ОБЛАСТНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ПРОФЕССИОНАЛЬНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

**«ТОМСКИЙ ЭКОНОМИКО-ПРОМЫШЛЕННЫЙ КОЛЛЕДЖ»**

К ЗАЩИТЕ ДОПУЩЕН

Директор ОГБПОУ «ТЭПК»

\_\_\_\_\_\_\_\_\_\_\_\_ / О.Н. Пояркова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

**ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА REST API СЕРВЕРА НА ФРЕЙМВОРКЕ LARAVEL И САЙТА «ТАЙНЫ ВОСТОКА» НА VUE**

КОМПЛЕКСНЫЙ ДИПЛОМНЫЙ ПРОЕКТ

*Специальность 09.02.07 Информационные системы и программирование*

Выполнили:

Студенты IV курса группы 1912с

\_\_\_\_\_\_\_\_\_\_\_\_ / Федоров А.А.

\_\_\_\_\_\_\_\_\_\_\_\_ / Рюгин А.И.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

Проверили:

Руководители КДП

\_\_\_\_\_\_\_\_\_\_\_\_ / Уляхин В.А.

\_\_\_\_\_\_\_\_\_\_\_\_ / Летягина И.Е.

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2025 г.

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 4](#_Toc200562046)

[ТЕХНОЛОГИИ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ 5](#_Toc200562047)

[Используемые языки программирования и фреймворки 5](#_Toc200562048)

[Используемое программное обеспечение 7](#_Toc200562049)

[ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ИНФОРМАЦИОННОГО ПРОДУКТА 8](#_Toc200562050)

[Описание проекта и задач 8](#_Toc200562051)

[Ролевая модель 10](#_Toc200562052)

[Специфические требования к API 12](#_Toc200562053)

[Разработка прототипа пользовательского интерфейса 44](#_Toc200562054)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ 52](#_Toc200562055)

[Нормализация данных 52](#_Toc200562056)

[Концептуальная модель данных 53](#_Toc200562057)

[Логическая модель данных 54](#_Toc200562058)

[Диаграмма базы данных 56](#_Toc200562059)

[ГЛАВА 3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ФУНКЦИОНАЛА ИНФОРМАЦИОННОГО ПРОДУКТА 57](#_Toc200562060)

[Установка и настройка фреймворка Laravel12 57](#_Toc200562061)

[Создание миграций 58](#_Toc200562062)

[Создание исключений 61](#_Toc200562063)

[Создание моделей 62](#_Toc200562064)

[Создание контроллеров 67](#_Toc200562065)

[Валидация запроса формы 85](#_Toc200562066)

[Маршрутизация 92](#_Toc200562067)

[ГЛАВА 4. АВТОТЕСТИРОВАНИЕ API 94](#_Toc200562068)

[ГЛАВА 5. РЕАЛИЗАЦИЯ ФУНКЦИОНАЛА ВЕБ-ПРИЛОЖЕНИЯ 95](#_Toc200562069)

[Реализация API и вспомогательных модулей 95](#_Toc200562070)

[Реализация управления состоянием (Vuex Store) 96](#_Toc200562071)

[Реализация пользовательского интерфейса 99](#_Toc200562072)

[ГЛАВА 6. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 135](#_Toc200562073)

[ГЛАВА 7. РУКОВОДСТВО МЕНЕДЖЕРА 137](#_Toc200562074)

[ЗАКЛЮЧЕНИЕ 138](#_Toc200562075)

[СПИСОК ЛИТЕРАТУРЫ 139](#_Toc200562076)

# ВВЕДЕНИЕ

Актуальность проекта:

Электронная коммерция стремительно развивается, предлагая пользователям удобные способы совершения покупок в интернете. Восточная мода, включающая японскую, корейскую и китайскую одежду, становится всё более популярной благодаря уникальному дизайну и высоким стандартам качества. Однако на рынке представлено ограниченное количество специализированных магазинов, предлагающих широкий ассортимент восточной одежды с удобной системой заказов и доставки.

Разработка интернет-магазина «Тайны Востока», ориентированного на продажу одежды из стран Азии, актуальна по следующим причинам:

* Повышенный интерес к азиатскому стилю: Восточная одежда набирает популярность среди ценителей уникального дизайна и современных трендов.
* Рост онлайн-покупок: Все больше пользователей предпочитают удобные интернет-магазины с продуманной системой доставки.
* Оптимизация логистики: Внедрение системы пунктов выдачи заказов позволяет снизить затраты на доставку и ускорить получение товаров.
* Цифровая трансформация бизнеса: Интеграция REST API и автоматизация бизнес-процессов обеспечивают удобное управление заказами, товарами и взаимодействием с клиентами.

Целью дипломного проекта является проектирование и разработка REST API сервера на фреймворке Laravel и сайта «тайны востока» на Vue:

* Удобный каталог товаров с возможностью поиска и фильтрации по категориям, брендам и странам производства.
* Систему оформления заказов с выбором пунктов выдачи.
* Безопасную и гибкую систему оплаты.
* Функциональность личного кабинета для управления заказами и оставления отзывов.

Задачи проекта:

1. Разработка REST API на Laravel для обработки заказов, управления корзиной, пользователями и товарами.
2. Создание веб-интерфейса на Vue, обеспечивающего адаптивность и удобство использования.
3. Реализация механизма аутентификации и личного кабинета с историей заказов.
4. Интеграция пунктов выдачи с выбором удобного места для получения заказов.
5. Разработка системы отзывов для оценки товаров пользователями.

# ТЕХНОЛОГИИ И ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

## Используемые языки программирования и фреймворки

PHP - это широко используемый язык программирования общего назначения, который применяется в основном для разработки веб-приложений и веб-серверных скриптов. PHP обычно встраивается непосредственно в HTML код, и его скрипты выполняются на сервере, что позволяет генерировать динамический контент на веб-страницах.

Основные характеристики PHP включают в себя:

* Простота использования: PHP имеет синтаксис, который легко читать и понимать, что делает его доступным для новичков.
* Гибкость: PHP поддерживает различные парадигмы программирования, включая процедурное и объектно-ориентированное программирование, что позволяет разработчикам выбирать подход, который лучше всего подходит для их задач.
* Широкая поддержка: PHP является одним из наиболее популярных языков программирования для веб-разработки и имеет обширное сообщество разработчиков, библиотек и фреймворков.
* Интеграция с веб-серверами: PHP легко интегрируется с различными веб-серверами, такими как Apache, Nginx и другие.
* Динамическое создание контента: PHP позволяет создавать динамические веб-страницы, которые могут меняться в зависимости от пользовательского ввода, данных из базы данных и других факторов.
* Базы данных: PHP обладает встроенной поддержкой для работы с различными системами управления базами данных, такими как MySQL, PostgreSQL, SQLite и другими.
* PHP используется для создания широкого спектра веб-приложений, включая блоги, интернет-магазины, корпоративные веб-сайты, социальные сети и многое другое. Он остается одним из наиболее популярных и распространенных языков программирования для веб-разработки благодаря своей простоте, гибкости и широкой поддержке сообщества.

SQL - это специализированный язык программирования, который используется для работы с реляционными базами данных. Он предоставляет набор команд и инструкций для управления данными в базах данных, таких как добавление, удаление, изменение и извлечение данных.

Основные характеристики SQL:

* Декларативный язык: SQL является декларативным языком программирования, что означает, что вы определяете, что вы хотите сделать с данными, а не как это делать. Вы описываете желаемый результат, а не последовательность действий.
* Манипуляция данными: SQL предоставляет команды для манипуляции данными в базе данных. Например, с помощью SQL вы можете создавать новые таблицы, добавлять новые записи, обновлять существующие записи и удалять записи.
* Язык запросов: для извлечения данных из базы данных. Вы можете создавать запросы, чтобы выбирать данные из одной или нескольких таблиц, фильтровать данные, сортировать их и группировать.

Laravel - это бесплатный и с открытым исходным кодом фреймворк для создания веб-приложений на языке программирования PHP. Он предоставляет разработчикам инструменты и структуру для быстрой и эффективной разработки веб-приложений, используя современные практики разработки и популярные шаблоны проектирования.

Основные характеристики Laravel:

* Элегантный синтаксис: Laravel предлагает простой и выразительный синтаксис, который делает код легким для чтения и понимания.
* Мощные функции: Laravel обладает богатым набором функций, включая маршрутизацию, контроллеры, валидацию данных, работу с базой данных, аутентификацию пользователей, кэширование, отправку электронной почты и многое другое.
* Быстрый старт: Laravel предлагает множество готовых компонентов и инструментов, которые позволяют быстро начать работу над проектом, не тратя время на настройку инфраструктуры.

Vue.js — это прогрессивный фреймворк для создания пользовательских интерфейсов и одностраничных приложений (SPA). Он используется для разработки клиентской части (фронтенда) веб-приложений и позволяет создавать динамические и интерактивные сайты.

Основные особенности Vue.js:

* Компонентный подход — код разбивается на переиспользуемые компоненты, что упрощает разработку и поддержку.
* Реактивность — изменения данных автоматически обновляют пользовательский интерфейс.
* Гибкость — можно использовать как для отдельных элементов на странице, так и для создания сложных веб-приложений.
* Легкость в освоении — синтаксис понятен даже для начинающих разработчиков.
* Интеграция с другими технологиями — Vue можно легко использовать с Laravel, что удобно при разработке интернет-магазинов.

## Используемое программное обеспечение

Draw.io — это бесплатный онлайн инструмент для создания диаграмм и схем. Он предоставляет широкий набор инструментов и элементов, которые позволяют пользователям создавать различные типы диаграмм, такие как организационные диаграммы, блок-схемы, ER-диаграммы, сетевые диаграммы, диаграммы потока процессов и многое другое.

Open Server Panel — это бесплатное программное обеспечение для Windows, которое предоставляет локальную среду разработки для веб-приложений. Он включает в себя веб-сервер Apache, базу данных MySQL, сервер приложений PHP и другие инструменты, необходимые для разработки и тестирования веб-приложений.

PhpStorm — это интегрированная среда разработки (IDE) для языка программирования PHP, разработанная компанией JetBrains. PhpStorm предоставляет разработчикам мощные инструменты для написания кода, отладки, тестирования и поддержки веб-приложений на PHP.

Postman — это популярный инструмент для разработки, тестирования и документирования API. Он предоставляет разработчикам удобный способ отправлять HTTP-запросы к веб-сервисам и анализировать ответы.

Figma — это облачный инструмент для дизайна пользовательских интерфейсов (UI) и совместной работы дизайнеров, разработчиков и других участников команды. Он позволяет создавать макеты сайтов, мобильных приложений, интерактивные прототипы и работать над дизайном в режиме реального времени.

# ГЛАВА 1. ТЕХНИЧЕСКОЕ ЗАДАНИЕ НА РАЗРАБОТКУ ИНФОРМАЦИОННОГО ПРОДУКТА

## Описание проекта и задач

**ОБЩЕЕ ОПИСАНИЕ ПРОЕКТА**

В условиях стремительного развития цифровых технологий и роста популярности онлайн-торговли, создание удобной, надёжной и функциональной платформы для интернет-магазина становится важной задачей. Проект "Тайны Востока" представляет собой специализированный веб-сервис по продаже одежды из азиатских стран (Японии, Кореи, Китая), ориентированный на широкий круг пользователей.

Разрабатываемая информационная система призвана обеспечить удобный доступ к товарам, упростить процесс оформления заказов, а также обеспечить эффективную работу персонала и администрирование магазина. Основной упор сделан на создание гибкой архитектуры, построенной по принципам клиент-серверного взаимодействия с использованием REST API.

В рамках проекта предусмотрено чёткое разграничение прав и функциональных возможностей пользователей. Система разделяет пользователей на три основные группы: клиентов, менеджеров и администраторов, каждая из которых выполняет определённые функции в рамках платформы.

**Функционал неавторизированного пользователя:**

1. Авторизация.
2. Регистрация.
3. Просмотр всех товаров.
4. Просмотр конкретного товара.
5. Фильтр товаров по категории, стране, цене и полу.
6. Поиск товаров по описанию или названию.
7. Просмотр категорий.
8. Просмотр стран.
9. Просмотр отзывов у товаров.
10. Просмотр пунктов выдачи.
11. Просмотр цветов.
12. Просмотр размеров.

**Функционал клиента:**

1. Просмотр, редактирование своей корзины.
2. Добавление и удаление товаров из своей корзины.
3. Оформление, просмотр, отмена своих заказов.
4. Просмотр конкретных заказов и их статуса.
5. Просмотр, добавление и удаление своих отзывов на купленные товары.

**Функционал менеджера:**

1. Просмотр, создание (с возможностью добавить новый цвет и размер), редактирование и удаление товаров.
2. Просмотр, редактирование (изменение статуса) и отмена заказов.
3. Просмотр всех заказов.
4. Удаление отзывов клиентов.

**Функционал администратора:**

1. Просмотр, редактирование всех пользователей.
2. Просмотр конкретного пользователя.
3. Создание категорий.
4. Просмотр, регистрация и удаление менеджеров.
5. Просмотр, создание (с возможностью добавить новый цвет и размер), редактирование и удаление товаров.
6. Просмотр, редактирование (изменение статуса) и отмена заказов.
7. Добавление, редактирование и удаление пункта выдачи.
8. Удаление отзывов клиентов.

**Функционал авторизированного пользователя:**

1. Выход из системы.
2. Просмотр и редактирование своего профиля.

**ОБЩИЕ ТРЕБОВАНИЯ**

Сервер не может или не будет обрабатывать запрос из-за чего-то, что очевидно является ошибкой клиента:

|  |
| --- |
| Неверный формат запроса |
| "code": 400,  "message": "Bad Request" |

Для доступа к ресурсу требуется аутентификация, и клиент не предоставил действительные учетные данные:

|  |
| --- |
| Не авторизован |
| «code»: 401,  «message»: «Unauthorized» |

Сервер понял запрос, но отказывается его авторизовать. Это может быть связано с отсутствием необходимых прав доступа:

|  |
| --- |
| Доступ запрещен |
| «code»: 403,  «message»: «Forbidden» |

Сервер не может найти запрошенный ресурс. Это может быть связано с неправильным URL или удалением ресурса:

|  |
| --- |
| Ресурс не найден |
| «code»: 404,  «message»: «Not Found» |

В случае ошибок, связанных с валидацией данных во всех запросах необходимо возвращать следующее:

|  |
| --- |
| Тело ответа |
| “code”: <code>,  “message”: <message> |

Вместо <code> и <message> необходимо указывать соответствующее значение, определенное в описании ответа на соответствующий запрос.

## Ролевая модель

Исходя из основных требований функциональности системы можно построить таблицу, разделяя роли и функциональность (см. таблицу 1):

Таблица 1. Ролевая модель

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Раздел** | **Функционал** | **Гость** | **Клиент** | **Менеджер** | **Администратор** |
| Авторизация | Авторизация | Нет | Да | Да | Да |
| Регистрация | Да | Да | Нет | Нет |
| Выход | Нет | Да | Да | Да |
| Товары | Просмотр всех товаров | Да | Да | Да | Да |
| Создание товара | Нет | Нет | Да | Да |
| Просмотр товара | Да | Да | Да | Да |
| Обновление товара | Нет | Нет | Да | Да |
| Удаление товара | Нет | Нет | Да | Да |
| Профиль | Просмотр профиля | Нет | Да | Да | Да |
| Редактирование профиля | Нет | Да | Да | Да |
| Фильтры | По параметрам | Да | Да | Да | Да |
| По названию или описанию | Да | Да | Да | Да |
| Страны | Просмотр всех стран | Да | Да | Да | Да |
| Категории | Просмотр всех категорий | Да | Да | Да | Да |
| Создание категории | Нет | Нет | Да | Да |
| Отзывы | Просмотр всех отзывов по товару | Да | Да | Да | Да |
| Создание отзыва | Нет | Да | Нет | Нет |
| Удаление отзыва | Нет | Да | Да | Да |
| Пункты выдачи | Просмотр всех пунктов выдачи | Да | Да | Да | Да |
| Создание пункта выдачи | Нет | Нет | Нет | Да |
| Обновление пункта выдачи | Нет | Нет | Нет | Да |
| Удаление пункта выдачи | Нет | Нет | Нет | Да |
| Корзина | Просмотр корзины | Нет | Да | Нет | Нет |
| Добавление товара в корзину | Нет | Да | Нет | Нет |
| Обновление количества товара | Нет | Да | Нет | Нет |
| Удаление товара из корзины | Нет | Да | Нет | Нет |
| Заказы | Просмотр своих заказов | Нет | Да | Нет | Нет |
| Просмотр всех заказов | Нет | Нет | Да | Да |
| Создание заказа | Нет | Да | Нет | Нет |
| Просмотр своего заказа | Нет | Да | Нет | Нет |
| Просмотр чужого заказа | Нет | Нет | Да | Да |
| Обновление заказа | Нет | Нет | Да | Да |
| Отмена заказа | Нет | Да | Да | Да |
| Менеджеры | Просмотр всех менеджеров | Нет | Нет | Нет | Да |
| Создание менеджера | Нет | Нет | Нет | Да |
| Удаление менеджера | Нет | Нет | Нет | Да |
| Пользователи | Просмотр всех пользователей | Нет | Нет | Нет | Да |
| Просмотр пользователя | Нет | Нет | Нет | Да |
| Обновление пользователя | Нет | Нет | Нет | Да |
| Статусы | Просмотр всех статусов | Да | Да | Да | Да |

## Специфические требования к API

**Авторизация**

Запрос для аутентификации пользователя в системе. При отправке запроса необходимо передать объект с почтой и паролем. Если пользователь отправил корректные данные, то необходимо вернуть сгенерированный токен, а иначе сообщение об ошибке.

|  |
| --- |
| HTTP-метод и URL |
| Метод: post  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/login |
| Заголовки |
| Content-Type: application/json  Accept: application/json |
| Тело запроса |
| {     "password": "Password123@",     "email": "abdul@mail.ru"  } |
| Ответ на успешную авторизацию (200 OK) |
| {      "token": "$2y$12$22NQ9pW9FwbHX3tTWmShpurXq34vH7FGpogPBzQEykHYtE3fhREHq",      "user": {          "id": 9,          "name": "Абдул Абдурихманович",          "email": "abdul@mail.ru",          "telephone": "88005553509",          "sex": 1,          "role\_id": 3,          "created\_at": "2025-05-15T03:31:55.000000Z",          "updated\_at": "2025-06-11T06:08:34.000000Z"      }  } |
| Ошибка авторизации (401 Unauthorized) |
| {      "code": 401,      "message": "Неверная почта или пароль"  } |

**Регистрация**

Запрос для регистрации клиента в системе. При отправке запроса необходимо передать объект с именем, полом, номером телефона, почтой и паролем. Если пользователь отправил корректные данные, то необходимо вернуть сгенерированный токен, а иначе сообщение об ошибке.

|  |
| --- |
| HTTP-метод и URL |
| Метод: post  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/register |
| Заголовки |
| Authorization: application/json  Accept: application/json |
| Тело запроса |
| {     "name": "Абдул Абдурихманови",     "email": "abdulchikп@mail.ru",     "telephone": "88005553535",     "sex": 1,     "password": "Lexa2019@"  } |
| Ответ на успешную регистрацию (201 OK) |
| {      "user": {          "name": "Абдул Абдурихманови",          "email": "abdulchikп@mail.ru",          "telephone": "88005553535",          "sex": 1,          "role\_id": 3,          "updated\_at": "2025-06-11T06:12:57.000000Z",          "created\_at": "2025-06-11T06:12:57.000000Z",          "id": 14      },      "token": "$2y$12$MSVG5IKxWXWgqxSowxP55OGPUfAWFMkgNuDhhX.ZWAFNeG4dWFwNi"  } |
| Ошибка валидации (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

**Выход**

Запрос предназначен для очистки значение токена пользователя.

|  |
| --- |
| HTTP-метод и URL |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/logout |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Тело запроса |
| {  token: $2y$12$hgKj3HcOQDcVzbfZhRILq/ZGXVUAW2mYn5meSUC  } |
| Ответ на успешный выход из системы (200 OK) |
| {      "message": "Вы вышли из системы"  } |

**CRUD-операции по товару**

Get-запрос к /api/product с токеном в теле запроса возвращает список товаров (200 ok) или сообщение о том, что товары не найдены (404 not found).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/product |
| Заголовки |
| Authorization: application/json  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| [      {          "id": 1,          "photo": "http://secrets-of-the-east.ru/storage/products/girl8.jpg",          "name": "Футболки и топы 1",          "description": "Описание товара",          "sex": 1,          "quantity": 6,          "price": "4302.00",          "category\_id": 3,          "country\_id": 1,          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 2,          "photo": "http://secrets-of-the-east.ru/storage/products/girl5.jpg",          "name": "Свитшоты и худи 2",          "description": "Описание товара",          "sex": 1,          "quantity": 5,          "price": "1655.00",          "category\_id": 2,          "country\_id": 3,          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      }  ] |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |

Post-запрос к /api/product с описанием товара в теле и токеном в теле запроса создаёт новый товар (200 ok), либо доступа (403 forbidden) или валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: post  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/product |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "name": "Футболка Premium",      "description": "Качественная хлопковая футболка",      "sex": 1,      "price": 1999,      "category\_id": 4,      "country\_id": 1,      "colors": [          {              "color\_id": 3,              "sizes": [                  {                      "size\_id": 1,                      "quantity": 10                  },                  {                      "size\_id": 2,                      "quantity": 15                  }              ]          },          {              "new\_color\_name": "Мятный",              "new\_color\_hex": "#A2F2D5",              "sizes": [                  {                      "new\_size\_name": "XL",                      "quantity": 8                  },                  {                      "size\_id": 2,                      "quantity": 12                  }              ]          }      ]  } |
| Ответ на успешное создание (200 OK) |
| {      "product": {          "photo": **null**,          "name": "Футболка Premium",          "description": "Качественная хлопковая футболка",          "sex": 1,          "quantity": 45,          "price": 1999,          "category\_id": 4,          "country\_id": 1,          "updated\_at": "2025-06-11T06:34:31.000000Z",          "created\_at": "2025-06-11T06:34:31.000000Z",          "id": 78      },      "message": "Товар успешно создан с цветами и размерами"  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "У вас нет прав на выполнение этого действия"  } |
| Ошибка валидации (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

Patch-запрос к /api/product/{id} с обновленным описанием товара и токеном в теле запроса обновляет существующий товар (200 ok) или доступа (403 forbidden), сообщение о том, что товар не найден (404 not found) или валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: patch  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/product/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "name": "Футболка Premium",      "description": "Качественная хлопковая футболка",      "sex": 1,      "price": 1999,      "category\_id": 4,      "country\_id": 1,      "colors": [          {              "color\_id": 3,              "sizes": [                  {                      "size\_id": 1,                      "quantity": 10                  },                  {                      "size\_id": 2,                      "quantity": 15                  }              ]          },          {              "new\_color\_name": "Мятный",              "new\_color\_hex": "#A2F2D5",              "sizes": [                  {                      "new\_size\_name": "XL",                      "quantity": 8                  },                  {                      "size\_id": 2,                      "quantity": 12                  }              ]          }      ]  } |
| Ответ на успешное обновление (200 OK) |
| {      "product": {          "id": 9,          "photo": "http://secrets-of-the-east.ru/storage/products/girl2.jpg",          "name": "Футболка Premium",          "description": "Качественная хлопковая футболка",          "sex": 1,          "quantity": 45,          "price": 1999,          "category\_id": 4,          "country\_id": 1,          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-06-11T06:37:56.000000Z",          "product\_color\_sizes": [              {                  "id": 582,                  "quantity": 10,                  "product\_id": 9,                  "color\_id": 3,                  "size\_id": 1,                  "created\_at": "2025-06-11T06:37:56.000000Z",                  "updated\_at": "2025-06-11T06:37:56.000000Z",                  "color": {                      "id": 3,                      "name": "Красный",                      "hex": "#FF0000",                      "created\_at": "2025-05-15T03:31:56.000000Z",                      "updated\_at": "2025-05-15T03:31:56.000000Z"                  },                  "size": {                      "id": 1,                      "name": "26-27",                      "created\_at": "2025-05-15T03:31:56.000000Z",                      "updated\_at": "2025-05-15T03:31:56.000000Z"                  }  }          ]      },      "message": "Товар успешно обновлен"  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "У вас нет прав на выполнение этого действия"  } |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |
| Ошибка валидации (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

Delete-запрос к /api/product/{id} с токеном в теле запроса удаляет товар (200 ok) или доступа (403 forbidden) или сообщение о том, что товар не найден (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: delete  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/product/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешное удаление (200 OK) |
| {      "message": "Товар успешно удален",      "success": **true**  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "У вас нет прав на выполнение этого действия"  } |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |

Get-запрос к /api/product/{id} с токеном в теле запроса возвращает товар по его id (200 ok) или сообщение о том, что товар не найден (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/product/{id} |
| Заголовки |
| Authorization: application/json  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| {      "id": 50,      "photo": "http://secrets-of-the-east.ru/storage/products/girl2.jpg",      "name": "Рубашки и блузы 50",      "description": "Описание товара",      "sex": 0,      "quantity": 9,      "price": "3602.00",      "category\_id": 4,      "country\_id": 2,      "created\_at": "2025-05-15T03:31:56.000000Z",      "updated\_at": "2025-05-15T03:31:56.000000Z",      "category": {          "id": 4,          "name": "Рубашки и блузы",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      "country": {          "id": 2,          "name": "Китай",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      "product\_color\_sizes": [          {              "id": 410,              "quantity": 6,              "product\_id": 50,              "color\_id": 6,              "size\_id": 6,              "created\_at": "2025-05-15T03:31:56.000000Z",              "updated\_at": "2025-05-15T03:31:56.000000Z",              "color": {                  "id": 6,                  "name": "Жёлтый",                  "hex": "#FFFF00",                  "created\_at": "2025-05-15T03:31:56.000000Z",                  "updated\_at": "2025-05-15T03:31:56.000000Z"              },              "size": {                  "id": 6,                  "name": "36-37",                  "created\_at": "2025-05-15T03:31:56.000000Z",                  "updated\_at": "2025-05-15T03:31:56.000000Z"              }          }      ]  } |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |

**CRUD-операции по профилю**

Get-запрос к /api/profile с токеном в теле запроса возвращает информацию об авторизированном пользователе (200 ok).

|  |
| --- |
| HTTP-метод и URL |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/profile |
| Заголовки |
| Authorization: application/json  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| {      "id": 10,      "name": "Алексей Петрович",      "email": "ruginpetr@mail.ru",      "telephone": "88005553510",      "sex": 1,      "role": "Пользователь",      "created\_at": "2025-05-15T03:31:55.000000Z",      "updated\_at": "2025-05-23T09:10:47.000000Z"  } |

Patch-запрос к /api/profile с описанием профилем авторизированного пользователя и токеном в теле запроса обновляет существующего пользователя (200 ok) или ошибку валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: patch  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/profile |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "name": "Лёха"  } |
| Ответ на успешное обновление (200 OK) |
| {      "message": "Профиль успешно обновлен",      "data": {          "id": 10,          "name": "Лёха",          "email": "ruginpetr@mail.ru",          "telephone": "88005553510",          "sex": 1,          "role\_id": 3,          "created\_at": "2025-05-15T03:31:55.000000Z",          "updated\_at": "2025-06-11T07:00:30.000000Z",          "role": {              "id": 3,              "name": "Пользователь",              "code": "user",              "created\_at": "2025-05-15T03:31:53.000000Z",              "updated\_at": "2025-05-15T03:31:53.000000Z"          }      }  } |
| Ошибка валидации (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

**Фильтрация**

Get-запрос к /api/products/filter возвращает товары по заданным параметрам (200 ok) или сообщение о том, что товары по заданным фильтрам не найдены (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/products/filter?country\_id=1&category\_id=1&min\_price=1000&max\_price=4000&sex=1 |
| Заголовки |
| Authorization: application/json  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| {      "success": **true**,      "filters": {          "category\_id": "1",          "country\_id": "1",          "min\_price": "1000",          "max\_price": "4000",          "sex": "1"      },      "count": 1,      "data": [          {              "id": 55,              "photo": "http://secrets-of-the-east.ru/storage/products/girl5.jpg",              "name": "Верхняя одежда 55",              "description": "Описание товара",              "sex": 1,              "quantity": 5,              "price": "3549.00",              "category\_id": 1,              "country\_id": 1,              "created\_at": "2025-05-15T03:31:56.000000Z",              "updated\_at": "2025-05-15T03:31:56.000000Z",              "category": {                  "id": 1,                  "name": "Верхняя одежда",                  "created\_at": "2025-05-15T03:31:56.000000Z",                  "updated\_at": "2025-05-15T03:31:56.000000Z"              },              "country": {                  "id": 1,                  "name": "Корея",                  "created\_at": "2025-05-15T03:31:56.000000Z",                  "updated\_at": "2025-05-15T03:31:56.000000Z"              }          }      ]  } |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Товары по заданным фильтрам не найдены"  } |

Get-запрос к /api/products/search с названием или описанием возвращает список товаров (200 ok) или сообщение о том, что товары не найдены (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/products/search |
| Заголовки |
| Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "q": "Футболки и топы 18"  } |
| Ответ на успешный просмотр (200 OK) |
| {      "success": **true**,      "query": "Футболки и топы 18",      "count": 1,      "data": [          {              "id": 18,              "photo": "http://secrets-of-the-east.ru/storage/products/girl1.jpg",              "name": "Футболки и топы 18",              "description": "Описание товара",              "sex": 0,              "quantity": 8,              "price": "3543.00",              "category\_id": 3,              "country\_id": 3,              "created\_at": "2025-05-15T03:31:56.000000Z",              "updated\_at": "2025-05-15T03:31:56.000000Z",              "category": {                  "id": 3,                  "name": "Футболки и топы",                  "created\_at": "2025-05-15T03:31:56.000000Z",                  "updated\_at": "2025-05-15T03:31:56.000000Z"              },              "country": {                  "id": 3,                  "name": "Япония",                  "created\_at": "2025-05-15T03:31:56.000000Z",                  "updated\_at": "2025-05-15T03:31:56.000000Z"              }          }      ]  } |
| Ошибка поиска (404 Not Found) |
| {      "message": "Товары по вашему запросу не найдены"  } |

**CRUD-операции по стране**

Get-запрос к /api/country с токеном в теле запроса возвращает список стран (200 ok) или сообщение о том, что страны не найдены (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/country |
| Заголовки |
| Accept: application/json  Content-Type: application/json |
| Ответ на успешный просмотр (200 OK) |
| {  “id”: 1,  “name”: “Китай”,  “created\_at”: “2025-02-26T10:15:30.000000Z”,  “updated\_at”: “2025-02-26T10:15:30.000000Z”  },  {  “id”: 2,  “name”: “Япония”,  “created\_at”: “2025-02-26T10:15:30.000000Z”,  “updated\_at”: “2025-02-26T10:15:30.000000Z”  },  {  “id”: 3,  “name”: “Корея”,  “created\_at”: “2025-02-26T10:15:30.000000Z”,  “updated\_at”: “2025-02-26T10:15:30.000000Z”  } |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |

**CRUD-операции по категории**

Get-запрос к /api/category с токеном в теле запроса возвращает список категорий одежды (200 ok) или сообщение о том, что категории одежды не найдены (404 not found).

|  |
| --- |
| Метод: get  Url: [http://secrets-of-the-east.ru/api/category](http://secrets-of-the-east.ru/api/categoryff) |
| Заголовки |
| Authorization: application/json  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| [      {          "id": 1,          "name": "Верхняя одежда",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 2,          "name": "Свитшоты и худи",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 3,          "name": "Футболки и топы",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 4,          "name": "Рубашки и блузы",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 5,          "name": "Джинсы и брюки",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 6,          "name": "Спортивная одежда",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 7,          "name": "Топчик",          "created\_at": "2025-06-04T09:47:21.000000Z",          "updated\_at": "2025-06-04T09:47:21.000000Z"      },      {          "id": 8,          "name": "Топчик",          "created\_at": "2025-06-04T12:02:13.000000Z",          "updated\_at": "2025-06-04T12:02:13.000000Z"      }  ] |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |

Post-запрос к /api/category с названием категории одежды в теле и токеном в теле запроса создаёт новую категорию одежды (200 ok), либо возвращает ошибку доступа (403 forbidden) или валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: post  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/category |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "name": "Новая категория"  } |
| Ответ на успешное создание (200 OK) |
| {      "name": "Новая категория",      "updated\_at": "2025-06-11T07:39:47.000000Z",      "created\_at": "2025-06-11T07:39:47.000000Z",      "id": 9  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "У вас нет прав на выполнение этого действия"  } |
| Ошибка валидации (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

**CRUD-операции по отзыву**

Get-запрос к /api/product/{id}/review с токеном в теле запроса возвращает список отзывов по id товара (200 ok) или сообщение о том, что отзывы у выбранного товара не найдены (404 not found) или о том, что сам товар не найден (404 not found).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/product/{id}/review |
| Заголовки |
| Authorization: application/json  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| [      {          "id": 11,          "rating": 4,          "description": "хихиххи",          "user\_id": 1,          "product\_id": 1,          "created\_at": "2025-05-15T11:44:14.000000Z",          "updated\_at": "2025-05-15T11:44:14.000000Z",          "user": {              "id": 1,              "name": "Дурачок"          }      },      {          "id": 1,          "rating": 5,          "description": "Идеальная куртка! Смотрится дороже своей цены, все швы аккуратные. Носила всю зиму — не промокает.",          "user\_id": 3,          "product\_id": 1,          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z",          "user": {              "id": 3,              "name": "Иванов Иван Иванович"          }      }  ] |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Отзывы для данного товара не указаны"  } |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Товар не найден"  } |

Post-запрос к /api/product/{id}/review с описанием отзыва в теле и токеном в теле запроса создаёт новый отзыв (200 ok), либо ошибку плохого запроса (400 bad request), либо возвращает ошибку доступа (403 forbidden) или валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: post  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/product/{id}/review |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "rating": 5,      "description": "Хорошая вещь"  } |
| Ответ на успешное создание (201 Created) |
| {      "rating": 5,      "description": "Хорошая вещь",      "user\_id": 10,      "product\_id": "60",      "updated\_at": "2025-06-11T07:59:31.000000Z",      "created\_at": "2025-06-11T07:59:31.000000Z",      "id": 12  } |
| Ошибка доступа (400 Bad Request) |
| {      "message": "Вы уже оставили отзыв на этот товар"  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "Вы можете оставить отзыв только после покупки и завершения заказа"  } |
| Ошибка валидации (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

Delete-запрос к /api/product/{productId}/review/{reviewId} с токеном в теле запроса удаляет категорию одежды (200 ok) или возвращает ошибку плохого запроса (400 bad request) или возвращает ошибку доступа (403 forbidden) или сообщение о том, что отзыв не найден (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: delete  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/product/{productId}/review/{reviewId} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешное удаление (200 OK) |
| {      "message": "Отзыв успешно удален"  } |
| Ошибка авторизации (400 Bad Request) |
| {      "code": 400,      "message": "Этот отзыв не принадлежит указанному товару"  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "Недостаточно прав для удаления отзыва"  } |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Отзыв не найден"  } |

**CRUD-операции по пунктам выдачи**

Get-запрос к /api/point возвращает список пунктов выдачи (200 ok) или сообщение о том, что пунктов выдачи нет (404 not found).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/point |
| Заголовки |
| Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| [      {          "id": 1,          "city": "Томск",          "street": "Иркутский тракт",          "house": "175",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 2,          "city": "Томск",          "street": "Иркутский тракт",          "house": "173",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 3,          "city": "Москва",          "street": "Тверская",          "house": "10",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      }  ] |
| Ошибка поиска (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |

Post-запрос к /api/point с описанием адреса в теле запроса создаёт новый пункт выдачи (200 ok), либо ошибку плохого запроса (400 bad request), либо возвращает ошибку доступа (403 forbidden) или валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: post  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/point |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "city": "Город",      "street": "Улица",      "house": "Дом"  } |
| Ответ на успешное создание (201 Created) |
| {      "city": "Город",      "street": "Улица",      "house": "Дом",      "updated\_at": "2025-06-11T08:26:41.000000Z",      "created\_at": "2025-06-11T08:26:41.000000Z",      "id": 12  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "У вас нет прав на выполнение этого действия"  } |
| Ошибка валидации (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

Patch-запрос к /api/point/{id} с описанием пункта выдачи авторизированного пользователя и токеном в теле запроса обновляет существующий пункт выдачи (200 ok) или возвращает ошибку доступа (403 forbidden).

|  |
| --- |
| HTTP-метод и URL |
| Метод: patch  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/point/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "city": "Томск"  } |
| Ответ на успешное обновление (200 OK) |
| {      "id": 12,      "city": "Томск",      "street": "Улица",      "house": "Дом",      "created\_at": "2025-06-11T08:26:41.000000Z",      "updated\_at": "2025-06-11T08:30:40.000000Z"  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "У вас нет прав на выполнение этого действия"  } |
| Ошибка валидации (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

Delete-запрос к /api/point/{id} авторизированного пользователя и токеном в теле запроса удаляет существующий пункт выдачи (200 ok) или возвращает ошибку доступа (403 forbidden) или возвращает ошибку валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: delete  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/point/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Ответ на успешное удаление (200 OK) |
| {      "message": "Пункт выдачи удалён."  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "У вас нет прав на выполнение этого действия"  } |

**CRUD-операции по корзине**

Get-запрос к /api/cart с токеном в теле запроса возвращает список товаров корзины текущего пользователя (200 ok) или сообщение о том, что корзина пуста (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/cart |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| {      "total\_cost": 23585,      "created\_at": "2025-06-11T08:40:18.000000Z",      "updated\_at": "2025-06-11T08:40:18.000000Z",      "products": [          {              "id": 501,              "name": "Спортивная одежда 60",              "price": "4717.00",              "photo": "http://secrets-of-the-east.ru/storage/products/girl6.jpg",              "description": "Описание товара",              "color": "Белый",              "size": "30-31",              "quantity": 5          }      ]  } |
| Ошибка поиска (404 Not Found) |
| {      "message": "Корзина пуста"  } |

Post-запрос к /api/cart/product/{productColorSizeId} с указанием количества в теле запроса добавляет товар в корзину (200 ok), либо ошибку плохого запроса (400 bad request), либо возвращает ошибку доступа (403 forbidden) или валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: post  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/cart/product/{productColorSizeId} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "quantity": 5  } |
| Ответ на успешное добавление (200 OK) |
| {      "status": "success",      "message": "Товар добавлен в корзину"  } |
| Ошибка поиска (404 Not Found) |
| {      "message": "Товар не найден"  } |

Patch-запрос к /api/cart/product/{productColorSizeId} с указанием количества и токеном в теле запроса обновляет количество товара в корзине (200 ok) или возвращает ошибку поиска (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: patch  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/cart/product/{productColorSizeId} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "quantity": 5  } |
| Ответ на успешное обновление (200 OK) |
| {      "status": "success",      "message": "Количество товара в корзине успешно обновлено",      "data": {          "total\_cost": 36887      }  } |
| Ошибка поиска (404 Not Found) |
| {      "status": "error",      "message": "Товар не найден в корзине"  } |

Delete-запрос к /api/cart/product/{productColorSizeId} авторизированного пользователя и токеном в теле запроса удаляет товар из корзины (200 ok) или возвращает ошибку поиска (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: delete  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/cart/product/{productColorSize} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Ответ на успешное удаление (200 OK) |
| {      "status": "success",      "message": "Товар успешно удалён из корзины",      "data": {          "total\_cost": 32170,          "remaining\_items": 1      }  } |
| Ошибка поиска (404 Not Found) |
| {      "status": "error",      "message": "Указанный товар не найден в вашей корзине"  } |

**CRUD-операции по заказам**

Get-запрос к /api/order с токеном в теле запроса возвращает список заказов текущего пользователя или список всех заказов (в зависимости, кто выполнил запрос) (200 ok) или сообщение о том, что заказы отсутсвуют (404 not found).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/order |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| {      "status": "success",      "data": [          {              "id": 11,              "order\_date": "2025-06-04 11:57:30",              "status": "Отменён",              "total\_cost": 47170,              "user": "Дурачок",              "address": {                  "city": "Томск",                  "street": "Иркутский тракт",                  "house": "175"              },              "items\_count": 2          },          {              "id": 12,              "order\_date": "2025-06-11 07:57:15",              "status": "Доставлен",              "total\_cost": 23585,              "user": "Лёха",              "address": {                  "city": "Томск",                  "street": "Иркутский тракт",                  "house": "175"              },              "items\_count": 1          }      ]  } |
| Ошибка поиска (для клиента) (404 Not Found) |
| {      "message": "У вас пока нет заказов"  } |
| Ошибка поиска (для менеджеров и администраторов) (404 Not Found) |
| {      "message": "В системе пока нет заказов"  } |

Post-запрос к /api/order с указанием пункта выдачи в теле запроса возвращает ссылку на сайт для тестовой оплаты (200 ok), либо ошибку плохого запроса (400 bad request), либо возвращает ошибку доступа (403 forbidden) или валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: post  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/order |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {    "point\_id": 1  } |
| Ответ на успех (200 OK) |
| {      "status": "payment\_required",      "message": "Требуется подтверждение оплаты",      "payment\_url": "https://checkout.stripe.com/c/pay/cs\_test\_a1hQXany8puo5NJq3XRTnzQjwrbvHTHuKNIzKU5lLC0haLBxYY73rtKBff#fidkdWxOYHwnPyd1blpxYHZxWjA0V1U1ajRDVWJvMENWM0JMdkZxf2tJbHZNVU9kMDNfSn1wMW10XzwySVxgfGJQY3NyVzNidWlIU1Fta2BvTzZOV0ZnUlZIM19ib3F%2FS258TldIcDJiU3BsNTVpTXVpdm9AaycpJ2N3amhWYHdzYHcnP3F3cGApJ2lkfGpwcVF8dWAnPyd2bGtiaWBabHFgaCcpJ2BrZGdpYFVpZGZgbWppYWB3dic%2FcXdwYHgl",      "session\_id": "cs\_test\_a1hQXany8puo5NJq3XRTnzQjwrbvHTHuKNIzKU5lLC0haLBxYY73rtKBff"  } |
| Ошибка плохого запроса (400 Bad Request) |
| {      "status": "error",      "message": "Не указан адрес доставки."  } |
| Ошибка плохого запроса (400 Bad Request) |
| {      "status": "error",      "message": "Корзина пуста. Добавьте товары для оформления заказа."  } |

Patch-запрос к /api/order/{id} с указанием нового статуса и токеном в теле запроса обновляет статус заказа (200 ok) или возвращает ошибку поиска (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: patch  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/order/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {      "status\_id": 1  } |
| Ответ на успешное обновление (200 OK) |
| {      "message": "Заказ обновлён",      "order": {          "id": 12,          "order\_date": "2025-06-11 07:57:15",          "total": "23585.00",          "user\_id": 10,          "status\_id": 1,          "point\_id": 1,          "created\_at": "2025-06-11T07:57:15.000000Z",          "updated\_at": "2025-06-11T09:19:31.000000Z"      }  } |
| Ошибка поиска (404 Not Found) |
| {      "error": "Заказ не найден"  } |

Patch-запрос к /api/order/cancelled/{id} с токеном в теле запроса обновляет статус заказа на «Отменён» (200 ok) или возвращает ошибку поиска (404 not found).

|  |
| --- |
| HTTP-метод и URL |
| Метод: patch  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/order/cancelled/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Ответ на успешное обновление (200 OK) |
| {      "message": "Заказ отменён"  } |
| Ошибка поиска (404 Not Found) |
| {      "error": "Заказ не найден"  } |

Get-запрос к /api/order/{id} с токеном в теле запроса возвращает информацию о заказе (в зависимости, кто выполнил запрос) (200 ok) или сообщение о том, что заказ не найден (404 not found).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/order/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| {      "id": 12,      "order\_date": "2025-06-11 07:57:15",      "status": "Отменён",      "total\_cost": 23585,      "user": "Лёха",      "address": {          "city": "Томск",          "street": "Иркутский тракт",          "house": "175"      },      "items": [          {              "product\_id": 60,              "photo": "http://secrets-of-the-east.ru/storage/products/girl6.jpg",              "name": "Спортивная одежда 60",              "price": "4717.00",              "quantity": 5,              "color": "Белый",              "size": "30-31",              "total": 23585          }      ]  } |
| Ошибка поиска (404 Not Found) |
| {      "error": "Заказ не найден"  } |

**CRUD-операции по менеджерам**

Get-запрос к /api/manager с токеном в теле запроса возвращает список менеджеров (200 ok) или сообщение о том, что у пользователя нет доступа (403 forbidden).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/manager |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| [      {          "id": 2,          "name": "Менеджер Отдела",          "email": "manager@mail.ru",          "telephone": "88005553502",          "role": "Менеджер",          "created\_at": "2025-05-15 03:31:53"      },      {          "id": 13,          "name": "manager",          "email": "manage@mail.ru",          "telephone": "88888888888",          "role": "Менеджер",          "created\_at": "2025-05-29 09:05:02"      }  ] |
| Ошибка доступа (403 Forbidden) |
| {      "message": "Доступ запрещен"  } |

Post-запрос к /api/manager с указанием информации о новом менеджере в теле запроса возвращает успешный ответ (200 ok), либо возвращает ошибку доступа (403 forbidden) или валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: post  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/manager |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {     "name": "Абдул Абдурихманови",     "email": "abdulchi@mail.ru",     "telephone": "88005553535",     "sex": 1,     "password": "Lexa2019@"  } |
| Ответ на успешное создание (201 Created) |
| {      "user": {          "name": "Абдул Абдурихманови",          "email": "abdulchi@mail.ru",          "telephone": "88005553535",          "sex": 1,          "role\_id": 2,          "updated\_at": "2025-06-11T09:40:14.000000Z",          "created\_at": "2025-06-11T09:40:14.000000Z",          "id": 15      },      "token": "$2y$12$cWz0fEZVYZTuLAXe.Ib16OvSV5/3ZXLhq8E20PS/o8FXhRHZo1hy2",      "message": "Менеджер успешно создан"  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "У вас нет прав на выполнение этого действия"  } |
| Ошибка плохого запроса (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

Delete-запрос к /api/manager/{id} авторизированного пользователя и токеном в теле запроса удаляет товар из корзины (200 ok) или возвращает ошибку плохого запроса (400 bad request) или возвращает ошибку доступа (403 forbidden).

|  |
| --- |
| HTTP-метод и URL |
| Метод: delete  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/manager/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Ответ на успешное удаление (200 OK) |
| {      "message": "Менеджер успешно удален",      "deleted\_manager": {          "id": 15,          "name": "Абдул Абдурихманови",          "email": "abdulchi@mail.ru",          "telephone": "88005553535",          "role": "Менеджер",          "created\_at": "2025-06-11 09:40:14"      }  } |
| Ошибка плохого запроса (400 Bad Request) |
| {      "message": "Можно удалять только менеджеров"  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "Недостаточно прав для удаления менеджера"  } |

**CRUD-операции по клиентам**

Get-запрос к /api/user с токеном в теле запроса возвращает список пользователей (200 ok) или сообщение о том, что у пользователя нет доступа (403 forbidden).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/user |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| [      {          "id": 1,          "name": "Дурачок",          "email": "admin@mail.ru",          "telephone": "88005553501",          "sex": 1,          "role": {              "id": 1,              "name": "Администратор",              "code": "admin"          },          "created\_at": "2025-05-15T03:31:53.000000Z",          "updated\_at": "2025-06-11T06:18:30.000000Z"      },      {          "id": 2,          "name": "Менеджер Отдела",          "email": "manager@mail.ru",          "telephone": "88005553502",          "sex": 1,          "role": {              "id": 2,              "name": "Менеджер",              "code": "manager"          },          "created\_at": "2025-05-15T03:31:53.000000Z",          "updated\_at": "2025-05-15T03:31:53.000000Z"      },  …  ] |
| Ошибка доступа (403 Forbidden) |
| {      "message": "Доступ запрещен"  } |

Get-запрос к /api/user/{id} с токеном в теле запроса возвращает пользователя (200 ok) или сообщение о том, что у пользователя нет доступа (403 forbidden).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/user/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| {      "id": 1,      "name": "Дурачок",      "email": "admin@mail.ru",      "telephone": "88005553501",      "sex": 1,      "role": {          "id": 1,          "name": "Администратор",          "code": "admin"      },      "created\_at": "2025-05-15T03:31:53.000000Z",      "updated\_at": "2025-06-11T06:18:30.000000Z"  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "Доступ запрещен"  } |

Patch-запрос к /api/user/{id} с указанием новой информации о пользователе в теле запроса возвращает успешный ответ (200 ok), либо возвращает ошибку доступа (403 forbidden) или валидации (422 unprocessable entity).

|  |
| --- |
| HTTP-метод и URL |
| Метод: patch  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/user/{id} |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json  Content-Type: application/json |
| Тело запроса |
| {     "name": "Абдул Абдурихманови",     "email": "abdulchi@mail.ru",     "telephone": "88005553535",     "sex": 1,     "password": "Lexa2019@"  } |
| Ответ на успешное создание (201 Created) |
| {      "user": {          "id": 1,          "name": "Абдул Абдурихманови",          "email": "abdulcfefefhi@mail.ru",          "telephone": "88005553535",          "sex": 1,          "role": {              "id": 1,              "name": "Администратор",              "code": "admin"          },          "created\_at": "2025-05-15T03:31:53.000000Z",          "updated\_at": "2025-06-11T11:10:08.000000Z"      },      "message": "Данные пользователя обновлены"  } |
| Ошибка доступа (403 Forbidden) |
| {      "message": "Доступ запрещен"  } |
| Ошибка плохого запроса (422 Unprocessable Entity) |
| {      "code": 422,      "message": "Ошибка валидации данных"  } |

**Вывод цветов, размеров и статусов**

Get-запрос к /api/colors с токеном в теле запроса возвращает список цветов (200 ok) или сообщение о том, что у пользователя нет доступа (403 forbidden).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/colors |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| [      {          "id": 1,          "name": "Чёрный",          "hex": "#000000",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 2,          "name": "Белый",          "hex": "#FFFFFF",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },  …  ] |
| Ошибка доступа (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |

Get-запрос к /api/sizes с токеном в теле запроса возвращает список размеров (200 ok) или сообщение о том, что у пользователя нет доступа (403 forbidden).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/sizes |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| [      {          "id": 1,          "name": "26-27",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 2,          "name": "28-29",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },  …  ] |
| Ошибка доступа (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |

Get-запрос к /api/statuses с токеном в теле запроса возвращает список статусов (200 ok) или сообщение о том, что у пользователя нет доступа (403 forbidden).

|  |
| --- |
| Метод: get  Url: [http://](http://courseproject)secrets-of-the-east.ru/api/statuses |
| Заголовки |
| Authorization: Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9  Accept: application/json |
| Ответ на успешный просмотр (200 OK) |
| [      {          "id": 1,          "name": "Принят",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },      {          "id": 2,          "name": "В сборке",          "created\_at": "2025-05-15T03:31:56.000000Z",          "updated\_at": "2025-05-15T03:31:56.000000Z"      },  …  ] |
| Ошибка доступа (404 Not Found) |
| {      "code": 404,      "message": "Не найдено"  } |

## Разработка прототипа пользовательского интерфейса

Прототип пользовательского интерфейса интернет-магазина «Тайны Востока» разработан для визуализации структуры и функционала будущего веб-сайта. Он включает главную страницу с каталогом товаров, систему фильтрации по категориям, страницу товара с подробным описанием, а также модули оформления заказа и личного кабинета. Макет ориентирован на удобство навигации, интуитивный дизайн и адаптивность для разных устройств.

***Главная страница***

При входе на сайт пользователь попадает на **главную страницу**, где сразу отображаются товары. Дизайн ориентирован на удобство навигации и быстрый доступ к нужным категориям.

***Верхняя панель навигации***

В верхней части сайта расположена **основная навигационная панель**, включающая:

**Справа**:

* **«Пункт выдачи»** – выбор удобного места получения заказа.
* **«Профиль»** – личный кабинет пользователя.
* **«Корзина»** – просмотр добавленных товаров и оформление заказа.

**Слева** (рядом с логотипом):

* **Кнопка «Инструменты»** – отображается только у **администраторов и менеджеров**. При нажатии открывается **выпадающее меню** с возможностью управления контентом сайта.

***Функционал кнопки «Инструменты»***

* **Менеджеры могут:**
  + Управлять товарами (создавать, редактировать, удалять).
  + Редактировать статусы заказов (подтверждение, отмена).
  + Просматривать все заказы.
  + Управлять отзывами (просмотр и удаление).
* **Администраторы могут:**
  + Управлять пользователями (просмотр, редактирование, регистрация и удаление менеджеров).
  + Полностью управлять товарами, заказами, странами, категориями одежды, цветами и размерами.

***Структура категорий***

Категории товаров расположены **в зоне товаров**, которое находится с лево:

**Слева от товаров** – категории по **типу одежды**: худи, толстовки, свитшоты, футболки и т. д. **размер цвет страна и фильтр по цене.**

**Поиск товаров**

В верхней части страницы предусмотрена **поисковая строка**, позволяющая пользователю быстро находить товары по названию, бренду или характеристикам.

Таким образом, главная страница объединяет удобную навигацию, фильтрацию товаров и инструменты управления. **Все элементы интерфейса представлены на рисунке 1.**

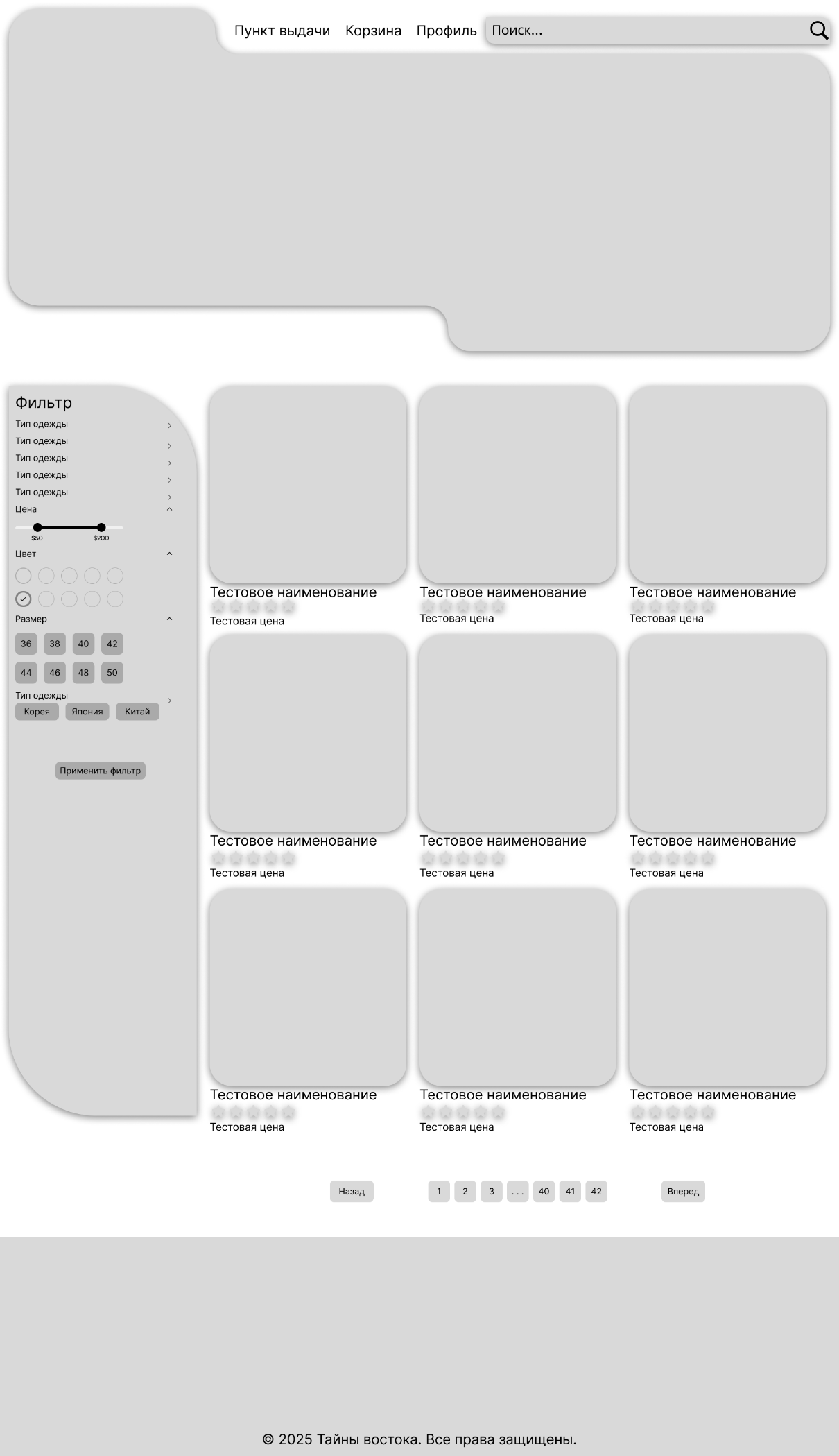


Рисунок - Главная страница

**Страница товара**

При нажатии на любой товар пользователь попадает на **страницу с подробной информацией о нем**. Данный раздел включает все ключевые данные, необходимые для принятия решения о покупке.

**Основная информация на странице товара**

**Фотография товара** – основное изображение, возможно с возможностью просмотра дополнительных фото.

* **Доступные размеры** – отображаются все имеющиеся размеры товара.
* **Пол** – указано, для кого предназначен товар (мужской, женский, унисекс).
* **Доступные цвета** – выбор цветовых вариаций товара.
* **Стоимость** – актуальная цена товара.
* **Оценка товара** – средний рейтинг на основе отзывов пользователей.
* **Комментарии от покупателей** – отзывы пользователей, которые уже приобрели данный товар.

**Функциональные элементы**

* **Кнопка «Добавить в корзину»** – позволяет покупателю быстро добавить товар в корзину для дальнейшего оформления заказа.
* **Функция удаления комментариев (для менеджеров)** – у каждого отзыва есть кнопка удаления, которая доступна только менеджерам. Если в комментарии содержится **ненормативная лексика**, менеджер может удалить его.

Таким образом, **страница товара** предоставляет пользователям полную информацию о продукте, а менеджерам – инструменты для модерирования отзывов. Данные элементы интерфейса **представлены на рисунке 2.**

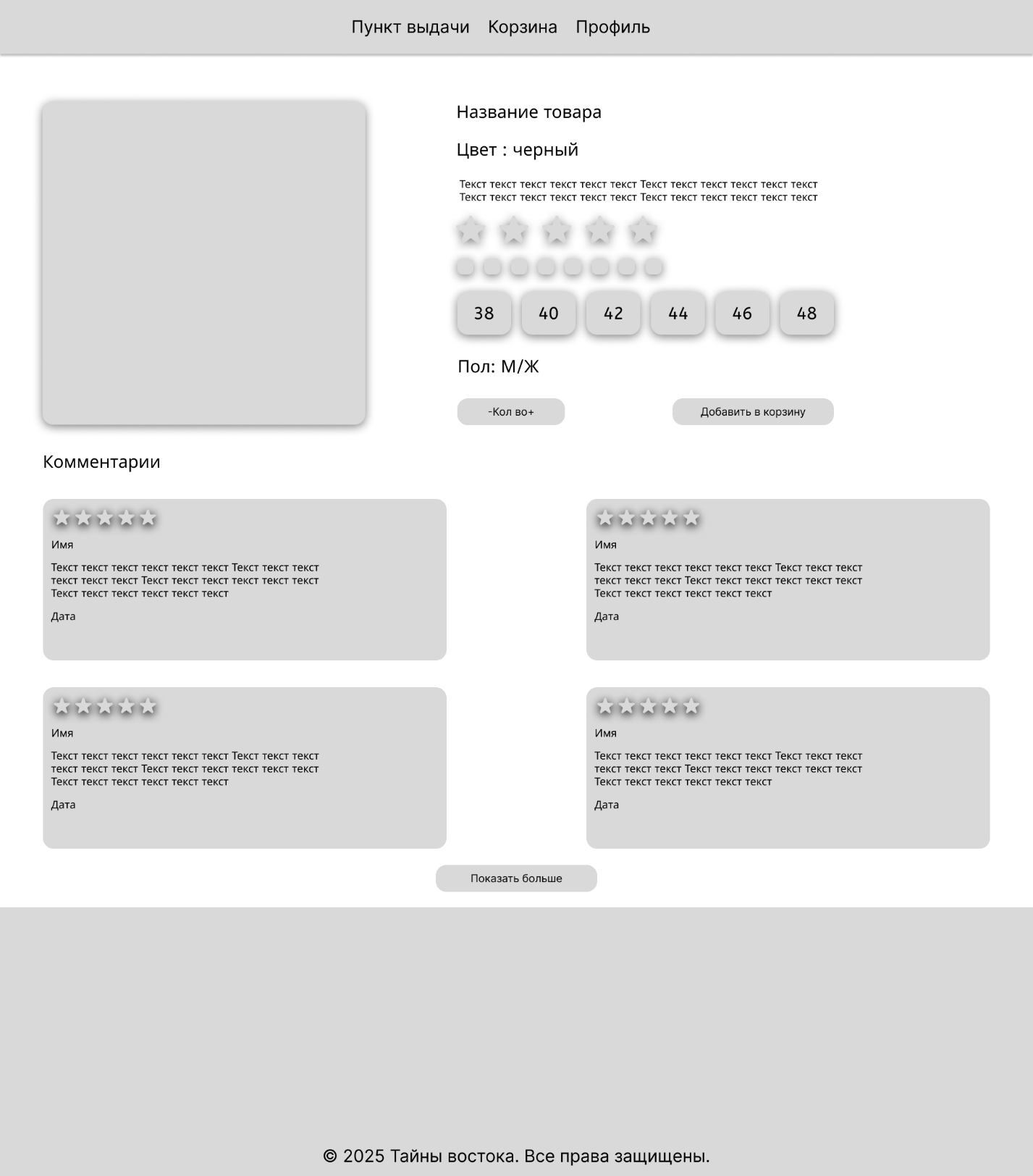


Рисунок 2 - Подробнее о товаре

**Страница корзины**

После добавления товара в корзину пользователь может перейти в **корзину**, нажав на кнопку **«Корзина»** в верхней панели навигации.

**Функционал страницы корзины**

На странице корзины отображаются:

* **Список добавленных товаров** – каждый товар с фото, названием, ценой и выбранными параметрами (размер, цвет).
* **Общее количество товаров** – отображается количество всех товаров в корзине.
* **Изменение количества товара** – возможность **увеличивать или уменьшать** количество товара.
* **Удаление товаров** – кнопка **«Удалить»** для удаления товара из корзины.
* **Выбор пункта выдачи** – позволяет выбрать место, куда будет доставлен заказ.

**Оформление заказа**

После проверки содержимого корзины пользователь нажимает **«Перейти к оформлению»**, после чего открывается **окно оплаты** (с выбором оплаты**)**, где нужно ввести данные банковской карты для завершения покупки. После успешной оплаты заказ подтверждается и обрабатывается, корзина представлена на рисунке 3.

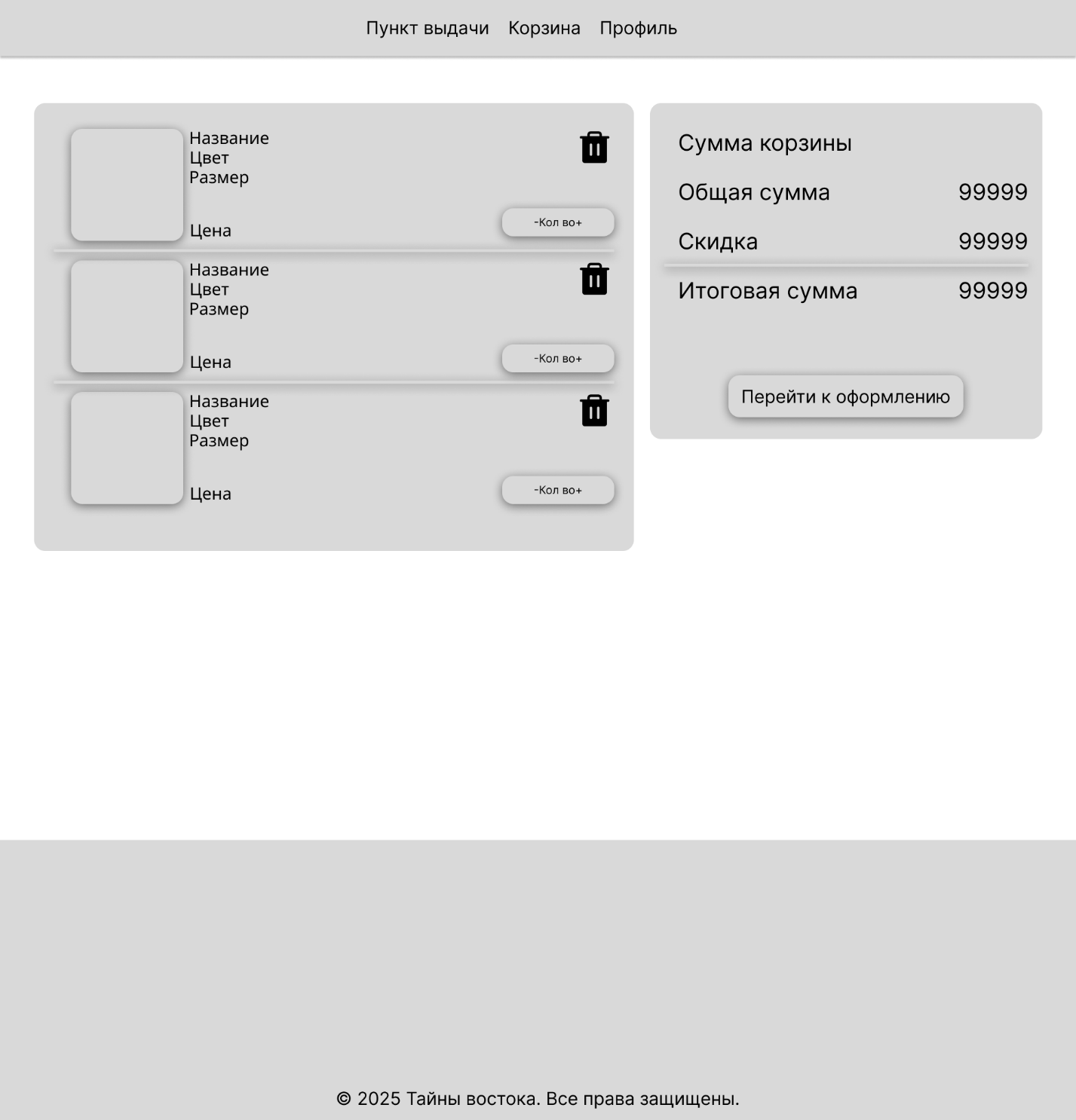


Рисунок 3 - Страница заказа

Модальное окно "Создание/Редактирование менеджера" (Рисунок 4) содержит идентичные формы с полями для ФИО, email, пароля и телефона, а также кнопка создания, обеспечивая единый интерфейс для регистрации и изменения данных менеджеров в системе.

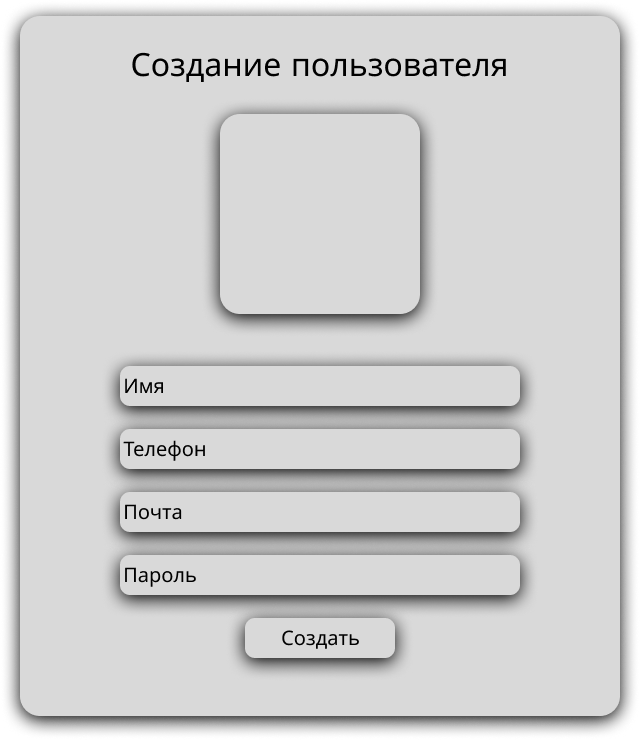


Рисунок 4- Создание и редактирование

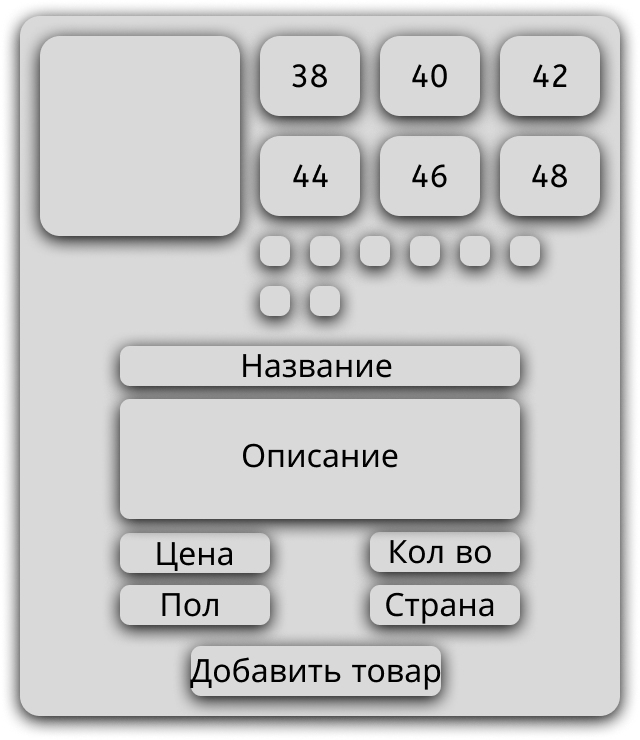
Модальное окно "Создание/редактирование товара" (Рисунок 5) содержит форму с полями для ввода названия, описания, цены и количества товара, выбором параметров (пол, категория, размеры, цвета), загрузкой фотографии и дополнительными полями (страна производства, артикул), сохраняя единый интерфейс для добавления и изменения позиций.

Рисунок 5 - Создание и редактирование одежды

Модальное окно "Управление заказами" (Рисунок 6) отображает список заказов с возможностью фильтрации и изменения их статусов (в обработке, отправлен, отменен) через выпадающее меню для каждого заказа, предоставляя менеджеру инструмент для оперативного контроля выполнения заказов.

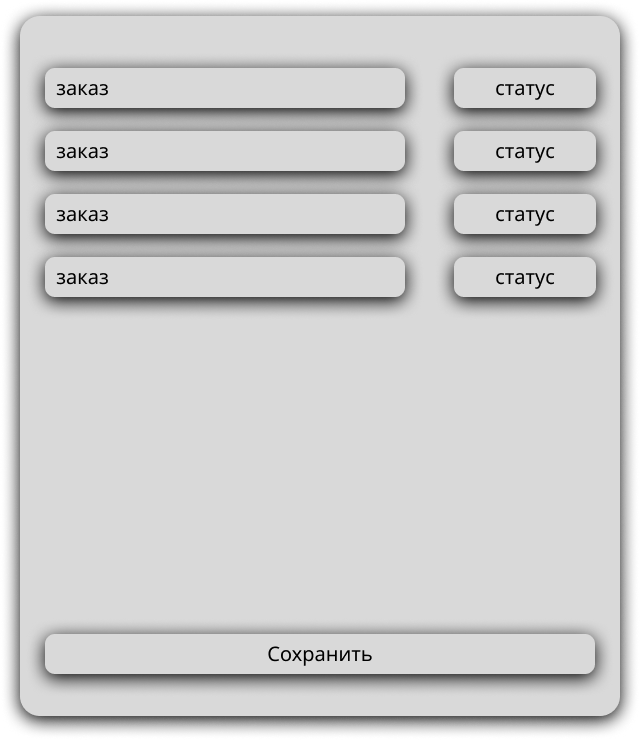


Рисунок 6 – Заказы

# ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ

## Нормализация данных

Нормальная форма — свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных. Нормальная форма определяется как совокупность требований, которым должно удовлетворять отношение.

Процесс преобразования отношений базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры БД к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных информации.

Устранение избыточности производится, как правило, за счёт декомпозиции отношений таким образом, чтобы в каждом отношении хранились только первичные факты.

Первая нормальная форма (1NF)

Переменная отношения находится в первой нормальной форме (1НФ) тогда и только тогда, когда в любом допустимом значении отношения каждый его кортеж содержит только одно значение для каждого из атрибутов.

Вторая нормальная форма (2NF)

Переменная отношения находится во второй нормальной форме тогда и только тогда, когда она находится в первой нормальной форме и каждый не ключевой атрибут неприводимо (функционально полно) зависит от её потенциального ключа. Функционально полная зависимость означает, что если потенциальный ключ является составным, то атрибут зависит от всего ключа и не зависит от его частей.

Третья нормальная форма (3NF)

Переменная отношения находится в третьей нормальной форме тогда и только тогда, когда она находится во второй нормальной форме, и отсутствуют транзитивные функциональные зависимости не ключевых атрибутов от ключевых. При проектировании БД третья нормальная форма схем отношений достаточна в большинстве случаев, и приведением к третьей нормальной форме процесс проектирования реляционной базы данных обычно заканчивается. Однако иногда полезно продолжить процесс нормализации.

## Концептуальная модель данных

Концептуальная модель – это отражение предметной области, для которой разрабатывается база данных. Не вдаваясь в теорию, отметим, что это некая диаграмма с принятыми обозначениями элементов. Все объекты, обозначающие вещи, обозначаются в виде прямоугольника. Атрибуты, характеризующие объект - в виде овала, а связи между объектами - ромбами. Мощность связи обозначаются стрелками (в направлении, где мощность равна многим - двойная стрелка, а со стороны, где она равна единице - одинарная).

Анализируя предметную область, можно выделить следующие сущности с их атрибутами (см. рисунок 6):

1. Сущность «Пользователи» с атрибутами «Имя», «Email», «Телефон», «Пол», «Токен».
2. Сущность «Роли» с атрибутами «Название», «Код».
3. Сущность «Заказы» с атрибутами «Дата и время заказа», «Общая стоимость».
4. Сущность «Статусы» с атрибутом «Название».
5. Сущность «Пункты выдачи» с атрибутами «Город», «Улица», «Дом».
6. Сущность «Составы заказов» с атрибутами «Количество», «Общая стоимость».
7. Сущность «Товары со цветом и размером» с атрибутом «Количество».
8. Сущность «Цвета» с атрибутами «Название», «HEX».
9. Сущность «Размеры» с атрибутом «Название».
10. Сущность «Товары» с атрибутами «Фото», «Название», «Описание», «Пол», «Цена».
11. Сущность «Категории» с атрибутом «Название».
12. Сущность «Страны» с атрибутом «Название».
13. Сущность «Отзывы» с атрибутами «Оценка», «Комментарий».
14. Сущность «Корзины» с атрибутами «Количество», «Общая стоимость».

## Логическая модель данных

Рисунок 6 - Концептуальная модель данных

Логическая модель базы данных — схема базы данных, выраженная в понятиях модели данных. Этим отличается от концептуальной модели, описывающей семантику предметной области без указания технологии (конкретных методов реализации), и от физической модели, которая описывает конкретные физические механизмы, применяемые для хранения данных в накопителях.

Получившиеся логическая модель данных представлена на рисунке 7.

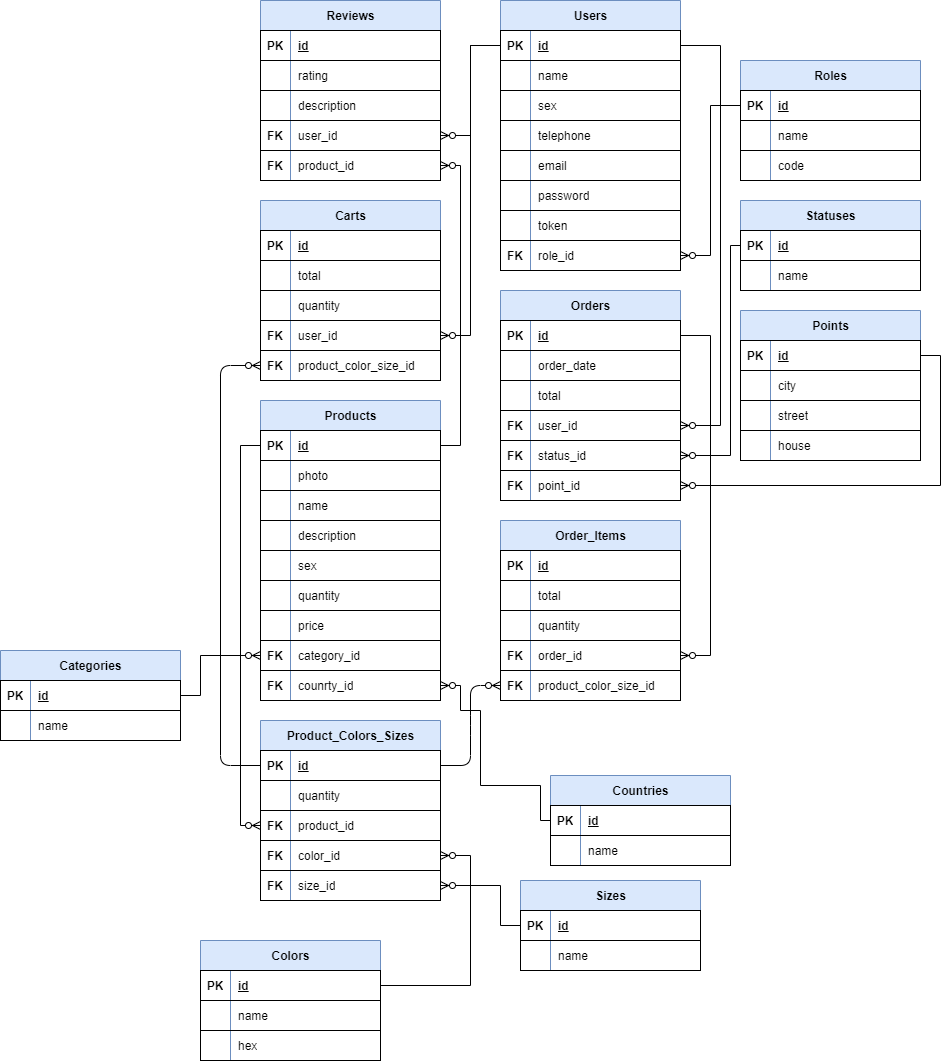


Рисунок 7 - Логическая модель данных

Физическая модель данных – это модель данных, описанная с помощью средств конкретной системы управления базами данных (СУБД). Физическая модель данных строится на базе даталогической путем добавления особенностей конкретной СУБД. К таким особенностям могут относиться поддерживаемые СУБД типы данных, соглашения о присвоении имен таблицам, атрибутам и т.д. Физическая модель данных фактически является готовым заданием на создание БД, имея которое можно реализовать БД в выбранной СУБД.

На рисунке 8 представленная физическая модель данных.

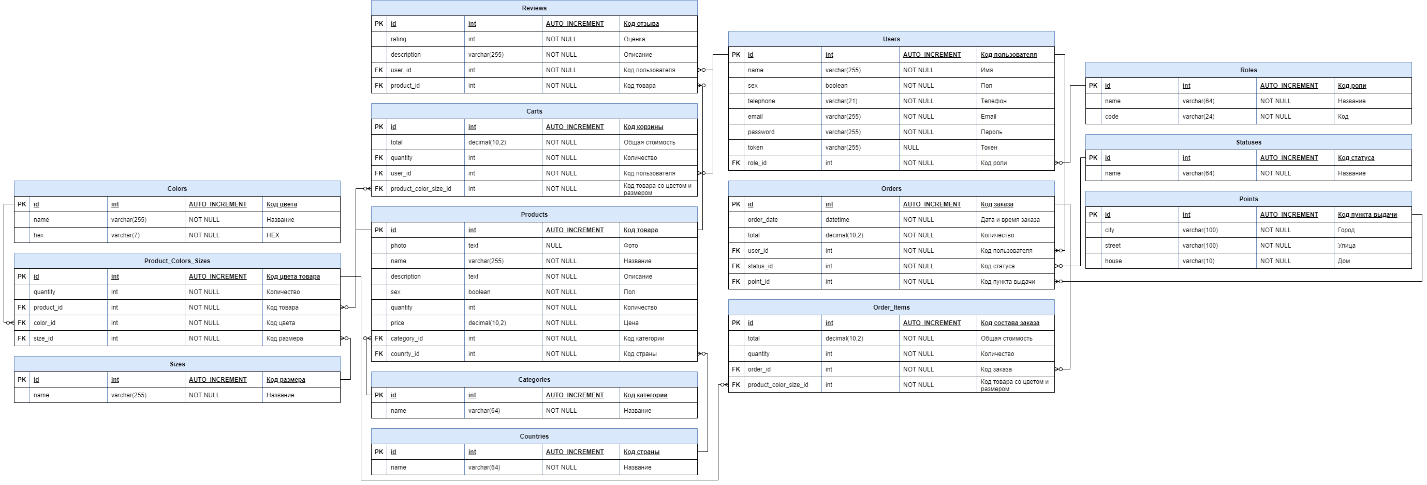


Рисунок 8 - Физическая модель данных

## Диаграмма базы данных

На рисунке 9 представлена итоговая диаграмма базы данных.

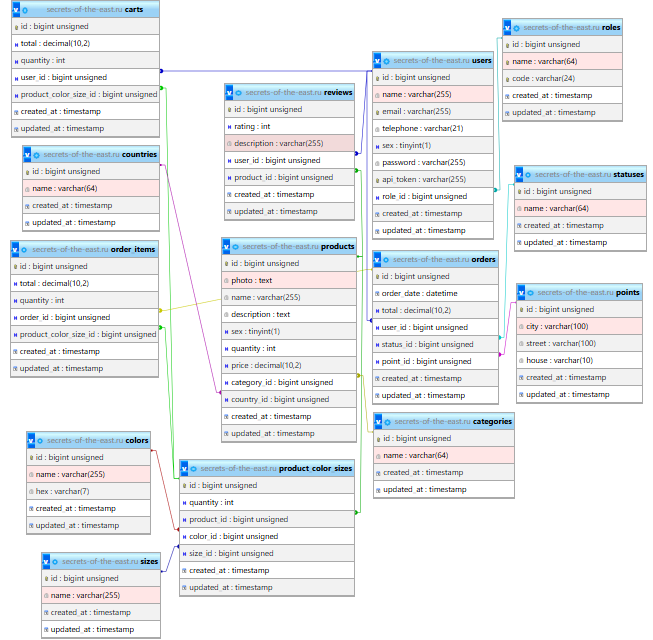


Рисунок 9 - Диаграмма базы данных

# ГЛАВА 3. ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ФУНКЦИОНАЛА ИНФОРМАЦИОННОГО ПРОДУКТА

## Установка и настройка фреймворка Laravel12

Для установки фреймворка требуется установка Composer: Laravel использует Composer для управления зависимостями. Можно скачать и установить Composer с официального сайта.

Установка Laravel с помощью Composer: после установки Composer перейдите в терминале в каталог, в который вы хотите установить Laravel, и выполните следующую команду: composer create-project laravel/laravel .

Где (.) – директория проекта. Composer создаст новый проект Laravel в указанном каталоге.

В корневой директории сайта создаем файл .htaccess и прописываем правила перенаправления запросов в точку входа в приложение (в папку public).

RewriteEngine On  
RewriteRule ^(.\*)$ public/$1 [L]

Этот фрагмент кода предназначен для настройки правил перезаписи URL веб-сервера Apache.

RewriteEngine On: - Эта строка включает модуль перезаписи модуля mod\_rewrite в Apache.

RewriteRule ^(.\*)$ public/$1 [L] - Это правило перезаписи, которое перенаправляет все запросы на веб-сервере к корневому каталогу /public. Знак (.\*)? соответствует любой строке в URL-адресе, а $1 обозначает часть URL, совпавшую с выражением в скобках. [L] - Этот флаг означает “если это правило совпало, остановите обработку дальнейших правил перезаписи”. Это важно, чтобы избежать бесконечных циклов перезаписи. Таким образом, эта строка перенаправляет все запросы на сервере к файлам в каталоге /public.

Параметры в файле .env, можно увидеть на листинге 1, являются настройками подключения к базе данных MySQL для проекта Laravel:

***Листинг 1. Настройки подключения к базе данных***

DB\_CONNECTION=mysql - Этот параметр указывает, какой вы используете драйвер базы данных для соединения с базой данных.  
DB\_HOST=127.0.0.1 - Здесь указывается хост базы данных.  
DB\_PORT=3306 - Этот параметр указывает порт.  
DB\_DATABASE= secrets-of-the-east.ru - Это имя базы данных.  
DB\_USERNAME=root - Это имя пользователя для доступа к базе данных MySQL.  
DB\_PASSWORD= - Это пароль для пользователя базы данных MySQL.

SESSION\_DRIVER=file - Это строка конфигурации, которая определяет, какой драйвер используется для хранения данных сессий в Laravel.

## Создание миграций

Миграции в Laravel - это способ создания и изменения структуры базы данных с использованием кода PHP вместо прямых SQL-запросов. Это помогает сохранить структуру базы данных в репозитории проекта и упростить совместную работу над базой данных.

Создаем миграции для таблиц fine, statuses, roles, bonuses, users, accesses для этого в корневой директории проекта выполняем следующие команды:

php artisan make: migration create\_roles\_table

php artisan make: migration create\_users\_table

php artisan make: migration create\_statuses\_table

php artisan make: migration create\_points\_table

php artisan make: migration create\_categories\_table

php artisan make: migration create\_countries\_table

php artisan make: migration create\_colors\_table

php artisan make: migration create\_sizes\_table

php artisan make: migration create\_products\_table

php artisan make: migration create\_reviews\_table

php artisan make: migration create\_orders\_table

php artisan make: migration create\_product\_color\_sizes\_table

php artisan make: migration create\_carts\_table

php artisan make: migration create\_order\_items\_table

Миграции похожи на контроль версий для вашей базы данных, позволяют вашей команде определять схемы базы данных приложения и совместно использовать их определение.

Фасад Schema обеспечивает независимую от базы данных поддержку для создания и управления таблицами во всех поддерживаемых Laravel системах баз данных. В обычной ситуации, этот фасад используется для создания и изменения таблиц / столбцов базы данных во время миграции. Итоговые миграции представлены в листинге 2.

**Листинг 2. Миграции базы данных**

Schema::create('roles', function (Blueprint $table) {  
 $table->id();  
 $table->string('name', 64)->unique();  
 $table->string('code', 24)->unique();  
 $table->timestamps();  
});

Schema::create('users', function (Blueprint $table) {  
 $table->id();  
 $table->string('name',255);  
 $table->string('email', 255)->unique();  
 $table->string('telephone', 21);  
 $table->boolean('sex');  
 $table->string('password',255);  
 $table->string('api\_token', 255)->nullable()->unique();  
 $table->foreignId('role\_id')->constrained();  
 $table->timestamps();  
});

Schema::create('statuses', function (Blueprint $table) {  
 $table->id();  
 $table->string('name', 64);  
 $table->timestamps();  
});

Schema::create('points', function (Blueprint $table) {  
 $table->id();  
 $table->string('city', 100);  
 $table->string('street', 100);  
 $table->string('house', 10);  
 $table->timestamps();  
});

Schema::create('categories', function (Blueprint $table) {  
 $table->id();  
 $table->string('name', 64);  
 $table->timestamps();  
});

Schema::create('countries', function (Blueprint $table) {  
 $table->id();  
 $table->string('name', 64);  
 $table->timestamps();  
});

Schema::create('colors', function (Blueprint $table) {  
 $table->id();  
 $table->string('name', 255);  
 $table->string('hex', 7);  
 $table->timestamps();  
});

Schema::create('sizes', function (Blueprint $table) {  
 $table->id();  
 $table->string('name', 255);  
 $table->timestamps();  
});

Schema::create('products', function (Blueprint $table) {  
 $table->id();  
 $table->text('photo')->nullable();  
 $table->string('name', 255);  
 $table->text('description');  
 $table->boolean('sex');  
 $table->integer('quantity');  
 $table->decimal('price', 10, 2);  
 $table->foreignId('category\_id')->constrained();  
 $table->foreignId('country\_id')->constrained();  
 $table->timestamps();  
});

Schema::create('reviews', function (Blueprint $table) {  
 $table->id();  
 $table->integer('rating');  
 $table->string('description', 255);  
 $table->foreignId('user\_id')->constrained();  
 $table->foreignId('product\_id')->constrained();  
 $table->timestamps();  
});

Schema::create('orders', function (Blueprint $table) {  
 $table->id();  
 $table->dateTime('order\_date');  
 $table->decimal('total', 10, 2);  
 $table->foreignId('user\_id')->constrained();  
 $table->foreignId('status\_id')->constrained();  
 $table->foreignId('point\_id')->constrained();  
 $table->timestamps();  
});

Schema::create('product\_color\_sizes', function (Blueprint $table) {  
 $table->id();  
 $table->integer('quantity');  
 $table->foreignId('product\_id')->constrained();  
 $table->foreignId('color\_id')->constrained();  
 $table->foreignId('size\_id')->constrained();  
 $table->timestamps();  
});

Schema::create('carts', function (Blueprint $table) {  
 $table->id();  
 $table->decimal('total', 10, 2);  
 $table->integer('quantity');  
 $table->foreignId('user\_id')->constrained();  
 $table->foreignId('product\_color\_size\_id')->constrained();  
 $table->timestamps();  
});

Schema::create('order\_items', function (Blueprint $table) {  
 $table->id();  
 $table->decimal('total', 10, 2);  
 $table->integer('quantity');  
 $table->foreignId('order\_id')->constrained();  
 $table->foreignId('product\_color\_size\_id')->constrained();  
 $table->timestamps();  
});

Выполняем миграцию с помощью команды: php artisan migrate.

## Создание исключений

Во время выбрасывания исключений Laravel проверяет, есть ли в классе исключения метод render(), если да, то он использует метод этого исключения для отображения результата. Если вы не хотите полагаться на глобальную систему отлова исключений Laravel, то можете вернуть ответ в JSON напрямую из контроллера.

Illuminate\Http\Exceptions\HttpResponseException - это особое исключение, встроенное в Laravel. Особенность этого исключения заключается в том, что оно уже содержит шаблон ответа для клиента, поэтому Laravel просто возвращает ответ из этого исключения.

Созданный класс исключения ApiException расширяет базовый класс Exception. В конструкторе он принимает код и сообщение исключения, а также, при необходимости, массив ошибок. При создании объекта исключения он вызывает конструктор родительского класса Exception, передавая ему сообщение и код исключения.

Создадим особое исключение app\Exceptions\Api\ApiException.php, можно увидеть на листинге 3, которое будет определять шаблон выдаваемых пользователю ошибок (будь то ошибка авторизации или ошибка валидации входных данных, либо ошибка доступа к определенному функционалу приложения) со следующим содержимым:

Конструктор принимает три параметра:

$message: Сообщение об ошибке (строка).

$code: HTTP-статус код (по умолчанию 500).

$errors: Дополнительные ошибки (например, ошибки валидации).

Создается массив $response, который содержит:

code: Статус-код.

message: Сообщение об ошибке.

errors: Ошибки, если они переданы.

Затем вызывается конструктор родительского класса (HttpResponseException) с JSON-ответом, установленным на указанный статус-код.

***Листинг 3. Класс исключения ApiException***

class ApiException extends HttpResponseException  
{  
 public function \_\_construct(string $message, int $code = 500, $errors = [])  
 {  
 $response = [  
 'code' => $code,  
 'message' => $message,  
 ];  
 if (!empty($errors)) {  
 $response['errors'] = $errors;  
 }  
 parent::*\_\_construct*( response()->json($response)->setStatusCode($code));  
 }  
}

## Создание моделей

Модели в фреймворке Laravel играют ключевую роль в работе с данными вашего приложения, например:

* Работа с данными базы данных: Модели представляют собой объектно-ориентированный способ взаимодействия с таблицами базы данных. Каждая модель обычно связана с определенной таблицей в базе данных, и она предоставляет удобные методы для выполнения запросов к этой таблице, таких как создание, чтение, обновление и удаление записей.
* Организация бизнес-логики: Модели часто содержат бизнес-логику вашего приложения, связанную с данными. Например, методы модели могут включать валидацию данных, вычисления или другие операции, связанные с обработкой данных.
* Отношения между таблицами: В Laravel отношения между таблицами базы данных могут быть легко определены в моделях. Это позволяет вам работать с данными, которые связаны между собой, например, получать связанные записи или устанавливать связи между ними.
* Использование в контроллерах и представлениях: Модели обычно используются в контроллерах для выполнения операций с данными, а также в представлениях для отображения этих данных. Они обеспечивают удобный способ доступа к данным вашего приложения из любой его части.
* Тестирование: Использование моделей позволяет упростить процесс тестирования вашего приложения. Вы можете создавать экземпляры моделей в тестах и проверять их поведение в различных сценариях.

Чтобы сгенерировать новую модель Eloquent, необходимо использовать команду:

php artisan make:model.

Создадим модели ко всем таблицам нашей базы данных, для этого поочередно выполняем следующие команды:

php artisan make:model Cart

php artisan make:model Category

php artisan make:model Color

php artisan make:model Country

php artisan make:model Order

php artisan make:model OrderItem

php artisan make:model Point

php artisan make:model Product

php artisan make:model ProductColorSize

php artisan make:model Review

php artisan make:model Role

php artisan make:model Size

php artisan make:model Status

php artisan make:model User

Далее отредактируем наши модели, можно увидеть на листинге 4, указав связи:

***Листинг 4. Модели базы данных***

class Cart extends Model  
{  
 protected $fillable = ['total', 'quantity', 'user\_id', 'product\_color\_size\_id'];  
  
 public function order()  
 {  
 return $this->belongsTo(Order::class);  
 }  
 public function productColorSize()  
 {  
 return $this->belongsTo(ProductColorSize::class);  
 }  
}

class Category extends Model  
{  
 protected $fillable = ['name'];  
  
 public function products(){  
 return $this->hasMany(Product::class);  
 }  
}

class Color extends Model  
{  
 protected $fillable = ['name', 'hex'];  
  
 public function productColorSizes(){  
 return $this->hasMany(ProductColorSize::class);  
 }  
}

class Country extends Model  
{  
 protected $fillable = ['name'];  
  
 public function products(){  
 return $this->hasMany(Product::class);  
 }  
}

class Order extends Model  
{  
 protected $fillable = ['order\_date', 'total', 'user\_id', 'status\_id', 'point\_id'];  
  
 public function user()  
 {  
 return $this->belongsTo(User::class);  
 }  
 public function status()  
 {  
 return $this->belongsTo(Status::class);  
 }  
 public function point()  
 {  
 return $this->belongsTo(Point::class);  
 }  
 public function orderItems(){  
 return $this->hasMany(OrderItem::class);  
 }  
}

class OrderItem extends Model  
{  
 protected $fillable = ['total', 'quantity', 'order\_id', 'product\_color\_size\_id'];  
  
 public function order()  
 {  
 return $this->belongsTo(Order::class);  
 }  
 public function productColorSize()  
 {  
 return $this->belongsTo(ProductColorSize::class);  
 }  
}

class Point extends Model  
{  
 protected $fillable = ['city', 'street', 'house'];  
  
 public function orders(){  
 return $this->hasMany(Order::class);  
 }  
}

class Product extends Model  
{  
 protected $fillable = ['photo', 'name', 'description', 'sex', 'quantity', 'price', 'category\_id', 'country\_id'];  
  
 public function category()  
 {  
 return $this->belongsTo(Category::class);  
 }  
 public function country()  
 {  
 return $this->belongsTo(Country::class);  
 }  
  
 public function productColorSizes()  
 {  
 return $this->hasMany(ProductColorSize::class);  
 }  
 public function reviews()  
 {  
 return $this->hasMany(Review::class);  
 }  
}

class ProductColorSize extends Model  
{  
 protected $fillable = ['quantity', 'product\_id', 'color\_id', 'size\_id'];  
  
 public function product()  
 {  
 return $this->belongsTo(Product::class);  
 }  
 public function color()  
 {  
 return $this->belongsTo(Color::class);  
 }  
 public function size()  
 {  
 return $this->belongsTo(Size::class);  
 }  
 public function orderItems()  
 {  
 return $this->hasMany(OrderItem::class);  
 }  
 public function carts()  
 {  
 return $this->hasMany(Cart::class);  
 }  
}

class Review extends Model  
{  
 protected $fillable = ['rating', 'description', 'user\_id', 'product\_id'];  
  
 public function user()  
 {  
 return $this->belongsTo(User::class);  
 }  
 public function product()  
 {  
 return $this->belongsTo(Product::class);  
 }  
}

class Role extends Model  
{  
 protected $fillable = ['name', 'code'];  
  
 public function users(){  
 return $this->hasMany(User::class);  
 }  
}

class Size extends Model  
{  
 protected $fillable = ['name'];  
  
 public function productColorSizes(){  
 return $this->hasMany(ProductColorSize::class);  
 }  
}

class Status extends Model  
{  
 protected $fillable = ['name'];  
  
 public function orders(){  
 return $this->hasMany(Order::class);  
 }  
}

class User extends *Authenticatable*{  
 use HasApiTokens, HasFactory, Notifiable;  
  
 protected $fillable = [  
 'name', 'email', 'telephone', 'sex', 'password', 'api\_token', 'role\_id',  
 ];  
 protected $hidden = [  
 'password',  
 'api\_token',  
 ];  
 protected function casts(): array  
 {  
 return [  
 'password' => 'hashed',  
 ];  
 }  
 public function role()  
 {  
 return $this->belongsTo(Role::class);  
 }  
 public function orders()  
 {  
 return $this->hasMany(Order::class);  
 }  
 public function reviews(){  
 return $this->hasMany(Review::class);  
 }  
}

Класс User содержит поля $fillable и представляет собой модель пользователя в Laravel, которая использует трейты HasApiTokens, HasFactory и Notifiable и связана с моделями Role, Fine, Access, Status.

С помощью $hidden определяются, атрибуты, которые должны быть скрыты при сериализации модели в JSON.

С помощью метода casts определяются, атрибуты модели, которые должны быть приведены к определенным типам. В нашем случае поле password автоматически кэшируется при сохранении в базу данных.

## Создание контроллеров

В фреймворке Laravel контроллеры играют важную роль в организации логики приложения. Они обрабатывают HTTP-запросы и возвращают соответствующие HTTP-ответы.

Выполним генерация контроллеров, необходимых для решения технического задания, для этого выполним следующие команды:

php artisan make:controller AuthController

php artisan make:controller CategoryController

php artisan make:controller ColorController

php artisan make:controller FilterController

php artisan make:controller OrderController

php artisan make:controller PointController

php artisan make:controller ProductController

php artisan make:controller ProfileController

php artisan make:controller ReviewController

php artisan make:controller SizeController

php artisan make:controller StatusController

php artisan make:controller UserController

php artisan make:controller ViewFilterController

php artisan make:controller CartController

Далее отредактируем наши котроллеры, можно увидеть на листингах 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, написав необходимые методы:

***Листинг 5. Контроллер AuthController для авторизации***

class AuthController  
{  
 // Регистрация пользователя  
 public function register(RegisterRequest $request)  
 {  
 // Извлекаем role\_id для роли 'Пользователь'  
 $role\_id = Role::*where*('code', 'user')->first()->id;  
  
 // Извлекаем валидированные данные  
 $validated = $request->validated();  
  
 // Создаем нового пользователя  
 $user = User::*create*([  
 ...$validated,  
 'role\_id' => $role\_id]);  
  
 // Создание токена для пользователя  
 $user->api\_token = Hash::make(Str::*random*(60));  
 $user->save();  
  
 // Возвращаем ответ с токеном  
 return response()->json([  
 'user' => $user,  
 'token' => $user->api\_token  
 ])->setStatusCode(201);  
 }  
 // Авторизация  
 public function login(Request $request) {  
 if (!Auth::attempt($request->only('email', 'password'))) {  
 throw new ApiException("Неверная почта или пароль", 401);  
 }  
 // Получение текущего пользователя  
 $user = Auth::user();  
 // Создание нового токена для пользователя  
 $user->api\_token = Hash::make(Str::*random*(60));  
 $user->save();  
 return response()->json(['token' => $user->api\_token, 'user' => $user])->setStatusCode(200);  
 }  
 // Выход  
 public function logout(Request $request) {  
 // Получение текущего пользователя  
 $user = Auth::user();  
 // Создание нового токена для пользователя  
 $user->api\_token = null;  
 $user->save();  
 return response()->json(['message' => 'Вы вышли из системы'])->setStatusCode(200);  
 }  
}

***Листинг 6. Контроллер CategoryController для категорий***

class CategoryController  
{  
 public function store(CreateCategoryRequest $request) {  
  
 if (Auth::user()->role->code != 'admin') {  
 return response()->json(['message' => 'У вас нет прав на выполнение этого действия'], 403);  
 }  
  
 $validated = $request->validated();  
 // Создание нового адреса  
 $category = Category::*create*([  
 ...$validated,  
 'name' => $request->name,  
 ]);  
  
 // Возвращаем ответ с созданным объектом и статусом 201 (создано)  
 return response()->json($category, 201);  
 }  
}

***Листинг 7. Контроллер ColorController для цветов***

class ColorController  
{  
 public function index()  
 {  
 $colors = Color::*all*();  
 if ($colors->isEmpty()) {  
 throw new ApiException( 'Не найдено', 404);  
 }  
 return response()->json($colors)->setStatusCode(200);  
 }  
}

***Листинг 8. Контроллер FilterController для фильтров по параметрам или по названию, описанию***

class FilterController  
{  
 public function filterProducts(FilterRequest $request)  
 {  
 // Валидация входных параметров  
 $validated = $request->validated();  
  
 // Начинаем построение запроса  
 $query = Product::*query*()->with(['category', 'country']);  
  
 // Применяем фильтры  
 if ($request->has('category\_id')) {  
 $category = Category::*find*($request->category\_id);  
 if (!$category) {  
 throw new ApiException('Категория не найдена', 404);  
 }  
 $query->where('category\_id', $request->category\_id);  
 }  
  
 if ($request->has('country\_id')) {  
 $country = Country::*find*($request->country\_id);  
 if (!$country) {  
 throw new ApiException('Страна не найдена', 404);  
 }  
 $query->where('country\_id', $request->country\_id);  
 }  
  
 if ($request->has('min\_price')) {  
 $query->where('price', '>=', $request->min\_price);  
 }  
  
 if ($request->has('max\_price')) {  
 $query->where('price', '<=', $request->max\_price);  
 }  
  
 if ($request->has('sex')) {  
 $query->where('sex', $request->sex);  
 }  
  
 // Получаем и возвращаем результаты  
 $products = $query->get();  
  
 if ($products->isEmpty()) {  
 throw new ApiException('Товары по заданным фильтрам не найдены', 404);  
 }  
  
 return response()->json([  
 'success' => true,  
 'filters' => $validated,  
 'count' => $products->count(),  
 'data' => $products  
 ]);  
 }  
 // Поиск товаров по названию или описанию  
 public function search(Request $request)  
 {  
 $request->validate([  
 'q' => 'nullable|string|max:255',  
 ]);  
  
 $search = $request->input('q');  
  
 $query = Product::*with*(['category', 'country']);  
  
 if ($search) {  
 $query->where(function ($q) use ($search) {  
 $q->where('name', 'like', "%{$search}%")  
 ->orWhere('description', 'like', "%{$search}%");  
 });  
 }  
  
 $products = $query->get();  
  
 return response()->json([  
 'success' => true,  
 'query' => $search,  
 'count' => $products->count(),  
 'data' => $products,  
 ], 200);  
 }  
}

***Листинг 9. Контроллер OrderController для заказов***

class OrderController  
{  
 // Получение списка всех заказов пользователя  
 public function index()  
 {  
 try {  
 $user = auth()->user();  
 $isAdmin = $user->role->code === 'admin' || $user->role->code === 'manager';  
  
 // Загружаем заказы с нужными связями  
 $orders = $isAdmin  
 ? Order::*with*([  
 'user', 'point', 'status',  
 'orderItems.productColorSize.product',  
 'orderItems.productColorSize.color',  
 'orderItems.productColorSize.size'  
 ])->get()  
 : $user->orders()  
 ->with([  
 'point', 'status',  
 'orderItems.productColorSize.product',  
 'orderItems.productColorSize.color',  
 'orderItems.productColorSize.size'  
 ])->get();  
  
 // Если заказов нет - возвращаем 404  
 if ($orders->isEmpty()) {  
 return response()->json([  
 'status' => 'error',  
 'message' => $isAdmin  
 ? 'В системе пока нет заказов'  
 : 'У вас пока нет заказов'  
 ], 404);  
 }  
  
 // Трансформируем заказы  
 $ordersTransformed = $orders->map(function ($order) use ($isAdmin) {  
 $totalCost = $order->orderItems->sum(function ($item) {  
 return $item->productColorSize->product->price \* $item->quantity;  
 });  
  
 return [  
 'id' => $order->id,  
 'order\_date' => $order->order\_date,  
 'status' => $order->status->name ?? null,  
 'total\_cost' => $totalCost,  
 'user' => $isAdmin ? $order->user->name : null,  
 'address' => [  
 'city' => $order->point->city ?? null,  
 'street' => $order->point->street ?? null,  
 'house' => $order->point->house ?? null,  
 ],  
 'items\_count' => $order->orderItems->count()  
 ];  
 });  
  
 return response()->json([  
 'status' => 'success',  
 'data' => $ordersTransformed  
 ]);  
  
 } catch (\Exception $e) {  
 return response()->json([  
 'status' => 'error',  
 'message' => 'Произошла ошибка при получении списка заказов'  
 ], 500);  
 }  
 }  
  
 // Создание нового заказа с тестовой оплатой  
 public function store(Request $request)  
 {  
 $clientId = $request->user()->id;  
 $addressId = $request->input('point\_id');  
  
 // Проверка наличия адреса  
 if (!$addressId) {  
 return response()->json([  
 'status' => 'error',  
 'message' => 'Не указан адрес доставки.',  
 ], 400);  
 }  
  
 // Загружаем корзину с нужными связями  
 $cartProducts = Cart::*where*('user\_id', $clientId)  
 ->with('productColorSize.product')  
 ->get();  
  
 if ($cartProducts->isEmpty()) {  
 return response()->json([  
 'status' => 'error',  
 'message' => 'Корзина пуста. Добавьте товары для оформления заказа.',  
 ], 400);  
 }  
  
 // Расчёт общей стоимости  
 $totalCost = $cartProducts->sum(function ($item) {  
 return $item->productColorSize->product->price \* $item->quantity;  
 });  
  
 try {  
 Stripe::*setApiKey*(env('STRIPE\_SECRET'));  
  
 // 1. Создаем сессию оплаты Stripe Checkout  
 $session = Session::*create*([  
 'payment\_method\_types' => ['card'],  
 'line\_items' => [[  
 'price\_data' => [  
 'currency' => 'rub',  
 'product\_data' => [  
 'name' => 'Заказ от ' . $request->user()->name,  
 ],  
 'unit\_amount' => $totalCost \* 100,  
 ],  
 'quantity' => 1,  
 ]],  
 'mode' => 'payment',  
 'success\_url' => route('checkout.success') . '?session\_id={CHECKOUT\_SESSION\_ID}',  
 'cancel\_url' => route('checkout.cancel'),  
 'metadata' => [  
 'user\_id' => $clientId,  
 'address\_id' => $addressId,  
 // Передаем только необходимые минимальные данные  
 'cart\_summary' => json\_encode([  
 'items\_count' => $cartProducts->count(),  
 'total' => $totalCost  
 ])  
 ],  
 ]);  
  
 // 2. Возвращаем URL для перенаправления на страницу оплаты Stripe  
 return response()->json([  
 'status' => 'payment\_required',  
 'message' => 'Требуется подтверждение оплаты',  
 'payment\_url' => $session->url,  
 'session\_id' => $session->id  
 ], 200);  
  
 } catch (\Exception $e) {  
 return response()->json([  
 'status' => 'error',  
 'message' => 'Ошибка при создании сессии оплаты: ' . $e->getMessage(),  
 ], 500);  
 }  
 }  
  
 public function handleSuccess(Request $request)  
 {  
 try {  
 Stripe::*setApiKey*(env('STRIPE\_SECRET'));  
 $sessionId = $request->get('session\_id');  
 $session = Session::*retrieve*($sessionId);  
  
 // Получаем данные из metadata  
 $clientId = $session->metadata->user\_id;  
 $addressId = $session->metadata->address\_id;  
  
 // 1. Получаем корзину пользователя из базы данных  
 $cartProducts = Cart::*where*('user\_id', $clientId)  
 ->with('productColorSize.product')  
 ->get();  
  
 if ($cartProducts->isEmpty()) {  
 throw new \Exception('Корзина пуста или уже обработана');  
 }  
  
 // 2. Создаем заказ  
 $order = Order::*create*([  
 'user\_id' => $clientId,  
 'point\_id' => $addressId,  
 'order\_date' => now(),  
 'total' => $session->amount\_total / 100,  
 'status\_id' => 2,  
 'payment\_id' => $session->payment\_intent,  
 'payment\_status' => 'succeeded',  
 ]);  
  
 // 3. Добавляем товары в заказ  
 foreach ($cartProducts as $item) {  
 OrderItem::*create*([  
 'order\_id' => $order->id,  
 'product\_color\_size\_id' => $item->product\_color\_size\_id,  
 'quantity' => $item->quantity,  
 'total' => $item->productColorSize->product->price \* $item->quantity,  
 ]);  
 }  
  
 // 4. Очищаем корзину  
 Cart::*where*('user\_id', $clientId)->delete();  
  
 return response()->json([  
 'status' => 'success',  
 'message' => 'Заказ успешно оформлен и оплачен',  
 'order\_id' => $order->id,  
 ], 201);  
  
 } catch (\Exception $e) {  
 return response()->json([  
 'status' => 'error',  
 'message' => 'Ошибка при обработке платежа: ' . $e->getMessage(),  
 ], 500);  
 }  
 }  
  
 public function handleCancel(Request $request)  
 {  
 return response()->json([  
 'status' => 'canceled',  
 'message' => 'Оплата отменена',  
 ], 200);  
 }  
  
 public function show($id)  
 {  
 $user = auth()->user();  
 $isAdmin = $user->role->code === 'admin' || $user->role->code === 'manager';  
  
 // Загружаем заказ с вложенными связями  
 $order = Order::*with*([  
 'user',  
 'point',  
 'status',  
 'orderItems.productColorSize.product',  
 'orderItems.productColorSize.color',  
 'orderItems.productColorSize.size'  
 ])  
 ->when(!$isAdmin, function ($query) use ($user) {  
 // Ограничиваем по пользователю, если не админ  
 $query->where('user\_id', $user->id);  
 })  
 ->find($id);  
  
 if (!$order) {  
 return response()->json(['error' => 'Заказ не найден'], 404);  
 }  
  
 // Список товаров  
 $items = $order->orderItems->map(function ($item) {  
 $pcs = $item->productColorSize;  
 $product = $pcs->product;  
  
 return [  
 'product\_id' => $product->id,  
 'photo' => $product->photo,  
 'name' => $product->name,  
 'price' => $product->price,  
 'quantity' => $item->quantity,  
 'color' => $pcs->color->name ?? null,  
 'size' => $pcs->size->name ?? null,  
 'total' => round($product->price \* $item->quantity, 2),  
 ];  
 });  
  
 $totalCost = $items->sum('total');  
  
 return response()->json([  
 'id' => $order->id,  
 'order\_date' => $order->order\_date,  
 'status' => $order->status->name ?? null,  
 'total\_cost' => $totalCost,  
 'user' => $isAdmin ? $order->user->name : null,  
 'address' => [  
 'city' => $order->point->city ?? null,  
 'street' => $order->point->street ?? null,  
 'house' => $order->point->house ?? null,  
 ],  
 'items' => $items,  
 ]);  
 }  
  
 // Обновление заказа (статус)  
 public function update(Request $request, $id)  
 {  
 if (Auth::user()->role->code != 'admin' && 'manager') {  
 return response()->json(['message' => 'У вас нет прав на выполнение этого действия'], 403);  
 }  
  
 // Загружаем заказ  
 $order = Order::*find*($id);  
  
 if (!$order) {  
 return response()->json(['error' => 'Заказ не найден'], 404);  
 }  
  
 // Валидируем данные  
 $validated = $request->validate([  
 'status\_id' => 'nullable|integer',  
 ]);  
  
 $order->update($validated);  
  
 return response()->json([  
 'message' => 'Заказ обновлён',  
 'order' => $order,  
 ]);  
 }  
  
 // Отмена заказа  
 public function destroy($id)  
 {  
 $user = auth()->user();  
  
 // Находим заказ текущего пользователя  
 $order = $user->orders()->find($id);  
  
 if (!$order) {  
 return response()->json(['error' => 'Заказ не найден'], 404);  
 }  
  
 $order->status\_id = 4;  
 $order->save();  
  
 return response()->json(['message' => 'Заказ отменён']);  
 }  
}

***Листинг 10. Контроллер PointController для пунктов выдачи***

class PointController  
{  
 // Получение списка всех адресов  
 public function index()  
 {  
 $points = Point::*all*();  
 if ($points->isEmpty()) {  
 throw new ApiException( 'Не найдено', 404);  
 }  
 return response()->json($points)->setStatusCode(200);  
 }  
  
 // Создание нового адреса  
 public function store(CreatePointRequest $request)  
 {  
 if(Auth::user()->role->code != 'admin'){  
 return response()->json(['message' => 'У вас нет прав на выполнение этого действия'], 403);  
 }  
 $validated = $request->validated();  
 // Создание нового адреса  
 $point = Point::*create*([  
 ...$validated,  
 'city' => $request->city,  
 'street' => $request->street,  
 'house' => $request->house,  
 ]);  
  
 // Возвращаем ответ с созданным объектом и статусом 201 (создано)  
 return response()->json($point, 201);  
 }  
  
 // Обновление данных адреса  
 public function update(UpdatePointRequest $request, $id)  
 {  
 if (Auth::user()->role->code != 'admin') {  
 return response()->json(['message' => 'У вас нет прав на выполнение этого действия'], 403);  
 }  
  
 $point = Point::*find*($id);  
 if (!$point) {  
 return response()->json(['error' => 'Пункт выдачи не найден'], 404);  
 }  
 $validated = $request->validated();  
 // Обновляем данные адреса  
 $point->update($validated);  
  
 // Возвращаем обновленный адрес  
 return response()->json($point);  
 }  
   
 // Удаление адреса по id  
 public function destroy($id)  
 {  
 if (Auth::user()->role->code != 'admin') {  
 return response()->json(['message' => 'У вас нет прав на выполнение этого действия'], 403);  
 }  
  
 $point = Point::*find*($id);  
  
 if (!$point) {  
 return response()->json(['message' => 'Пункт выдачи не найден'], 404);  
 }  
  
 // Проверка на наличие связанных записей  
 if (  
 $point->orders()->exists()  
 // добавь сюда другие связи, если есть  
 ) {  
 return response()->json([  
 'message' => 'Нельзя удалить пункт выдачи, так как есть заказы, связанные с ним.'  
 ], 409);  
 }  
  
 $point->delete();  
  
 return response()->json(['message' => 'Пункт выдачи удалён.'], 200);  
 }  
}

***Листинг 11. Контроллер ProductController для продуктов***

class ProductController  
{  
 // Метод для просмотра всех товаров  
 public function index()  
 {  
 $products = Product::*all*();  
 if ($products->isEmpty()) {  
 throw new ApiException( 'Не найдено', 404);  
 }  
 return response()->json($products)->setStatusCode(200);  
 }  
 public function store(CreateProductRequest $request)  
 {  
 if (Auth::user()->role->code != 'admin' && Auth::user()->role->code != 'manager'){  
 return response()->json(['message' => 'У вас нет прав на выполнение этого действия'], 403);  
 }  
  
 // Обработка фото товара  
 $photoUrl = null;  
 if ($request->hasFile('photo') && $request->file('photo')->isValid()) {  
 $photoPath = $request->file('photo')->store('products', 'public');  
 $photoUrl = url('storage/' . $photoPath);  
 }  
  
 // Создаём товар с начальным quantity = 0  
 $product = Product::*create*([  
 'photo' => $photoUrl,  
 'name' => $request->name,  
 'description' => $request->description,  
 'sex' => $request->sex,  
 'quantity' => 0, // Устанавливаем начальное значение  
 'price' => $request->price,  
 'category\_id' => $request->category\_id,  
 'country\_id' => $request->country\_id,  
 ]);  
  
 $totalQuantity = 0;  
  
 // Обработка цветов и размеров  
 foreach ($request->colors as $colorData) {  
 // Обработка цвета (существующий или новый)  
 if (isset($colorData['color\_id'])) {  
 // Используем существующий цвет  
 $colorId = $colorData['color\_id'];  
 } else {  
 // Создаем новый цвет  
 $color = Color::*create*([  
 'name' => $colorData['new\_color\_name'],  
 'hex' => $colorData['new\_color\_hex'],  
 ]);  
 $colorId = $color->id;  
 }  
  
 // Обработка размеров для текущего цвета  
 foreach ($colorData['sizes'] as $sizeData) {  
 // Обработка размера (существующий или новый)  
 if (isset($sizeData['size\_id'])) {  
 // Используем существующий размер  
 $sizeId = $sizeData['size\_id'];  
 } else {  
 // Создаем новый размер  
 $size = Size::*create*([  
 'name' => $sizeData['new\_size\_name'],  
 ]);  
 $sizeId = $size->id;  
 }  
  
 // Создаем связь продукта с цветом и размером  
 ProductColorSize::*create*([  
 'product\_id' => $product->id,  
 'color\_id' => $colorId,  
 'size\_id' => $sizeId,  
 'quantity' => $sizeData['quantity'],  
 ]);  
  
 $totalQuantity += $sizeData['quantity'];  
 }  
 }  
  
 // Обновляем общее количество товара  
 $product->update(['quantity' => $totalQuantity]);  
  
 return response()->json([  
 'product' => $product,  
 'message' => 'Товар успешно создан с цветами и размерами'  
 ], 201);  
 }  
 // Метод для просмотра товара  
 public function show($id)  
 {  
 $product = Product::*with*(['category','country','productColorSizes.color', 'productColorSizes.size'])->find($id);  
  
 if (!$product) {  
 return response()->json(['message' => 'Товар не найден'], 404);  
 }  
  
 return response()->json($product, 200);  
 }  
 public function update(UpdateProductRequest $request, $id)  
 {  
 // Проверка прав доступа  
 if (Auth::user()->role->code != 'admin' && Auth::user()->role->code != 'manager') {  
 return response()->json(['message' => 'У вас нет прав на выполнение этого действия'], 403);  
 }  
  
 // Поиск товара с его вариантами  
 $product = Product::*with*(['productColorSizes.carts', 'productColorSizes.orderItems'])->find($id);  
 if (!$product) {  
 return response()->json(['message' => 'Товар не найден'], 404);  
 }  
  
 // Обработка основного фото  
 if ($request->hasFile('photo') && $request->file('photo')->isValid()) {  
 $photoPath = $request->file('photo')->store('products', 'public');  
 $photoUrl = url('storage/' . $photoPath);  
 $product->photo = $photoUrl;  
 }  
  
 $totalQuantity = 0;  
 $usedPcsIds = [];  
  
 // Обработка цветов и размеров  
 foreach ($request->colors as $colorData) {  
 $colorId = $this->getOrCreateColor($colorData);  
  
 foreach ($colorData['sizes'] as $sizeData) {  
 $sizeId = $this->getOrCreateSize($sizeData);  
  
 // Ищем или создаем связь товара с цветом и размером  
 $pcs = ProductColorSize::*firstOrNew*([  
 'product\_id' => $product->id,  
 'color\_id' => $colorId,  
 'size\_id' => $sizeId,  
 ]);  
  
 $pcs->quantity = $sizeData['quantity'];  
 $pcs->save();  
 $usedPcsIds[] = $pcs->id;  
 $totalQuantity += $sizeData['quantity'];  
 }  
 }  
  
 // Удаляем только те варианты, которых:  
 // 1. Нет в новом запросе  
 // 2. Нет в корзинах  
 // 3. Нет в заказах  
 $toDelete = $product->productColorSizes()  
 ->whereNotIn('id', $usedPcsIds)  
 ->whereDoesntHave('carts')  
 ->whereDoesntHave('orderItems')  
 ->get();  
  
 foreach ($toDelete as $pcs) {  
 $pcs->delete();  
 }  
  
 // Обновляем основные данные товара  
 $product->update([  
 'name' => $request->name,  
 'description' => $request->description,  
 'sex' => $request->sex,  
 'price' => $request->price,  
 'category\_id' => $request->category\_id,  
 'country\_id' => $request->country\_id,  
 'quantity' => $totalQuantity,  
 ]);  
  
 return response()->json([  
 'product' => $product->fresh()->load(['productColorSizes.color', 'productColorSizes.size']),  
 'message' => 'Товар успешно обновлен',  
 'warning' => $toDelete->count() < count($product->productColorSizes) - count($usedPcsIds)  
 ? 'Некоторые варианты не были удалены, так как они используются в заказах или корзинах'  
 : null  
 ], 200);  
 }  
  
 protected function getOrCreateColor($colorData)  
 {  
 if (isset($colorData['color\_id'])) {  
 return $colorData['color\_id'];  
 }  
  
 return Color::*create*([  
 'name' => $colorData['new\_color\_name'],  
 'hex' => $colorData['new\_color\_hex'],  
 ])->id;  
 }  
  
 protected function getOrCreateSize($sizeData)  
 {  
 if (isset($sizeData['size\_id'])) {  
 return $sizeData['size\_id'];  
 }  
  
 return Size::*create*([  
 'name' => $sizeData['new\_size\_name'],  
 ])->id;  
 }  
 public function destroy($id)  
 {  
 if (Auth::user()->role->code != 'admin' && Auth::user()->role->code != 'manager'){  
 return response()->json(['message' => 'У вас нет прав на выполнение этого действия'], 403);  
 }  
  
 $product = Product::*find*($id);  
  
 if (!$product) {  
 return response()->json(['message' => 'Товар не найден'], 404);  
 }  
  
 try {  
 // Поиск товара с учетом связанных данных  
 $product = Product::*with*('productColorSizes')->find($id);  
  
 if (!$product) {  
 return response()->json([  
 'message' => 'Товар не найден',  
 'success' => false  
 ], 404);  
 }  
  
 // Удаление связанных записей (цвета/размеры)  
 $product->productColorSizes()->delete();  
  
 // Удаление самого товара  
 $product->delete();  
  
 return response()->json([  
 'message' => 'Товар успешно удален',  
 'success' => true  
 ], 200);  
  
 } catch (\Illuminate\Database\QueryException $e) {  
 // Код ошибки для нарушения ограничения внешнего ключа  
 if ($e->getCode() == '23000') {  
 return response()->json([  
 'message' => 'Невозможно удалить товар: существуют связанные записи в корзине или других таблицах',  
 'success' => false  
 ], 409); // 409 Conflict - подходящий код статуса  
 }  
  
 // Другие ошибки БД  
 return response()->json([  
 'message' => 'Произошла ошибка при удалении товара',  
 'success' => false,  
 'error' => $e->getMessage()  
 ], 500);  
 }  
 }  
}

***Листинг 12. Контроллер ProfileController для профиля***

class ProfileController  
{  
 // Просмотр профиля  
 public function show()  
 {  
 $user = Auth::user()->load('role');  
  
 return response()->json([  
 'id' => $user->id,  
 'name' => $user->name,  
 'email' => $user->email,  
 'telephone' => $user->telephone,  
 'sex' => $user->sex,  
 'role' => $user->role->name ?? null,  
 'created\_at' => $user->created\_at,  
 'updated\_at' => $user->updated\_at,  
 ]);  
 }  
 // Обновление профиля  
 public function update(UpdateProfileRequest $request)  
 {  
 $user = Auth::user();  
  
 // Извлекаем валидированные данные  
 $validated = $request->validated();  
  
 $user->update($validated);  
  
 return response()->json([  
 'message' => 'Профиль успешно обновлен',  
 'data' => $user->fresh()->load('role')  
 ]);  
 }  
}

***Листинг 13. Контроллер ReviewController для отзывов***

class ReviewController  
{  
 public function index($id) {  
 $product = Product::*find*($id);  
 if (!$product) {  
 throw new ApiException('Товар не найден', 404);  
 }  
  
 $reviews = Review::*with*(['user' => function($query) {  
 $query->select('id', 'name');  
 }])  
 ->where('product\_id', $id)  
 ->orderBy('created\_at', 'desc')  
 ->get();  
  
 if ($reviews->isEmpty()) {  
 throw new ApiException('Отзывы для данного товара не указаны', 404);  
 }  
  
 return response()->json($reviews);  
 }  
 public function store(CreateReviewRequest $request, $id)  
 {  
 $user = Auth::user();  
  
 // Только обычные пользователи могут оставлять отзывы  
 if ($user->role->code != 'user') {  
 return response()->json(['message' => 'Только обычные пользователи могут оставлять отзывы'], 403);  
 }  
  
 // Проверка: уже есть отзыв на этот товар от пользователя  
 $existingReview = Review::*where*('user\_id', $user->id)  
 ->where('product\_id', $id)  
 ->first();  
  
 if ($existingReview) {  
 return response()->json(['message' => 'Вы уже оставили отзыв на этот товар'], 400);  
 }  
  
 // Проверка: есть ли у пользователя заказ с этим товаром и статусом id = 3  
 $hasCompletedOrder = Order::*where*('user\_id', $user->id)  
 ->where('status\_id', 3)  
 ->whereHas('orderItems.productColorSize', function ($query) use ($id) {  
 $query->where('product\_id', $id);  
 })  
 ->exists();  
  
 if (!$hasCompletedOrder) {  
 return response()->json(['message' => 'Вы можете оставить отзыв только после покупки и завершения заказа'], 403);  
 }  
  
 // Создаём отзыв  
 $review = Review::*create*([  
 'rating' => $request->rating,  
 'description' => $request->description,  
 'user\_id' => $user->id,  
 'product\_id' => $id,  
 ]);  
  
 return response()->json($review, 201);  
 }  
  
  
 public function destroy($productId, $reviewId)  
 {  
 $user = Auth::user();  
 $product = Product::*find*($productId);  
 $review = Review::*find*($reviewId);  
  
 // Проверка существования товара и отзыва  
 if (!$product) {  
 throw new ApiException('Товар не найден', 404);  
 }  
 if (!$review) {  
 throw new ApiException('Отзыв не найден', 404);  
 }  
  
 // Проверка соответствия отзыва и товара  
 if ($review->product\_id != $product->id) {  
 throw new ApiException('Этот отзыв не принадлежит указанному товару', 400);  
 }  
  
 // Проверка прав доступа  
 $isAdminOrManager = in\_array($user->role->code, ['admin', 'manager']);  
 $isReviewAuthor = $review->user\_id == $user->id;  
  
 if (!$isAdminOrManager && !$isReviewAuthor) {  
 return response()->json([  
 'message' => 'Недостаточно прав для удаления отзыва'  
 ], 403);  
 }  
  
 // Удаление отзыва  
 $review->delete();  
  
 return response()->json([  
 'message' => 'Отзыв успешно удален'  
 ], 200);  
 }  
}

***Листинг 14. Контроллер SizeController для размеров***

class SizeController  
{  
 public function index()  
 {  
 $sizes = Size::*all*();  
 if ($sizes->isEmpty()) {  
 throw new ApiException( 'Не найдено', 404);  
 }  
 return response()->json($sizes)->setStatusCode(200);  
 }  
}

***Листинг 15. Контроллер StatusController для статусов***

class StatusController  
{  
 public function index()  
 {  
 $statuses = Status::*all*();  
 if ($statuses->isEmpty()) {  
 throw new ApiException( 'Не найдено', 404);  
 }  
 return response()->json($statuses)->setStatusCode(200);  
 }  
}

***Листинг 16. Контроллер UserController для обычных пользователей и менеджеров***

public function createManager(RegisterRequest $request)  
 {  
 // Извлекаем role\_id для роли 'Менеджер' (предполагаем, что у менеджера code = 'manager')  
 $role\_id = Role::*where*('code', 'manager')->first()->id;  
  
 // Если роль не найдена, используем явное значение 2  
 if (!$role\_id) {  
 $role\_id = 2;  
 }  
  
 // Извлекаем валидированные данные  
 $validated = $request->validated();  
  
 // Создаем нового пользователя-менеджера  
 $user = User::*create*([  
 ...$validated,  
 'role\_id' => $role\_id  
 ]);  
  
 // Создание токена для пользователя  
 $user->api\_token = Hash::make(Str::*random*(60));  
 $user->save();  
  
 // Возвращаем ответ с токеном  
 return response()->json([  
 'user' => $user,  
 'token' => $user->api\_token,  
 'message' => 'Менеджер успешно создан'  
 ])->setStatusCode(201);  
 }  
 // Удаление менеджера (для админов и менеджеров)  
 public function deleteManager($id)  
 {  
 $currentUser = Auth::user();  
 $managerToDelete = User::*with*('role')->findOrFail($id);  
  
 // Проверяем, что удаляемый пользователь - менеджер  
 if ($managerToDelete->role->code !== 'manager') {  
 return response()->json(['message' => 'Можно удалять только менеджеров'], 400);  
 }  
  
 // Проверяем права текущего пользователя  
 if (!in\_array($currentUser->role->code, ['admin'])) {  
 return response()->json(['message' => 'Недостаточно прав для удаления менеджера'], 403);  
 }  
  
 $managerToDelete->delete();  
  
 return response()->json([  
 'message' => 'Менеджер успешно удален',  
 'deleted\_manager' => $this->formatManagerData($managerToDelete)  
 ]);  
 }  
  
 // Список всех менеджеров (для админов и менеджеров)  
 public function listManagers()  
 {  
 $currentUser = Auth::user();  
  
 if (!in\_array($currentUser->role->code, ['admin', 'manager'])) {  
 return response()->json(['message' => 'Доступ запрещен'], 403);  
 }  
  
 $managers = User::*whereHas*('role', function($query) {  
 $query->where('code', 'manager');  
 })->get()->map(function($manager) {  
 return $this->formatManagerData($manager);  
 });  
  
 return response()->json($managers);  
 }  
  
 // Форматирование данных менеджера  
 protected function formatManagerData($user)  
 {  
 return [  
 'id' => $user->id,  
 'name' => $user->name,  
 'email' => $user->email,  
 'telephone' => $user->telephone,  
 'role' => $user->role->name,  
 'created\_at' => $user->created\_at->format('Y-m-d H:i:s'),  
 ];  
 }  
 // Просмотр всех пользователей (для админа)  
 public function index()  
 {  
 if (Auth::user()->role->code !== 'admin') {  
 return response()->json(['message' => 'Доступ запрещен'], 403);  
 }  
  
 $users = User::*with*('role')->get()->map(function ($user) {  
 return $this->formatUserData($user);  
 });  
  
 return response()->json($users);  
 }  
  
 // Просмотр конкретного пользователя  
 public function show($id)  
 {  
 $currentUser = Auth::user();  
 $requestedUser = User::*with*('role')->findOrFail($id);  
  
 // Проверка прав: админ или просмотр своего профиля  
 if ($currentUser->role->code !== 'admin' && $currentUser->id != $id) {  
 return response()->json(['message' => 'Доступ запрещен'], 403);  
 }  
  
 return response()->json($this->formatUserData($requestedUser));  
 }  
  
 // Обновление пользователя  
 public function update(Request $request, $id)  
 {  
 $currentUser = Auth::user();  
 $user = User::*findOrFail*($id);  
  
 // Только админ может изменять других пользователей  
 if ($currentUser->role->code !== 'admin' && $currentUser->id != $id) {  
 return response()->json(['message' => 'Доступ запрещен'], 403);  
 }  
  
 // Валидация  
 $validated = $request->validate([  
 'name' => 'sometimes|string|max:255',  
 'email' => 'sometimes|email|unique:users,email,' . $id,  
 'telephone' => 'sometimes|string|max:20',  
 'sex' => 'sometimes|boolean',  
 'role\_id' => 'sometimes|integer|exists:roles,id'  
 ]);  
  
 // Только админ может менять роль  
 if (isset($validated['role\_id'])) {  
 if ($currentUser->role->code !== 'admin') {  
 return response()->json(['message' => 'Недостаточно прав для изменения роли'], 403);  
 }  
  
 // Запрещаем изменять роль последнему админу  
 if ($user->role->code === 'admin' &&  
 User::*where*('role\_id', $user->role\_id)->count() <= 1) {  
 return response()->json(['message' => 'Нельзя изменить роль последнему администратору'], 400);  
 }  
 }  
  
 $user->update($validated);  
  
 return response()->json([  
 'user' => $this->formatUserData($user->fresh('role')),  
 'message' => 'Данные пользователя обновлены'  
 ]);  
 }  
  
 // Форматирование данных пользователя  
 protected function formatUserData($user)  
 {  
 return [  
 'id' => $user->id,  
 'name' => $user->name,  
 'email' => $user->email,  
 'telephone' => $user->telephone,  
 'sex' => $user->sex,  
 'role' => [  
 'id' => $user->role->id,  
 'name' => $user->role->name,  
 'code' => $user->role->code  
 ],  
 'created\_at' => $user->created\_at,  
 'updated\_at' => $user->updated\_at,  
 ];  
 }  
}

***Листинг 17. Контроллер ViewFilterController для вывода категорий и стран***

class ViewFilterController  
{  
 public function allCountryIndex()  
 {  
 $countries = Country::*all*();  
 if ($countries->isEmpty()) {  
 throw new ApiException('Не найдено', 404);  
 }  
 return response()->json($countries)->setStatusCode(200);  
 }  
  
 public function allCategoryIndex()  
 {  
 $categories = Category::*all*();  
 if ($categories->isEmpty()) {  
 throw new ApiException( 'Не найдено', 404);  
 }  
 return response()->json($categories)->setStatusCode(200);  
 }  
}

***Листинг 18. Контроллер CartController для корзины***

class СartController  
{  
 // Получение корзины с товарами для пользователя  
 public function show(Request $request)  
 {  
 $clientId = $request->user()->id;  
  
 $cartProducts = Cart::*with*(['productColorSize.product', 'productColorSize.color', 'productColorSize.size'])  
 ->where('user\_id', $clientId)  
 ->get();  
  
 if ($cartProducts->isEmpty()) {  
 return response()->json(['message' => 'Корзина пуста'], 404);  
 }  
  
 $products = $cartProducts->map(function ($cartProduct) {  
 $pcs = $cartProduct->productColorSize;  
  
 return [  
 'id' => $pcs->id,  
 'name' => $pcs->product->name,  
 'price' => $pcs->product->price,  
 'photo' => $pcs->product->photo,  
 'description' => $pcs->product->description,  
 'color' => $pcs->color->name ?? null,  
 'size' => $pcs->size->name ?? null,  
 'quantity' => $cartProduct->quantity,  
 ];  
 });  
  
 $totalCost = $products->sum(function ($product) {  
 return $product['price'] \* $product['quantity'];  
 });  
  
 return response()->json([  
 'total\_cost' => $totalCost,  
 'created\_at' => $cartProducts->first()->created\_at,  
 'updated\_at' => $cartProducts->first()->updated\_at,  
 'products' => $products,  
 ]);  
 }  
  
 // Добавление товара в корзину  
 public function addProduct(Request $request, $productColorSizeId)  
 {  
 $clientId = $request->user()->id;  
 $quantity = $request->input('quantity', 1);  
  
 $pcs = ProductColorSize::*find*($productColorSizeId);  
  
 if (!$pcs) {  
 return response()->json(['message' => 'Товар не найден'], 404);  
 }  
  
 $existingCartProduct = Cart::*where*('user\_id', $clientId)  
 ->where('product\_color\_size\_id', $productColorSizeId)  
 ->first();  
  
 if ($existingCartProduct) {  
 $existingCartProduct->quantity += $quantity;  
  
 // Пересчёт total  
 $existingCartProduct->total = $pcs->product->price \* $existingCartProduct->quantity;  
  
 $existingCartProduct->save();  
 } else {  
 Cart::*create*([  
 'user\_id' => $clientId,  
 'product\_color\_size\_id' => $productColorSizeId,  
 'quantity' => $quantity,  
 'total' => $pcs->product->price \* $quantity,  
 ]);  
 }  
  
  
 return response()->json([  
 'status' => 'success',  
 'message' => 'Товар добавлен в корзину',  
 ]);  
 }  
  
 // Обновление количества товара в корзине  
 public function updateProduct(Request $request, $productColorSizeId)  
 {  
 $clientId = $request->user()->id;  
 $quantity = $request->input('quantity');  
  
 if ($quantity < 1) {  
 return response()->json([  
 'status' => 'error',  
 'message' => 'Количество товара должно быть больше 0'  
 ], 400);  
 }  
  
 $cartProduct = Cart::*where*('user\_id', $clientId)  
 ->where('product\_color\_size\_id', $productColorSizeId)  
 ->with('productColorSize.product')  
 ->first();  
  
 if (!$cartProduct) {  
 return response()->json([  
 'status' => 'error',  
 'message' => 'Товар не найден в корзине'  
 ], 404);  
 }  
  
 $cartProduct->quantity = $quantity;  
 $cartProduct->total = $cartProduct->productColorSize->product->price \* $quantity;  
 $cartProduct->save();  
  
 // Пересчёт всей корзины  
 $cartProducts = $this->getUserCartProducts($clientId);  
 $totalCost = $cartProducts->sum('total');  
  
 return response()->json([  
 'status' => 'success',  
 'message' => 'Количество товара в корзине успешно обновлено',  
 'data' => [  
 'total\_cost' => $totalCost  
 ]  
 ]);  
 }  
  
 // Удаление товара из корзины  
 public function removeProduct(Request $request, $productColorSizeId)  
 {  
 $clientId = $request->user()->id;  
  
 try {  
 // Проверяем существование товара в корзине пользователя  
 $cartItem = Cart::*where*('user\_id', $clientId)  
 ->where('product\_color\_size\_id', $productColorSizeId)  
 ->firstOrFail();  
  
 // Удаляем товар из корзины  
 $cartItem->delete();  
  
 // Получаем обновленную корзину  
 $cartProducts = $this->getUserCartProducts($clientId);  
 $totalCost = $cartProducts->sum('total');  
  
 return response()->json([  
 'status' => 'success',  
 'message' => 'Товар успешно удалён из корзины',  
 'data' => [  
 'total\_cost' => $totalCost,  
 'remaining\_items' => $cartProducts->count()  
 ]  
 ]);  
  
 } catch (\Illuminate\Database\Eloquent\ModelNotFoundException $e) {  
 return response()->json([  
 'status' => 'error',  
 'message' => 'Указанный товар не найден в вашей корзине'  
 ], 404);  
 } catch (\Exception $e) {  
 return response()->json([  
 'status' => 'error',  
 'message' => 'Произошла ошибка при удалении товара из корзины: ' . $e->getMessage()  
 ], 500);  
 }  
 }  
  
 // Вспомогательный метод для получения корзины пользователя  
 private function getUserCartProducts($clientId)  
 {  
 return Cart::*where*('user\_id', $clientId)->get();  
 }  
}

## Валидация запроса формы

В Laravel валидация запроса формы обычно выполняется в контроллере, используя встроенные инструменты валидации.

Form Request - это класс где мы описываем правила валидации для входящих данных. Обычно класс содержит набор правил под запрос из клиента. Мы можем его декларировать в контроллере, и через контейнер в Laravel он автоматически проверит данные на соответствии нашим правилам и через внутренние механизмы фреймворка выдаст ответ клиенту.

Для более сложных сценариев валидации вы можете создать «запрос формы». Запрос формы – это ваш класс запроса, который инкапсулирует свою собственную логику валидации и авторизации. Сгенерируем родительский запрос формы ApiRequest c методами непрошедшей авторизации и непрошедшей валидации.

Класс ApiRequest, можно увидеть на листинге 19, расширяет FormRequest и переопределяет два метода: failedValidation и failedAuthorization.

***Листинг 19. Класс ApiRequest c методами непрошедшей авторизации и непрошедшей валидации***

class ApiRequest extends FormRequest  
{  
 // Вызов исключения при провале валидации данных  
 public function failedValidation(Validator $validator)  
 {  
 throw new ApiException('Ошибка валидации данных', 422, $validator->errors());  
 }  
 // Вызов исключения при провале авторизации  
 public function failedAuthorization()  
 {  
 throw new ApiException('Ошибка доступа пользователя', 403);  
 }  
}

Класс RegisterRequest, виден на листинге 20, расширяет ApiRequest и определяет правила валидации для регистрации пользователя.

***Листинг 20. Класс RegisterRequest с правилами валидации для регистрации пользователя***

class RegisterRequest extends ApiRequest  
{  
 public function authorize(): bool  
 {  
 return true;  
 }  
 public function rules(): array  
 {  
 return [  
 'name' => 'required|string|max:64|min:3',  
 'email' => 'required|string|max:64|min:3|unique:users,email,',  
 'telephone' => 'nullable|string|max:25|min:10',  
 'sex' => 'required|boolean',  
 'password' => 'required|string|max:64|regex:/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/'  
 ];  
 }  
 public function messages(): array{  
 return [  
 'name.required' => 'Поле "Имя" обязательно для заполнения.',  
 'name.max' => 'Поле "Имя" не должно превышать 64 символа.',  
 'name.min' => 'Поле "Имя" не должно быть меньше 3 символов.',  
  
 'email.required' => 'Поле "Email" обязательно для заполнения.',  
 'email.max' => 'Поле "Email" не должно превышать 64 символа.',  
 'email.min' => 'Поле "Email" не должно быть меньше 3 символов.',  
 'email.unique' => 'Пользователь с вашей почтой уже зарегистрирован',  
  
 'telephone.max' => 'Поле "Телефон" не должно превышать 25 символов.',  
 'telephone.min' => 'Поле "Телефон" не должно быть меньше 3 символов.',  
  
 'sex.required' => 'Поле "Пол" обязательно для заполнения.',  
  
 'password.required' => 'Поле "Пароль" обязательно для заполнения.',  
 'password.max' => 'Поле "Пароль" не должно превышать 64 символа.',  
 'password.regex' => 'Пароль должен содержать как минимум одну заглавную букву, одну строчную букву, одну цифру и один специальный символ (@$!%\*?&).',  
 ];  
 }  
}

Класс FilterRequest, виден на листинге 21, расширяет ApiRequest и определяет правила валидации для фильтра поиска товаров по параметрам.

***Листинг 21. Класс FilterRequest с правилами валидации для фильтра поиска товаров по параметрам***

class FilterRequest extends ApiRequest  
{  
 public function authorize(): bool  
 {  
 return true;  
 }  
  
 public function rules(): array  
 {  
 return [  
 'category\_id' => 'sometimes|integer|exists:categories,id',  
 'country\_id' => 'sometimes|integer|exists:countries,id',  
 'min\_price' => 'sometimes|numeric|min:0',  
 'max\_price' => 'sometimes|numeric|min:1|gt:min\_price',  
 'sex' => 'sometimes|boolean',  
 ];  
 }  
 public function messages(): array  
 {  
 return [  
 'exists' => 'Указанный :attribute не существует',  
 'gt' => 'Максимальная цена должна быть больше минимальной',  
  
 'category\_id.integer' => 'ID категории должно быть целым числом',  
 'country\_id.integer' => 'ID страны должно быть целым числом',  
  
 'min\_price.numeric' => 'Минимальная цена должна быть числом',  
 'max\_price.numeric' => 'Максимальная цена должна быть числом',  
 'min\_price.min' => 'Минимальная цена не может быть меньше 0',  
 'max\_price.min' => 'Максимальная цена не может быть меньше 1',  
  
 'sex.boolean' => 'Пол должен быть true или false, 1 или 0'  
 ];  
 }  
}

Класс CreatePointRequest, виден на листинге 22, расширяет ApiRequest и определяет правила валидации для создания пункта выдачи.

***Листинг 22. Класс CreatePointRequest с правилами валидации для создания пункта выдачи***

class CreatePointRequest extends ApiRequest  
{  
 public function authorize(): bool  
 {  
 return true;  
 }  
  
 public function rules(): array  
 {  
 return [  
 'city' => 'required|string|max:100|min:2',  
 'street' => 'required|string|max:100|min:1',  
 'house' => 'required|string|max:10|min:1',  
 ];  
 }  
 public function messages(): array{  
 return [  
 'city.required' => 'Поле "Город" обязательно для заполнения.',  
 'city.max' => 'Поле "Город" не должно превышать 100 символов.',  
 'city.min' => 'Поле "Город" не должно быть меньше 2 символов.',  
  
 'street.required' => 'Поле "Улица" обязательно для заполнения.',  
 'street.max' => 'Поле "Улица" не должно превышать 100 символов.',  
 'street.min' => 'Поле "Улица" не должно быть меньше 1 символа.',  
  
 'house.required' => 'Поле "Дом" обязательно для заполнения.',  
 'house.max' => 'Поле "Дом" не должно превышать 25 символов.',  
 'house.min' => 'Поле "Дом" не должно быть меньше 1 символа.',  
 ];  
 }  
}

Класс UpdatePointRequest, виден на листинге 23, расширяет ApiRequest и определяет правила валидации для редактирования пункта выдачи.

***Листинг 23. Класс UpdatePointRequest с правилами валидации для редактирования пункта выдачи***

class UpdatePointRequest extends ApiRequest  
{  
 public function authorize(): bool  
 {  
 return true;  
 }  
  
 public function rules(): array  
 {  
 return [  
 'city' => 'string|max:100|min:2',  
 'street' => 'string|max:100|min:1',  
 'house' => 'string|max:10|min:1',  
 ];  
 }  
 public function messages(): array{  
 return [  
 'city.max' => 'Поле "Город" не должно превышать 100 символов.',  
 'city.min' => 'Поле "Город" не должно быть меньше 2 символов.',  
  
 'street.max' => 'Поле "Улица" не должно превышать 100 символов.',  
 'street.min' => 'Поле "Улица" не должно быть меньше 1 символа.',  
  
 'house.max' => 'Поле "Дом" не должно превышать 25 символов.',  
 'house.min' => 'Поле "Дом" не должно быть меньше 1 символа.',  
 ];  
 }  
}

Класс CreateProductRequest, виден на листинге 24, расширяет ApiRequest и определяет правила валидации для создания товара.

***Листинг 24. Класс CreateProductRequest с правилами валидации для создания товара***

class CreateProductRequest extends ApiRequest  
{  
 public function authorize(): bool  
 {  
 return true;  
 }  
  
 public function rules(): array  
 {  
 return [  
 'photo' => 'nullable|file|image|max:2048',  
 'name' => 'required|string|max:255',  
 'description' => 'required|string',  
 'sex' => 'required|boolean',  
 'price' => 'required|numeric|min:0',  
 'category\_id' => 'required|integer|exists:categories,id',  
 'country\_id' => 'required|integer|exists:countries,id',  
 'colors' => 'required|array',  
 'colors.\*.color\_id' => 'nullable|exists:colors,id',  
 'colors.\*.new\_color\_name' => 'nullable|required\_with:colors.\*.new\_color\_hex|string|max:255',  
 'colors.\*.new\_color\_hex' => 'nullable|required\_with:colors.\*.new\_color\_name|string|regex:/^#[a-fA-F0-9]{6}$/',  
 'colors.\*.sizes' => 'required|array',  
 'colors.\*.sizes.\*.size\_id' => 'nullable|exists:sizes,id',  
 'colors.\*.sizes.\*.new\_size\_name' => 'nullable|string|max:255',  
 'colors.\*.sizes.\*.quantity' => 'required|integer|min:0',  
 ];  
 }  
 public function messages(): array{  
 return [  
 'name.required' => 'Поле "Название" обязательно для заполнения.',  
 'name.max' => 'Поле "Название" не должно превышать 255 символов.',  
  
 'description.required' => 'Поле "Описание" обязательно для заполнения.',  
  
 'sex.required' => 'Поле "Пол" обязательно для заполнения.',  
  
 'price.required' => 'Поле "Цена" обязательно для заполнения.',  
 'price.min' => 'Поле "Цена" не может быть ниже 0.',  
  
 'category\_id' => 'Категория должна быть выбрана',  
 'category\_id.exists' => 'Выбранной категории не существует',  
 'country\_id' => 'Страна производителя должна быть выбрана',  
 'country\_id.exists' => 'Выбранной страны не существует',  
  
 'colors.required' => 'Необходимо указать хотя бы один цвет.',  
 'colors.min' => 'Необходимо указать хотя бы один цвет.',  
 'colors.max' => 'Нельзя указать более 20 цветов.',  
  
 'colors.\*.color\_id.exists' => 'Выбранный цвет не существует или был удален.',  
 'colors.\*.color\_id.required\_without\_all' => 'Укажите существующий цвет или создайте новый.',  
  
 'colors.\*.new\_color\_name.string' => 'Название нового цвета должно быть строкой.',  
 'colors.\*.new\_color\_name.max' => 'Название нового цвета не должно превышать 100 символов.',  
 'colors.\*.new\_color\_name.required\_with' => 'Название цвета обязательно при указании HEX-кода.',  
 'colors.\*.new\_color\_name.required\_without' => 'Укажите название нового цвета или выберите существующий.',  
  
 'colors.\*.new\_color\_hex.regex' => 'HEX-код цвета должен быть в формате #FFFFFF.',  
 'colors.\*.new\_color\_hex.required\_with' => 'HEX-код цвета обязателен при указании названия.',  
 'colors.\*.new\_color\_hex.required\_without' => 'Укажите HEX-код нового цвета или выберите существующий.',  
  
 'colors.\*.sizes.required' => 'Для каждого цвета укажите хотя бы один размер.',  
 'colors.\*.sizes.min' => 'Для каждого цвета укажите хотя бы один размер.',  
 'colors.\*.sizes.max' => 'Для каждого цвета нельзя указать более 20 размеров.',  
  
 'colors.\*.sizes.\*.size\_id.exists' => 'Выбранный размер не существует или был удален.',  
 'colors.\*.sizes.\*.size\_id.required\_without' => 'Укажите существующий размер или создайте новый.',  
  
 'colors.\*.sizes.\*.new\_size\_name.string' => 'Название нового размера должно быть строкой.',  
 'colors.\*.sizes.\*.new\_size\_name.max' => 'Название нового размера не должно превышать 50 символов.',  
 'colors.\*.sizes.\*.new\_size\_name.required\_without' => 'Укажите название нового размера или выберите существующий.',  
  
 'colors.\*.sizes.\*.quantity.required' => 'Укажите количество товара для размера.',  
 'colors.\*.sizes.\*.quantity.integer' => 'Количество должно быть целым числом.',  
 'colors.\*.sizes.\*.quantity.min' => 'Количество не может быть отрицательным.',  
 'colors.\*.sizes.\*.quantity.max' => 'Количество не может превышать 10000.',  
 ];  
 }  
}

Класс UpdateProductRequest, виден на листинге 25, расширяет ApiRequest и определяет правила валидации для редактирования товара.

***Листинг 25. Класс UpdateProductRequest с правилами валидации для редактирования товара***

class UpdateProductRequest extends ApiRequest  
{  
 public function authorize(): bool  
 {  
 return true;  
 }  
  
 public function rules(): array  
 {  
 return [  
 'photo' => 'nullable|file|image|max:2048',  
 'name' => 'string|max:255',  
 'description' => 'string',  
 'sex' => 'boolean',  
 'price' => 'numeric|min:0',  
 'category\_id' => 'integer|exists:categories,id',  
 'country\_id' => 'integer|exists:countries,id',  
 'colors' => 'nullable|array',  
 'colors.\*.color\_id' => 'nullable|integer|exists:colors,id',  
 'colors.\*.sizes' => 'nullable|array',  
 'colors.\*.sizes.\*.size\_id' => 'nullable|integer|exists:sizes,id',  
 'colors.\*.sizes.\*.quantity' => 'nullable|integer|min:0',  
 ];  
 }  
 public function messages(): array{  
 return [  
 'name.max' => 'Поле "Название" не должно превышать 255 символов.',  
  
 'price.min' => 'Поле "Цена" не должно быть ниже 0.',  
  
 'category\_id' => 'Категория должна быть выбрана',  
 'category\_id.exists' => 'Выбранной категории не существует',  
 'country\_id' => 'Страна производителя должна быть выбрана',  
 'country\_id.exists' => 'Выбранной страны не существует',  
  
 'colors.required' => 'Необходимо указать хотя бы один цвет.',  
 'colors.min' => 'Необходимо указать хотя бы один цвет.',  
 'colors.max' => 'Нельзя указать более 20 цветов.',  
  
 'colors.\*.color\_id.exists' => 'Выбранный цвет не существует или был удален.',  
 'colors.\*.color\_id.required\_without\_all' => 'Укажите существующий цвет или создайте новый.',  
  
 'colors.\*.new\_color\_name.string' => 'Название нового цвета должно быть строкой.',  
 'colors.\*.new\_color\_name.max' => 'Название нового цвета не должно превышать 100 символов.',  
 'colors.\*.new\_color\_name.required\_with' => 'Название цвета обязательно при указании HEX-кода.',  
 'colors.\*.new\_color\_name.required\_without' => 'Укажите название нового цвета или выберите существующий.',  
  
 'colors.\*.new\_color\_hex.regex' => 'HEX-код цвета должен быть в формате #FFFFFF.',  
 'colors.\*.new\_color\_hex.required\_with' => 'HEX-код цвета обязателен при указании названия.',  
 'colors.\*.new\_color\_hex.required\_without' => 'Укажите HEX-код нового цвета или выберите существующий.',  
  
 'colors.\*.sizes.required' => 'Для каждого цвета укажите хотя бы один размер.',  
 'colors.\*.sizes.min' => 'Для каждого цвета укажите хотя бы один размер.',  
 'colors.\*.sizes.max' => 'Для каждого цвета нельзя указать более 20 размеров.',  
  
 'colors.\*.sizes.\*.size\_id.exists' => 'Выбранный размер не существует или был удален.',  
 'colors.\*.sizes.\*.size\_id.required\_without' => 'Укажите существующий размер или создайте новый.',  
  
 'colors.\*.sizes.\*.new\_size\_name.string' => 'Название нового размера должно быть строкой.',  
 'colors.\*.sizes.\*.new\_size\_name.max' => 'Название нового размера не должно превышать 50 символов.',  
 'colors.\*.sizes.\*.new\_size\_name.required\_without' => 'Укажите название нового размера или выберите существующий.',  
  
 'colors.\*.sizes.\*.quantity.required' => 'Укажите количество товара для размера.',  
 'colors.\*.sizes.\*.quantity.integer' => 'Количество должно быть целым числом.',  
 'colors.\*.sizes.\*.quantity.min' => 'Количество не может быть отрицательным.',  
 'colors.\*.sizes.\*.quantity.max' => 'Количество не может превышать 10000.',  
 ];  
 }  
}

Класс UpdateProfileRequest, виден на листинге 26, расширяет ApiRequest и определяет правила валидации для редактирования профиля.

***Листинг 26. Класс UpdateProfileRequest с правилами валидации для редактирования профиля***

class UpdateProfileRequest extends ApiRequest  
{  
 public function authorize(): bool  
 {  
 return true;  
 }  
  
 public function rules(): array  
 {  
 return [  
 'name' => 'string|max:64|min:3',  
 'email' => 'string|max:64|min:3|unique:users,email,' . $this->user()->id,  
 'telephone' => 'string|max:25|min:10',  
 'sex' => 'boolean',  
 'password' => 'string|max:64|regex:/^(?=.\*[a-z])(?=.\*[A-Z])(?=.\*\d)(?=.\*[@$!%\*?&])[A-Za-z\d@$!%\*?&]{8,}$/',  
 ];  
 }  
 public function messages(): array{  
 return [  
 'name.max' => 'Поле "Имя" не должно превышать 64 символа.',  
 'name.min' => 'Поле "Имя" не должно быть меньше 3 символов.',  
  
 'email.max' => 'Поле "Email" не должно превышать 64 символа.',  
 'email.min' => 'Поле "Email" не должно быть меньше 3 символов.',  
 'email.unique' => 'Пользователь с вашей почтой уже существует',  
  
 'telephone.max' => 'Поле "Телефон" не должно превышать 25 символов.',  
 'telephone.min' => 'Поле "Телефон" не должно быть меньше 3 символов.',  
  
 'sex.required' => 'Поле "Пол" обязательно для заполнения.',  
  
 'password.max' => 'Поле "Пароль" не должно превышать 64 символа.',  
 'password.regex' => 'Пароль должен содержать как минимум одну заглавную букву, одну строчную букву, одну цифру и один специальный символ (@$!%\*?&).',  
 ];  
 }  
}

Класс CreateReviewRequest, виден на листинге 27, расширяет ApiRequest и определяет правила валидации для создания отзыва.

***Листинг 27. Класс CreateReviewRequest с правилами валидации для создания отзыва***

class CreateReviewRequest extends ApiRequest  
{  
 public function authorize(): bool  
 {  
 return true;  
 }  
  
 public function rules(): array  
 {  
 return [  
 'rating' => 'required|integer|max:5|min:1',  
 'description' => 'required|string|max:255',  
 ];  
 }  
 public function messages(): array{  
 return [  
 'rating.required' => 'Поле "Рейтинг" обязательно для заполнения.',  
 'rating.max' => 'Поле "Рейтинг" не должно превышать 5.',  
 'rating.min' => 'Поле "Рейтинг" не должно быть меньше 1.',  
  
 'description.required' => 'Поле "Описание" обязательно для заполнения.',  
 'description.max' => 'Поле "Описание" не должно превышать 255 символов.',  
 ];  
 }  
}

Класс UpdateReviewRequest, виден на листинге 28, расширяет ApiRequest и определяет правила валидации для редактирования отзыва.

***Листинг 28. Класс UpdateReviewRequest с правилами валидации для редактирования отзыва***

class UpdateReviewRequest extends ApiRequest  
{  
 public function authorize(): bool  
 {  
 return true;  
 }  
  
 public function rules(): array  
 {  
 return [  
 'rating' => 'integer|max:5|min:1',  
 'description' => 'string|max:255',  
 ];  
 }  
 public function messages(): array{  
 return [  
 'rating.max' => 'Поле "Рейтинг" не должно превышать 5.',  
 'rating.min' => 'Поле "Рейтинг" не должно быть меньше 1.',  
  
 'description.max' => 'Поле "Описание" не должно превышать 255 символов.',  
 ];  
 }  
}

## Маршрутизация

Маршрутизация в Laravel - это процесс определения того, как приложение должно отвечать на различные HTTP-запросы. В фреймворке Laravel маршруты определяются в файле routes/web.php для веб-маршрутов и routes/api.php для API-маршрутов.

Маршруты, определенные в файле routes/api.php, вложены в группу маршрутов в RouteServiceProvider. Внутри этой группы автоматически применяется префикс URI/api, поэтому не нужно вручную добавлять его к каждому маршруту в файле маршрутов.

Исходя из технического задания были разработаны маршруты, которые записаны в файле routes\api.php и можно увидеть на листинге 29:

***Листинг 29. Маршрутизация***

// Регистрация пользователя  
Route::post('/register', [AuthController::class, 'register']);  
// Авторизация  
Route::post('/login', [AuthController::class, 'login']);  
// Выход  
Route::post('/logout', [AuthController::class, 'logout'])->middleware('auth:api');  
  
// Товары  
Route::get('/product', [ProductController::class, 'index']); // Просмотр всех товаров  
Route::post('/product', [ProductController::class, 'store'])->middleware('auth:api'); // Создание товара  
Route::get('/product/{id}', [ProductController::class, 'show']); // Просмотр конкретного товара  
Route::patch('/product/{id}', [ProductController::class, 'update'])->middleware('auth:api'); // Обновление товара  
Route::delete('/product/{id}', [ProductController::class, 'destroy'])->middleware('auth:api'); // Удаление товара  
  
// Профиль  
Route::get('/profile', [ProfileController::class, 'show'])->middleware('auth:api'); // Просмотр профиля  
Route::patch('/profile', [ProfileController::class, 'update'])->middleware('auth:api'); // Обновление своего профиля  
  
// Товары по фильтрам  
Route::get('/products/filter', [FilterController::class, 'filterProducts']); // Фильтр по категории, стране, цене и полу  
Route::get('/products/search', [FilterController::class, 'search']); // Поиск по тексту  
  
// Вывод категорий и стран  
Route::get('/country', [ViewFilterController::class, 'allCountryIndex']);  
Route::get('/category', [ViewFilterController::class, 'allCategoryIndex']);  
  
// Создание категории  
Route::post('/category', [CategoryController::class, 'store'])->middleware('auth:api');  
  
// Отзывы  
Route::get('/product/{id}/review', [ReviewController::class, 'index']); // Просмотр отзывов по товару  
Route::post('/product/{id}/review', [ReviewController::class, 'store'])->middleware('auth:api'); // Создание отзыва  
Route::delete('/product/{productId}/review/{reviewId}', [ReviewController::class, 'destroy'])->middleware('auth:api'); // Удаление отзыва  
  
// Пункты выдачи  
Route::get('/point', [PointController::class, 'index']); // Просмотр всех пунктов выдачи  
Route::post('/point', [PointController::class, 'store'])->middleware('auth:api'); // Создание пункта выдачи  
Route::patch('/point/{id}', [PointController::class, 'update'])->middleware('auth:api'); // Редактирование пункта выдачи  
Route::delete('/point/{id}', [PointController::class, 'destroy'])->middleware('auth:api'); // Удаление пункта выдачи  
  
// Корзина  
Route::get('/cart', [СartController::class, 'show'])->middleware('auth:api'); // Просмотр товаров всех товаров в своей корзине  
Route::post('/cart/product/{productColorSizeId}', [СartController::class, 'addProduct'])->middleware('auth:api'); // Добаление товара в корзину  
Route::patch('/cart/product/{productColorSizeId}', [СartController::class, 'updateProduct'])->middleware('auth:api'); // Обновление количества товара в корзине  
Route::delete('/cart/product/{productColorSizeId}', [СartController::class, 'removeProduct'])->middleware('auth:api'); // Удаление товара из корзины  
  
// Заказы  
Route::get('/order', [OrderController::class, 'index'])->middleware('auth:api'); // Просмотр всех заказов или только своих (в зависимости от роли)  
Route::post('/order', [OrderController::class, 'store'])->middleware('auth:api'); // Создание заказа  
Route::get('/order/{id}', [OrderController::class, 'show'])->middleware('auth:api'); // Просмотр конкретного заказа  
Route::patch('/order/{id}', [OrderController::class, 'update'])->middleware('auth:api'); // Обновление статуса заказа  
Route::patch('/order/cancelled/{id}', [OrderController::class, 'destroy'])->middleware('auth:api'); // Отмена заказа  
Route::get('/checkout/success', [OrderController::class, 'handleSuccess'])->name('checkout.success'); // Оплачено  
Route::get('/checkout/cancel', [OrderController::class, 'handleCancel'])->name('checkout.cancel'); // Отменён  
  
// Менеджеры  
Route::post('/manager', [UserController::class, 'createManager'])->middleware('auth:api'); // Создание менеджера  
Route::delete('/manager/{id}', [UserController::class, 'deleteManager'])->middleware('auth:api'); // Удаление менеджера  
Route::get('/manager', [UserController::class, 'listManagers'])->middleware('auth:api'); // Список менеджеров  
  
// Пользователи  
Route::get('/user', [UserController::class, 'index'])->middleware('auth:api');  
Route::get('/user/{id}', [UserController::class, 'show'])->middleware('auth:api');  
Route::patch('/user/{id}', [UserController::class, 'update'])->middleware('auth:api');  
  
// Цвета и размеры  
Route::get('/colors', [ColorController::class, 'index']);  
Route::get('/sizes', [SizeController::class, 'index']);  
  
// Статусы  
Route::get('/statuses', [StatusController::class, 'index']);

# ГЛАВА 4. АВТОТЕСТИРОВАНИЕ API

В рамках разработки системы было проведено комплексное автоматизированное тестирование API с применением Postman. Были разработаны тестовые сценарии, охватывающие основные эндпоинты и сценарии взаимодействия с сервером.

## 401 Unauthorized

Код состояния HTTP 401 Unauthorized указывает на то, что запрос к API не может быть выполнен, поскольку отсутствует действующая аутентификация; это означает, что клиент должен предоставить учетные данные для доступа к защищенному ресурсу, например, предоставить верный токен в заголовке Authorization, либо у текущего пользователя недостаточно прав для доступа.

pm.test ("Проверка кода ответа", function () {

pm.response.to.have.status(401);

});

pm.test ("Проверка сообщения", function () {

pm.expect(pm.response.json()).to.have.property("message");

});

|  |  |
| --- | --- |
| **URL** | **МЕТОД** |
| http://{path}/api/login | POST |
| http://{path}/api/logout | POST |
| http://{path}/api/product | POST |
| http://{path}/api/product/{id} | PATCH |
| DELETE |
| http://{path}/api/profile | GET |
| PATCH |
| http://{path}/api/category | POST |
| http://{path}/api/product/{id}/review | POST |
| http://{path}/api/product/{productId}/review/{reviewId} | DELETE |
| http://{path}/api/point | POST |
| http://{path}/api/point/{id} | PATCH |
| DELETE |
| http://{path}/api/cart | GET |
| http://{path}/api/cart/{productColorSizeId} | POST |
| PATCH |
| DELETE |
| http://{path}/api/order | GET |
| POST |
| http://{path}/api/order/{id} | GET |
| PATCH |
| http://{path}/api/order/cancelled/{id} | PATCH |
| http://{path}/api/accesses-partial-cancel/{id} | POST |
| http://{path}/api/manager | POST |
| GET |
| http://{path}/api/manager/{id} | DELETE |
| http://{path}/api/user | GET |
| http://{path}/api/user/{id} | GET |
| PATCH |

## [403 Forbidden](https://ru.wikipedia.org/wiki/%D0%9E%D1%88%D0%B8%D0%B1%D0%BA%D0%B0_403)

Код состояния HTTP 403 Forbidden указывает на то, что сервер понимает запрос, но отказывается его выполнять, даже если клиент аутентифицирован; это означает, что у пользователя, даже после успешной аутентификации, недостаточно прав доступа к запрошенному ресурсу или эндпоинту, либо доступ к этому ресурсу запрещен по другим причинам, не связанным с аутентификацией.

pm.test ("Проверка кода ответа", function () {

pm.response.to.have.status(403);

});

pm.test ("Проверка сообщения", function () {

pm.expect(pm.response.json()).to.have.property("message");

});

|  |  |
| --- | --- |
| **URL** | **МЕТОД** |
| http://{path}/api/product | POST |
| http://{path}/api/product/{id} | PATCH |
| DELETE |
| http://{path}/api/category | POST |
| http://{path}/api/product/{id}/review | POST |
| http://{path}/api/point | POST |
| http://{path}/api/point/{id} | PATCH |
| http://{path}/api/point/{id} | DELETE |
| http://{path}/api/order/{id} | PATCH |
| http://{path}/api/manager | POST |
| GET |
| http://{path}/api/manager/{id} | DELETE |
| http://{path}/api/user | GET |
| http://{path}/api/user/{id} | GET |
| PATCH |

## 404 Not Found

Код состояния HTTP 404 Not Found указывает на то, что сервер не смог найти ресурс, соответствующий запрошенному URL; это означает, что запрошенный эндпоинт или ресурс не существует на сервере, либо был удален, либо URL был введен с ошибкой, и сервер не может обработать запрос, так как целевой ресурс не идентифицирован.

pm.test ("Проверка кода ответа", function () {

pm.response.to.have.status(404);

});

pm.test ("Проверка сообщения", function () {

pm.expect(pm.response.json()).to.have.property("message");

});

|  |  |
| --- | --- |
| **URL** | **МЕТОД** |
| http://{path}/api/product | GET |
| http://{path}/api/product/{id} | GET |
| PATCH |
| DELETE |
| http://{path}/api/products/filter | GET |
| http://{path}/api/products/search | GET |
| http://{path}/api/country | GET |
| http://{path}/api/category | GET |
| http://{path}/api/product/{id}/review | GET |
| http://{path}/api/product/{productId}/review/{reviewId} | DELETE |
| http://{path}/api/point | GET |
| http://{path}/api/point/{id} | PATCH |
| DELETE |
| http://{path}/api/cart | GET |
| http://{path}/api/cart/{productColorSizeId} | POST |
| PATCH |
| DELETE |
| http://{path}/api/order | GET |
| http://{path}/api/order/{id} | GET |
| PATCH |
| http://{path}/api/order/cancelled/{id} | PATCH |
| http://{path}/api/colors | GET |
| http://{path}/api/sizes | GET |
| http://{path}/api/statuses | GET |

## 200 OK

Код состояния HTTP 200 OK указывает на то, что запрос был успешно обработан сервером; это означает, что запрошенное действие было выполнено, и сервер возвращает запрошенные данные или подтверждение успешного выполнения операции в теле ответа.

pm.test ("Проверка кода ответа", function () {

pm.response.to.have.status(200);

});

pm.test ("Проверка сообщения", function () {

pm.expect(pm.response.json()).to.have.property("message");

});

|  |  |
| --- | --- |
| **URL** | **МЕТОД** |
| http://{path}/api/login | POST |
| http://{path}/api/logout | POST |
| http://{path}/api/product | GET |
| http://{path}/api/product/{id} | GET |
| PATCH |
| DELETE |
| http://{path}/api/profile | GET |
| PATCH |
| http://{path}/api/products/filter | GET |
| http://{path}/api/products/search | GET |
| http://{path}/api/country | GET |
| http://{path}/api/category | GET |
| http://{path}/api/product/{id}/review | GET |
| http://{path}/api/product/{productId}/review/{reviewId} | DELETE |
| http://{path}/api/point | GET |
| http://{path}/api/point/{id} | PATCH |
| DELETE |
| http://{path}/api/cart | GET |
| http://{path}/api/cart/{productColorSizeId} | POST |
| PATCH |
| DELETE |
| http://{path}/api/order | GET |
| POST |
| http://{path}/api/order/{id} | GET |
| PATCH |
| http://{path}/api/order/cancelled/{id} | PATCH |
| http://{path}/api/manager/{id} | DELETE |
| http://{path}/api/manager | GET |
| http://{path}/api/user | GET |
| http://{path}/api/user/{id} | GET |
| PATCH |
| http://{path}/api/colors | GET |
| http://{path}/api/sizes | GET |
| http://{path}/api/statuses | GET |

## 201 Created

Код состояния HTTP 201 Created указывает на то, что запрос был успешно обработан и в результате на сервере был создан новый ресурс; это означает, что сервер успешно создал новый элемент (например, запись в базе данных) на основе данных, предоставленных в запросе, и возвращает URI нового ресурса в заголовке Location.

pm.test ("Проверка кода ответа", function () {

pm.response.to.have.status(201);

});

pm.test ("Проверка сообщения", function () {

pm.expect(pm.response.json()).to.have.property("message");

});

|  |  |
| --- | --- |
| **URL** | **МЕТОД** |
| http://{path}/api/register | POST |
| http://{path}/api/product | POST |
| http://{path}/api/category | POST |
| http://{path}/api/product/{id}/review | POST |
| http://{path}/api/point | POST |
| http://{path}/api/manager | POST |

# ГЛАВА 5. РЕАЛИЗАЦИЯ ФУНКЦИОНАЛА ВЕБ-ПРИЛОЖЕНИЯ

Перед стартом разработки веб-приложения интернет-магазина "Тайны Востока" была проведена подготовка программной среды и настроены основные инструменты разработки.

В качестве базовой технологии был выбран JavaScript-фреймворк Vue 3, отличающийся гибкостью, производительностью и удобством для создания современных интерфейсов. Данный инструмент позволяет быстро реализовывать компоненты и эффективно управлять состоянием приложения.

Для установки необходимых библиотек и модулей применялся менеджер пакетов npm, благодаря которому проект получил все зависимости, сгруппированные в папке node\_modules.

В процессе работы использовался инструмент Vite, обеспечивающий мгновенный запуск проекта, поддержку модульной архитектуры и функцию горячей перезагрузки (Hot Module Replacement), что значительно ускоряет цикл разработки и тестирования.

Выбор этих технологий позволил создать стабильную и легко масштабируемую архитектуру клиентской части интернет-магазина.

## Реализация API и вспомогательных модулей

Для обеспечения взаимодействия клиентской части интернет-магазина "Тайны Востока" с серверной логикой на Laravel была реализована система API-запросов с использованием библиотеки Axios, которая широко применяется в разработке Vue-приложений для работы с HTTP-запросами.

Создание отдельного модуля для настройки API позволило централизовать конфигурацию всех запросов, стандартизировать их выполнение, а также упростить масштабирование и сопровождение проекта. Все обращения к серверу проходят через единый экземпляр Axios, что делает код чище и управляемее.

Модуль настройки API-запросов (api.js)

В модуле api.js создаётся настраиваемый экземпляр Axios, в который включены следующие параметры:

* базовый URL, указывающий на backend (http://secrets-of-the-east.ru/api);
* заголовок по умолчанию 'Content-Type': 'application/json' для корректной передачи данных в формате JSON;
* перехватчик запроса (interceptor), автоматически добавляющий токен авторизации Bearer из localStorage ко всем исходящим запросам, если пользователь авторизован.

Такой подход обеспечивает безопасную авторизацию на стороне клиента и упрощает доступ к защищённым маршрутам API Laravel.

Листинг 1 - Модуль api.js

import axios from 'axios';

const api = axios.create({

baseURL: 'http://secrets-of-the-east.ru/api',

headers: {

'Content-Type': 'application/json',

},

});

// Interceptor: добавляет токен к каждому запросу

api.interceptors.request.use(config => {

const token = localStorage.getItem('auth\_token');

if (token) {

config.headers.Authorization = `Bearer ${token}`;

}

return config;

}, error => {

return Promise.reject(error);

});

export default api;

## Реализация управления состоянием (Vuex Store)

Для централизованного управления данными на клиентской части интернет-магазина использовалась библиотека Vuex. Она обеспечивает глобальное хранилище, к которому имеют доступ все компоненты, и реализует реактивный подход к работе с данными. Такой подход упрощает взаимодействие между модулями приложения, особенно при наличии авторизации, фильтрации и отображения пользовательских данных.

Vuex используется для хранения информации о текущем пользователе, состоянии загрузки, а также взаимодействия с локальным хранилищем браузера (localStorage), что необходимо для сохранения авторизации между сессиями.

Хранилище было реализовано в файле store/index.js. Оно включает четыре ключевых раздела: state, mutations, actions и getters.

**State — хранение состояния**

State — это централизованное хранилище всех данных, нужных для работы и отображения интерфейса приложения. В нашем проекте сюда попадают такие данные, как статус загрузки, сведения о текущем пользователе, его имя, email и роль. При старте приложения данные пользователя извлекаются из localStorage, чтобы сохранить сессию между перезагрузками страницы и обеспечить плавный пользовательский опыт.

Такой подход помогает избежать рассогласования данных и дублирования, даёт единый источник правды, что особенно важно для больших приложений.

Листинг 4 – Состояние (state) Vuex-хранилища

state: {

isLoading: false, // Флаг для отображения состояния загрузки

user: {

loggedIn: !!localStorage.getItem('auth\_token'), // Проверка авторизации по наличию токена

name: localStorage.getItem('userName') || '', // Имя пользователя из localStorage

email: localStorage.getItem('userEmail') || '', // Email пользователя из localStorage

role: Number(localStorage.getItem('userRole')) || null, // Роль пользователя (1-админ, 2-менеджер, 3-пользователь)

},

**Mutations — синхронное изменение состояния**

Mutations обеспечивают единственный способ изменить состояние приложения. Они работают синхронно, что упрощает отладку и делает поведение приложения более предсказуемым. Все изменения state проходят через вызов мутаций.

В проекте мутации реализуют установку состояния загрузки, сохранение данных пользователя в state и localStorage, а также очистку данных при выходе из системы. Это обеспечивает согласованность данных и их сохранность.

Листинг 4 – Мутации Vuex-хранилища

mutations: {

setLoading(state, value) {

state.isLoading = value; // Устанавливаем флаг загрузки

},

setUser(state, user) {

// Обновляем состояние пользователя

state.user = {

loggedIn: true,

name: user.name,

email: user.email,

role: Number(user.role),

};

// Сохраняем данные в localStorage

localStorage.setItem('auth\_token', user.token);

localStorage.setItem('userName', user.name);

localStorage.setItem('userEmail', user.email);

localStorage.setItem('userRole', user.role);

},

logout(state) {

// Сбрасываем состояние пользователя

state.user = { loggedIn: false, name: '', email: '', role: null };

// Удаляем данные из localStorage

localStorage.removeItem('auth\_token');

localStorage.removeItem('userName');

localStorage.removeItem('userEmail');

localStorage.removeItem('userRole');

},

**Actions — асинхронная работа с API**

Actions позволяют выполнять асинхронные операции, например, запросы к серверу, и по окончании вызывать мутации для обновления состояния. Это поддерживает однонаправленный поток данных и разделение логики.

В нашем приложении actions реализуют процесс логина (отправку данных на backend, получение токена и данных пользователя), выход из системы и управление индикатором загрузки. Обработка ошибок внутри actions повышает надёжность и информативность.

Листинг 4 – Действия Vuex-хранилища

actions: {

async login({ commit }, userCredentials) {

try {

// Отправляем запрос на авторизацию

const response = await api.post('/login', userCredentials);

// Проверка структуры ответа

if (!response.data || !response.data.token) {

throw new Error('Неверная структура ответа сервера');

}

// Сохраняем данные пользователя

commit('setUser', {

token: response.data.token,

name: response.data.user.name || userCredentials.email,

email: response.data.user.email || userCredentials.email,

role: response.data.user.role\_id,

});

} catch (error) {

console.error('Ошибка авторизации:', error); // Логируем ошибку

throw error; // Прокидываем исключение

}

},

async logout({ commit }) {

try {

// Отправляем запрос на выход

await api.post('/logout');

} catch (error) {

console.warn('Ошибка при выходе с сервера:', error); // Логируем, но не блокируем выход

} finally {

commit('logout'); // Очищаем состояние

}

},

startLoading({ commit }) {

commit('setLoading', true); // Включаем индикатор загрузки

},

stopLoading({ commit }) {

commit('setLoading', false); // Выключаем индикатор загрузки

},

**Getters — получение производных данных**

Getters позволяют получать вычисленные или отфильтрованные данные из state без его изменения. Они выступают прослойкой между состоянием и компонентами, предоставляя удобный доступ к производным значениям.

В нашем случае getters возвращают, например, статус авторизации пользователя, имя, роль, а также логические проверки типа — админ, менеджер или пользователь. Это позволяет компонентам управлять отображением элементов интерфейса на основе роли и статуса пользователя.

Листинг 4 – Геттеры Vuex-хранилища

getters: {

isLoading: (state) => state.isLoading, // Текущий статус загрузки

isLoggedIn: (state) => state.user.loggedIn, // Авторизован ли пользователь

userName: (state) => state.user.name, // Имя пользователя

userRole: (state) => state.user.role, // ID роли пользователя

isAdmin: (state) => state.user.role === 1, // Является ли админом

isManager: (state) => state.user.role === 2, // Является ли менеджером

isUser: (state) => state.user.role === 3, // Является ли обычным пользователем

},

Использование Vuex обеспечило централизованное, структурированное и реактивное управление состоянием. За счёт интеграции с localStorage удалось реализовать сохранение пользовательской сессии между обновлениями страницы. Геттеры позволяют эффективно управлять доступом к функциональности в зависимости от роли пользователя, а мутации и действия обеспечивают надёжную обработку авторизации и взаимодействия с сервером.

## Реализация пользовательского интерфейса

**Компонент App**

Файл App отвечает за общую компоновку пользовательского интерфейса интернет-магазина. В нём задаются постоянные элементы, присутствующие на всех страницах приложения, включая шапку, подвал и область отображения содержимого текущего маршрута.

В верхней части расположен компонент MainHeader, который отображает шапку сайта со ссылками навигации, поиском и доступом к профилю и корзине. Нижняя часть содержит компонент MainFooter с контактной информацией и ссылками на основные разделы. Между этими компонентами располагается элемент <router-view />, в котором динамически отображается содержимое активной страницы (например, главная, корзина, страница товара и т.д.).

Для корректного отображения интерфейса на всех устройствах применяется flex-верстка с вертикальной ориентацией и автоматическим заполнением пространства между шапкой и футером. Общий код компонента приведён в Листинге 1.

Листинг 1 – App.vue

<template>

<div class="layout">

<MainHeader />

<main class="main-content">

<router-view />

</main>

<MainFooter />

</div>

</template>

<script>

import MainHeader from './components/Headers/MainHeader.vue'

import MainFooter from './components/MainFooter.vue'

export default {

name: 'App',

components: {

MainHeader,

MainFooter

}

}

</script>

**Компонент** **MainHeader**

Компонент MainHeader отвечает за верхнюю часть интерфейса и присутствует на всех страницах приложения. В шапке расположены:

* Название магазина с переходом на главную страницу («Тайны Востока»),
* Главное навигационное меню с ссылками на пункты выдачи, корзину и профиль пользователя,
* Поле поиска товаров с поддержкой отправки запроса по нажатию клавиши Enter,
* Секретная кнопка, видимая только для пользователей с ролью администратора или менеджера,
* Кнопка выхода для авторизованных пользователей.

Шапка также управляет открытием модальных окон для регистрации, входа в систему, просмотра профиля, выбора пунктов выдачи и панели инструментов. Это реализовано через локальные флаги видимости и события, что обеспечивает удобство и плавность взаимодействия пользователя с сайтом.

Внутри компонента используется реактивное состояние и вычисляемые свойства для проверки авторизации пользователя и его роли, что позволяет показывать или скрывать определённые элементы интерфейса, например, панель инструментов. Подробный код компонента приведён в Листинге 2. Визуальное представление этого модального окна показано на Рисунке 7.

Листинг 2 – MainHeader.vue

<template>

<header class="header">

<div class="nav-bar">

<div class="nav-left">

<!-- Название магазина, клик переводит на главную -->

<a href="#" class="store-name" @click.prevent="goHome">Тайны Востока</a>

</div>

<div class="nav-center">

<!-- Навигационные ссылки -->

<a href="#" class="nav-link" @click.prevent="showDeliveryPoints">Пункт выдачи</a>

<a href="#" class="nav-link" @click.prevent="goToCart">Корзина</a>

<a href="#" class="nav-link" @click.prevent="handleProfileClick">Профиль</a>

</div>

<div class="nav-right">

<!-- Поле поиска с отправкой запроса по Enter -->

<div class="search-container">

<input

type="text"

class="search-input"

placeholder="Поиск..."

v-model="searchQuery"

@keyup.enter="handleSearch"

/>

</div>

<!-- Секретная кнопка для админов/менеджеров -->

<button

v-if="isAdmin"

class="secret-button"

@click="handleSecretClick"

title="Инструменты"

>

Инструменты

</button>

<!-- Кнопка выхода для авторизованных -->

<button v-if="isAuthenticated" class="logout-button" @click="handleLogout">

Выйти

</button>

</div>

</div>

<!-- Модальные окна -->

<RegisterModal v-if="showRegister" @close="showRegister = false" @switchToLogin="switchToLogin" />

<LoginModal v-if="showLogin" @close="showLogin = false" @switchToRegister="switchToRegister" @loginSuccess="showProfileModal" />

<ProfileModal v-if="showProfile" @close="showProfile = false" @logout="handleLogout" />

<PointsModal v-show="showDeliveryModal" @close="showDeliveryModal = false" />

<AdminPanelModal v-if="showAdminModal" @close="showAdminModal = false" />

</header>

</template>

<script>

import RegisterModal from "../auth/RegisterModal.vue";

import LoginModal from "../auth/LoginModal.vue";

import ProfileModal from "../auth/ProfileModal.vue";

import PointsModal from "../Points/PointsModal.vue";

import AdminPanelModal from "../admin/AdminPanelModal.vue";

import { useRouter } from "vue-router";

import { ref, computed, watch } from "vue";

import { useStore } from "vuex";

export default {

name: "MainHeader",

components: {

RegisterModal,

LoginModal,

ProfileModal,

PointsModal,

AdminPanelModal,

},

setup() {

const router = useRouter();

const store = useStore();

const searchQuery = ref("");

// Обработка поиска: при нажатии Enter отправляем событие для обновления товаров

const handleSearch = () => {

if (router.currentRoute.value.path !== "/") {

router.push("/");

}

window.dispatchEvent(

new CustomEvent("search-request", {

detail: { query: searchQuery.value.trim() },

})

);

};

// Проверка, авторизован ли пользователь

const isAuthenticated = computed(() => store.state.user.loggedIn);

// Получение роли пользователя из Vuex

const userRole = computed(() => store.state.user?.role || null);

// Определяем, является ли пользователь администратором или менеджером (роль 1 или 2)

const isAdmin = computed(() => {

const role\_id = Number(userRole.value);

return role\_id === 1 || role\_id === 2;

});

// Отладка для проверки роли пользователя в консоли

watch(

userRole,

(newRole) => {

console.log("Текущая роль пользователя:", newRole);

},

{ immediate: true }

);

return {

searchQuery,

handleSearch,

router,

isAuthenticated,

store,

userRole,

isAdmin,

};

},

data() {

return {

// Флаги видимости модальных окон

showRegister: false,

showLogin: false,

showProfile: false,

showDeliveryModal: false,

showAdminModal: false,

};

},

methods: {

// Переход на главную страницу

goHome() {

this.$router.push("/");

},

// Открытие панели админа

handleSecretClick() {

this.showAdminModal = true;

},

// Переход в корзину

goToCart() {

this.$router.push("/cart");

},

// Обработка клика по профилю: если авторизован, открыть профиль, иначе окно входа

handleProfileClick() {

if (this.isAuthenticated) {

this.showProfile = true;

} else {

this.showLogin = true;

}

},

// Переключение с регистрации на вход

switchToLogin() {

this.showRegister = false;

this.showLogin = true;

},

// Переключение с входа на регистрацию

switchToRegister() {

this.showLogin = false;

this.showRegister = true;

},

// Показать профиль после успешного входа

showProfileModal() {

this.showLogin = false;

this.showProfile = true;

},

// Выход пользователя из системы

handleLogout() {

this.store.dispatch("logout");

this.showProfile = false;

},

// Открытие модального окна с пунктами выдачи

showDeliveryPoints() {

this.showDeliveryModal = true;

},

},

};

</script>

****

Рисунок 7 – MainHeader

**Компонент модально окно входа (****LoginModal)**

Это окно предназначено для авторизации пользователей. Оно содержит форму с двумя полями: email и пароль. При ошибках авторизации (например, неверный пароль) пользователю выводится сообщение об ошибке. Отправка данных происходит асинхронно через API. Окно закрывается по кнопке "×" или клику на затемнённый фон. Есть ссылка для переключения на регистрацию.

Подробный код компонента приведён в Листинге 3. Визуальное представление этого модального окна показано на Рисунке 8.

Листинг 3 – LoginModal.vue

<template>

<!-- Обертка модального окна -->

<div class="modal-overlay" @click.self="close">

<div class="modal-content">

<button class="close-button" @click="close">×</button>

<h2>Вход</h2>

<form @submit.prevent="submitLogin">

<!-- Поле для ввода email -->

<input

type="email"

v-model="form.email"

placeholder="Email"

required

/>

<!-- Поле для ввода пароля -->

<input

type="password"

v-model="form.password"

placeholder="Пароль"

required

/>

<!-- Отображение ошибки с сервера -->

<p v-if="error" class="error-message">{{ error }}</p>

<button type="submit">Войти</button>

</form>

<!-- Ссылка для переключения на окно регистрации -->

<p>

Нет аккаунта?

<a href="#" @click.prevent="$emit('switchToRegister')">Зарегистрироваться</a>

</p>

</div>

</div>

</template>

<script>

export default {

name: "LoginModal",

data() {

return {

// Объект для хранения введенных данных формы

form: {

email: "",

password: "",

},

// Строка для хранения сообщения об ошибке

error: "",

};

},

methods: {

// Метод для отправки данных на сервер

async submitLogin() {

this.error = "";

try {

// Асинхронный запрос на API для входа пользователя

const response = await fetch("/api/login", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(this.form),

});

if (!response.ok) {

// Если сервер вернул ошибку, получаем текст ошибки

const data = await response.json();

this.error = data.message || "Ошибка авторизации";

return;

}

// При успешном входе эмитируем событие закрытия модального окна

this.$emit("loginSuccess");

this.close();

} catch (e) {

this.error = "Ошибка сети";

}

},

// Метод закрытия модального окна

close() {

this.$emit("close");

},

</script>

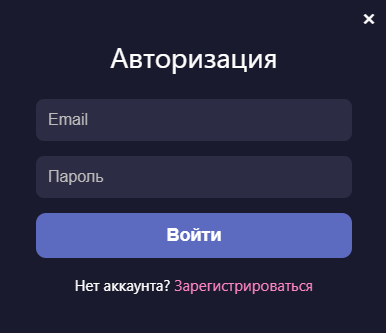
****

Рисунок 8 - LoginModal

**Компонент модального окна регистрации (****RegisterModal)**

Это окно предназначено для регистрации новых пользователей. В нем есть форма с полями для ФИО, email, телефона, пароля и выбора пола. При ошибках валидации или серверных ошибках показываются соответствующие сообщения. Отправка данных происходит асинхронно через API. Окно закрывается по кнопке "×" или клику на затемнённый фон. Есть ссылка для переключения на окно входа.

Визуальное представление этого модального окна показано на Рисунке 9, а полный код компонента — в Листинге 4.

Листинг 4 – RegisterModal.vue

<template>

<!-- Обертка модального окна с затемнением -->

<div class="modal-overlay" @click.self="close">

<div class="modal-content">

<!-- Кнопка закрытия -->

<button class="close-button" @click="close">×</button>

<h2>Регистрация</h2>

<form @submit.prevent="submitRegister">

<!-- Поле для ФИО -->

<input

type="text"

v-model="form.name"

placeholder="ФИО"

required

/>

<p v-if="errors.name" class="error-message">{{ errors.name[0] }}</p>

<!-- Поле для email -->

<input

type="email"

v-model="form.email"

placeholder="Email"

required

/>

<p v-if="errors.email" class="error-message">{{ errors.email[0] }}</p>

<!-- Поле для телефона -->

<input

type="text"

v-model="form.phone"

placeholder="Телефон"

required

/>

<p v-if="errors.phone" class="error-message">{{ errors.phone[0] }}</p>

<!-- Поле для пароля -->

<input

type="password"

v-model="form.password"

placeholder="Пароль"

required

/>

<p v-if="errors.password" class="error-message">{{ errors.password[0] }}</p>

<!-- Выбор пола -->

<select v-model="form.gender" required>

<option value="male">Мужской</option>

<option value="female">Женский</option>

</select>

<p v-if="errors.gender" class="error-message">{{ errors.gender[0] }}</p>

<!-- Кнопка отправки формы -->

<button type="submit">Зарегистрироваться</button>

</form>

<!-- Ссылка для переключения на окно входа -->

<p>

Уже есть аккаунт?

<a href="#" @click.prevent="$emit('switchToLogin')">Войти</a>

</p>

</div>

</div>

</template>

<script>

export default {

name: "RegisterModal",

data() {

return {

// Данные формы регистрации

form: {

name: "",

email: "",

phone: "",

password: "",

gender: "",

},

// Объект для ошибок валидации

errors: {},

};

},

methods: {

// Отправка данных регистрации на сервер

async submitRegister() {

this.errors = {};

try {

const response = await fetch("/api/register", {

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify({

name: this.form.name,

email: this.form.email,

telephone: this.form.phone,

sex: this.form.gender === "male",

password: this.form.password,

}),

});

if (!response.ok) {

// Получаем ошибки с сервера

const data = await response.json();

if (data.errors) {

this.errors = data.errors;

} else {

alert(data.message || "Ошибка регистрации");

}

return;

}

// При успешной регистрации закрываем окно и вызываем событие успеха

this.$emit("registerSuccess");

this.close();

} catch (e) {

alert("Ошибка сети");

}

},

// Закрытие модального окна

close() {

this.$emit("close");

},

},

};

</script>

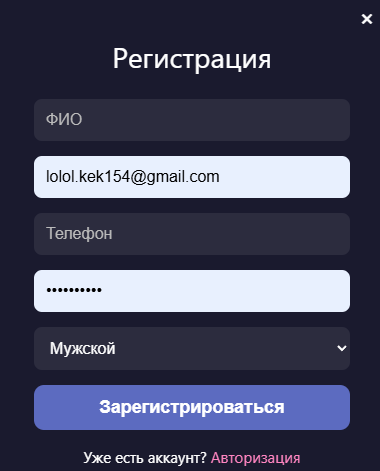
****

Рисунок 9 - RegisterModal

**Компонент модального окна профиля (****ProfileModal)**

Этот компонент отвечает за отображение информации текущего пользователя и позволяет редактировать его данные. В модальном окне показываются поля для ФИО, email и номера телефона. Также доступна кнопка выхода из системы. При нажатии на кнопку «Редактировать» поля становятся доступными для изменения. После внесения изменений пользователь может сохранить их, отправив запрос на сервер через API. При возникновении ошибок отображаются соответствующие сообщения. Окно закрывается по кнопке "×" или клику на затемнённый фон.

Визуальное представление модального окна профиля показано на Рисунке 10, а полный код компонента приведён в Листинге 5.

Листинг 5 – ProfileModal.vue

<template>

<!-- Обертка модального окна с затемнением -->

<div v-if="isVisible" class="modal-overlay" @click.self="handleClose">

<div class="modal-content">

<h2>Профиль</h2>

<!-- Кнопка закрытия -->

<button @click="handleClose" class="close-btn">×</button>

<!-- Состояние загрузки -->

<div v-if="loading" class="loading-state">

<div class="spinner"></div>

<p>Загрузка данных...</p>

</div>

<!-- Просмотр профиля -->

<div v-else-if="user && !isEditing" class="profile-info">

<!-- Заглушка для аватара с инициалами -->

<div class="avatar-placeholder">

{{ getInitials(user.name) }}

</div>

<div class="info-item">

<span class="label">Имя:</span>

<span class="value">{{ user.name }}</span>

</div>

<div class="info-item">

<span class="label">Email:</span>

<span class="value">{{ user.email }}</span>

</div>

<div v-if="user.telephone" class="info-item">

<span class="label">Телефон:</span>

<span class="value">{{ user.telephone }}</span>

</div>

<!-- Кнопки редактирования и выхода -->

<div class="actions">

<button @click="startEditing" class="edit-btn">Редактировать</button>

<button @click="logout" class="logout-btn">Выйти</button>

</div>

</div>

<!-- Форма редактирования профиля -->

<div v-else-if="isEditing" class="edit-form">

<div class="form-group">

<label for="name">Имя:</label>

<input id="name" v-model="editForm.name" type="text" class="form-input" />

</div>

<div class="form-group">

<label for="email">Email:</label>

<input id="email" v-model="editForm.email" type="email" class="form-input" />

</div>

<div class="form-group">

<label for="telephone">Телефон:</label>

<input id="telephone" v-model="editForm.telephone" type="tel" class="form-input" />

</div>

<!-- Кнопки сохранения и отмены -->

<div class="form-actions">

<button @click="saveChanges" class="save-btn" :disabled="isSaving">

{{ isSaving ? 'Сохранение...' : 'Сохранить' }}

</button>

<button @click="cancelEditing" class="cancel-btn">Отмена</button>

</div>

</div>

<!-- Сообщение об ошибке -->

<div v-else class="error-state">

<p class="error-message">⚠️ Не удалось загрузить данные профиля</p>

<button @click="retryLoading" class="retry-btn">Повторить попытку</button>

<button @click="logout" class="logout-btn">Выйти</button>

</div>

</div>

</div>

</template>

<script>

import api from "@/services/api";

export default {

name: "ProfileModal",

data() {

return {

user: null,

isVisible: true,

loading: true,

error: null,

isEditing: false,

isSaving: false,

editForm: {

name: "",

email: "",

telephone: ""

}

};

},

async mounted() {

await this.loadUserData();

},

methods: {

// Загрузка данных пользователя

async loadUserData() {

try {

this.loading = true;

this.error = null;

const token = localStorage.getItem("auth\_token");

if (!token) throw new Error("Токен отсутствует");

const response = await api.get("/profile", {

headers: { Authorization: `Bearer ${token}` }

});

this.user = response.data;

this.editForm = {

name: this.user.name,

email: this.user.email,

telephone: this.user.telephone || ""

};

} catch (error) {

this.error = error.response?.data?.message || error.message;

if (error.response?.status === 401) this.handleForceLogout();

} finally {

this.loading = false;

}

},

// Получение инициалов из имени

getInitials(name) {

return name.split(" ").map(part => part[0]).join("").toUpperCase();

},

// Переход в режим редактирования

startEditing() {

this.isEditing = true;

},

// Отмена редактирования

cancelEditing() {

this.isEditing = false;

this.editForm = {

name: this.user.name,

email: this.user.email,

telephone: this.user.telephone || ""

};

},

// Сохранение изменений

async saveChanges() {

try {

this.isSaving = true;

const token = localStorage.getItem("auth\_token");

const response = await api.patch("/profile", this.editForm, {

headers: { Authorization: `Bearer ${token}` }

});

const updatedUser = response.data.data;

this.user = updatedUser;

this.isEditing = false;

this.$emit("profile-updated", this.user);

} catch (error) {

alert(error.response?.data?.message || "Ошибка при сохранении данных");

} finally {

this.isSaving = false;

}

},

// Выход из аккаунта

logout() {

localStorage.removeItem("auth\_token");

this.$emit("logout");

this.$emit("close");

},

// Принудительный выход при ошибке авторизации

handleForceLogout() {

this.logout();

},

// Повторная попытка загрузки данных

retryLoading() {

this.loadUserData();

},

// Закрытие модального окна

handleClose() {

if (this.isEditing && !confirm("У вас есть несохранённые изменения. Закрыть окно?")) return;

this.$emit("close");

}

}

};

</script>

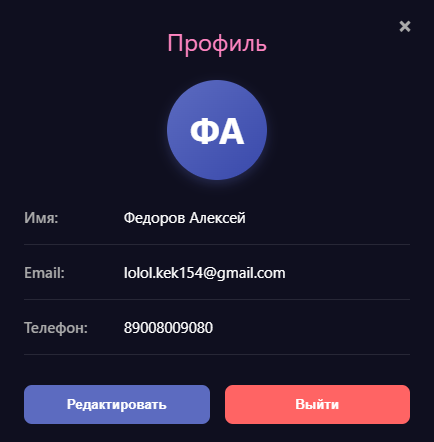
****

Рисунок 10 - ProfileModal

**Компонент подвала сайта (MainFooter)**

Компонент отвечает за отображение подвала сайта с кнопками "О нас" и "Контакты". При клике на кнопку открывается модальное окно с соответствующей информацией.

Модальные окна содержат текст о магазине и контакты сотрудников с телефонами и ссылками на соцсети (ВК, Telegram, GitHub). Окно можно закрыть кликом на фон или кнопкой "×".

Визуальное представление подвала и модальных окон показано на Рисунке 11,12 и 13 а полный код компонента приведён в Листинге 6.

Листинг 6 – MainFooter.vue

<template>

<!-- Подвал сайта с кнопками "О нас" и "Контакты" -->

<div>

<footer class="footer">

<div class="footer-content">

<!-- Кнопки для открытия модальных окон -->

<button @click="openModal('about')" class="footer-link">О нас</button>

<button @click="openModal('contacts')" class="footer-link">Контакты</button>

</div>

<p class="footer-copy">© 2025 Тайны востока. Все права защищены.</p>

</footer>

<!-- Модальное окно, показывается при выборе -->

<div v-if="activeModal" class="modal-overlay" @click.self="closeModal">

<div class="modal-window">

<!-- Кнопка закрытия модального окна -->

<button class="modal-close" @click="closeModal">×</button>

<div class="modal-content">

<!-- Содержимое "О нас" -->

<template v-if="activeModal === 'about'">

<h3>О нас</h3>

<p>

«Тайны Востока» — интернет-магазин стильной и доступной одежды из Китая, Кореи и Японии.

Мы вдохновляемся восточной культурой и привозим лучшие тренды прямо к вам.

</p>

</template>

<!-- Содержимое "Контакты" -->

<template v-else-if="activeModal === 'contacts'">

<h3>Контакты</h3>

<p>Свяжитесь с нами:</p>

<!-- Карточки сотрудников с контактами и ссылками -->

<div class="employee-cards">

<div class="employee-card">

<h4>Федоров Алексей Алексеевич</h4>

<p>Frontend-разработчик</p>

<p>Телефон: <a>+7 996 205 0982</a></p>

<div class="social-links">

<a href="https://vk.com/a\_d\_e\_q\_u\_a\_t\_e\_7" target="\_blank" class="social-link vk">ВКонтакте</a>

<a href="https://t.me/AriStoCraTT7" target="\_blank" class="social-link tg">Telegram</a>

<a href="https://github.com/DeadPage7" target="\_blank" class="social-link gh">GitHub</a>

</div>

</div>

<div class="employee-card">

<h4>Рюгин Алексей Иваночив</h4>

<p>Backend-разработчик</p>

<p>Телефон: <a>+7 952 804 6123</a></p>

<div class="social-links">

<a href="https://vk.com/name\_unknooown" target="\_blank" class="social-link vk">ВКонтакте</a>

<a href="https://t.me/name\_unknooown" target="\_blank" class="social-link tg">Telegram</a>

<a href="https://github.com/Fatalex2005" target="\_blank" class="social-link gh">GitHub</a>

</div>

</div>

</div>

</template>

</div>

</div>

</div>

</div>

</template>

<script setup>

import { ref } from 'vue';

// Состояние активного модального окна (null - нет окна)

const activeModal = ref(null);

// Функция открытия модального окна по имени

const openModal = (modalName) => {

activeModal.value = modalName;

};

// Функция закрытия модального окна

const closeModal = () => {

activeModal.value = null;

};

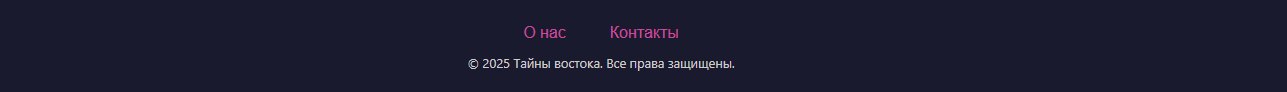
****

Рисунок 11 - MainFooter

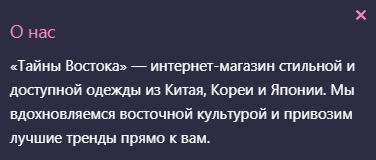
****

Рисунок 1 - MainFooter

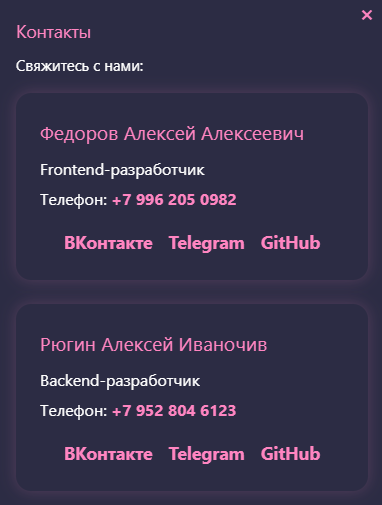
****

Рисунок 1 - MainFooter

**Страница** **HomeView**

На странице HomeView реализован базовый макет интернет-магазина, состоящий из трёх ключевых частей.

Сверху размещён компонент HeaderInfo, который служит информационным блоком с основными данными или акцентами сайта.

Основной контент разделён на две части по горизонтали:

* Левая часть — это компонент ProductFilter, отвечающий за фильтрацию товаров по различным параметрам (категории, цена, цвет и т.п.). Пользователь может выбрать нужные фильтры, и они передаются в компонент отображения товаров.
* Правая часть — блок ProductCards, в котором отображаются карточки товаров с учётом текущих фильтров и поискового запроса. Компонент получает объект фильтров и строку поиска и обновляет список товаров динамически.

Связь между фильтрами и карточками товаров реализована через передачу данных с помощью реактивных переменных. Обновление поискового запроса происходит по событию из шапки сайта, что обеспечивает единое управление отображаемыми товарами.

Такой подход позволяет удобно разделить логику фильтрации и отображения, а также облегчает масштабирование и поддержку кода. Полный код компонента приведён в Листинге 7.

Листинг 7 – HomeView.vue

<template>

<!-- Информационный блок в шапке -->

<HeaderInfo></HeaderInfo>

<div class="home-container">

<!-- Левая часть — компонент фильтрации товаров -->

<ProductFilter @filter-changed="handleFilterChange" />

<!-- Правая часть — отображение карточек товаров с применением фильтров и поиска -->

<div class="products-section">

<ProductCards

:filters="activeFilters"

:search-query="searchQuery"

/>

</div>

</div>

</template>

<script setup>

// Импорт Vue API и компонентов

import { onMounted, ref } from 'vue';

import ProductFilter from '@/components/Product/ProductFilter.vue';

import ProductCards from '@/components/Product/ProductCards.vue';

import HeaderInfo from "@/components/Headers/HeaderInfo.vue";

// Хранение активных фильтров (реактивный объект)

const activeFilters = ref({});

// Хранение поискового запроса (реактивная строка)

const searchQuery = ref('');

// Обработчик события изменения фильтра, обновляет активные фильтры

const handleFilterChange = (filters) => {

activeFilters.value = filters;

};

// Добавляем слушатель пользовательского события search-request для обновления поискового запроса

onMounted(() => {

window.addEventListener('search-request', (event) => {

searchQuery.value = event.detail.query;

});

});

</script>

**Компонент** **ProductCards**

Компонент ProductCards отвечает за отображение списка товаров на странице. Он показывает индикатор загрузки во время получения данных, выводит сообщение, если товары не найдены, и отображает сетку карточек товаров с пагинацией.

Каждая карточка содержит изображение товара, название, страну производителя, категорию и цену. По клику на карточку происходит переход на страницу с детальной информацией о товаре.

Данные загружаются с сервера с учётом активных фильтров и поискового запроса. При изменении фильтров или поискового запроса происходит автоматический повторный запрос товаров.

Пагинация реализована с кнопками для перехода по страницам, отображается максимум 12 товаров на странице. Это обеспечивает удобный и понятный интерфейс для пользователя при просмотре каталога.

Подробно визуальное представление компонента показано на Рисунке 14, а полный код компонента приведён в Листинге 8.

Листинг 8 – ProductCards.vue

<template>

<div class="products-container">

<!-- Индикатор загрузки, показывается во время запроса данных -->

<div v-if="loading" class="loading-overlay">

<div class="loader"></div>

</div>

<!-- Сообщение, если товары не найдены -->

<div v-if="!loading && products.length === 0" class="no-products">

<h3>Товары не найдены</h3>

<p>Попробуйте изменить параметры фильтрации</p>

</div>

<!-- Сетка карточек товаров -->

<div v-else class="products-grid">

<div

v-for="product in paginatedProducts"

:key="product.id"

class="product-card"

@click="goToProduct(product.id)"

>

<!-- Отображение скидки, если она есть -->

<div class="product-badge" v-if="product.discount">

-{{ product.discount }}%

</div>

<!-- Изображение товара -->

<div class="product-image-container">

<img

:src="getImageUrl(product.photo)"

:alt="product.name"

class="product-image"

@error="handleImageError" <!-- Подмена изображения при ошибке загрузки -->

/>

</div>

<!-- Основная информация о товаре -->

<div class="product-info">

<!-- Страна и категория товара -->

<div class="product-meta">

<span class="product-country">{{ product.country?.name }}</span>

<span class="product-category">{{ product.category?.name }}</span>

</div>

<!-- Название товара -->

<h3 class="product-name">{{ product.name }}</h3>

<!-- Цены: текущая и старая (если есть скидка) -->

<div class="product-price">

<span class="current-price">{{ formatPrice(calculatePrice(product)) }}</span>

<span v-if="product.discount" class="old-price">{{ formatPrice(product.price) }}</span>

</div>

</div>

</div>

</div>

<!-- Пагинация -->

<div v-if="totalPages > 1 && !loading" class="pagination">

<!-- Кнопка "назад" -->

<button

@click="prevPage"

:disabled="currentPage === 1"

class="pagination-btn"

>

&lt;

</button>

<!-- Кнопки страниц -->

<button

v-for="page in visiblePages"

:key="page"

@click="goToPage(page)"

:class="{ active: currentPage === page }"

class="pagination-btn"

>

{{ page }}

</button>

<!-- Кнопка "вперёд" -->

<button

@click="nextPage"

:disabled="currentPage === totalPages"

class="pagination-btn"

>

&gt;

</button>

</div>

</div>

</template>

<script setup>

import { ref, computed, onMounted, watch } from 'vue';

import { useRouter } from 'vue-router';

import api from "@/services/api";

const router = useRouter();

// Получаем фильтры и поисковый запрос из пропсов

const props = defineProps({

filters: {

type: Object,

default: () => ({})

},

searchQuery: {

type: String,

default: ''

}

});

const products = ref([]); // Список товаров

const loading = ref(false); // Флаг загрузки

const currentPage = ref(1); // Текущая страница пагинации

const totalProducts = ref(0); // Общее количество товаров

const itemsPerPage = 12; // Количество товаров на странице

// Функция загрузки товаров с сервера с учётом фильтров и поиска

const fetchProducts = async () => {

loading.value = true; // Включаем индикатор загрузки

try {

let response;

if (props.searchQuery) {

// Если есть поисковый запрос — отправляем запрос на поиск

response = await api.get('/products/search', {

params: { q: props.searchQuery }

});

products.value = response.data.data || response.data;

totalProducts.value = response.data.count || products.value.length;

} else {

// Иначе загружаем товары по фильтрам с пагинацией

const params = {

...props.filters,

page: currentPage.value,

per\_page: itemsPerPage

};

// Убираем пустые параметры из запроса

Object.keys(params).forEach(key => {

if (params[key] === null || params[key] === undefined || params[key] === '') {

delete params[key];

}

});

response = await api.get('/products/filter', { params });

products.value = response.data.data || response.data;

totalProducts.value = response.data.count || products.value.length;

}

} catch (error) {

console.error('Ошибка при загрузке товаров:', error);

products.value = [];

totalProducts.value = 0;

} finally {

loading.value = false; // Выключаем индикатор загрузки

}

};

// Вычисляем общее количество страниц

const totalPages = computed(() => Math.ceil(products.value.length / itemsPerPage));

// Вычисляем товары, которые нужно показать на текущей странице

const paginatedProducts = computed(() =>

products.value.slice((currentPage.value - 1) \* itemsPerPage, currentPage.value \* itemsPerPage)

);

// Вычисляем видимые кнопки пагинации (максимум 5)

const visiblePages = computed(() => {

const pages = [];

const maxVisible = 5;

let start = Math.max(1, currentPage.value - Math.floor(maxVisible / 2));

let end = Math.min(totalPages.value, start + maxVisible - 1);

if (end - start + 1 < maxVisible) {

start = Math.max(1, end - maxVisible + 1);

}

for (let i = start; i <= end; i++) {

pages.push(i);

}

return pages;

});

// Переход на выбранную страницу пагинации

const goToPage = (page) => {

currentPage.value = page;

window.scrollTo({ top: 0, behavior: 'smooth' }); // Плавный скролл вверх

};

// Переход на предыдущую страницу

const prevPage = () => {

if (currentPage.value > 1) {

currentPage.value--;

window.scrollTo({ top: 0, behavior: 'smooth' });

}

};

// Переход на следующую страницу

const nextPage = () => {

if (currentPage.value < totalPages.value) {

currentPage.value++;

window.scrollTo({ top: 0, behavior: 'smooth' });

}

};

// Формируем корректный URL для изображения товара

const getImageUrl = (path) => {

if (!path) return '/placeholder-product.jpg'; // Запасное изображение

if (path.startsWith('http')) return path; // Если ссылка внешняя

return `http://secrets-of-the-east.ru/storage/${path}`; // Путь на сервере

};

// Обработчик ошибки загрузки изображения — подменяем на запасное

const handleImageError = (e) => {

e.target.src = '/placeholder-product.jpg';

};

// Форматируем цену в рублях

const formatPrice = (price) => {

return new Intl.NumberFormat('ru-RU', {

style: 'currency',

currency: 'RUB',

maximumFractionDigits: 0

}).format(price);

};

// Вычисляем цену с учётом скидки

const calculatePrice = (product) => {

return product.discount ? product.price \* (1 - product.discount / 100) : product.price;

};

// Переход на страницу с подробной информацией о товаре

const goToProduct = (id) => {

router.push(`/product/${id}`);

};

// При изменении фильтров сбрасываем страницу на первую и обновляем список товаров

watch(() => props.filters, () => {

currentPage.value = 1;

fetchProducts();

}, { deep: true });

// При изменении поискового запроса сбрасываем страницу на первую и обновляем список товаров

watch(() => props.searchQuery, () => {

currentPage.value = 1;

fetchProducts();

});

// При монтировании компонента загружаем товары

onMounted(() => {

fetchProducts();

});

</script>

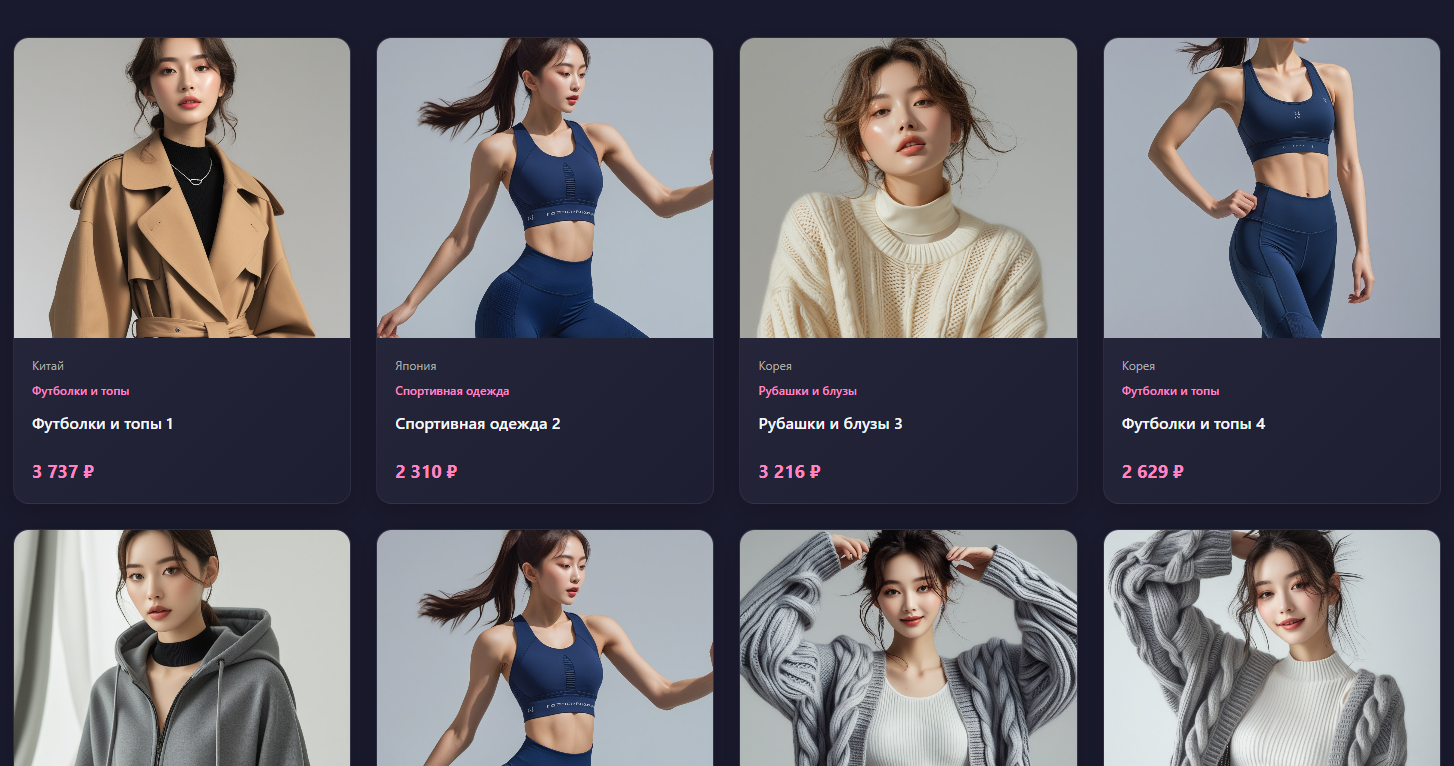
****

Рисунок 1 - ProductCards

**Компонент ProductFilter**

Компонент ProductFilter предоставляет пользователю интерфейс для фильтрации товаров по различным критериям: категории, стране, полу и ценовому диапазону. Данные для фильтров загружаются с сервера при инициализации компонента. Пользователь может применить или сбросить фильтры, после чего генерируется событие filter-changed с актуальными параметрами, которое можно обработать в родительском компоненте для запроса данных.

Компонент включает анимацию секций, контроль ввода диапазона цен, а также очистку пустых значений перед отправкой фильтров. Это делает фильтрацию более гибкой и устойчивой к ошибкам.

Подробное визуальное представление компонента показано на Рисунке 15, а полный код приведён в Листинге 9.

Листинг 9 – ProductFilter.vue

<template>

<div class="filter-container">

<h2>Фильтры</h2>

<!-- Фильтр по категориям -->

<div class="filter-section" :class="sectionClass">

<h3>Категории</h3>

<select v-model="selectedCategory" class="filter-select">

<option value="">Все категории</option>

<!-- Категории подгружаются из API -->

<option v-for="category in categories" :key="category.id" :value="category.id">

{{ category.name }}

</option>

</select>

</div>

<!-- Фильтр по странам -->

<div class="filter-section" :class="sectionClass">

<h3>Страны</h3>

<select v-model="selectedCountry" class="filter-select">

<option value="">Все страны</option>

<!-- Страны подгружаются из API -->

<option v-for="country in countries" :key="country.id" :value="country.id">

{{ country.name }}

</option>

</select>

</div>

<!-- Фильтр по полу -->

<div class="filter-section" :class="sectionClass">

<h3>Пол</h3>

<select v-model="selectedGender" class="filter-select">

<option value="">Не важно</option>

<option value="0">Мужской</option>

<option value="1">Женский</option>

</select>

</div>

<!-- Фильтр по цене -->

<div class="filter-section" :class="sectionClass">

<h3>Цена</h3>

<div class="price-range">

<div class="price-inputs">

<input

type="number"

v-model="priceRange[0]"

:min="minPrice"

:max="maxPrice"

@input="updatePriceText"

class="price-input"

placeholder="От"

/>

<span>—</span>

<input

type="number"

v-model="priceRange[1]"

:min="minPrice"

:max="maxPrice"

@input="updatePriceText"

class="price-input"

placeholder="До"

/>

</div>

<!-- Отображение выбранного диапазона -->

<div class="price-values">

<span>{{ priceRange[0] }} ₽</span>

<span>{{ priceRange[1] }} ₽</span>

</div>

</div>

</div>

<!-- Кнопки управления -->

<button @click="applyFilters" class="apply-btn">Применить фильтры</button>

<button @click="resetFilters" class="reset-btn">Сбросить фильтры</button>

</div>

</template>

<script setup>

import { ref, onMounted } from 'vue'

import api from "@/services/api"

const selectedCategory = ref('') // Выбранная категория

const selectedCountry = ref('') // Выбранная страна

const selectedGender = ref('') // Выбранный пол

const priceRange = ref([0, 100000]) // Диапазон цены

const minPrice = 0 // Минимально возможная цена

const maxPrice = 100000 // Максимально возможная цена

const categories = ref([]) // Список категорий

const countries = ref([]) // Список стран

const emit = defineEmits(['filter-changed']) // Объявление события

// Загрузка данных фильтров с сервера

const loadFilterData = async () => {

try {

const [cats, cntrs] = await Promise.all([

api.get('/category'),

api.get('/country')

])

categories.value = cats.data

countries.value = cntrs.data

} catch (error) {

console.error('Ошибка загрузки фильтров:', error)

}

}

// Применение фильтров и генерация события

const applyFilters = () => {

const filters = {

category\_id: selectedCategory.value || null,

country\_id: selectedCountry.value || null,

min\_price: priceRange.value[0] !== minPrice ? priceRange.value[0] : null,

max\_price: priceRange.value[1] !== maxPrice ? priceRange.value[1] : null,

sex: selectedGender.value || null

}

// Удаляем пустые значения из объекта

const cleanFilters = Object.fromEntries(

Object.entries(filters).filter(([\_, value]) => value !== null && value !== undefined)

)

emit('filter-changed', cleanFilters)

}

// Сброс всех фильтров до значений по умолчанию

const resetFilters = () => {

selectedCategory.value = ''

selectedCountry.value = ''

selectedGender.value = ''

priceRange.value = [minPrice, maxPrice]

emit('filter-changed', {

category\_id: '',

country\_id: '',

price\_min: minPrice,

price\_max: maxPrice,

sex: ''

})

}

// Проверка и коррекция диапазона цен

const updatePriceText = () => {

if (priceRange.value[0] > priceRange.value[1]) {

priceRange.value[0] = priceRange.value[1]

}

}

// Загружаем данные при монтировании компонента

onMounted(() => {

loadFilterData()

})

// Класс для анимации секций фильтра

const sectionClass = 'filter-section-anim'

</script>

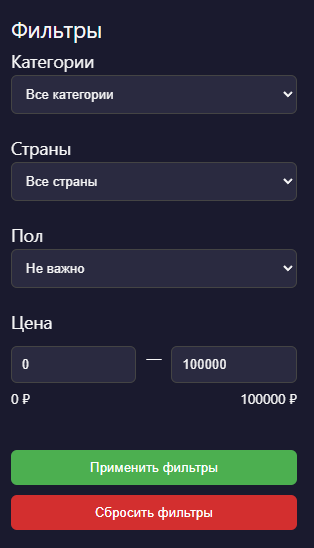
****

Рисунок 1 - ProductFilter

**Страница ProductDetailView**

Страница ProductDetailView отвечает за отображение подробной информации о выбранном товаре. Она автоматически открывается при нажатии на карточку товара в компоненте ProductCards. Основная задача — предоставить пользователю расширенную информацию о товаре и отзывы других покупателей.

Страница состоит из двух основных компонентов:

* ProductDetailCard — отображает подробные сведения о товаре: изображение, описание, цену, скидки, характеристики и т.п.
* ProductReviews — показывает отзывы пользователей, а также форму для добавления нового отзыва.

Переход на эту страницу осуществляется программно через router.push() в компоненте ProductCards, используя идентификатор выбранного товара. Полный код страницы приведён в Листинге 10.

Листинг 10 – ProductDetailView.vue

<template>

<div class="product-detail-view">

<!-- Компонент карточки с подробной информацией о товаре -->

<ProductDetailCard />

<!-- Компонент отзывов к товару -->

<ProductReviews />

</div>

</template>

<script setup>

// Импорт компонента карточки товара

import ProductDetailCard from '@/components/Product/ProductDetailCard.vue'

// Импорт компонента отзывов

import ProductReviews from '@/components/Product/ProductReviews.vue'

</script>

**Компонент ProductDetailCard**

Компонент ProductDetailCard отображает детальную информацию о товаре, включая его изображение, название, описание, категорию, страну производства, пол и цену. Пользователь может выбрать цвет и размер из доступных вариантов, а также указать количество для добавления в корзину. Компонент обеспечивает проверку корректности выбора и информирует пользователя об ошибках или успешном добавлении товара.

Для удобства пользователя реализованы визуальные эффекты при наведении на цветовые и размерные опции, а также динамическое обновление доступных размеров в зависимости от выбранного цвета. При загрузке данных отображается индикатор загрузки, а в случае ошибки — соответствующее сообщение.

Подробное визуальное представление компонента показано на Рисунке 16, а полный код приведён в Листинге 11.

Листинг 11 – ProductDetailCard.vue

<template>

<div>

<!-- Индикатор загрузки -->

<div v-if="loading" class="loading-wrapper">

<div class="spinner"></div>

<div class="loading-text">Загрузка товара...</div>

</div>

<transition name="fade">

<template v-if="!loading">

<div v-if="product" class="product-detail">

<!-- Основное изображение товара -->

<div class="image-block">

<img :src="getImageUrl(product.photo)" :alt="product.name" class="main-image" />

</div>

<div class="info-block">

<h1>{{ product.name }}</h1>

<p class="description">{{ product.description }}</p>

<!-- Информация о категории, стране и поле -->

<p><strong>Категория:</strong> {{ product.category?.name || 'Неизвестная категория' }}</p>

<p><strong>Страна:</strong> {{ product.country?.name || 'Неизвестная страна' }}</p>

<p><strong>Пол:</strong> {{ formatSex(product.sex) }}</p>

<p><strong>Цена:</strong> {{ formatPrice(product.price) }}</p>

<!-- Выбор цвета -->

<div v-if="uniqueColors.length">

<h3>Доступные цвета:</h3>

<div class="color-options">

<div

v-for="color in uniqueColors"

:key="color.id"

class="color-box"

:style="{ backgroundColor: color.hex, border: selectedColor?.id === color.id ? '3px solid #ff46a0' : '2px solid #2e2e3f' }"

:title="color.name"

@click="selectColor(color)"

></div>

</div>

</div>

<!-- Выбор размера -->

<div v-if="availableSizes.length">

<h3 style="margin-top: 20px;">Доступные размеры:</h3>

<div class="size-options">

<div

v-for="size in availableSizes"

:key="size.id"

class="size-box"

:class="{ selected: selectedSize?.id === size.id }"

:title="'Размер: ' + size.name"

@click="selectSize(size)"

>

{{ size.name }}

</div>

</div>

</div>

<!-- Количество и кнопка добавления в корзину -->

<div class="add-to-cart-block">

<div class="quantity-block">

<label for="quantity">Количество:</label>

<div class="quantity-controls">

<button class="quantity-btn" @click="decreaseQuantity">-</button>

<input

type="number"

id="quantity"

v-model="quantity"

min="1"

class="quantity-input"

readonly

/>

<button class="quantity-btn" @click="increaseQuantity">+</button>

</div>

</div>

<button

class="btn-add-to-cart"

:disabled="!canAddToCart"

@click="addToCart"

title="Выберите цвет и размер"

>

Добавить в корзину

</button>

</div>

<!-- Сообщения об ошибках и успехе -->

<p v-if="error" class="error-message">{{ error }}</p>

<p v-if="successMessage" class="success-message">{{ successMessage }}</p>

</div>

</div>

<!-- Сообщение при ошибке загрузки товара -->

<div v-else class="error">Ошибка загрузки товара</div>

</template>

</transition>

</div>

</template>

<script setup>

import { ref, onMounted, computed } from 'vue';

import { useRoute } from 'vue-router';

import api from '@/services/api';

// Получаем ID товара из маршрута

const route = useRoute();

const productId = route.params.id;

// Состояния компонента

const product = ref(null);

const loading = ref(true);

const quantity = ref(1);

const error = ref('');

const successMessage = ref('');

// Выбранные пользователем цвет и размер

const selectedColor = ref(null);

const selectedSize = ref(null);

// Загрузка данных о товаре с сервера

const fetchProduct = async () => {

try {

const response = await api.get(`/product/${productId}`);

product.value = response.data;

} catch (e) {

console.error('Ошибка при загрузке данных о товаре:', e);

} finally {

loading.value = false;

}

};

// Уникальные цвета товара для отображения

const uniqueColors = computed(() => {

const seen = new Set();

return (product.value?.product\_color\_sizes || [])

.map(pcs => pcs.color)

.filter(color => color && !seen.has(color.id) && seen.add(color.id));

});

// Размеры, доступные для выбранного цвета

const availableSizes = computed(() => {

if (!selectedColor.value) return [];

const sizes = product.value.product\_color\_sizes

.filter(pcs => pcs.color.id === selectedColor.value.id)

.map(pcs => pcs.size);

const seen = new Set();

return sizes.filter(size => size && !seen.has(size.id) && seen.add(size.id));

});

// Проверка возможности добавления товара в корзину

const canAddToCart = computed(() => {

return selectedColor.value && selectedSize.value && quantity.value > 0;

});

// Форматирование цены в рублях

const formatPrice = (price) => {

return new Intl.NumberFormat('ru-RU', {

style: 'currency',

currency: 'RUB',

maximumFractionDigits: 0

}).format(price);

};

// Форматирование пола товара для вывода

const formatSex = (sex) => {

if (sex === 0) return 'Женский';

if (sex === 1) return 'Мужской';

return 'Не указан';

};

// Формируем полный URL изображения

const getImageUrl = (path) => {

if (!path) return '/placeholder-product.jpg';

return path.startsWith('http') ? path : `http://secrets-of-the-east.ru/storage/${path}`;

};

// Уменьшение количества

const decreaseQuantity = () => {

if (quantity.value > 1) quantity.value--;

};

// Увеличение количества

const increaseQuantity = () => {

quantity.value++;

};

// Обработка выбора цвета

const selectColor = (color) => {

selectedColor.value = color;

selectedSize.value = null; // сброс выбранного размера при смене цвета

error.value = '';

successMessage.value = '';

};

// Обработка выбора размера

const selectSize = (size) => {

selectedSize.value = size;

error.value = '';

successMessage.value = '';

};

// Добавление товара в корзину

const addToCart = async () => {

error.value = '';

successMessage.value = '';

if (!selectedColor.value || !selectedSize.value) {

error.value = 'Пожалуйста, выберите цвет и размер';

return;

}

// Находим связку color+size для передачи серверу

const pcs = product.value.product\_color\_sizes.find(pcs =>

pcs.color.id === selectedColor.value.id && pcs.size.id === selectedSize.value.id

);

if (!pcs) {

error.value = 'Выбранный цвет и размер недоступны';

return;

}

try {

await api.post(`/cart/product/${pcs.id}`, {

quantity: quantity.value

});

successMessage.value = 'Товар успешно добавлен в корзину';

} catch (e) {

error.value = 'Ошибка при добавлении товара в корзину';

console.error(e);

}

};

// Загружаем данные при монтировании компонента

onMounted(fetchProduct);

</script>

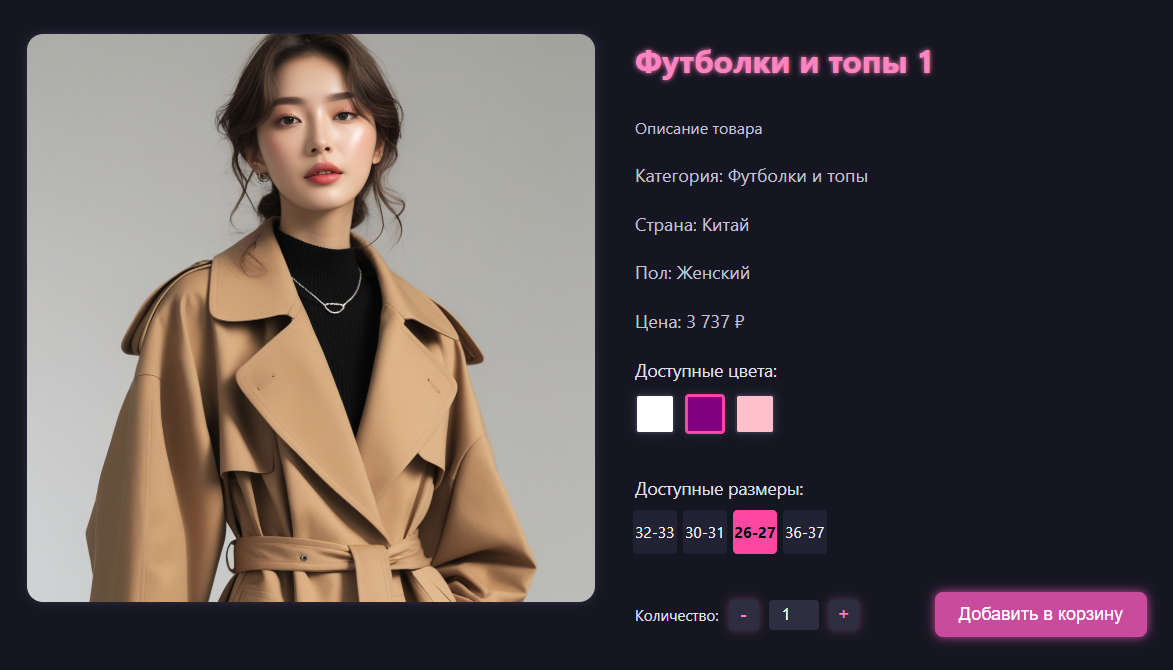
****

Рисунок 1 - ProductDetailCard

**Компонент ProductReviews**

Данный компонент отображает отзывы пользователей о конкретном товаре и позволяет авторизованным пользователям добавить свой отзыв с рейтингом и комментарием.

В верхней части выводится список уже оставленных отзывов с именем пользователя, рейтингом в виде звёзд и текстом комментария. Если отзывов нет, показывается сообщение об их отсутствии.

Ниже расположена форма для добавления нового отзыва, доступная только авторизованным пользователям. В форме можно выбрать рейтинг кликом по звёздам и ввести текст комментария. При попытке отправки происходит проверка заполнения полей, а при ошибках запросов или валидации пользователю выводятся соответствующие сообщения.

Для получения и отправки данных используются асинхронные запросы к API, при монтировании компонента происходит загрузка текущих отзывов с сервера.

В случае если пользователь не авторизован, вместо формы отображается сообщение с приглашением авторизоваться через профиль.

Подробно визуальное представление страницы показано на Рисунке 17, а полный код страницы приведён в Листинге 12.

Листинг 12 – ProductReviews.vue

<template>

<div class="product-reviews">

<!-- Секция с отзывами, если они есть -->

<div v-if="reviews.length">

<h3 class="reviews-title">Отзывы</h3>

<div class="reviews-container">

<div v-for="review in reviews" :key="review.id" class="review">

<div class="review-header">

<!-- Имя пользователя -->

<div class="review-user">{{ review.user.name }}</div>

<!-- Отображение рейтинга звёзд -->

<div class="review-stars">

<span

v-for="star in 5"

:key="star"

class="star"

:class="{ filled: star <= review.rating }"

>★</span>

</div>

</div>

<!-- Текст отзыва -->

<p class="review-text">{{ review.description }}</p>

</div>

</div>

</div>

<!-- Сообщение при отсутствии отзывов -->

<div v-else>

<p>Нет отзывов для этого товара.</p>

</div>

<!-- Форма добавления отзыва для авторизованных -->

<div v-if="isLoggedIn" class="review-form">

<h3>Оставьте свой отзыв</h3>

<form @submit.prevent="submitReview">

<div class="form-group rating-group">

<label>Оценка:</label>

<div class="rating-stars">

<!-- Выбор рейтинга кликом по звёздам -->

<span

v-for="star in 5"

:key="star"

class="star"

:class="{ selected: star <= newReview.rating }"

@click="newReview.rating = star"

role="button"

tabindex="0"

@keydown.enter.prevent="newReview.rating = star"

aria-label="Оценить {{ star }} звёзд"

>★</span>

</div>

</div>

<div class="form-group">

<label for="description">Комментарий:</label>

<textarea

v-model="newReview.description"

id="description"

rows="4"

placeholder="Напишите ваш отзыв здесь..."

required

></textarea>

</div>

<!-- Отображение ошибки валидации -->

<p v-if="errorMessage" class="error-message">{{ errorMessage }}</p>

<button type="submit">Отправить отзыв</button>

</form>

</div>

<!-- Сообщение для неавторизованных пользователей -->

<div v-else>

<p>Чтобы оставить отзыв, необходимо нажать на профиль и авторизоваться.</p>

</div>

</div>

</template>

<script setup>

import { ref, onMounted, computed } from 'vue';

import { useStore } from 'vuex';

import { useRoute } from 'vue-router';

import api from '@/services/api';

// Получение store и текущего ID товара из маршрута

const store = useStore();

const route = useRoute();

const productId = route.params.id;

// Массив отзывов и данные новой формы

const reviews = ref([]);

const newReview = ref({ rating: 0, description: '' });

const errorMessage = ref('');

// Вычисляемое значение для проверки авторизации

const isLoggedIn = computed(() => store.state.user.loggedIn);

// Функция загрузки отзывов с сервера

const fetchReviews = async () => {

try {

const response = await api.get(`/product/${productId}/review`);

reviews.value = response.data;

} catch (error) {

console.error('Ошибка при загрузке отзывов:', error);

}

};

// Функция отправки нового отзыва

const submitReview = async () => {

errorMessage.value = '';

// Проверка заполненности полей

if (!newReview.value.rating || !newReview.value.description.trim()) {

errorMessage.value = 'Пожалуйста, заполните оценку и комментарий.';

return;

}

try {

const response = await api.post(`/product/${productId}/review`, newReview.value);

reviews.value.push(response.data); // Добавление нового отзыва

newReview.value = { rating: 0, description: '' }; // Очистка формы

} catch (error) {

if (error.response && error.response.data && error.response.data.message) {

errorMessage.value = error.response.data.message;

} else if (error.response && error.response.data && typeof error.response.data === 'object') {

// Обработка ошибок валидации Laravel

const messages = [];

for (const key in error.response.data.errors) {

messages.push(...error.response.data.errors[key]);

}

errorMessage.value = messages.join(' ');

} else {

errorMessage.value = 'Произошла ошибка при отправке отзыва.';

}

}

};

// Загрузка отзывов при монтировании компонента

onMounted(fetchReviews);

</script>

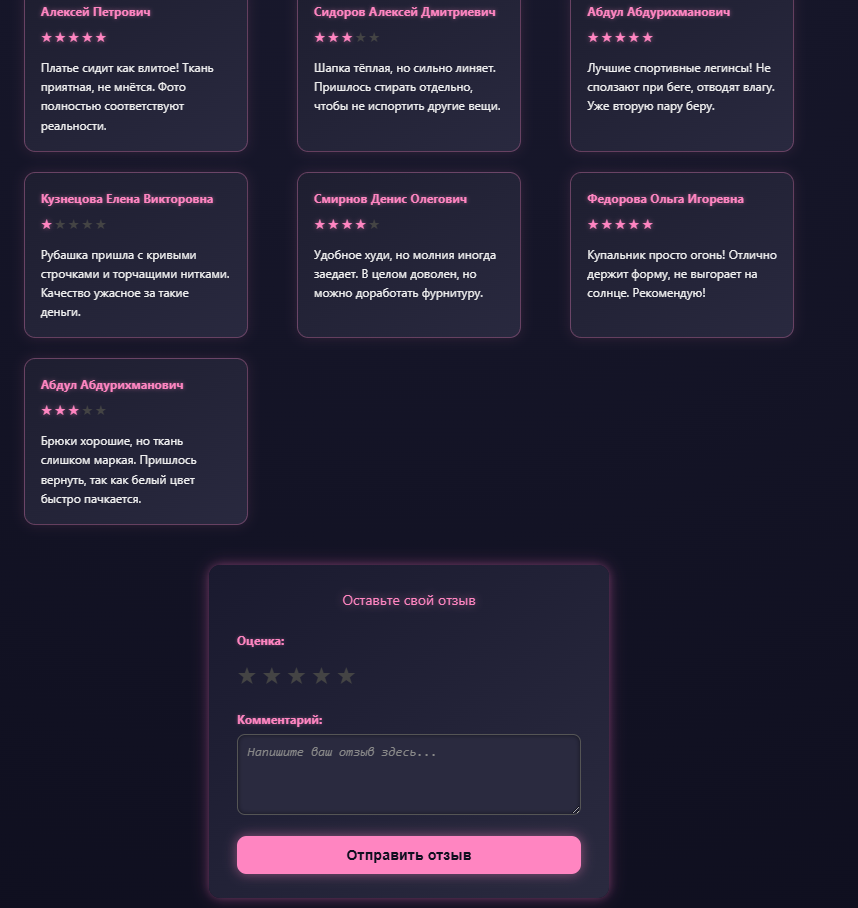
****

Рисунок 1 - ProductReviews

**Страница CartView**

Компонент страницы корзины разделён на две части: слева отображается список товаров в корзине, справа — блок оформления заказа с итоговой суммой.

В левой части используется дочерний компонент CartItems, которому передаётся массив продуктов из корзины. Компонент CartItems отвечает за визуализацию каждого товара, а также уведомляет родителя о необходимости обновить данные корзины через событие update.

Справа размещён компонент CartCheckout, который получает общую стоимость корзины и отображает блок с кнопками для оформления заказа.

Основные данные корзины (products и total\_cost) загружаются с сервера при монтировании компонента функцией fetchCart. Она делает запрос к API и обновляет состояние cart. При ошибках загрузки в консоль выводятся сообщения. Полный код страницы приведён в Листинге 13.

Листинг 13 – CartView.vue

<template>

<div class="cart-container">

<!-- Левая часть: список товаров -->

<div class="left-side">

<!-- Компонент, отображающий товары корзины -->

<CartItems :products="cart.products" @update="fetchCart" />

</div>

<!-- Правая часть: оформление заказа -->

<div class="right-side">

<!-- Компонент с итоговой суммой и кнопками -->

<CartCheckout :totalCost="cart.total\_cost" />

</div>

</div>

</template>

<script setup>

import { ref, onMounted } from "vue";

import api from "@/services/api";

import CartItems from '@/components/cart/CartItems.vue'

import CartCheckout from '@/components/cart/CartCheckout.vue'

// Состояние корзины: товары и итоговая стоимость

const cart = ref({

products: [],

total\_cost: 0,

});

// Функция загрузки корзины с сервера

async function fetchCart() {

try {

const response = await api.get("/cart");

if (response.data.products && Array.isArray(response.data.products)) {

cart.value = response.data; // Обновляем состояние корзины

} else {

console.error("В ответе API нет массива products");

}

} catch (error) {

console.error("Ошибка загрузки корзины:", error);

}

}

// Загружаем корзину при монтировании компонента

onMounted(() => {

fetchCart();

});

</script>

**Компонент** **CartItems**

Компонент CartItems отвечает за отображение списка товаров в корзине и управление их количеством. При загрузке данных отображается индикатор загрузки. Если корзина пуста, выводится соответствующее сообщение.

Для каждого товара выводится карточка с фотографией, названием, характеристиками (цвет, размер), ценой за единицу и общей стоимостью с учётом количества.

Пользователь может изменить количество товара с помощью кнопок «+» и «−». Кнопка «−» блокируется, если количество равно 1, чтобы нельзя было уменьшить ниже минимального значения. Также есть кнопка «Удалить» для удаления товара из корзины.

Изменение количества и удаление товара отправляют запросы на сервер через API, а при успешном ответе обновляют локальное состояние products для мгновенного отображения изменений.

Загрузка корзины происходит при монтировании компонента.

Подробно визуальное представление компонента показано на Рисунке 18, а полный код компонента — в Листинге 14.

Листинг 14 – CartItems.vue

<template>

<div class="cart-items-container">

<!-- Индикатор загрузки -->

<div v-if="loading" class="loading">Загрузка...</div>

<div v-else>

<!-- Сообщение, если корзина пуста -->

<div v-if="products.length === 0" class="empty">Корзина пуста</div>

<!-- Список товаров -->

<div class="products-list">

<div

v-for="product in products"

:key="product.productColorSizeId"

class="product-card"

tabindex="0"

aria-label="Товар в корзине"

>

<div class="product-image-container">

<!-- Фото товара -->

<img :src="product.photo" alt="Фото товара" class="product-photo" />

</div>

<div class="product-info">

<h3 class="product-name">{{ product.name }}</h3>

<p>

Цвет: <span class="highlight">{{ product.color || "—" }}</span>,

Размер: <span class="highlight">{{ product.size || "—" }}</span>

</p>

<p>Цена за шт: <span class="price">{{ product.price }} ₽</span></p>

<p>Итого: <span class="price-total">{{ product.price \* product.quantity }} ₽</span></p>

</div>

<div class="product-controls">

<!-- Управление количеством товара -->

<div class="quantity-controls" role="group" aria-label="Управление количеством">

<button

@click="updateQuantity(product, product.quantity - 1)"

:disabled="product.quantity <= 1"

aria-disabled="product.quantity <= 1"

aria-label="Уменьшить количество"

>

-

</button>

<span class="quantity" aria-live="polite">{{ product.quantity }}</span>

<button @click="updateQuantity(product, product.quantity + 1)" aria-label="Увеличить количество">+</button>

</div>

<!-- Кнопка удаления товара -->

<button class="remove-btn" @click="removeProduct(product)" aria-label="Удалить товар">

Удалить

</button>

</div>

</div>

</div>

</div>

</div>

</template>

<script setup>

import { ref, onMounted } from "vue";

import api from "@/services/api";

// Состояние списка товаров и индикатора загрузки

const products = ref([]);

const loading = ref(false);

// Загрузка корзины с сервера

async function loadCart() {

loading.value = true;

try {

const response = await api.get("/cart");

products.value = response.data.products.map((p) => ({

...p,

productColorSizeId: p.productColorSizeId || p.id, // Обеспечиваем уникальный ключ

}));

} catch (error) {

console.error("Ошибка загрузки корзины:", error);

products.value = [];

} finally {

loading.value = false;

}

}

// Обновление количества товара в корзине

async function updateQuantity(product, newQuantity) {

if (newQuantity < 1) return; // Минимум 1

try {

await api.patch(`/cart/product/${product.productColorSizeId}`, {

quantity: newQuantity,

});

product.quantity = newQuantity; // Обновляем локально

} catch (error) {

console.error("Ошибка обновления количества:", error);

}

}

// Удаление товара из корзины

async function removeProduct(product) {

try {

await api.delete(`/cart/product/${product.productColorSizeId}`);

products.value = products.value.filter(

(p) => p.productColorSizeId !== product.productColorSizeId

);

} catch (error) {

console.error("Ошибка удаления товара:", error);

}

}

// Загрузка корзины при монтировании компонента

onMounted(() => {

loadCart();

});

</script>



Рисунок 1 - CartItems

# ГЛАВА 6. ТЕСТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ

# ГЛАВА 7. РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

?????

1. Вход и регистрация

1.1 Вход на сайт и регистрация

1. Для работы с корзиной и оформлением заказа необходимо быть авторизованным пользователем.
2. Авторизация и регистрация выполняются через кнопку «Профиль» в верхней части сайта.
3. Если у вас ещё нет аккаунта — зарегистрируйтесь, заполнив форму. После регистрации вы автоматически войдёте в систему.
4. Без авторизации добавление товара в корзину и оформление заказа невозможно.

2. Каталог товаров и фильтрация

2.1 Фильтры и товары

1. Страница каталога разделена на две части:

* Слева — блок фильтрации (категории, цвет, цена и др.).
* Справа — список товаров, соответствующих фильтрам.

1. Выберите нужные параметры фильтра для удобного поиска товаров.
2. Выберите интересующий товар для просмотра подробной информации.

2.2 Выбор товара

1. На странице товара сначала выберите цвет.
2. Затем выберите размер.
3. После выбора всех параметров станет доступна кнопка «Добавить в корзину».

3. Корзина и оформление заказа

3.1 Корзина

1. Корзина отображает товары с выбранными цветом и размером.
2. Если в корзине есть товары, это значит, что вы уже авторизованы.
3. В корзине можно изменить количество товаров или удалить их.
4. Итоговая сумма заказа отображается справа.
5. Для перехода к оформлению заказа нажмите кнопку «Оформить заказ».

3.2 Оформление заказа

1. При оформлении заказа выберите пункт выдачи — магазин из доступного списка.
2. Ввод адреса доставки не требуется, так как доставка происходит через выбранный магазин.
3. Проверьте список товаров и итоговую сумму.
4. Подтвердите заказ кнопкой «Подтвердить».

4. Профиль пользователя

4.1 Управление профилем

1. Перейдите в раздел «Профиль» через кнопку в верхнем меню.
2. Здесь вы можете просматривать и редактировать личные данные, такие как имя, email и телефон.
3. В разделе профиля также доступны ваши текущие и прошлые заказы.

5. Отзывы

5.1 Просмотр и добавление отзывов

1. Просматривать отзывы можно без авторизации.
2. Оставлять отзывы могут только авторизованные пользователи.
3. Чтобы оставить отзыв, перейдите в профиль, авторизуйтесь и на странице товара заполните форму отзыва.

# ГЛАВА 8. РУКОВОДСТВО МЕНЕДЖЕРА

????

1. Вход в систему
2. Менеджер входит на сайт с учётными данными, выданными администратором. Самостоятельная регистрация отсутствует.

2. Доступ к панели инструментов

1. После входа в систему в шапке сайта появляется кнопка «Инструменты».
2. Нажмите на кнопку «Инструменты», чтобы открыть панель управления.

3. Создание товара с выбором цвета и размера

1. В панели инструментов нажмите кнопку «Создать товар».
2. Откроется форма создания товара, где необходимо заполнить основные параметры товара.
3. В процессе создания товара у вас будет возможность:
4. выбрать существующий цвет из списка;
5. или создать новый цвет прямо в этой же форме, если подходящего нет;
6. аналогично — выбрать существующий размер или добавить новый размер в процессе создания.
7. Это позволяет менеджеру быстро добавлять новые варианты цветов и размеров без выхода из формы создания товара.
8. После заполнения всех данных и выбора (или создания) цвета и размера подтвердите создание товара.

4. Редактирование товаров

1. В списке товаров нажмите кнопку «Редактировать» напротив нужного товара.
2. В открывшейся форме редактирования можно изменить данные товара, включая цвета и размеры.
3. Если необходимо, также можно создать новые цвета и размеры прямо из формы редактирования товара.
4. Сохраните внесённые изменения.

# ЗАКЛЮЧЕНИЕ

# СПИСОК ЛИТЕРАТУРЫ