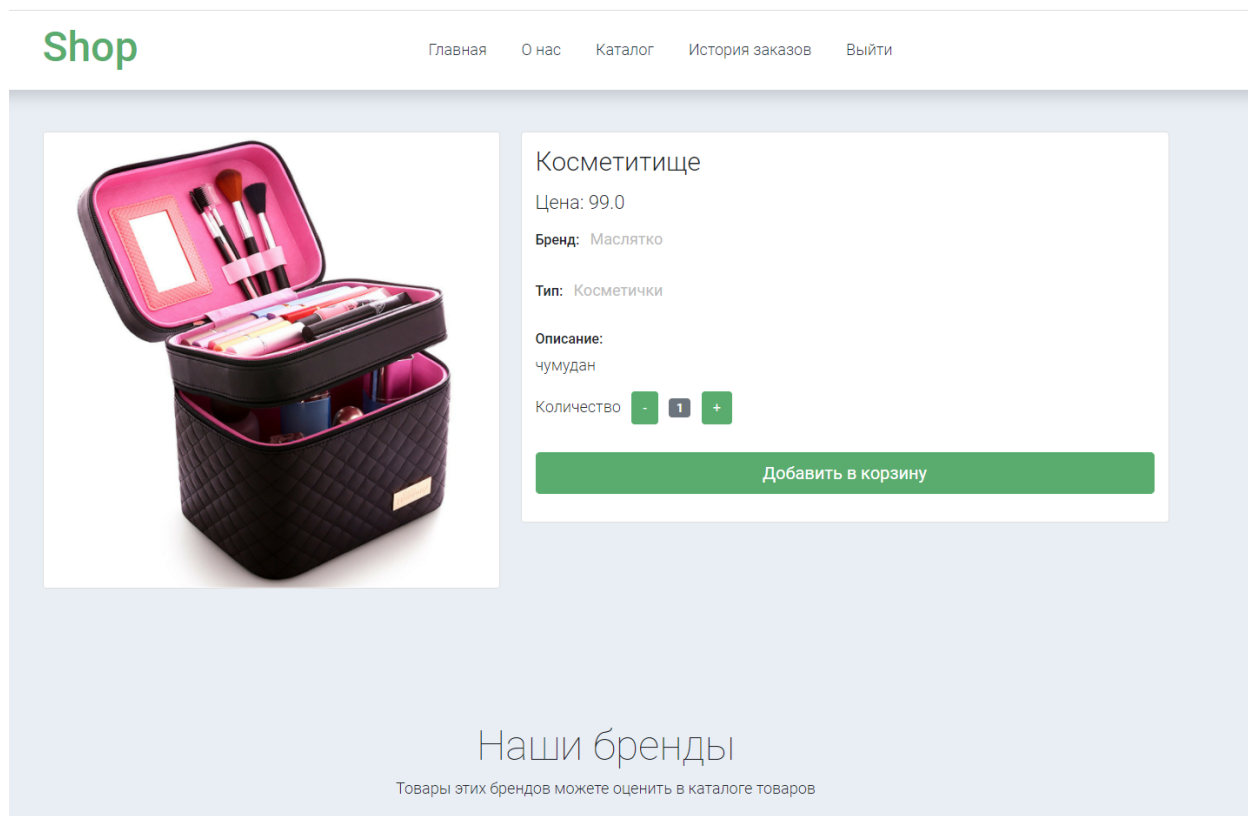


Рисунок 2.7 Страница подробностей о товаре.

Кроме того, пользователь на странице истории может взаимодействовать как с товарами из корзины, так и просматривать все сделанные собой ранее заказы.

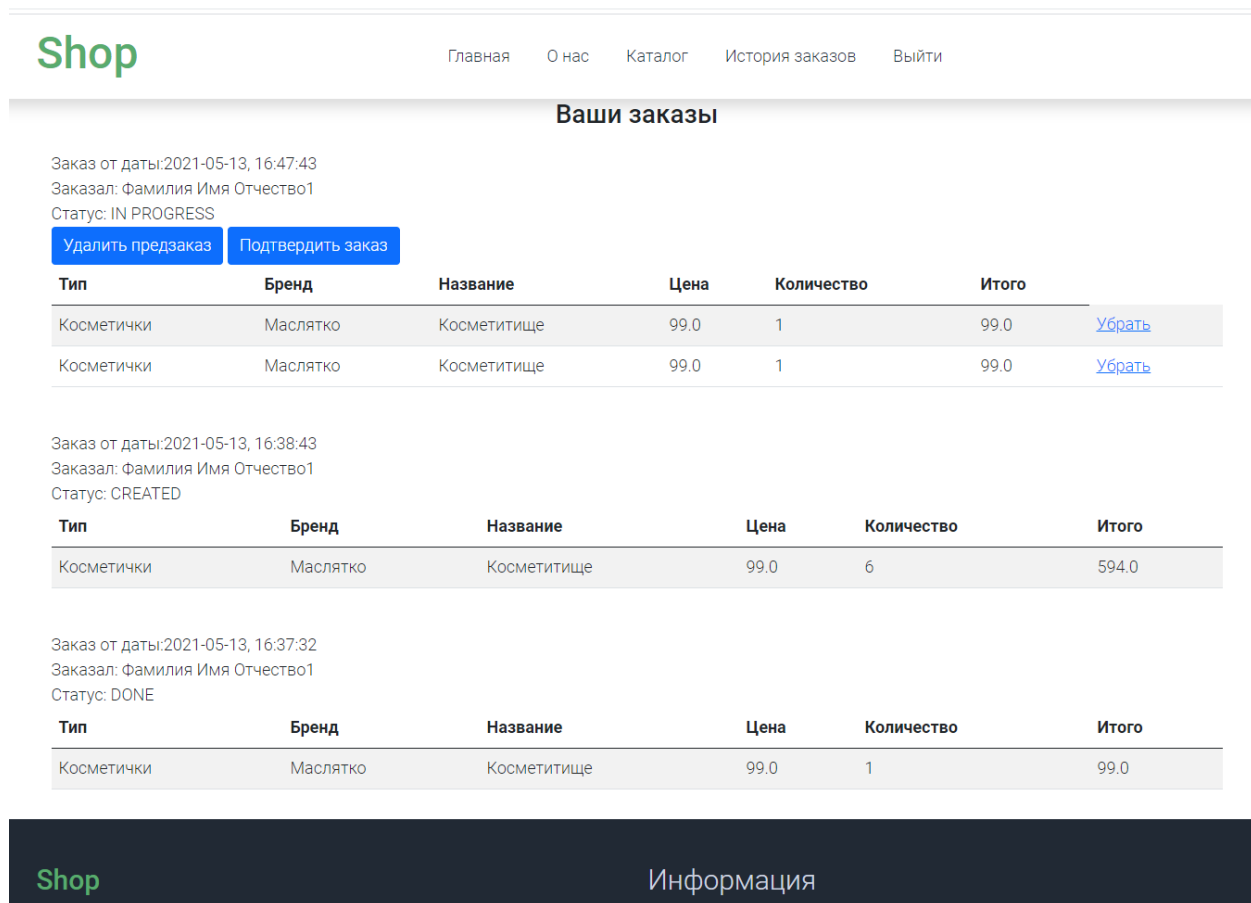


Рисунок 2.8 - Страница истории заказов

При входе от имени администратора будет показано уже другое меню

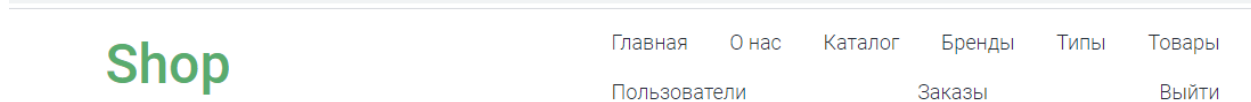


Рисунок 2.9 – Меню администратора

Так, на страницах взаимодействия с таблицами в табличной же форме будет отображена информация с возможностью редактирования данных.

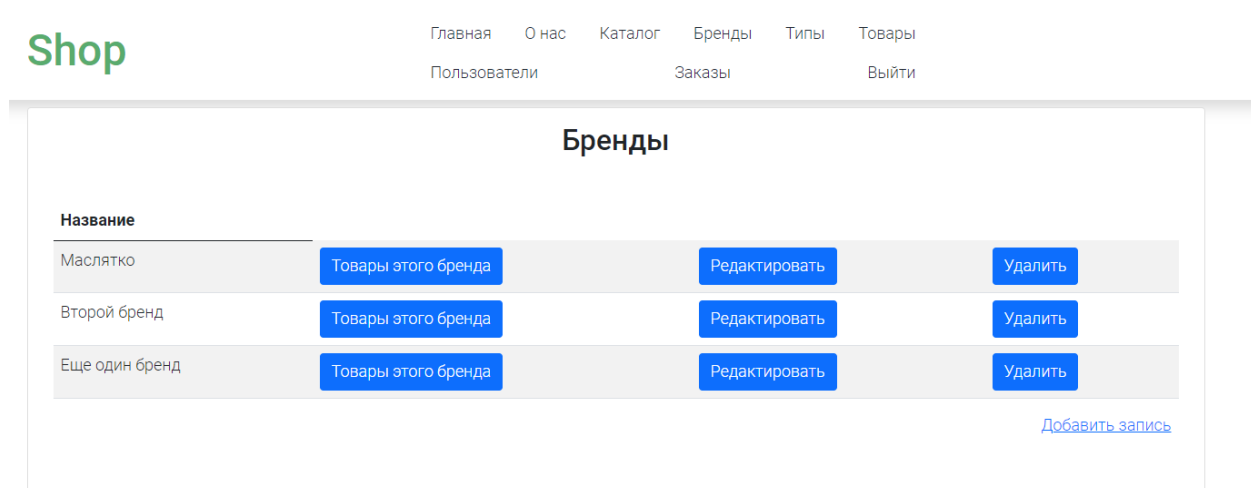


Рисунок 2.10 – Страница работы с брендами

Ниже показана страница редактирования бренда. Остальные страницы редактирования других таблиц сделаны по тому же принципу.

Рисунок 2.11- Страница редактирования бренда

На странице заказов будут отображены заказы пользователей с указанием их статусов. Так, администратор может либо подтвердить заказ, либо удалить его.

**Заказы пользователей**

Заказ от даты: 2021-05-13, 16:47:43  
Заказал: Фамилия Имя Отчество1  
Статус: IN PROGRESS  
[Удалить заказ](#)

Тип	Бренд	Название	Цена	Количество
Косметички	Маслятко	Косметитище	99.0	1
Косметички	Маслятко	Косметитище	99.0	1

Заказ от даты: 2021-05-13, 16:38:43  
Заказал: Фамилия Имя Отчество1  
Статус: CREATED  
[Удалить заказ](#) [Подтвердить заказ](#)

Тип	Бренд	Название	Цена	Количество
Косметички	Маслятко	Косметитище	99.0	6

Рисунок 2.12 – Страница просмотра заказов для администратора

## 2.5. Обоснование выбора компонентов и технологий для реализации программного средства.

Архитектура данного проекта — это веб-приложение. Архитектура приложения во многом предопределило и технологии, используемые для построения проекта.

Основные технологии, применяемые при создании приложения:

- серверный язык *JAVA*;
- язык запросов *SQL (MySQL)*;
- фреймворк *Spring*;
- язык разметки *HTML*;
- таблица стилей *CSS*;
- клиентский язык *JavaScript*.

Каждая из выбранных технологий отвечает за разные аспекты работы программы.

Язык Java появился в 1995 году – 90-е годы были вообще урожайными на новые языки и концепции программирования. В таком Эдеме языков важно было не заблудиться, по ошибке приняв за Священный Грааль технологию, которая не пройдет испытания временем. Java прошел испытания, хотя и очень долгие. Очень не рекомендуется путать этот язык с JavaScript – они по виду похожи, но это совсем разные языки [7].

Вероятно, в Java впервые реализовали концепцию того, что язык должен быть максимально изолирован от платформы разработки, чтобы применять его без изменений везде: в компьютерах, часах, сотовых телефонах, бытовой технике. С «железной частью» должна была справляться виртуальная машина (JVM), которая, собственно, и создавалась индивидуально под каждое устройство. Сам же язык был неизменен и в качестве результата выдавал байт-код. С самого начала было известно, что код не может исполняться очень быстро, но многие устройства не требовали высокой скорости исполнения. Кроме того, со временем появились оптимизирующие компиляторы, так что, в среднем, программа на Java работает раза в 2-3 медленнее, чем на C++. Постоянное сравнение с C/C++ здесь не случайно: многие современные языки взяли за основу его конструкции и синтаксис, так что, бывает, узнать сходу язык очень трудно. Вместе с тем, Java с тех пор сильно «размножилась», и даже J#, J и прочие аналоги являются не родными братьями, а лишь подобием.

Сама идея языка, вполне, кстати, достаточного для создания софта любой сложности, была сначала не понята: был ли, мол, смысл создавать между аппаратурой и кодом промежуточные слои исполняющих машин? Со временем сомнения рассеялись: появилась мультязычная платформа.

NET, и даже в Windows появились слои – аппаратно-зависимые, платформо-независимые. Самое же простое объяснение – софт стал очень сложным, а программисты очень ленивыми, чтобы переписывать программы под каждый отдельный аппарат.

Но вернемся к языку. Как уже говорилось, чем-то он похож на C++, чем-то на старый добрый Бейсик. Нет сейчас ни одного языка, который бы не хвалился своими возможностями ООП, и Java здесь не отличается от канонов: классы и объекты здесь используются везде, даже в самых примитивных задачах вроде вывода строки на экран. Из особенностей можно отметить, что все объекты в языке создаются только динамически, а все функции являются методами классов. Множественное наследование не поддерживается, как в C++, как и «опасные» указатели. ООП дает много преимуществ, но и требует слишком многого – в случае Java памяти устройства никогда не будет слишком много. В остальном же, имеются библиотеки классов для практически всех задач; преимущественно – под написание клиентских и серверных приложений. Хозяин Java – Oracle – успешно использует язык для использования в разработках своей одноименной СУБД. На сегодняшний день язык считается наиболее востребованным на рынке.

Spring Framework (или коротко Spring) — универсальный фреймворк с открытым исходным кодом для Java-платформы. Также существует форк для платформы .NET Framework, названный Spring.NET [8].

Первая версия была написана Родом Джонсоном, который впервые опубликовал её вместе с изданием своей книги «Expert One-on-One Java EE Design and Development» (Wrox Press, октябрь 2002 года).

Фреймворк был впервые выпущен под лицензией Apache 2.0 license в июне 2003 года. Первая стабильная версия 1.0 была выпущена в марте 2004. Spring 2.0 был выпущен в октябре 2006, Spring 2.5 — в ноябре 2007, Spring 3.0 в декабре 2009, и Spring 3.1 в декабре 2011. Текущая версия — 5.2.x.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как альтернатива и замена модели Enterprise JavaBeans. Spring предоставляет бóльшую свободу Java-разработчикам в проектировании; кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений корпоративного масштаба.

Между тем, особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Следующие две технологии — это *HTML* и *CSS*, которые предопределены архитектурой проекта. Имеется единственное обоснование их выбора — это веб-архитектура разрабатываемого программного средства.

*HTML* — язык гипертекстовой разметки, который используется для структурирования и отображения веб-страницы и ее контента. Для разработки данного программного средства будет применяться версия этого языка *HTML5*.

*CSS* — это формальный язык описания внешнего вида документа, написанного с использованием языка разметки. Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки *HTML*. Применяемая версия данной технологии — *CSS3*.



### 3. ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Для проверки работы спроектированной автоматизированной системы, разработаны тесты, с помощью которых можно оценить корректную работу веб-сервиса. В таблицах 3.1-3.2 приведены результаты тестирования.

Таблица 1.1 – Набор тестов для автоматизированной системы

Место проведения теста	Содержание теста	Ожидаемый результат	Отметка о прохождении теста
Главная страница	Загрузка главной страницы	Отображение страницы	Да
Страница добавления брендов	Перейти на страницу. Ввести новые данные. Нажать кнопку отправить	Переход на страницу брендов с выводом только что созданного значения	Да
Страница типов. Страница товаров.	Перейти на страницу типов. Добавить два новых типа. Перейти на страницу товаров. В строке типа выбрать любое из нового значения	При переходе на страницу добавления товаров, в списке типов отображаются новые значения	Да
Страница заказов администратора. Страница истории пользователя	Перейти на страницу заказов администратора. Подтвердить заказ. Перейти на страницу истории заказов пользователя.	На странице истории пользователя уже нельзя будет удалить заказ, так как он подтвержден администратором	Да

Таблица 4.2 – Набор тестов отображения информации

Место проведения теста	Содержание теста	Ожидаемый результат	Отметка о прохождении теста
Все страницы	Перейти на любую из вкладок в произвольном порядке.	Корректное отображение информации, отображение того, чего ожидает пользователь.	Да

Все вкладки	Увеличение масштаба отображения средствами браузера.	Появление скроллов справа и снизу, корректное отображение информации.	Да
Все вкладки	Обновление страницы используя средства браузера.	Состояние веб-сервиса до обновления и после не изменилось.	Да
Меню браузера	Нажатие функциональных кнопок браузера «Назад» и «Вперед» в произвольный момент работы веб-приложения.	Корректное поведение веб-сервиса. Отображение информации.	Да

## 4. РУКОВОДСТВО ПО РАЗВЕРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

Для установки веб-сервиса необходимым требованием является наличие Java Virtual Machine, Apache Tomcat 8, а также MySQL Server 8.0.1 в качестве целевой СУБД, maven 3.6.

База данных для веб-сервиса генерируется сама после первого запуска приложения.

Для запуска приложения на клиентском компьютере, достаточно в консоли перейти в папку проекта и написать код

```
mvn clean package spring-boot:run
```

Если запуск решено производить посредством war-архива, то его нужно переместить или скопировать в папку, где установлен Apache Tomcat 8. Далее необходимо запустить Apache Tomcat 8.

Для запуска веб-сервиса необходимо открыть браузер на вашем компьютере и написать в адресную строку следующий URL-адрес: <http://localhost:8080/>.

Далее опишем руководство по использованию программного средства.

При заходе на главную страницу, пользователь будет видеть общее количество групп и студентов в системе и пункт меню для перехода на другие страницы.

## Лучшие крема

Да-да, вы не ослышались

В нашем каталоге можно найти крема под любую кожу и требования, и мы гарантируем вам качество выбранного товара!



## Товары в каталоге доступны по категориям

При переходе на страницу каталога, вам будет доступен выбор категорий для удобства навигации. Ниже представлены возможные категории (конкретно этих может и не быть), и их явно больше чем 3!



Крема



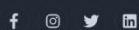
Косметички



Мази

### Shop

📍 Адрес компании  
☎ 010-020-0340  
✉ info@company.com



Copyright © 2021 by Hecha

### Информация

Главная  
О нас  
Каталог

Рисунок 4.1 – Главная страница

Можно посмотреть информацию о магазине на соответствующей странице.

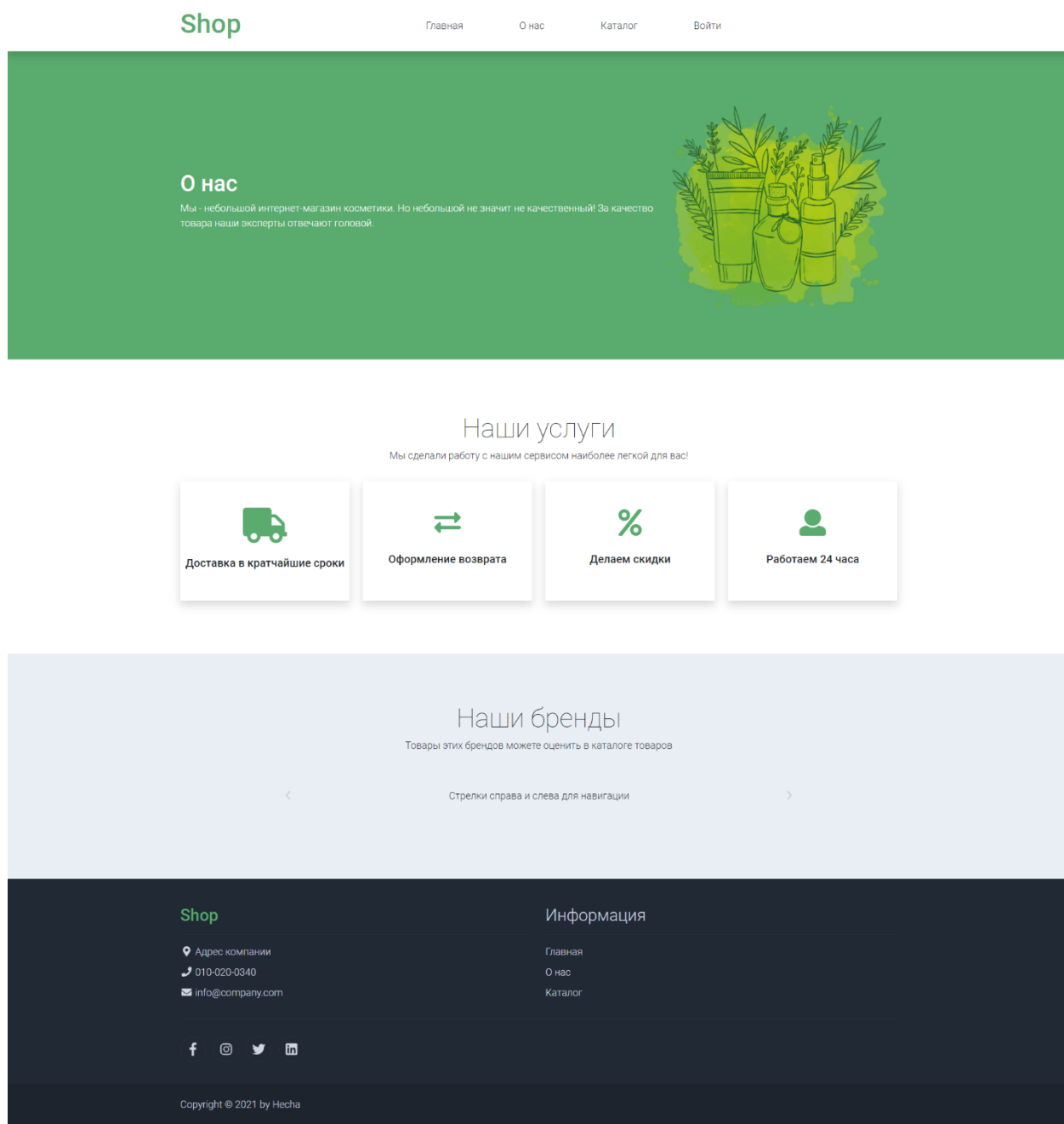


Рисунок 4.2 – Страница о нас

При переходе на страницу каталога будет отображен список всех товаров магазина

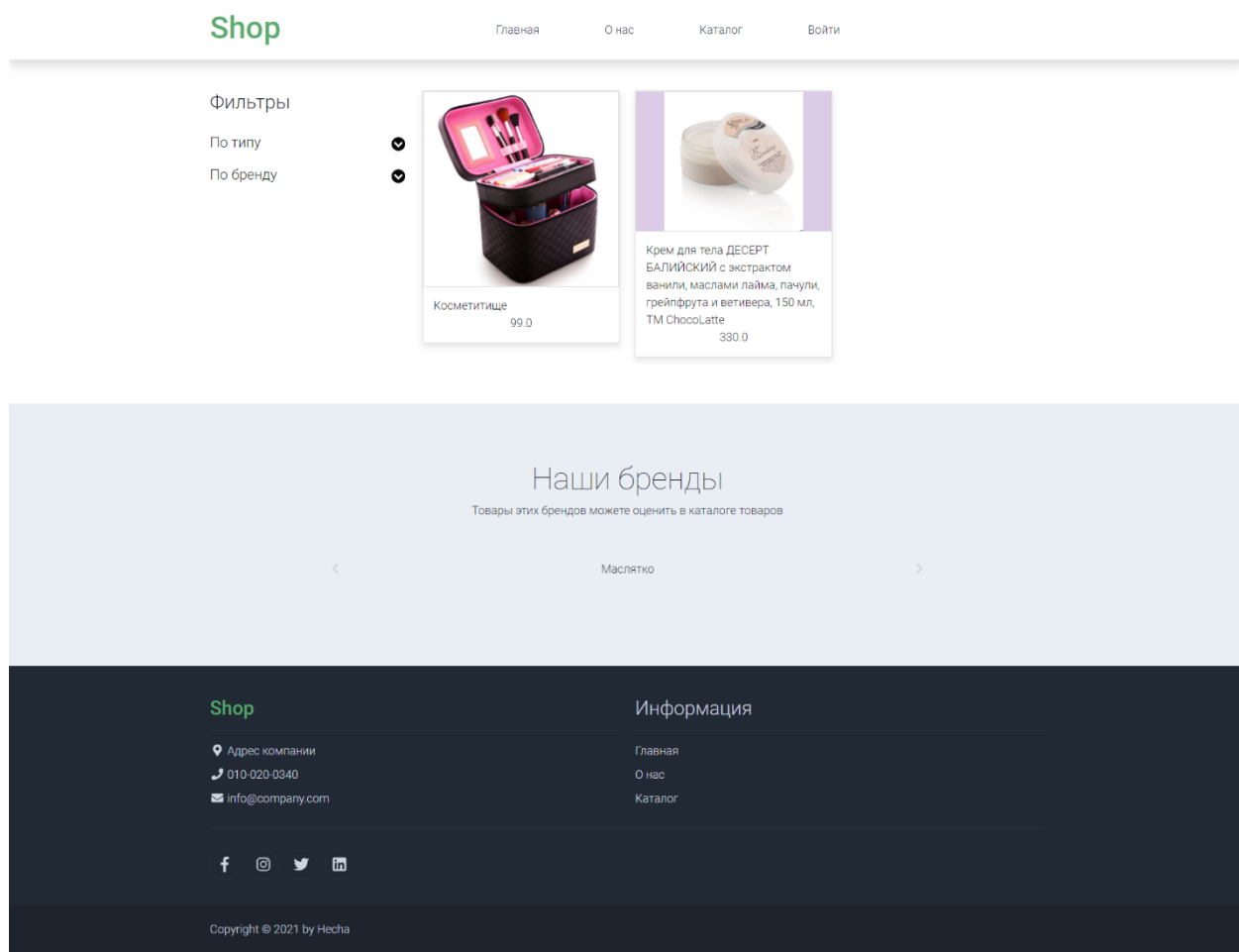


Рисунок 4.3 – Каталог товаров  
В левом меню есть фильтрация товаров по типу.

## Фильтры

По типу



Косметички

Крема

Лосьены

Помады

По бренду



Рисунок 4.4 – Фильтры типов  
Или по бренду. При добавлении новых типов и брендов, эти меню расширяются.

По типу



По бренду



Маслятко

Второй бренд

Еще один бренд

Рисунок 4.5 – Фильтр брендов  
К примеру, отобразим только косметички.

Эпо

Главная

О нас

Фильтры

По типу



По бренду



Косметичке

99.0

Рисунок 4.6 – Отображение товаров выбранного типа

При наведении на товар будет отображена кнопка для перехода на странице подробностей об том товаре.

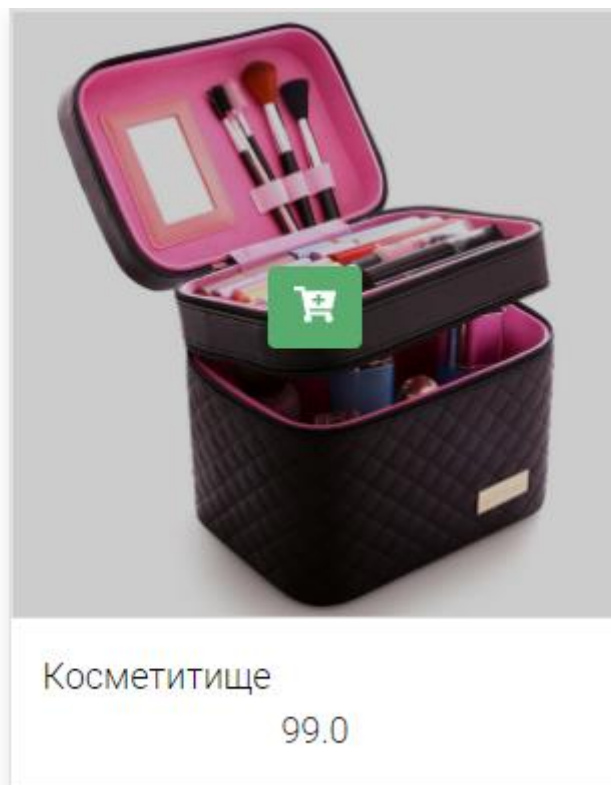


Рисунок 4.7 – Вид при наведении мышкой

Так, перейдем на страницу подробностей о креме для тела. Так как мы пока в режиме гостя (не вошли в систему), то мы не можем заказать данный товара, а можем только посмотреть информацию о нем.

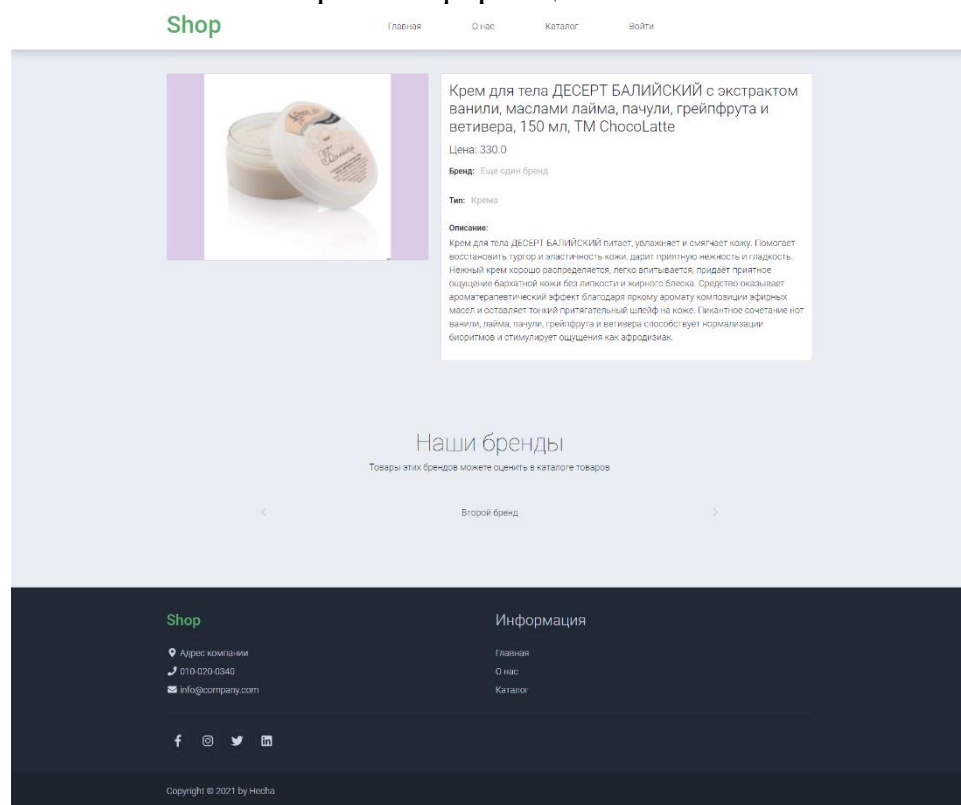


Рисунок 4.8 – Подробности товара



Перейдем на страницу входа для регистрации в системе.

Рисунок 4.9 – Страница входа и регистрации  
Зарегистрируем нового пользователя.

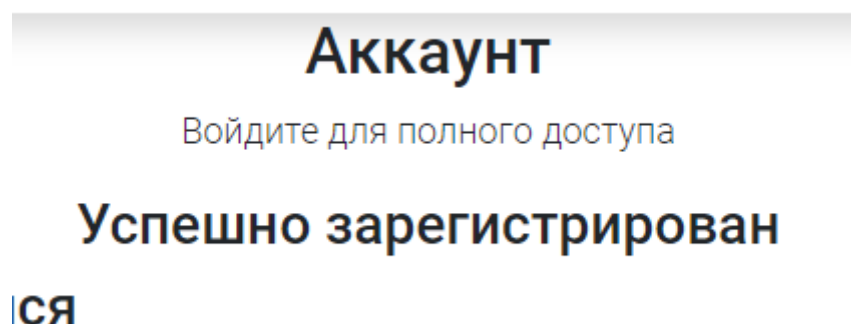


Рисунок 4.10 – Результат регистрации

Как видно из рисунка выше, система выдала уведомление об успешном регистрации нового пользователя, так что мы можем сейчас войти в систему и пользоваться ей уже как нормальный пользователь.

После входа в систему, нам будет показано уже другое меню.

Рисунок 4.11 – Меню пользователя после входа

Перейдем на подробности о товаре и увидим, что сейчас мы уже можем положить этот товар к себе в корзину.

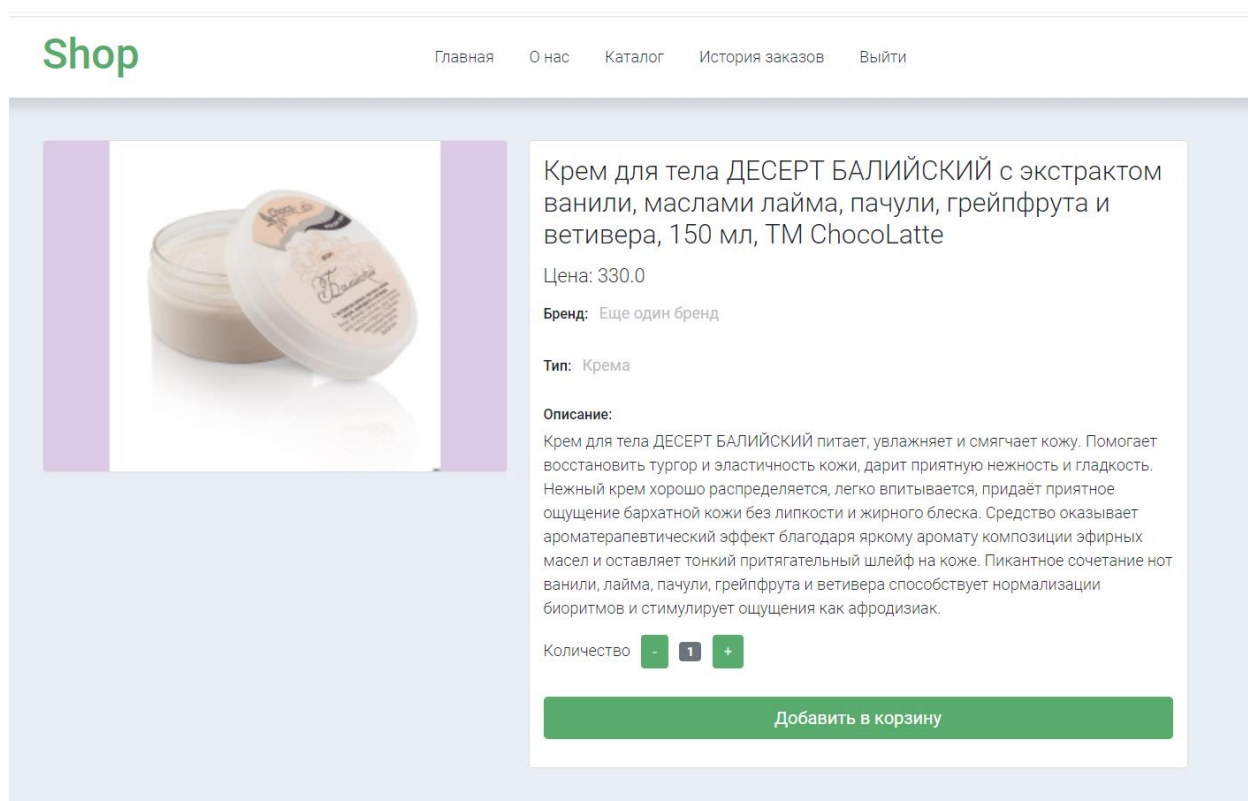


Рисунок 4.12 – Страница подробностей о товаре

Заполним корзину несколькими товарами, после чего нас перебросит на страницу истории, на которой мы можем взаимодействовать с текущей корзиной или просматривать сделанные ранее заказы.

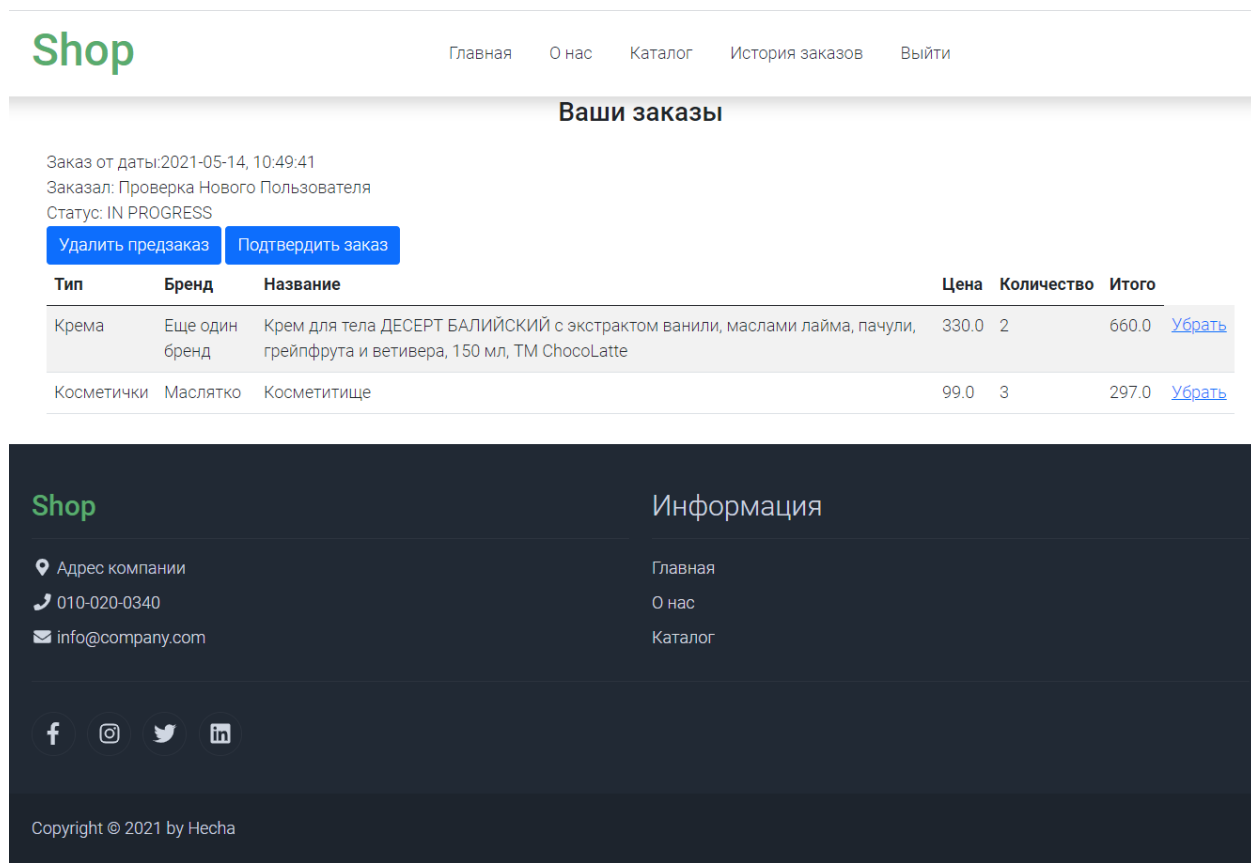


Рисунок 4.13 – Страница истории заказов

Удалим первый товар из нашей корзины, представив что мы его ошибочно добавили или же передумали его покупать.

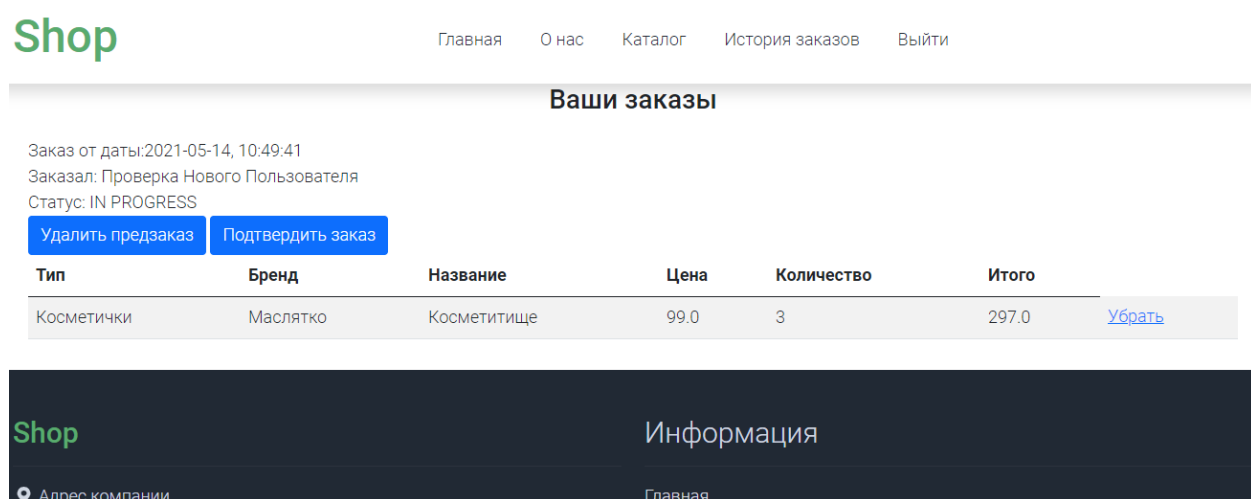


Рисунок 4.14 – Результат после удаления товара из корзины

Как видно, товар спокойно удалился. Пока мы не подтверждали заказ, мы можем добавлять или удалять товары из корзины, но как только заказ будет подтвержден, мы уже не сможем этого сделать. Подтверждаем свой заказ, нажав соответствующую кнопку.

Ваши заказы					
Заказ от даты:2021-05-14, 10:50:06					
Заказал: Проверка Нового Пользователя					
Статус: CREATED					
Тип	Бренд	Название	Цена	Количество	Итого
Косметички	Маслятко	Косметитище	99.0	3	297.0

### Рисунок 4.15 – Результат подтверждения заказа

Заказ подтвержден, как видно из рисунка выше, мы больше не можем изменять товары в заказе.

Войдем от имени администратора. У нас сразу же изменится отображение меню.

Главная	О нас	Каталог	Бренды	Типы	Товары
Пользователи		Заказы			Выйти

### Рисунок 4.16 – Меню администратора

Перейдем в заказы от имени администратора и увидим только что созданный пользователем заказ, его статус CERATED, что означает что пользователь подтвердил его.

Заказы пользователей					
Заказ от даты:2021-05-14, 10:50:06					
Заказал: Проверка Нового Пользователя					
Статус: CREATED					
<div>Удалить заказ</div> <div>Подтвердить заказ</div>					
Тип	Бренд	Название	Цена	Количество	
Косметички	Маслятко	Косметитище	99.0	3	

Заказ от даты:2021-05-13, 16:47:43					
Заказал: Фамилия Имя Отчество1					
Статус: IN PROGRESS					
<div>Удалить заказ</div>					

### Рисунок 4.17 – Страница заказов администратора

Теперь мы можем либо подтвердить выполнение этого заказа или же удалить его. Подтверждаем.

Заказы пользователей				
Заказ от даты: 2021-05-14, 10:50:06				
Заказал: Проверка Нового Пользователя				
Статус: DONE				
<a href="#">Удалить заказ</a>				
Тип	Бренд	Название	Цена	Количество
Косметички	Маслятко	Косметитище	99.0	3

Заказ от даты: 2021-05-13, 16:47:43

Рисунок 4.18 – Результат подтверждение заказа администратором

Заказ после подтверждения перешел в другой статус. Вернемся на страницу пользователя для просмотра.

Ваши заказы					
Заказ от даты: 2021-05-14, 10:50:06					
Заказал: Проверка Нового Пользователя					
Статус: DONE					
Тип	Бренд	Название	Цена	Количество	Итого
Косметички	Маслятко	Косметитище	99.0	3	297.0

Рисунок 4.19 – Вид страницы истории заказов пользователя после подтверждения заказа администратором

Статус заказа также изменился и на странице пользователя.

Осталось показать такие действия администратора, как добавление информации в таблицы приложения. Перейдем на страницу брендов.

Shop	<a href="#">Главная</a>	<a href="#">О нас</a>	<a href="#">Каталог</a>	<a href="#">Бренды</a>	<a href="#">Типы</a>	<a href="#">Товары</a>
	<a href="#">Пользователи</a>		<a href="#">Заказы</a>		<a href="#">Выйти</a>	
Бренды						
Название						
Маслятко	<a href="#">Товары этого бренда</a>		<a href="#">Редактировать</a>		<a href="#">Удалить</a>	
Второй бренд	<a href="#">Товары этого бренда</a>		<a href="#">Редактировать</a>		<a href="#">Удалить</a>	
Еще один бренд	<a href="#">Товары этого бренда</a>		<a href="#">Редактировать</a>		<a href="#">Удалить</a>	
<a href="#">Добавить запись</a>						

Рисунок 4.20 – Страница работы с брендами

Для добавления нажмем соответствующую кнопку.

### Создание / Редактирование бренда

Название

Добавляем новый бренд

Назад

Отправить

Рисунок 4.21 – Добавление нового бренда  
После добавления товара, он будет отображен внизу списка.

Бренды			
Название			
Маслятко	Товары этого бренда	Редактировать	Удалить
Второй бренд	Товары этого бренда	Редактировать	Удалить
Еще один бренд	Товары этого бренда	Редактировать	Удалить
Добавляем новый бренд	Товары этого бренда	Редактировать	Удалить
<a href="#">Добавить запись</a>			

Рисунок 4.22 – Результат добавления  
Прделаем те же действия с типами товаров.

Типы			
Название			
Косметички	<a href="#">Товары этого типа</a>	<a href="#">Редактировать</a>	<a href="#">Удалить</a>
Крема	<a href="#">Товары этого типа</a>	<a href="#">Редактировать</a>	<a href="#">Удалить</a>
Лосьены	<a href="#">Товары этого типа</a>	<a href="#">Редактировать</a>	<a href="#">Удалить</a>
Помады	<a href="#">Товары этого типа</a>	<a href="#">Редактировать</a>	<a href="#">Удалить</a>
<a href="#">Добавить запись</a>			

Рисунок 4.23 – Страница работы с типами товаров

### Создание / Редактирование типа

Название

Новый тип

Рисунок 4.24 – Добавление нового типа

Типы			
Название			
Косметички	<a href="#">Товары этого типа</a>	<a href="#">Редактировать</a>	<a href="#">Удалить</a>
Крема	<a href="#">Товары этого типа</a>	<a href="#">Редактировать</a>	<a href="#">Удалить</a>
Лосьены	<a href="#">Товары этого типа</a>	<a href="#">Редактировать</a>	<a href="#">Удалить</a>
Помады	<a href="#">Товары этого типа</a>	<a href="#">Редактировать</a>	<a href="#">Удалить</a>
Новый тип	<a href="#">Товары этого типа</a>	<a href="#">Редактировать</a>	<a href="#">Удалить</a>
<a href="#">Добавить запись</a>			

Рисунок 4.25 – Результат добавления типа

Так мы добавил новый бренд и тип товара. Пора добавить новый товар. Вид таблицы отличается только тем, что нам еще предварительно показывается изображение товара.

Shop

Главная

О нас

Каталог

Бренды

Типы

Товары

Пользователи

Заказы

Выйти

Товары



Название	Бренд	Тип	Цена	Описание	Изображение	
Косметитище	Маслятко	Косметички	99.0	чумудан		<a href="#">Редактировать</a> <a href="#">Удалить</a>
Крем для тела ДЕСЕРТ БАЛИЙСКИЙ с экстрактом ванили, маслами лайма, пачули, грейпфрута и ветивера, 150 мл, TM ChocoLatte	Еще один бренд	Крема	330.0	Крем для тела ДЕСЕРТ БАЛИЙСКИЙ питает, увлажняет и смягчает кожу. Помогает восстановить тургор и эластичность кожи, дарит приятную нежность и гладкость. Нежный крем хорошо распределяется, легко впитывается, придаёт приятное ощущение бархатной кожи без липкости и жирного блеска. Средство оказывает ароматерапевтический эффект благодаря яркому аромату композиции эфирных масел и оставляет тонкий притягательный шлейф на коже. Пикантное сочетание нот ванили, лайма, пачули, грейпфрута и ветивера способствует нормализации биоритмов и стимулирует ощущения как афродизиаки.		<a href="#">Редактировать</a> <a href="#">Удалить</a>
<a href="#">Добавить запись</a>						

Рисунок 4.26 – Страница работы с товарами

Перейдем на страницу добавления нового товара. Из списка брендов и типов выберем добавленные ранее данные.

Главная

О нас

Каталог

Бренды

Типы

Товары

Пользователи

Заказы

Выйти

Товар

Тип

Бренд

Название

Описание

Ссылка на изображение

Цена

Новый тип

Добавляем новый бренд

новый товар

Описание товара

Описание товара

Описание товара

Описание товара

Описание товара

Описание товара

https://lh3.googleusercontent.com/proxy/hRRh2jwowlXR3

888

Назад

Отправить

Рисунок 4.27 – Добавление нового товара  
После добавления, наш товар также отображен внизу списка.


нормализации биоритмов и стимулирует ощущения как афродизиак.								
новый товар	Добавляем новый бренд	Новый тип	888.0	Описание товара	Описание товара	Описание товара		<a href="#">Редактировать</a> <a href="#">Удалить</a>
<a href="#">Добавить запись</a>								

Рисунок 4.28 – Результат добавления нового товара  
Посмотрим, появился и он в каталоге товаров.

Shop

Главная

О нас

Каталог

Бренды

Типы

Товары

Пользователи


Заказы

Выйти

Фильтры


По типу

По бренду




Косметитище

99.0



Крем для тела ДЕСЕРТ БАЛИЙСКИЙ с экстрактом ванили, маслами лайма, пачули, грейпфрута и ветивера, 150 мл, TM ChocoLatte

330.0



новый товар

888.0

Рисунок 4.29 – Каталог товаров после добавление нового товара

41



Товар отображен в каталоге. Мы также можем перейти на страницу подробностей о товаре.

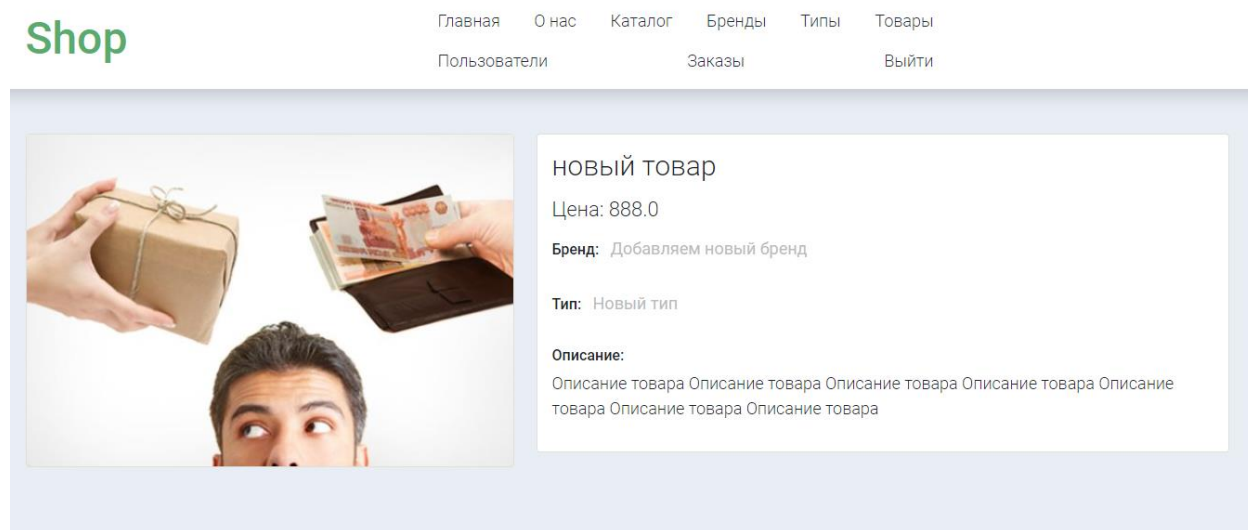


Рисунок 4.30 – Страница подробностей о новом товаре

На странице верно отображается вся добавленная нами ранее информация.

## **ЗАКЛЮЧЕНИЕ**

Итогом написанной курсовой работы является программное приложение интернет-магазина косметики. Данное приложение является доступным и понятным любому пользователю, располагает удобным и минималистичным интерфейсом, а также соответствует всем требованиям, предъявленным к курсовому проекту.

Данная программа является web- приложением, что говорит о том, что множество клиентов, расположенных на расстоянии друг от друга могут одновременно общаться с сервером.

Вся используемая информация хранится в базе данных, разработанной для данного проекта. Доступ к базе данных был реализован только со стороны веб-сервиса.

Проанализировав все моменты, можно сказать, что данная программа соответствует всем требованиям курсового проекта и все задачи, поставленные перед разработкой программы, соблюдены.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Атрибутивная информация [Электронный ресурс]. – Режим доступа: <http://geum.ru/next/art-199278.php>. – Дата доступа: 14.05.2021.
2. Диаграмма структуры программного приложения (SSD) [Электронный ресурс]. – Режим доступа: [https://studopedia.su/3\\_24465\\_SADT-diagrammi.html](https://studopedia.su/3_24465_SADT-diagrammi.html). – Дата доступа: 14.05.2021.
3. Цель анализа требований в проектах [Электронный ресурс]. – Режим доступа: <https://analytics.infozone.pro/requirements-analysis/analysis-of-requirements-wiegers-2004/>. – Дата доступа: 14.05.2021.
4. Разработка информационной модели данных [Электронный ресурс]. – Режим доступа: [https://studbooks.net/2239768/informatika/razrabotka\\_informatsionnoy\\_modeli\\_dannyh](https://studbooks.net/2239768/informatika/razrabotka_informatsionnoy_modeli_dannyh). – Дата доступа: 14.05.2021.
5. Фреймворк Spring MVC [Электронный ресурс]. – Режим доступа: <https://itnan.ru/post.php?c=1&p=336816>. – Дата доступа: 14.05.2021.
6. Проектирование пользовательского интерфейса [Электронный ресурс]. – Режим доступа: <https://www.visualpharm.ru/prototyping.html>. – Дата доступа: 14.05.2021.
7. Серверные языки: Java (обзор) [Электронный ресурс]. – Режим доступа: <https://codomaza.com/article/servernye-jazyki-java-obzor>. – Дата доступа: 14.05.2021.
8. Spring Framework [Электронный ресурс]. – Режим доступа: <https://google-info.org/325966/1/spring-framework.html>. – Дата доступа: 14.05.2021.