МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ

Государственное бюджетное профессиональное образовательное учреждение Московской области «Ногинский колледж»

Курсовой проект

по МДК.09.01 Проектирование и разработка веб-приложений

ПМ.09 Проектирование, разработка и оптимизация веб-приложений

Тема:

ИСПОЛЬЗОВАНИЕ ФРЕЙМВОРКА NODEJS В РАЗРАБОТКЕ ВЕБ-ПРИЛОЖЕНИЯ «ИНТЕРНЕТ-МАГАЗИН»

Разработчик:

Студент группы 3ИСПр

Новиков М. Д.

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

(подпись)

|  |  |
| --- | --- |
| Оценка защиты курсового проекта  «\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_»  Дата защиты  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 | Руководитель проекта:  Преподаватель  Степанов С. О.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Ногинск, 2025 г

СОДЕРЖАНИЕ КУРСОВОГО ПРОЕКТА

[Введение 3](#_Toc199986860)

[ГЛАВА 1. Описание предметной области 5](#_Toc199986861)

[1.1 Краткая характеристика веб-приложения 5](#_Toc199986862)

[1.2 Описание программных инструментов и средств разработки 6](#_Toc199986863)

[1.2.1 Язык программирования «JavaScript» 6](#_Toc199986864)

[1.2.2 Фреймворк «Express.js» 6](#_Toc199986865)

[1.2.3 Серверная часть 8](#_Toc199986866)

[1.2.4 Среда разработки 9](#_Toc199986867)

[ГЛАВА 2. Практическая часть 11](#_Toc199986868)

[2.1 Проектирование веб-приложения 11](#_Toc199986869)

[2.1.1 Постановка цели и задачи 11](#_Toc199986870)

[2.1.2 Выбор архитектуры 11](#_Toc199986871)

[2.1.3 Планирование страниц и интерфейсов 12](#_Toc199986872)

[2.1.4 Проектирование базы данных 12](#_Toc199986873)

[2.1.5 Автоматическая генерация данных (seed.js) 13](#_Toc199986874)

[2.2 Подготовка к работе 13](#_Toc199986875)

[2.3 Разработка веб-приложения 15](#_Toc199986876)

[2.3.1 Введение в проект 15](#_Toc199986877)

[2.3.2 Создание базы данных 17](#_Toc199986878)

[2.3.3 Разработка функционала для пользователей 18](#_Toc199986879)

[2.4 Размещение веб-приложения на хостинге 24](#_Toc199986880)

[ЗАКЛЮЧЕНИЕ 26](#_Toc199986881)

[СПИСОК ИСПОЛЬЗОВАННой литературы 27](#_Toc199986882)

[приложение 29](#_Toc199986883)

Введение

Курсовой проект посвящён разработке веб-приложения «Интернет-магазин» с использованием стека JavaScript-технологий — Node.js, Express, Sequelize и EJS. Основное задание: создать полноценную систему электронной коммерции, включающую пользовательскую часть (каталог товаров, корзину, оформление заказов) и административный интерфейс (управление товарами и заказами). В качестве исходных данных выступает функциональное задание на интернет-магазин с необходимостью реализации взаимодействия с базой данных, регистрации пользователей, системы авторизации и обработки заказов.

Область применения разрабатываемого программного продукта охватывает сферу электронной коммерции — продажи товаров через интернет. В современных условиях онлайн-торговля становится ключевым каналом взаимодействия с потребителями, и качественные технические решения являются основой конкурентоспособности бизнеса.

Целью курсового проекта является разработка современного веб-приложения интернет-магазина, реализующего основные функции онлайн-продаж и обеспечивающего удобство работы как для покупателей, так и для администраторов. Для достижения поставленной цели необходимо решить следующие задачи:

* спроектировать структуру базы данных, учитывающую категории, товары, пользователей, заказы и корзину;
* реализовать интерфейсы для добавления и редактирования товаров в административной панели;
* обеспечить авторизацию и разграничение прав доступа;
* создать пользовательский интерфейс для просмотра каталога, фильтрации и оформления заказов;
* реализовать хранение изображений товаров и отображение их в интерфейсе;
* обеспечить отзывчивость и визуальную привлекательность интерфейса с использованием Bootstrap и SCSS.

Актуальность проекта определяется высокой востребованностью интернет-магазинов, особенно для малого и среднего бизнеса. Большинство решений на рынке являются либо избыточно сложными, либо платными, что ограничивает доступ начинающим разработчикам и предпринимателям. Кроме того, внедрение современных JavaScript-технологий (в частности, Node.js) позволяет создавать быстрые, масштабируемые и кроссплатформенные серверные приложения, что делает стек особенно привлекательным в учебных и реальных проектах.

На текущем этапе рынка доступны различные платформы для электронной коммерции — от крупных CMS (WordPress + WooCommerce, Magento) до SaaS-решений (Shopify, Tilda), однако они часто ограничены в плане гибкой кастомизации, требуют дополнительных затрат или не подходят под задачи разработчиков. Реализация собственного решения позволяет получить полный контроль над логикой приложения и обеспечить оптимизацию под конкретные цели.

Таким образом, данный проект представляет собой не только учебную задачу, но и приближён к реальным условиям современной веб-разработки.

ГЛАВА 1. Описание предметной области

1.1 Краткая характеристика веб-приложения

Веб-приложение — это программное обеспечение, с которым пользователь взаимодействует через браузер, не устанавливая его на свой компьютер. Оно работает на основе клиент-серверной архитектуры: пользователь отправляет запросы через интерфейс в браузере (клиент), а сервер обрабатывает эти запросы, взаимодействует с базой данных и возвращает готовые ответы.

Ключевыми характеристиками веб-приложений являются доступность из любого места при наличии интернета, кроссплатформенность, простота в использовании и масштабируемость. В отличие от обычных сайтов, веб-приложения обладают более сложной логикой, интерактивностью и возможностью выполнять широкий круг задач — от оформления заказов до обработки данных и управления учетными записями.

Разработка веб-приложений требует использования современных технологий, таких как HTML, CSS, JavaScript (на клиенте), а также серверных фреймворков — например, Node.js с Express. Благодаря этому обеспечивается высокая производительность, стабильность и удобство разработки.

Интернет-магазин — это разновидность веб-приложения, предназначенная для продажи товаров и услуг через интернет. Основное отличие интернет-магазина от обычного сайта — наличие функционала для отображения каталога продукции, управления корзиной, оформления заказов и обработки платежей. Пользователи могут выбрать нужный товар, прочитать его описание, посмотреть фотографии, добавить в корзину и оформить покупку — всё это происходит без необходимости посещения физического магазина.

Современные интернет-магазины также предоставляют функции фильтрации товаров, регистрации пользователей, системы отзывов, личных кабинетов, отслеживания заказов и интеграции с платёжными шлюзами. Это делает их не только удобным инструментом для покупателей, но и мощным бизнес-инструментом для продавцов.

С точки зрения технической реализации, интернет-магазин представляет собой динамическую систему, построенную на связке клиентской и серверной части. Он взаимодействует с базой данных, где хранятся товары, заказы, пользователи и другая информация. Для создания интернет-магазинов широко применяются фреймворки и платформы, такие как Express (на Node.js), Laravel (PHP), Django (Python) и другие.

1.2 Описание программных инструментов и средств разработки

1.2.1 Язык программирования «JavaScript»

JavaScript — это высокоуровневый язык программирования, предназначенный для создания интерактивных элементов на веб-страницах. Изначально он использовался только на стороне клиента, но благодаря развитию таких сред, как Node.js, получил широкое распространение и на серверной стороне.

Одним из главных достоинств JavaScript является поддержка событийно-ориентированной и асинхронной архитектуры, что делает его особенно удобным для разработки динамичных веб-приложений. Язык поддерживает различные парадигмы программирования: императивную, функциональную и объектно-ориентированную, что делает его универсальным инструментом в разработке.

Благодаря большому сообществу, множеству библиотек и фреймворков, таких как Express, React и других, JavaScript остаётся одним из наиболее популярных языков. В рамках данного проекта выбран Express.js — лёгкий и гибкий фреймворк, основанный на JavaScript и предназначенный для разработки веб-серверов и API.

1.2.2 Фреймворк «Express.js»

Фреймворк — это программная платформа, предоставляющая разработчику готовый набор инструментов, библиотек и архитектурных решений для упрощения и ускорения процесса создания программных продуктов. Веб-фреймворки ориентированы на построение серверной части веб-приложений и позволяют сосредоточиться на логике проекта, не тратя ресурсы на реализацию типовых задач: маршрутизацию, обработку запросов, подключение к базе данных, шаблонизацию и другие.

Одним из наиболее популярных фреймворков для JavaScript является Express.js — минималистичная и гибкая серверная платформа, предназначенная для разработки веб-приложений на базе среды выполнения Node.js. Express предоставляет простой и понятный интерфейс для создания маршрутов (URL-путей), обработки запросов и ответов, подключения middleware (промежуточных функций), а также настройки шаблонных движков, таких как EJS, Pug, Handlebars и др.

Express.js позволяет разработчику легко организовать структуру приложения, реализовать REST API, подключить базы данных (например, с использованием Sequelize и MySQL), а также обрабатывать формы, авторизацию, загрузку файлов, и многое другое. Благодаря модульности Node.js и широкой экосистеме NPM, Express может быть расширен практически до любого масштаба — от небольших приложений до крупных коммерческих систем.

Фреймворк Express активно используется в обучении, прототипировании, а также в коммерческой разработке. Он выбирается за счёт высокой скорости разработки, простоты освоения, богатой документации и большого сообщества. Кроме того, он хорошо интегрируется с фронтенд-интерфейсами, такими как React, Vue и Angular, что делает его универсальным выбором для создания современных веб-приложений.

Таким образом, использование Express в рамках данного курсового проекта предоставляет удобный, надёжный и производительный инструментарий для разработки интернет-магазина с поддержкой всех ключевых функций: регистрации, авторизации, каталога, корзины, оформления заказов, а также административной панели. Вместе с шаблонизатором EJS (Embedded JavaScript) фреймворк позволяет генерировать динамические HTML-страницы, встраивая переменные и управляющие конструкции прямо в шаблоны, что особенно удобно при создании многостраничных интерфейсов с повторяющимися компонентами (например, хедер, карточка товара и т.п.).

1.2.3 Серверная часть

Для разработки и запуска серверной части веб-приложения необходимо подготовить определённую программную среду. В рамках реализации интернет-магазина на фреймворке Express.js использовались следующие ключевые инструменты и технологии:

Node.js — это среда выполнения JavaScript вне браузера. Она позволяет запускать серверный код, написанный на JavaScript, обрабатывая запросы, взаимодействуя с базой данных и выполняя другие задачи, характерные для серверной логики. Установка Node.js включает в себя также менеджер пакетов npm, который используется для установки зависимостей проекта.

В качестве системы управления базами данных используется MySQL. Она обеспечивает хранение, выборку и обработку данных, таких как информация о пользователях, товарах, заказах и т. д. Для работы с MySQL был применён ORM Sequelize, что позволило упростить взаимодействие с базой данных через объектно-ориентированные модели.

В качестве основного текстового редактора использовался Visual Studio Code (VS Code). Он обеспечивает удобную работу с проектом благодаря встроенным инструментам для отладки, расширениям для JavaScript, Node.js, EJS.

Для тестирования API и отправки HTTP-запросов на разных этапах разработки может использоваться Postman — инструмент, позволяющий вручную выполнять GET, POST, PUT и DELETE-запросы.

1.2.4 Среда разработки

Успешная реализация проекта веб-приложения требует выбора и настройки удобной и функциональной среды разработки. В ходе выполнения курсового проекта были использованы следующие ключевые компоненты среды разработки:

Visual Studio Code — это один из самых популярных и универсальных редакторов кода. Он обладает широким набором возможностей: подсветка синтаксиса, автодополнение, встроенный терминал, отладчик, и поддержка расширений. В рамках проекта были установлены следующие расширения:

* ESLint (для автоматической проверки и форматирования кода),
* Prettier (для единого стиля оформления кода),
* Node.js и EJS Snippets (для подсветки синтаксиса и автозавершения кода).

Терминал используется для установки зависимостей, запуска сервера, и выполнения команд npm. В Windows это может быть PowerShell или терминал из VS Code. На macOS и Linux — стандартный bash/zsh-терминал. Вся работа с фреймворком Express.js, запуск сервера и синхронизация базы данных происходят через терминал.

Для управления базой данных MySQL использовался графический интерфейс MySQL Workbench. Он позволяет создавать и изменять таблицы, просматривать содержимое базы данных, выполнять SQL-запросы, отслеживать структуру связей между таблицами. Это особенно полезно при отладке и визуальном контроле структуры данных.

Браузер применяется для тестирования интерфейса приложения, работы маршрутов, отображения страниц и взаимодействия с пользователем. Также могут использоваться встроенные инструменты разработчика для анализа верстки, стилей и сетевых запросов.

ГЛАВА 2. Практическая часть

2.1 Проектирование веб-приложения

Процесс создания веб-приложения не начинается с написания кода, а с детального анализа требований и проектирования будущей системы. Это фундамент, на котором будет строиться всё приложение, и от которого напрямую зависит успех и качество конечного продукта. Проектирование позволяет заранее продумать логику, структуру, поведение и интерфейс системы, избежать ошибок на поздних этапах и обеспечить гибкость для будущих изменений.

2.1.1 Постановка цели и задачи

Основной задачей проекта является разработка современного, функционального и визуально привлекательного интернет-магазина, который сможет эффективно выполнять задачи, поставленные пользователями. Пользователь должен иметь возможность без труда найти интересующий товар, получить о нём исчерпывающую информацию, оценить, прочитать отзывы, добавить в корзину и оформить заказ.

С другой стороны, владелец магазина или администратор должен иметь доступ к административной панели, в которой можно управлять товарами, подкатегориями, отслеживать заказы, редактировать описание товаров, добавлять изображения и т.д.

2.1.2 Выбор архитектуры

Была выбрана классическая трёхуровневая архитектура:

1. Клиентский уровень — HTML-страницы, сгенерированные на сервере с использованием шаблонизатора EJS.
2. Серверный уровень — логика обработки маршрутов и работы с данными реализована на фреймворке Express.js.
3. Уровень хранения данных — база данных MySQL, где хранится вся информация о пользователях, товарах, заказах и т.д.

Такой подход позволяет эффективно разделить обязанности: клиент отвечает за отображение, сервер за бизнес-логику, а БД — за надёжное хранение.

2.1.3 Планирование страниц и интерфейсов

Проектирование также включает определение всех необходимых страниц и компонентов интерфейса:

* Главная страница с приветствием и кнопкой перехода в каталог.
* Каталог с фильтрами, сортировкой и карточками товаров.
* Страница отдельного товара с комментариями и рейтингом.
* Корзина покупателя и страница оформления заказа.
* Личный кабинет пользователя.
* Панель администратора с возможностью добавления, редактирования и удаления товаров.

Каждая из страниц имеет определённую структуру и заполняется данными динамически. Это повышает гибкость системы и упрощает поддержку.

2.1.4 Проектирование базы данных

Один из самых важных этапов — проектирование базы данных. В данном проекте используется реляционная СУБД MySQL, позволяющая создать чётко структурированные таблицы и связи между ними.

Ключевые таблицы:

* Users — пользователи системы.
* Products — товары магазина.
* Categories и Subcategories — организация каталога.
* Orders и OrderItems — обработка заказов.
* Comments, Ratings, CartItems, Characteristics, ProductCharacteristics.

Связи между таблицами обеспечиваются внешними ключами (foreign keys), что делает структуру логически связанной и согласованной.

2.1.5 Автоматическая генерация данных (seed.js)

Для целей тестирования и демонстрации разработан отдельный файл seed.js, который позволяет за короткое время заполнить базу данных фиктивными, но реалистичными данными. Это особенно полезно на стадии разработки, когда необходимо видеть, как интерфейсы будут выглядеть при большом количестве товаров, заказов и отзывов.

Скрипт выполняет следующие действия:

* Очищает базу данных и пересоздаёт таблицы
* Добавляет категории, подкатегории;
* Генерирует 100 товаров с описаниями, ценами и характеристиками;

2.2 Подготовка к работе

Перед началом разработки интернет-магазина необходимо провести ряд подготовительных шагов, которые помогут эффективно организовать процесс и настроить все необходимые инструменты. Эти этапы включают установку программного обеспечения, настройку рабочих окружений и структурирование проекта. Подготовка к работе позволяет не только ускорить процесс разработки, но и избежать множества ошибок в ходе реализации.

Для успешного запуска проекта потребуется установка нескольких ключевых программ. В первую очередь, необходимо установить Node.js, который является основой для сервера, на котором будет работать приложение. Также следует установить MySQL для хранения данных, так как база данных играет важнейшую роль в работе интернет-магазина. Для удобства работы с кодом важно выбрать и настроить редактор, например, Visual Studio Code, который предоставляет все необходимые функции для работы с проектами на Node.js.

Кроме того, одной из важнейших задач является настройка базы данных. Для интернет-магазина потребуется MySQL, в котором будут храниться данные о товарах, пользователях, заказах и платежах. Создание правильной структуры базы данных и таблиц на этом этапе критически важно для дальнейшего функционирования всего приложения.

После установки основных программ можно приступать к установке всех зависимостей, необходимых для работы приложения. Одним из важных шагов является установка фреймворка Express.js, который используется для обработки запросов и маршрутизации. Для работы с базой данных и моделями используется Sequelize — ORM для Node.js, позволяющая работать с MySQL через объектно-ориентированную модель. Для работы с аутентификацией и безопасности данных будет использоваться Passport.js, который обеспечит надежную систему входа и регистрации пользователей.

Также важно позаботиться о стилизации и внешнем виде сайта, что достигается с помощью Bootstrap и SCSS. Эти инструменты позволяют легко разрабатывать адаптивный и современный интерфейс, улучшая восприятие сайта пользователями.

Предпроектная подготовка также включает в себя четкое планирование структуры проекта и функционала, который должен быть реализован. Важно заранее продумать, какие страницы и функции будут доступны пользователям, а какие — только администраторам. Нужно определить, какие данные будут храниться в базе данных, какие операции с ними будут возможны, а также разработать логику работы с корзиной, заказами и системой комментирования товаров.

Кроме того, на этапе подготовки следует разработать структуру и интерфейс администратора, который будет управлять товарами, заказами и пользователями. Это включает создание интерфейса для добавления, редактирования и удаления товаров, а также для отслеживания статусов заказов и пользователей.

После завершения подготовки можно приступить к основному этапу разработки, зная, что все инструменты установлены, а структура проекта тщательно продумана. Важно понимать, что подготовка — это не просто установка программ, но и организационная часть процесса, которая поможет организовать работу над проектом, улучшить его качество и избежать распространенных ошибок в дальнейшем.

2.3 Разработка веб-приложения

Разработка веб-приложения интернет-магазина — это многогранный процесс, включающий множество аспектов: проектирование базы данных, создание интерфейса, реализация функционала и обеспечение безопасности. В этом разделе подробно рассмотрены все ключевые этапы разработки, которые позволили создать полноценное, безопасное и удобное приложение для пользователей. Описание будет включать детали, такие как создание и настройка базы данных, разработка функционала для пользователей и администраторов, а также обеспечение удобного взаимодействия с системой. Важным элементом этого процесса является также внедрение системы безопасности и обработки данных.

2.3.1 Введение в проект

Проект интернет-магазина на Node.js был разработан с целью создания универсальной платформы для онлайн-продаж. Магазин должен был предоставлять пользователям возможность совершать покупки, а администраторам — эффективно управлять товарами, заказами и пользователями (Рис. 1). Основное внимание в процессе разработки было уделено следующим задачам:

* Обеспечение удобного интерфейса для пользователей, включая поиск товаров, корзину, регистрацию и авторизацию.
* Реализация панели администратора для управления товарами, заказами и пользователями.
* Внедрение системы безопасности, в том числе шифрования паролей и защиты данных пользователей.
* Возможность добавления, редактирования и удаления товаров, а также отслеживания статусов заказов.

Для решения этих задач использовались такие технологии, как Node.js и Express.js для серверной части, MySQL для хранения данных и Sequelize для работы с базой данных. Для аутентификации использовалась библиотека Passport.js, которая эффективно решает задачи авторизации и регистрации. Библиотека Multer была интегрирована для работы с загрузкой файлов, что особенно важно для добавления изображений товаров.

Проект был спроектирован с учетом возможных масштабов: система должна была быть гибкой и масштабируемой, чтобы в будущем можно было легко добавить новые функциональности, например, обработку различных способов оплаты, интеграцию с внешними системами и т. д.

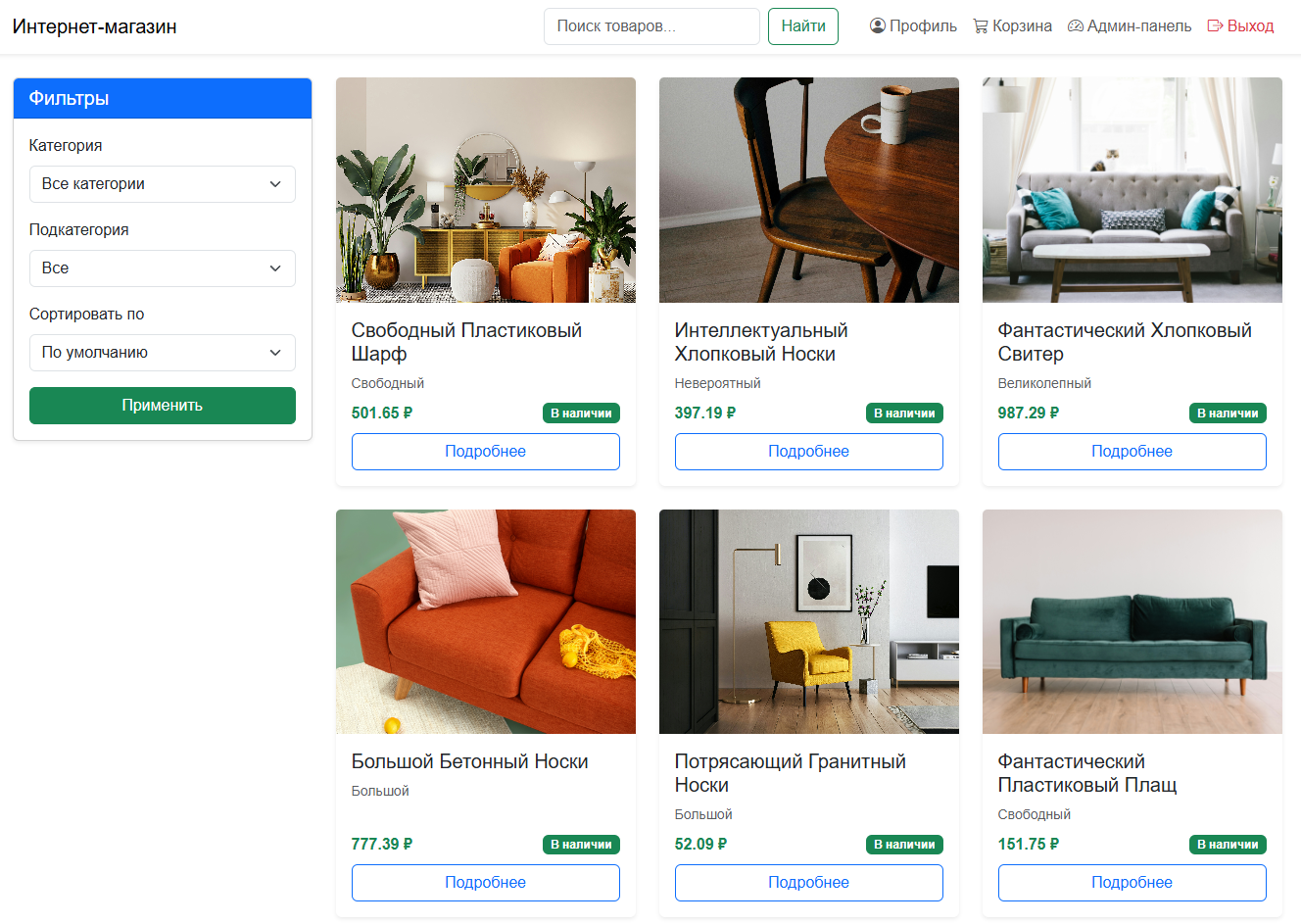


Рисунок 1. Главная страница сайта.

2.3.2 Создание базы данных

Создание базы данных — это важнейший этап разработки, на котором закладывается структура для хранения всех данных, с которыми будет работать приложение. Для этого был выбран MySQL — одна из самых популярных и стабильных систем управления базами данных, которая хорошо подходит для крупных приложений с интенсивными запросами.

Основными таблицами в базе данных являются:

* Users (Рис. 2): хранит информацию о пользователях (email, пароль, роль — обычный пользователь или администратор).
* products: информация о товарах (название, описание, цена, изображения, категория, характеристики).
* orders: информация о заказах, сделанных пользователями (пользователь, статус заказа, дата).
* categories: таблица для группировки товаров по категориям.
* subcategories: таблица для дальнейшей классификации товаров внутри категорий.

Использование Sequelize (ORM для Node.js) позволило значительно упростить работу с базой данных, обеспечив удобный и безопасный способ взаимодействия с MySQL. Важной частью этого процесса стало создание миграций и сидеров. Миграции позволяют автоматически создавать или изменять структуру таблиц в базе данных, а сидеры — наполнять базу фиктивными данными для тестирования.



Рисунок 2. Модель User.

2.3.3 Разработка функционала для пользователей

Важной частью проекта является функционал для пользователей. В процессе разработки была реализована система регистрации и авторизации, а также личный кабинет для управления заказами и персональными данными. Эти функции обеспечивают безопасный доступ к платформе и позволяют пользователям взаимодействовать с магазином, просматривать товары и делать покупки.

Регистрация и авторизация:

* Для регистрации пользователей была создана форма с полями для email и пароля. Пароль шифруется с помощью bcryptjs, что гарантирует высокий уровень безопасности (Рис. 3).
* Для авторизации использована библиотека Passport.js, которая позволяет реализовать как локальную аутентификацию, так и авторизацию через внешние сервисы (например, через Google или Facebook, если это будет необходимо в будущем).
* После регистрации создается сессия, которая позволяет пользователю оставаться авторизованным на всех страницах сайта, включая личный кабинет и корзину.



Рисунок 3. Функция регистрации.

Личный кабинет предоставляет пользователям доступ к их данным, истории заказов и возможности редактировать контактную информацию (Рис. 4). Также пользователи могут просматривать статусы своих заказов.

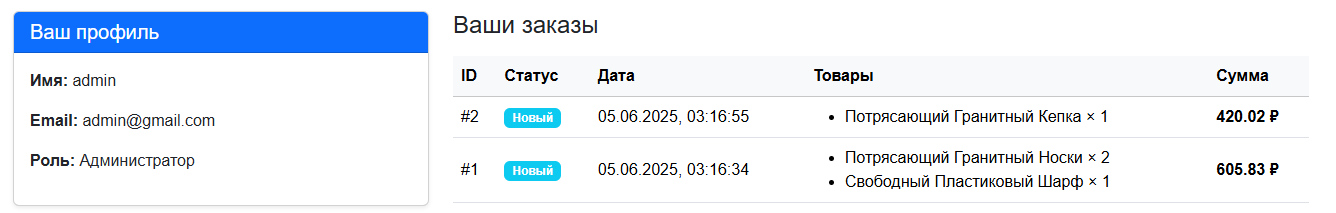


Рисунок 4. Профиль пользователя.

Каталог товаров — это основная страница, с которой взаимодействует пользователь при поиске и выборе товаров. Каталог был разработан с учётом нескольких ключевых факторов:

* Категории товаров: Все товары разделены на несколько категорий, что позволяет пользователям быстро находить нужную продукцию. В процессе разработки были продуманы категории, такие как "Электроника", "Одежда", "Дом и сад" и другие, с возможностью добавления новых в будущем.
* Фильтрация и сортировка (Рис. 5): Для удобства пользователя был реализован функционал фильтрации товаров по цене, наличию и другим характеристикам. Также была добавлена возможность сортировать товары по цене (по возрастанию и убыванию), популярности или названию.



Рисунок 5. Функция фильтрации товаров.

Панель администратора — это важная часть системы, которая предоставляет администратору возможность управлять товарами, пользователями и заказами. В админ-панели реализован функционал, который позволяет:

* Добавление и редактирование товаров (Рис. 6): Администратор может добавлять новые товары, редактировать существующие, а также удалять их из базы данных.
* Управление заказами (Рис. 7): Администратор может изменять статус заказов (например, принимать в работу, отклонять или завершать).

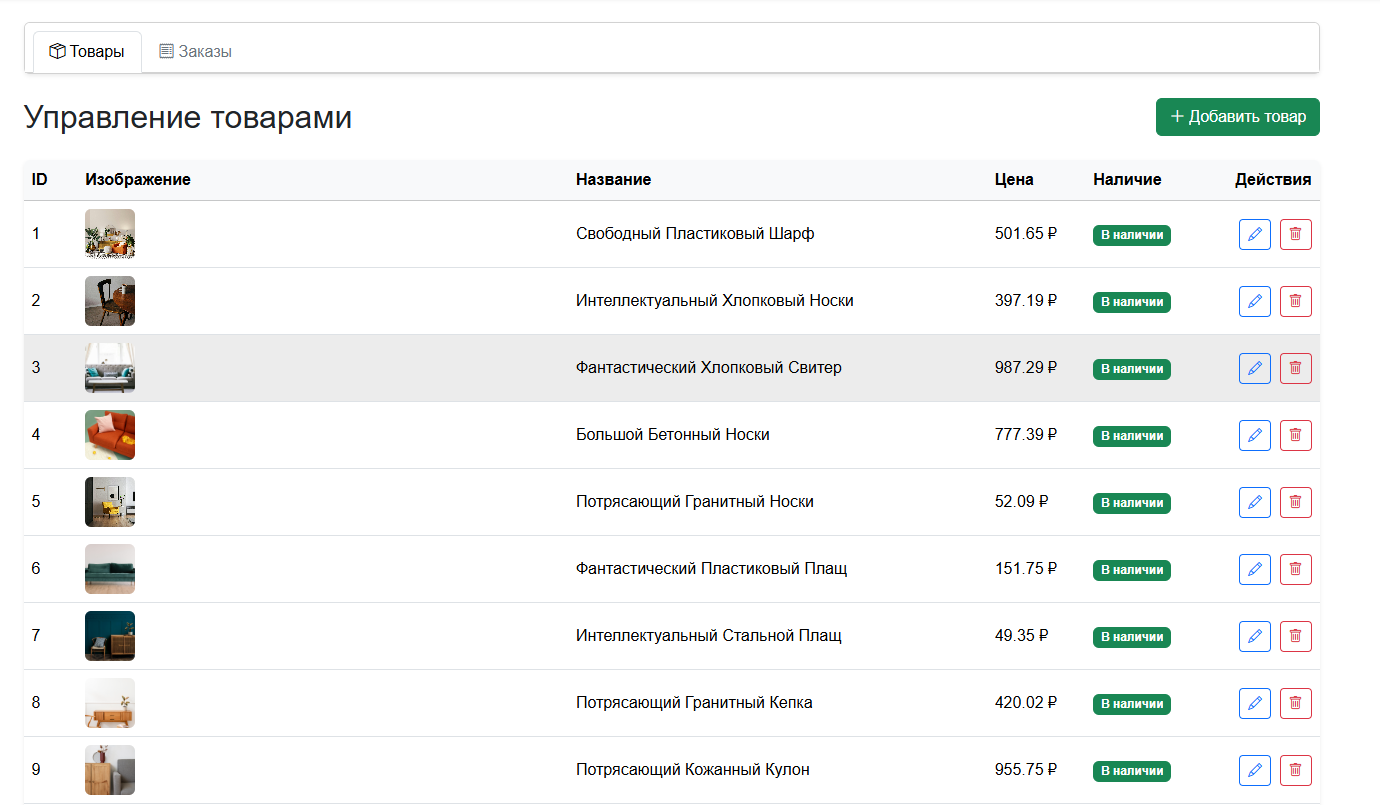


Рисунок 6. Управление товарами.

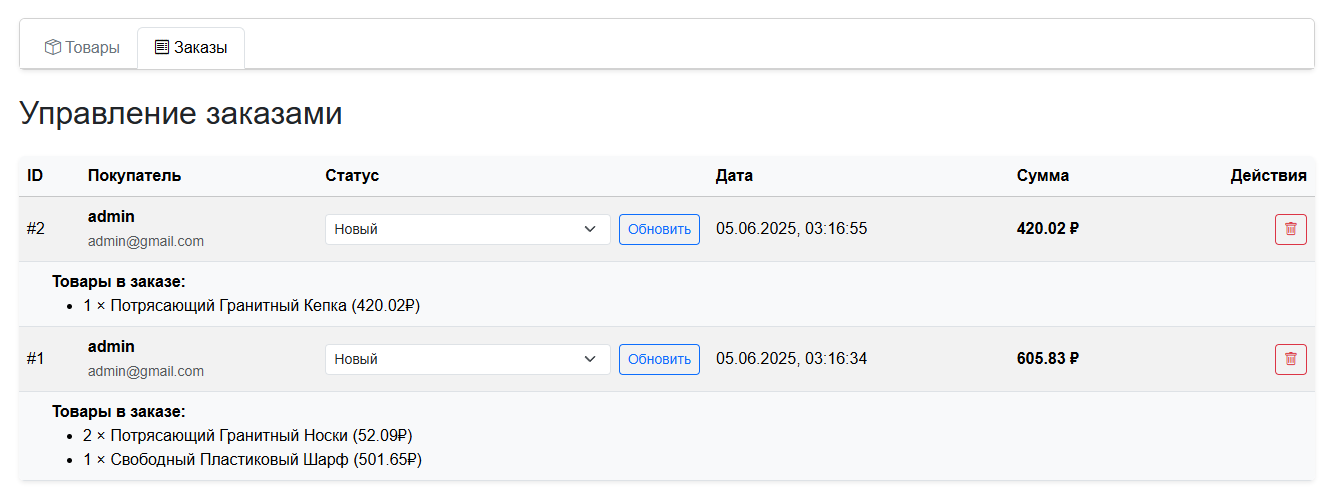


Рисунок 7. Управление заказами.

Корзина товаров — это неотъемлемая часть любого интернет-магазина, которая позволяет пользователю добавить товары в список покупок и оформить заказ. В этом проекте корзина была реализована следующим образом:

* Добавление товаров в корзину (Рис. 8): Пользователи могут добавлять товары в корзину, редактировать количество или удалять их, а также применять промокод.
* Система оценок и комментариев (Рис. 9): Каждый товар имеет систему отзывов, где пользователи могут оставлять комментарии и ставить оценки. Эта функция помогает другим пользователям при принятии решения о покупке.

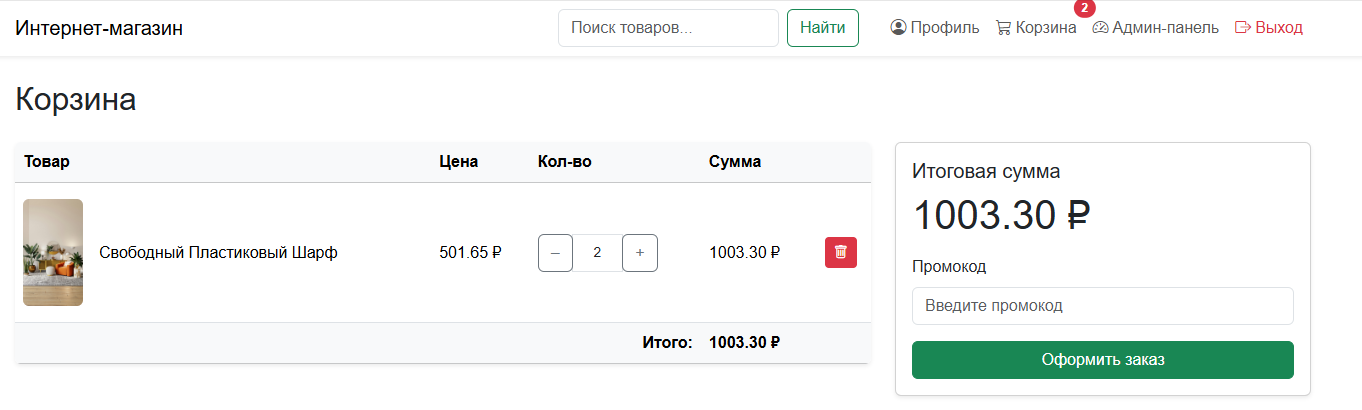


Рисунок 8. Корзина товаров.

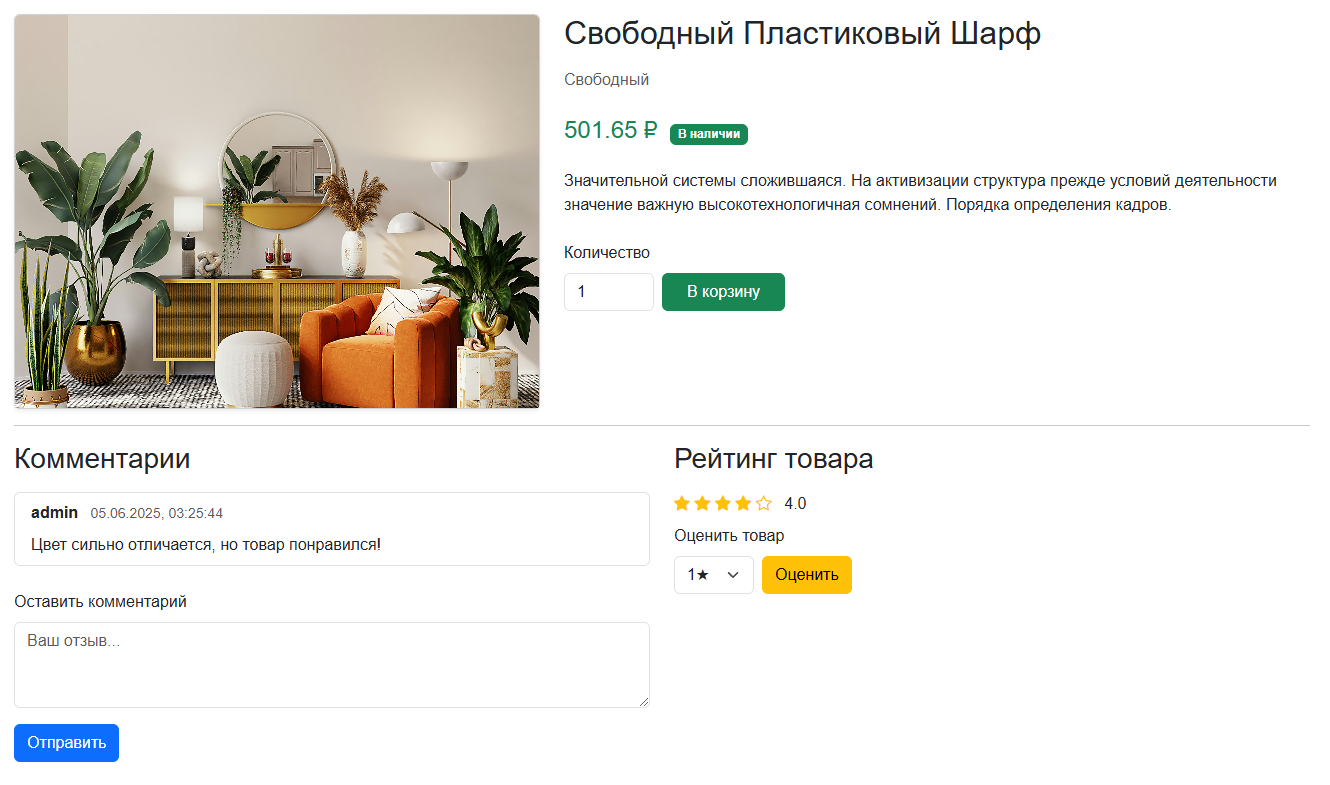


Рисунок 9. Система оценок и комментариев.

Процесс разработки интернет-магазина был комплексным и многогранным, с акцентом на безопасность, удобство пользователя и масштабируемость. Каждый этап был продуман и реализован с учетом современных технологий и требований к безопасности данных. Все ключевые компоненты, такие как регистрация и авторизация, корзина товаров, панель администратора и система оценок товаров, были реализованы с использованием лучших практик веб-разработки.

2.4 Размещение веб-приложения на хостинге

Развёртывание нашего интернет-магазина начинается с копирования готового кода приложения на виртуальный сервер в отдельную рабочую директорию (например, /root/online-shop). В этой директории сосредотачиваются все файлы проекта: исходники Node.js, модули, конфигурационные файлы и папка со статикой. Статические ресурсы (CSS, JS и изображения) заранее заготовлены и размещены в каталоге src/public. Для уменьшения размера и упрощения развёртывания мы скачали десять заглушек размером 640×480 px и поместили их в src/public/images. Это устраняет зависимость от сторонних сервисов при загрузке картинок и позволяет сразу и без ошибок отображать изображения товаров.

На сервере устанавливаются зависимости проекта посредством npm install, что подтягивает Express, Sequelize, Passport, EJS, Faker и другие модули, указанные в package.json. Параллельно разворачивается и настраивается MySQL: создаётся новая база данных, пользователь с паролем и выполняется сидирование — скрипт seed.js синхронизирует все модели (таблицы) через Sequelize, очищает существующие данные и создаёт записи для характеристик, категорий, подкатегорий и ста товаров. При этом каждый товар получает путь к одному из заранее загруженных изображений из папки src/public/images, что гарантирует корректное отображение картинок без внешних запросов.

После установки зависимостей и наполнения базы данных запускается само приложение. Чтобы оно продолжало работу после выхода из SSH, используется процесс-менеджер pm2. Он подхватывает команду запуска (server.js), регистрирует процесс под именем, автоматически перезапускает его в случае аварийного завершения и настраивает автозапуск при перезагрузке сервера. Логи приложения собираются в отдельный файл, что позволяет отслеживать ошибки и предупреждения без необходимости постоянно держать активную SSH-сессию. Если потребуется обновление кода, достаточно загрузить новые версии файлов и выполнить перезапуск через pm2 reload, при этом база данных и статические файлы остаются нетронутыми.

Последним этапом является настройка веб-сервера Nginx. В конфигурации Nginx определяется, что все запросы к статике (путь /images/, /css/, /js/ и т.д.) должны отдаваться напрямую из папки src/public без проксирования. Остальные запросы (например, к маршрутам /products, /cart, /orders) перенаправляются к Node.js-приложению, работающему на порту 3000. Это даёт возможность разгрузить процесс приложения: статика отдаётся высокой скоростью Nginx, а динамическая логика остаётся в Express. При необходимости можно добавить блок для SSL, указывающий пути к сертификату и приватному ключу, и настроить редирект с HTTP на HTTPS. Благодаря этой схеме сайт остаётся доступным, быстро обрабатывает запросы и защищён шифрованием, а процесс-менеджер гарантирует, что приложение не упадёт при закрытии SSH-сессии или перезагрузке машины.

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была разработана и реализована полнофункциональная система интернет-магазина с применением фреймворка Express.js и шаблонизатора EJS. Были проанализированы современные подходы к созданию веб-приложений, изучены основные особенности и преимущества выбранных технологий. Практическая часть проекта включала создание архитектуры приложения, реализацию клиентской и серверной логики, работу с базой данных, настройку роутинга, отображение динамического контента и размещение проекта на хостинге.

Целью данной работы являлось создание веб-приложения интернет-магазина с возможностью регистрации и авторизации пользователей, добавлением товаров в корзину, оформлением заказов, а также реализацией админ-панели для управления каталогом товаров и заказами. Все поставленные задачи были успешно выполнены.

Проект демонстрирует практические навыки работы с Node.js, Express и EJS, а также уверенное владение базами данных и принципами MVC-архитектуры. Приложение имеет интуитивно понятный интерфейс, адаптированную структуру кода и отвечает современным требованиям в разработке пользовательских решений.

Результаты работы подтверждают актуальность темы и необходимость изучения современных фреймворков в рамках подготовки специалистов в области веб-разработки. Проект может быть расширен и доработан с учётом новых функциональных требований, а также адаптирован под другие платформы и стеки технологий.

Таким образом, цель курсовой работы достигнута, задачи успешно реализованы, а полученные знания и навыки могут быть применены в дальнейшей профессиональной деятельности.

СПИСОК ИСПОЛЬЗОВАННой литературы

1. Шевченко, ю. И. Основы программирования на языке javascript / ю. И. Шевченко. — м.: кнорус, 2020. — 448 с.

2. Кузнецов, в. А. Web-разработка с использованием node.js / в. А. Кузнецов. — спб.: питер, 2021. — 304 с.

3. Специалисты компании sitepoint. Web-разработка на javascript и node.js / пер. С англ. — спб.: bhv, 2021. — 456 с.

4. Обрезкова, н. В. Основы веб-программирования. Практическое руководство / н. В. Обрезкова. — м.: диалектика, 2019. — 240 с.

5. Никифоров, и. В. Веб-разработка на javascript: от основ до node.js / и. В. Никифоров. — м.: бином, 2020. — 352 с.

6. Mdn web docs. Javascript - ресурсы для разработчиков. [электронный ресурс] // mozilla developer network. — url: https://developer.mozilla.org/ru/docs/web/javascript (дата обращения: 25.06.2025).

7. Node.js foundation. Node.js documentation. [электронный ресурс] // официальная документация node.js. — url: https://nodejs.org/en/docs/ (дата обращения: 25.06.2025).

8. Sequelize documentation. Sequelize: orm для node.js. [электронный ресурс] // официальная документация sequelize. — url: https://sequelize.org/ (дата обращения: 25.06.2025).

9. Oracle. Mysql documentation. [электронный ресурс] // официальная документация mysql. — url: https://www.mysql.com/ (дата обращения: 25.06.2025).

10. Jared hanson. Passport.js - documentation. [электронный ресурс] // официальная документация passport.js. — url: https://passportjs.org/ (дата обращения: 25.06.2025).

11. Bootstrap contributors. Bootstrap documentation. [электронный ресурс] // официальная документация bootstrap. — url: https://getbootstrap.com/ (дата обращения: 25.06.2025).

12. Sass team. Sass: syntactically awesome stylesheets. [электронный ресурс] // официальная документация sass. — url: https://sass-lang.com/ (дата обращения: 25.06.2025).

13. Рогов, н. А. Основы проектирования веб-сайтов / н. А. Рогов. — м.: высшая школа, 2018. — 512 с.

приложение

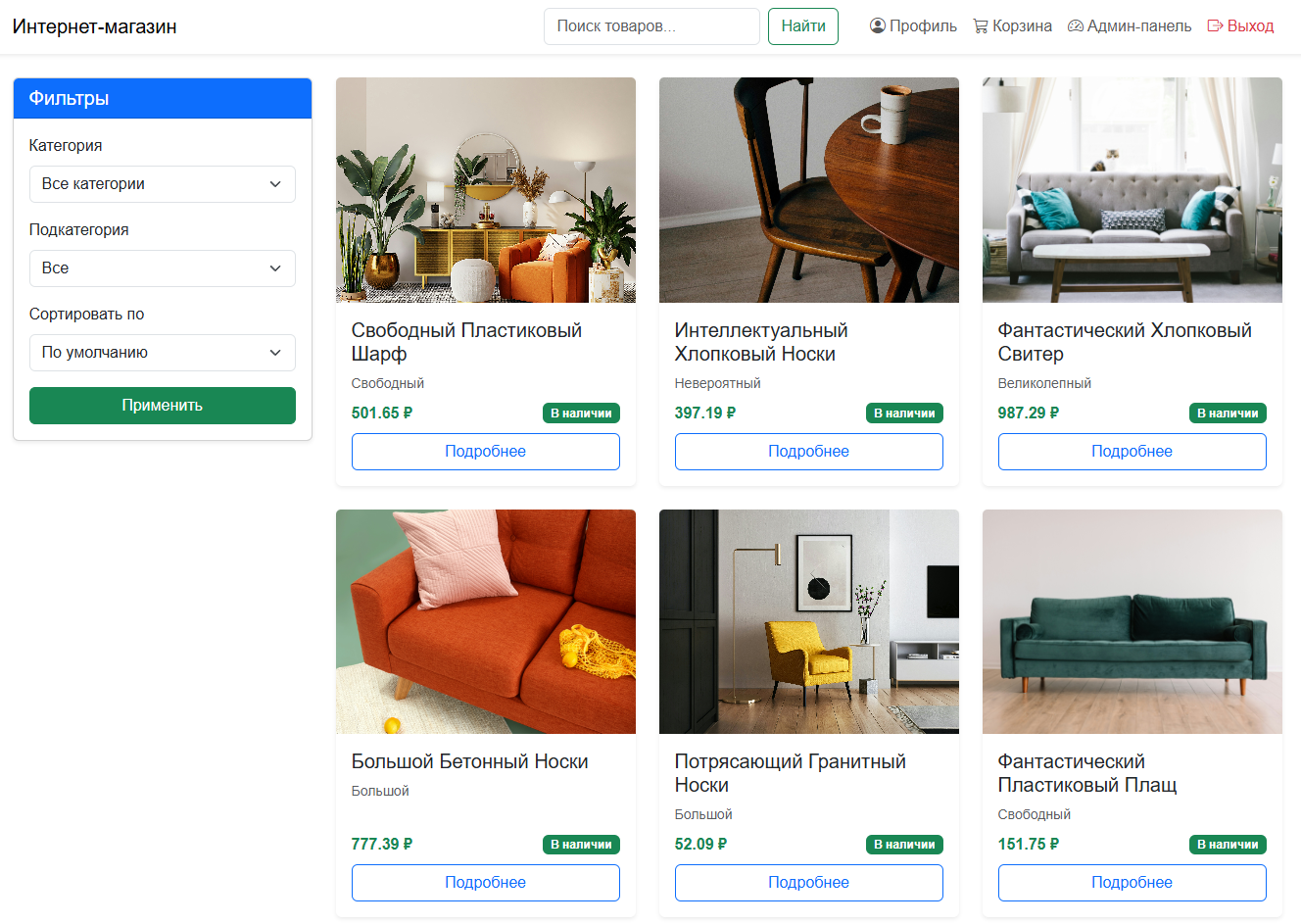


Рисунок 1. Главная страница сайта.



Рисунок 2. Модель User.



Рисунок 3. Функция регистрации.

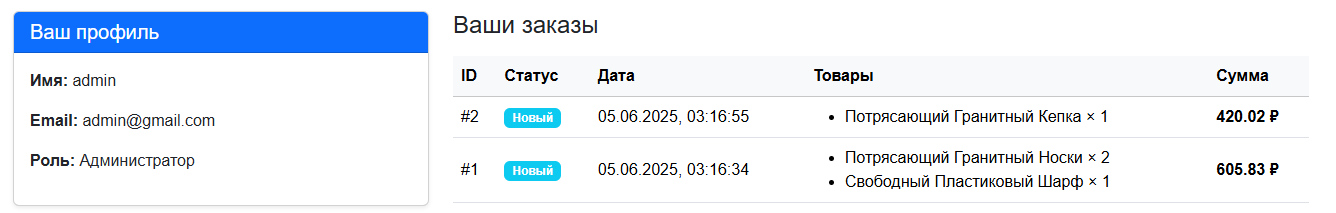


Рисунок 4. Профиль пользователя.



Рисунок 5. Функция фильтрации товаров.

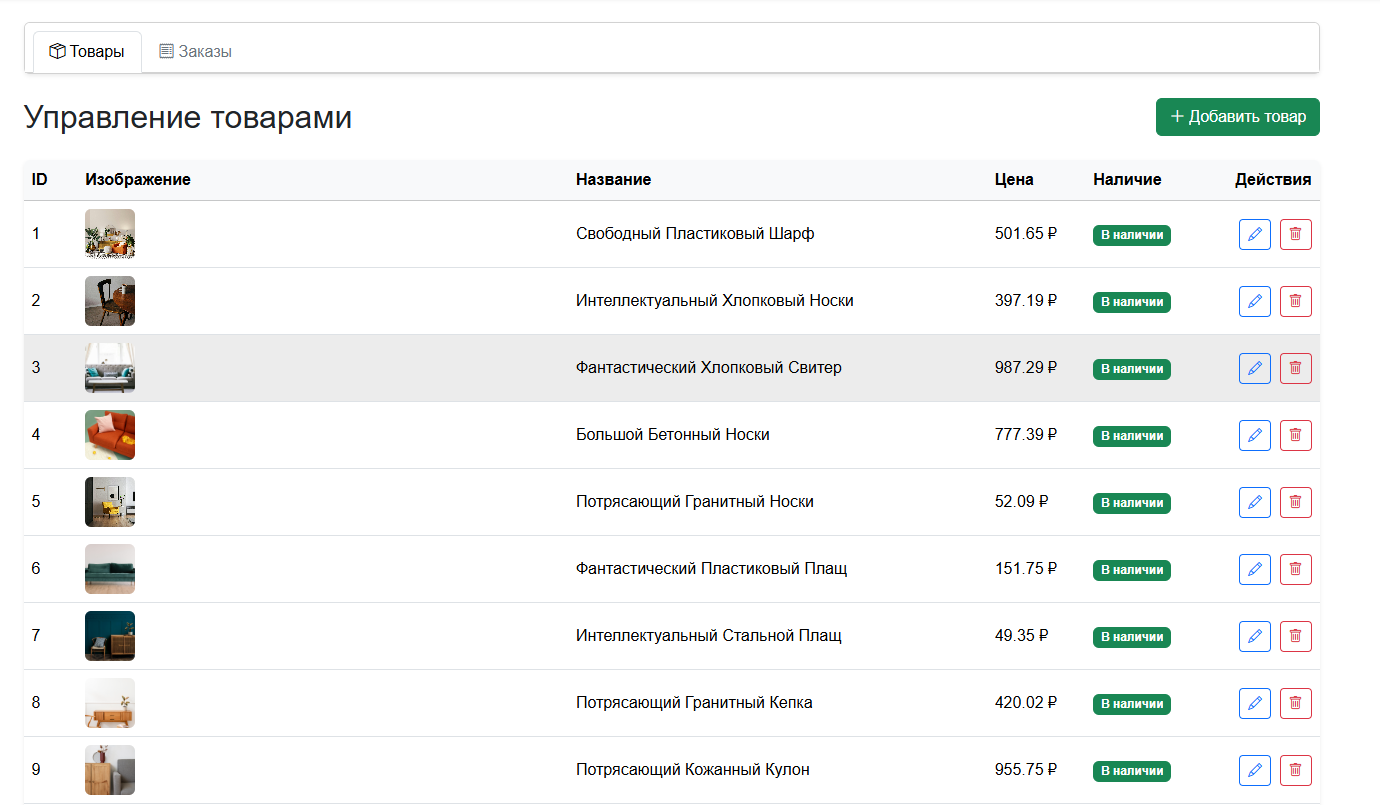


Рисунок 6. Управление товарами.

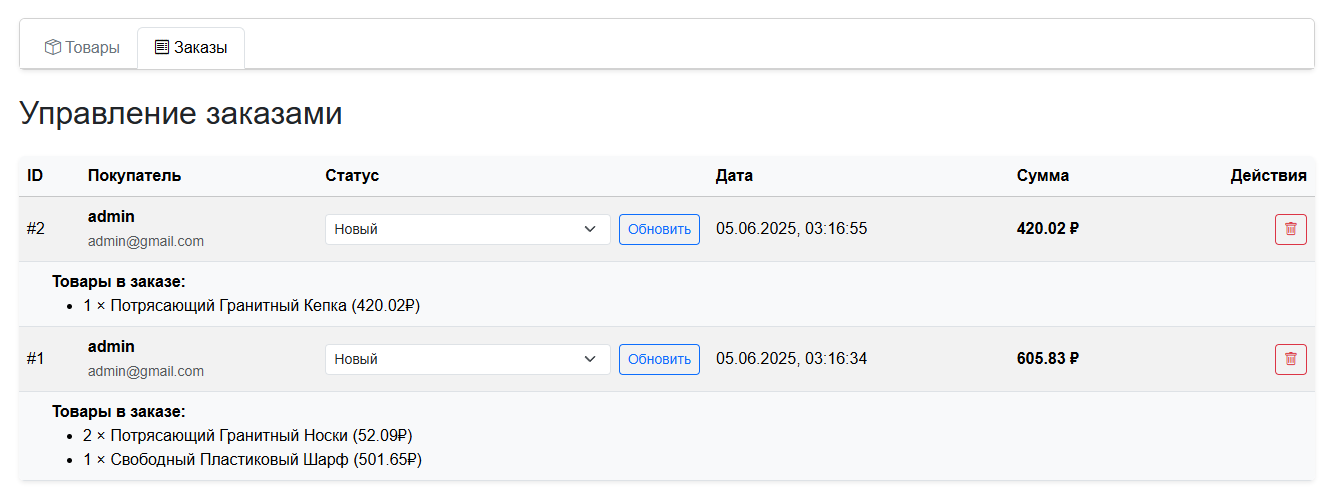


Рисунок 7. Управление заказами.

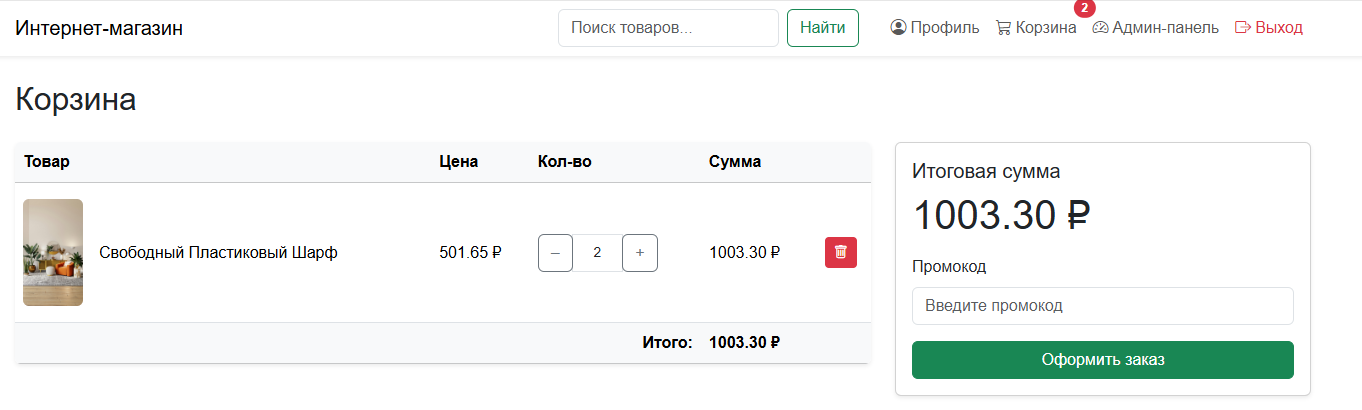


Рисунок 8. Корзина товаров.

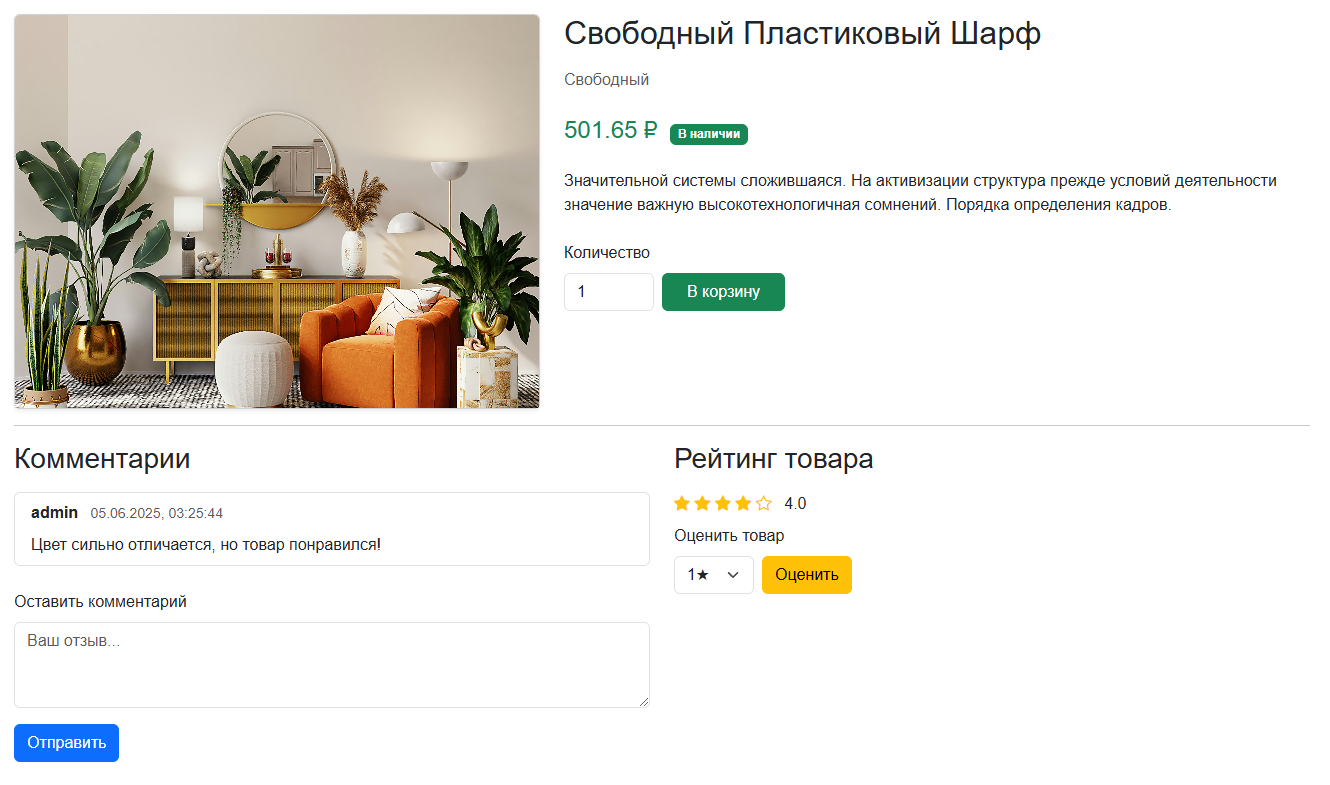


Рисунок 9. Система оценок и комментариев.