Федеральное государственное образовательное бюджетное
учреждение высшего образования
**«Финансовый университет
при Правительстве Российской Федерации»
(Финансовый университет)**

Институт открытого образования

Пояснительная записка к курсовой работе по дисциплине
«Современные технологии программирования»
на тему:

«Информационно-справочная система магазина цифровой техники»

Выполнил:
Студент группы ДПИ23-1
Кобзарь М. А.

Научный руководитель:
к.т.н., доцент
Антонов А. А.

Москва – 2025

ОГЛАВЛЕНИЕ

Введение

В современном мире цифровая техника занимает особое место в повседневной жизни людей. Согласно данным статистики, рынок цифровой техники в России демонстрирует устойчивый рост, несмотря на экономические вызовы последних лет [1]. В условиях высокой конкуренции и постоянно меняющихся потребительских предпочтений эффективное управление ассортиментом, запасами, продажами и клиентской базой становится ключевым фактором успеха для розничных магазинов электроники [2].

Современные информационные системы позволяют автоматизировать многие бизнес-процессы, значительно повышая эффективность работы предприятий торговли. Как показывают исследования, внедрение специализированных программных решений в розничной торговле цифровой техникой способствует сокращению издержек на 15-20%, увеличению скорости обработки заказов на 30% и повышению уровня удовлетворенности клиентов на 25% [3].

В настоящее время многие небольшие и средние магазины цифровой техники продолжают использовать устаревшие методы учета или упрощенные программные решения, что приводит к ошибкам в управлении запасами, потерям клиентов и снижению прибыли [4]. Актуальность разработки современной информационно-справочной системы для магазина цифровой техники обусловлена необходимостью создания комплексного решения, которое бы объединяло функции учета товаров, управления заказами, анализа продаж и взаимодействия с клиентами в единой платформе.

Объектом исследования является процесс управления деятельностью магазина цифровой техники.

Предметом исследования выступает информационно-справочная система для автоматизации бизнес-процессов магазина цифровой техники.

Целью курсовой работы является разработка информационно-справочной системы магазина цифровой техники, обеспечивающей эффективное управление ассортиментом товаров, заказами, клиентской базой и аналитикой продаж.

Для достижения поставленной цели необходимо решить следующие задачи:

- разработать клиент-серверное приложение на языке программирования Java, предназначенное для автоматизации деятельности магазина цифровой техники;
- создать серверную часть приложения на основе фреймворка Spring Boot, отвечающую за обработку запросов клиентов и управление данными;
- разработать клиентскую часть приложения с графическим интерфейсом, обеспечивающую удобное взаимодействие с системой для различных категорий пользователей;
- применить модель MVC (Model View Controller) для разделения управляющей логики на отдельные компоненты, что позволит улучшить читаемость кода и упростит его поддержку;
- реализовать функционал для работы с базой данных, включающий добавление, редактирование, удаление и поиск товаров, клиентов и заказов;
- обеспечить безопасность хранения данных и управление правами доступа для различных ролей пользователей;

– разработать механизм генерации отчетов и аналитических данных для принятия управленческих решений.

Для решения поставленных задач используются современные технологии и инструментальные средства: Java, Spring Boot [5-7], MySQL, ORM-технологии (Hibernate/JPA) [8]. Применение этих технологий позволяет создать надежную, масштабируемую и кроссплатформенную систему, соответствующую современным требованиям к информационным решениям для розничной торговли.

Таким образом, разработка информационно-справочной системы магазина цифровой техники является актуальной задачей, решение которой позволит автоматизировать ключевые бизнес-процессы и повысить эффективность работы предприятия.

1. Описание программы

1.1. Алгоритмические решения

1.1.1. Безопасность

В информационно-справочной системе магазина цифровой техники осуществлена многоуровневая система безопасности, обеспечивающая защиту пользовательских данных и нужную работу приложения. Основой архитектурой безопасности является фреймворк Spring Security [12], который предоставляет многоаспектные механизмы аутентификации и авторизации.

Система использует ролевую модель доступа (RBAC) с тремя рангами привелегий: CLIENT, MANAGER и ADMIN. При запуске приложения конфигурируется цепочка фильтров безопасности, устанавливающая доступ к всяческим ресурсам исходя из роли юзера. Для аутентификации пользователей осуществляется механизм DaoAuthenticationProvider, который использует UserDetailsService для загрузки сведений о пользователях из базы данных и PasswordEncoder для хеширования паролей с применением алгоритма BCrypt [12].

Пароли пользователей никогда не хранятся в открытом виде. При создании аккаунта или смене пароля система по умолчанию преобразует пароль в защищенный хеш-код, внося уникальную случайную последовательность (соль) для каждого участника системы, что предоставляет защиту даже в ситуации компрометации базы данных. Для дополнительной безопасности сделана защита от межсайтовой подделки запросов (CSRF), не допусткающая нелегальное выполнение действий от имени подтвержденного пользователя.

Для предотвращения встречающихся веб-атак, таких как SQL-инъекции и XSS-уязвимости, применяются такие меры: использование ORM Hibernate с параметризованными запросами вместо конкатенации SQL-строк, валидация и экранирование всех исходных данных, а также

применение шаблонизатора Thymeleaf, который автоматически экранирует HTML-спецсимволы при представлении пользовательского контента.

1.1.2. Клиент-серверное взаимодействие

Система исполнена в виде классической трехслойной архитектуры с ясным разделением на клиентский, серверный и уровень данных. Клиентская часть представлена веб-интерфейсом, разработанным с использованием HTML5, CSS3 и JavaScript. В качестве серверной технологии взят Spring Boot 3.4 [5-7], дающий высокую производительность и масштабируемость программы.

Общение между клиентом и сервером осуществляется по протоколу HTTP/HTTPS с использованием идей RESTful API для операций с данными. Все запросы от клиента обрабатываются контроллерами Spring MVC, которые составляют ответы в формате HTML для отображения в браузере пользователя. Для показа данных задействуется шаблонизатор Thymeleaf, который позволяет формировать динамические HTML-страницы, полагаясь на данные, полученные из сервисного слоя.

Серверная часть приложения составлена по принципу инверсии управления (IoC), где компоненты создаются и управляются контейнером Spring. Для доступа к данным используется технология Spring Data JPA с реализацией Hibernate в качестве провайдера, что гарантирует объектно-реляционное отображение (ORM) и абстрагирует разработчика от низкоуровневых действий с базой данных [8]. Программа использует MySQL 8.0 в качестве системы управления базами данных для хранения информации о товарах, категориях, заказах и пользователях [11].

Особое внимание уделено обработке ошибок и исключений. В системе реализован глобальный обработчик исключений

GlobalExceptionHandler, который перехватывает все необработанные ошибки и преобразует их в понятные пользователю сообщения. Для ошибок предусмотрены свои страницы с соответствующими HTTP-статусами. Все операции записи в базу данных выполняются в транзакциях с помощью аннотации @Transactional [8], что гарантирует целостность данных при возникновении ошибок.

Ключевым элементом архитектуры является разделение ответственности между слоями приложения: контроллеры отвечают за обработку HTTP-запросов, сервисы реализуют бизнес-логику, а репозитории обеспечивают доступ к данным.

1.1.3. Бизнес-логика приложения

Бизнес-логика информационно-справочной системы реализована в виде набора сервисов, каждый из которых отвечает за определенную функциональную область. Основные сервисы включают: управление пользователями (UserService), товаров (ProductService), категорий (CategoryService) и заказов (OrderService). Все сервисы разработаны с учетом принципов SOLID и обеспечивают инкапсуляцию бизнес-правил приложения [9].

В сервисе управления товарами реализована логика валидации, включающая проверку уникальности названия товара (без учета регистра), существования категории, корректности цены и количества на складе. При создании или обновлении товара система автоматически проверяет все бизнес-ограничения и генерирует понятные сообщения об ошибках в случае нарушения правил. Для поиска товаров реализован гибкий механизм фильтрации с поддержкой частичного совпадения по названию и бренду, точного совпадения по категории, а также фильтрации по диапазонам цен и количества.

Сервис управления заказами реализует полный жизненный цикл заказа: от создания до доставки или отмены. При создании заказа

система проверяет наличие товара на складе и автоматически рассчитывает общую стоимость (приложение А). Для изменения статуса заказа применены строгие правила доступа: клиенты могут изменять только статус своих заказов и только на "CANCELLED", менеджеры и администраторы имеют полный доступ к управлению статусами. При отмене заказа система автоматически возвращает товар на склад, корректно обновляя количество доступных единиц.

Важной частью бизнес-логики является механизм поиска и фильтрации данных. Для всех основных сущностей (пользователи, товары, заказы, категории) реализованы специализированные методы поиска с поддержкой множественных критериев. Например, при поиске заказов можно одновременно фильтровать по email пользователя, названию товара, статусу заказа и диапазонам дат создания и обновления. Все фильтры являются опциональными — если параметр не указан, он не учитывается при построении запроса.

Для повышения удобства использования системы реализованы механизмы автоматического расчета статистических показателей. Например, на страницах товаров и заказов отображается средняя цена товара, среднее количество товаров на складе, средняя сумма заказа и среднее время доставки. Эти данные рассчитываются динамически на стороне клиента на основе текущего набора отфильтрованных данных (приложение Б), что позволяет менеджерам и администраторам быстро оценивать ключевые показатели эффективности работы магазина.

1.2. Описание интерфейса программы

Пользовательский интерфейс информационно-справочной системы магазина цифровой техники разработан с использованием современных веб-технологий: HTML5, CSS3 и JavaScript. Основная цель интерфейса — предоставление интуитивно понятного и удобного способа взаимодействия с системой для пользователей с различными

ролями (клиент, менеджер, администратор). Все страницы системы имеют единый стиль оформления и последовательную навигацию, что обеспечивает комфортную работу с приложением.

1.2.1. Навигация и аутентификация

Интерфейс системы включает постоянное навигационное меню в верхней части экрана, которое отображается на всех страницах приложения (рисунок 1). Меню содержит ссылки на основные разделы системы: товары, категории, заказы, профиль пользователя и раздел "Об авторе". Доступность элементов меню динамически изменяется в зависимости от роли авторизованного пользователя: обычные клиенты видят только базовые функции, менеджеры получают доступ к управлению заказами и товарами, а администраторы — ко всем разделам системы, включая управление пользователями.

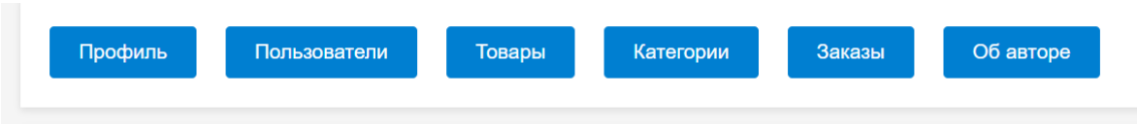


Рисунок 1 – Навигационное меню для пользователя с ролью ADMIN

В правом верхнем углу навигационной панели расположены кнопки аутентификации: для неавторизованных пользователей отображается кнопка входа в систему, для авторизованных — кнопка выхода.

Система реализует два основных сценария аутентификации: вход существующего пользователя и регистрация нового. На странице входа пользователю необходимо указать email и пароль, после чего система проверяет их корректность и предоставляет доступ к соответствующему функционалу в зависимости от роли (рисунок 2). При неверных данных отображается информативное сообщение об ошибке, подсказывающее пользователю о необходимости проверить введенные данные.

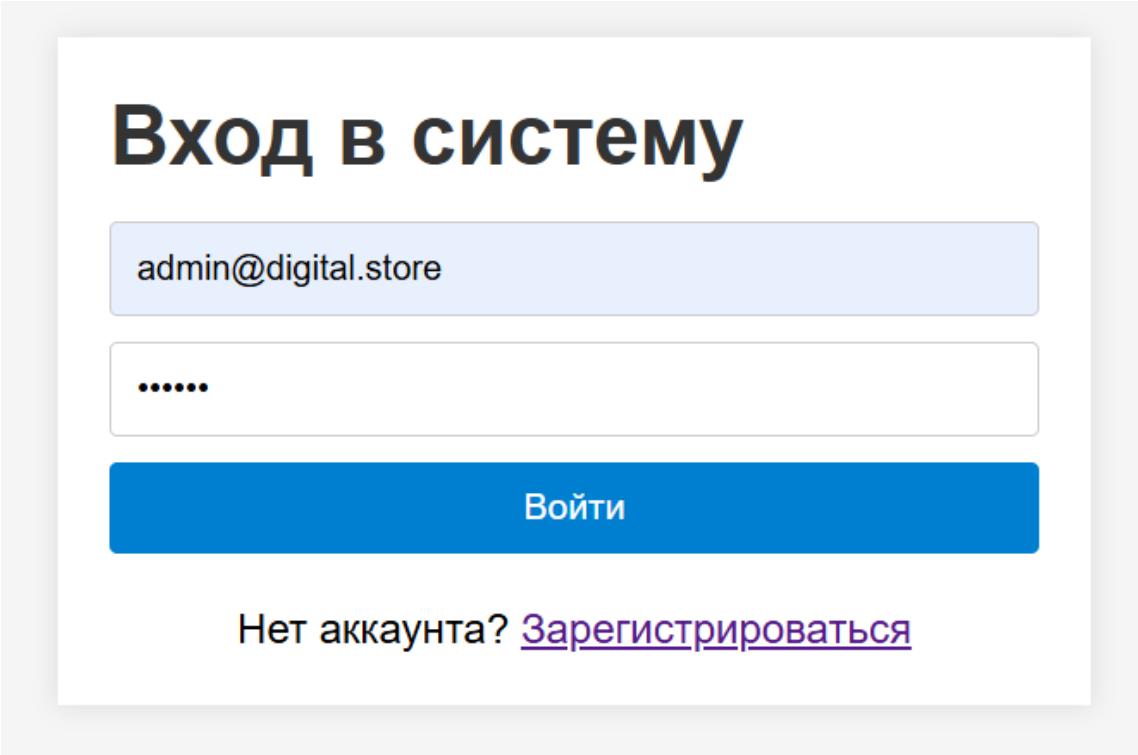


Рисунок 2 – Форма для входа пользователя

Страница регистрации содержит поля для ввода email, имени, фамилии и пароля (минимум 6 символов) (рисунок 3). Система автоматически проверяет корректность email-адреса и уникальность введенного значения в базе данных. После успешной регистрации пользователь перенаправляется на страницу входа с соответствующим уведомлением.

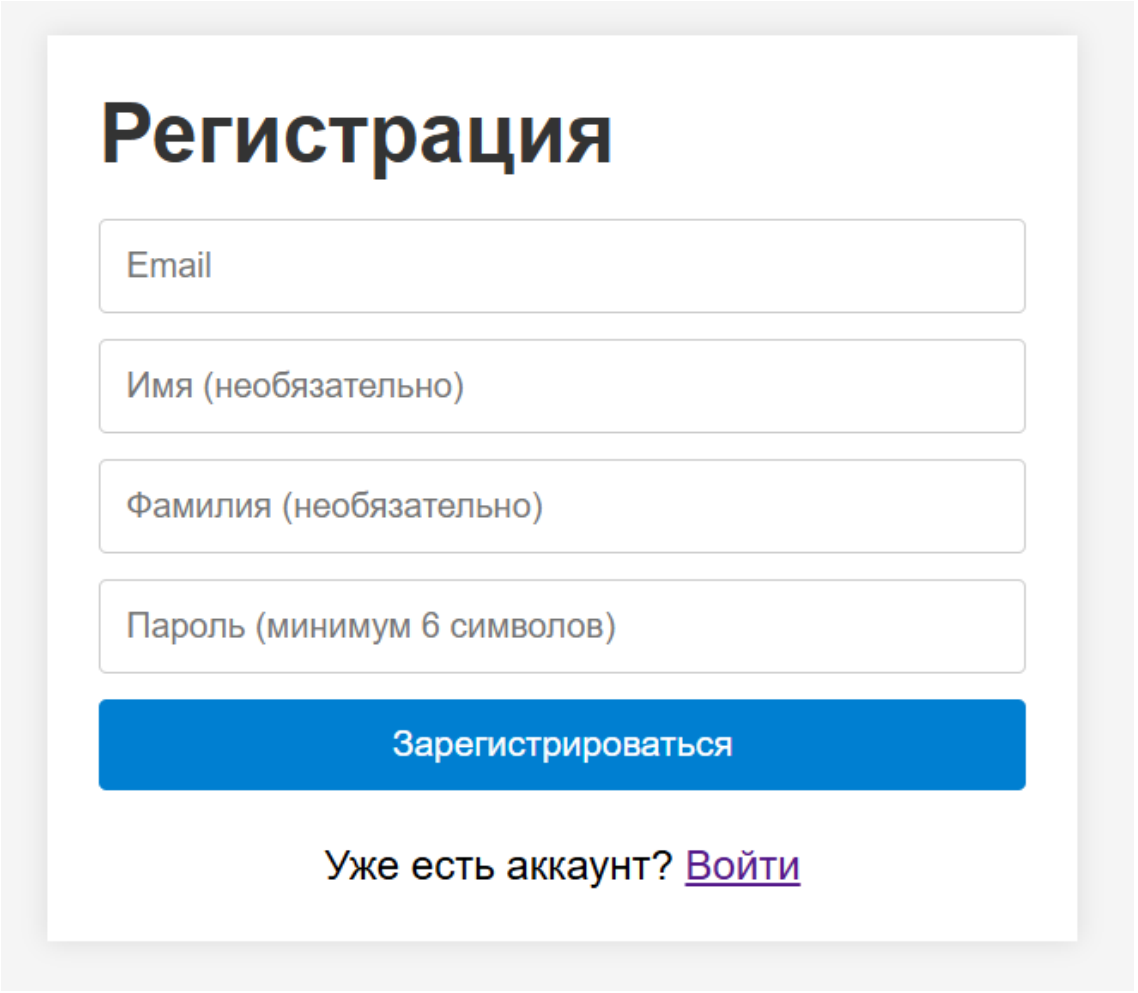


Рисунок 3 – Форма для регистрации пользователя

1.2.2. Страница товаров

Основной раздел системы — каталог товаров, который отображается по умолчанию при входе в приложение. Страница товаров (рисунок 4) содержит фильтры для поиска и отбора данных, статистические показатели и таблицу с информацией о товарах.

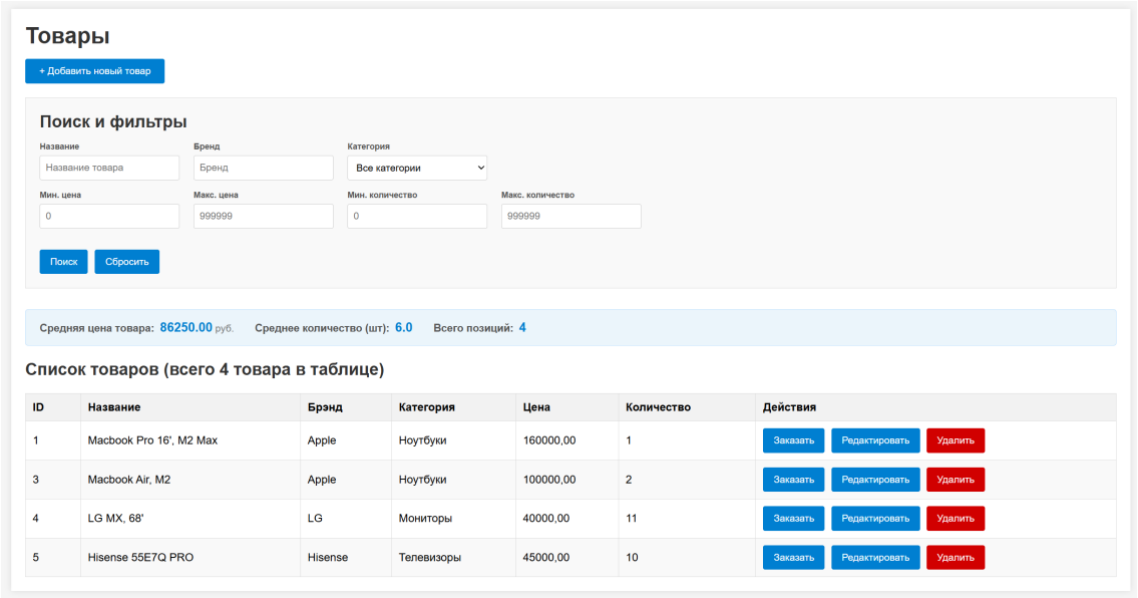


Рисунок 4 – Раздел товаров для пользователя с ролью ADMIN

В верхней части страницы расположен блок фильтров, позволяющий отбирать товары по различным критериям: названию, бренду, категории, ценовому диапазону и количеству на складе. Все фильтры являются опциональными и могут применяться как по отдельности, так и в комбинации. Результаты фильтрации обновляются немедленно после нажатия кнопки "Поиск", а сброс фильтров выполняется кнопкой "Сбросить".

Под блоком фильтров расположена панель статистики, отображающая ключевые показатели по текущему набору товаров: среднюю цену, среднее количество товаров на складе и общее количество позиций в выборке. Эти данные рассчитываются динамически на стороне клиента после загрузки страницы и автоматически обновляются при применении фильтров.

Основная часть страницы содержит таблицу с информацией о товарах. Каждая строка таблицы включает ID товара, название, бренд, категорию, цену и количество на складе. Для авторизованных пользователей в последнем столбце отображаются кнопки действий: для всех пользователей — кнопка "Заказать" (рисунок 5), для менеджеров и администраторов — дополнительные кнопки

"Редактировать" (рисунок 6) и "Удалить". Заголовки столбцов являются интерактивными — при клике на них выполняется сортировка данных по соответствующему полю, что обеспечивает удобную работу с большими объемами информации.

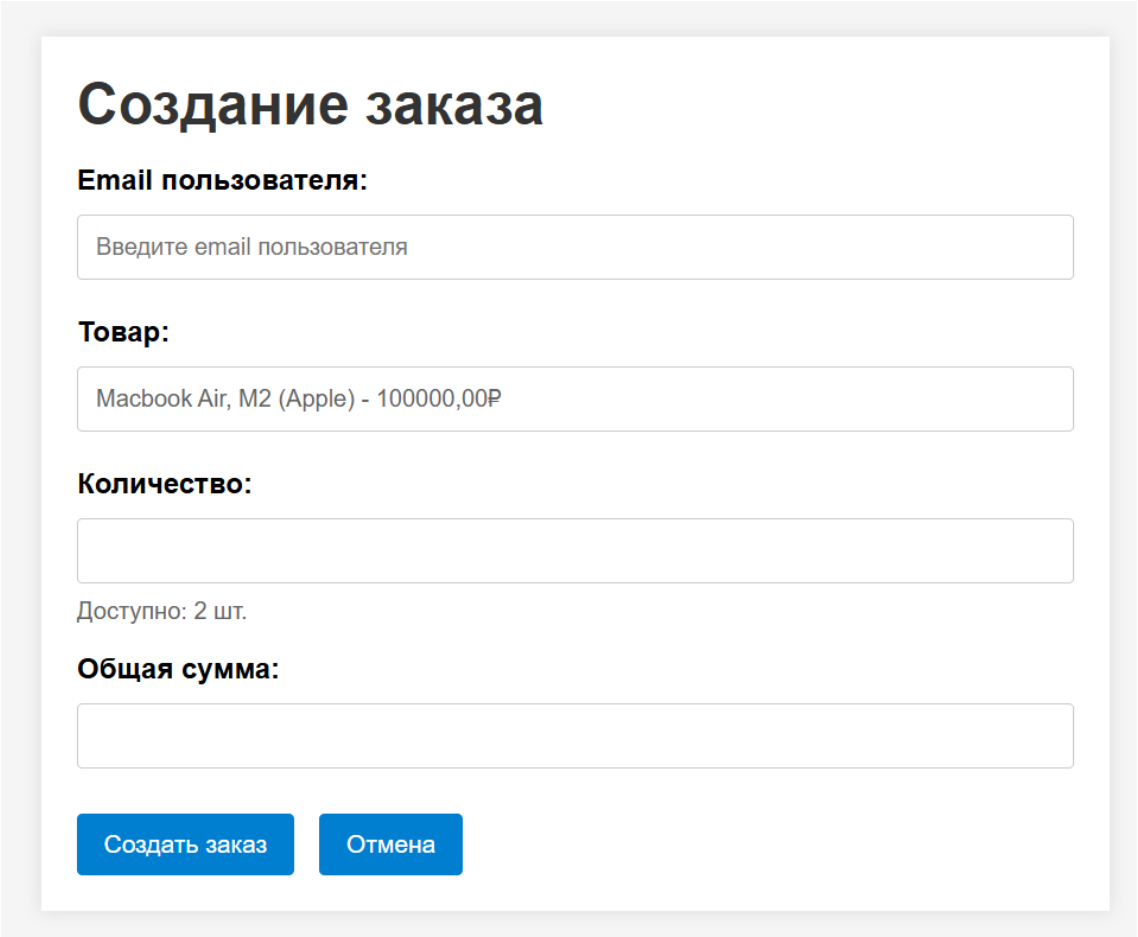


Рисунок 5 – Форма создания заказа

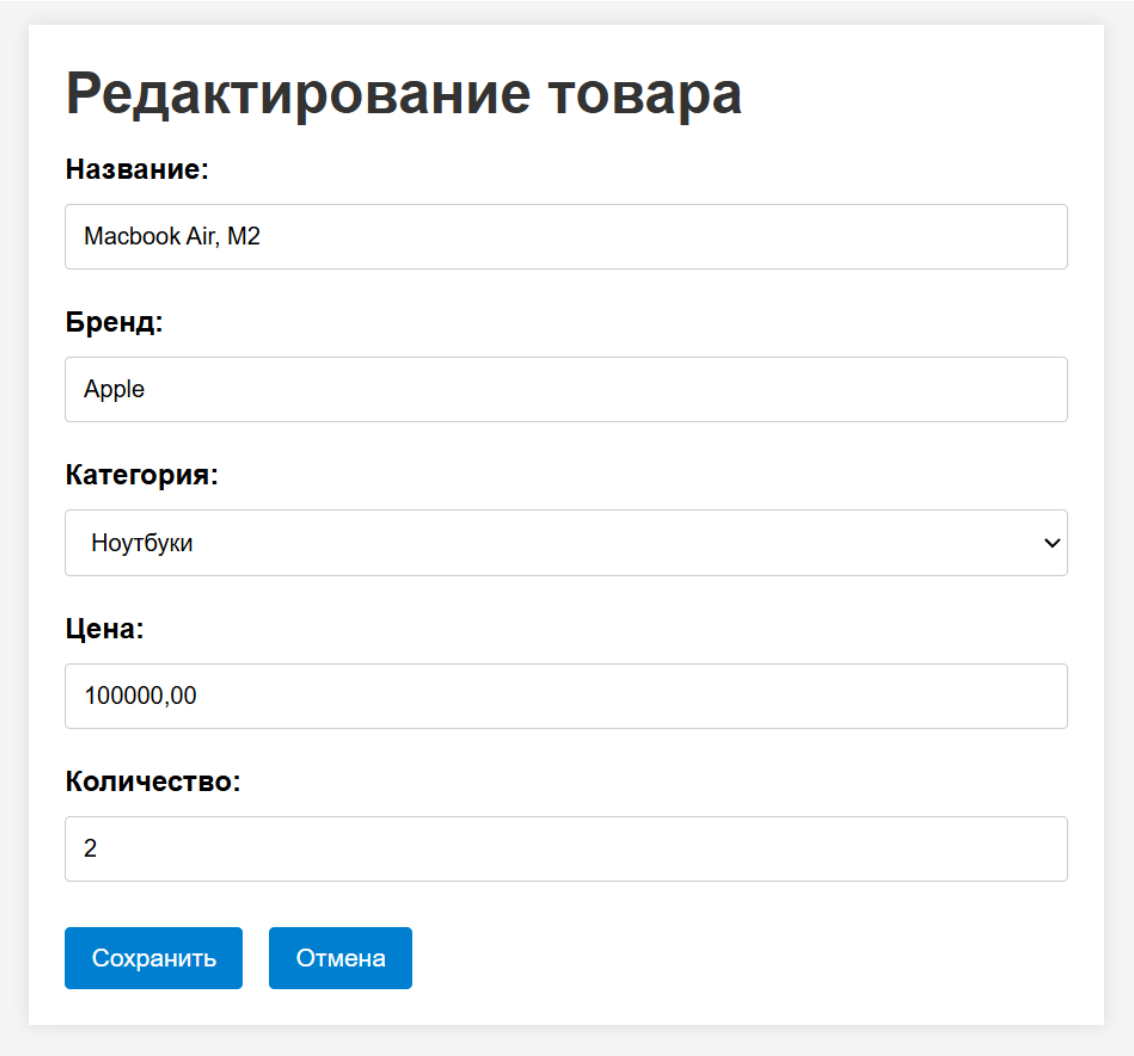


Рисунок 6 – Форма редактирования товара

1.2.3. Страница заказов

Раздел заказов (рисунок 7) предоставляет пользователям возможность просматривать существующие заказы, создавать новые и изменять статусы существующих. Для клиентов отображаются только их собственные заказы, в то время как менеджеры и администраторы видят все заказы в системе.

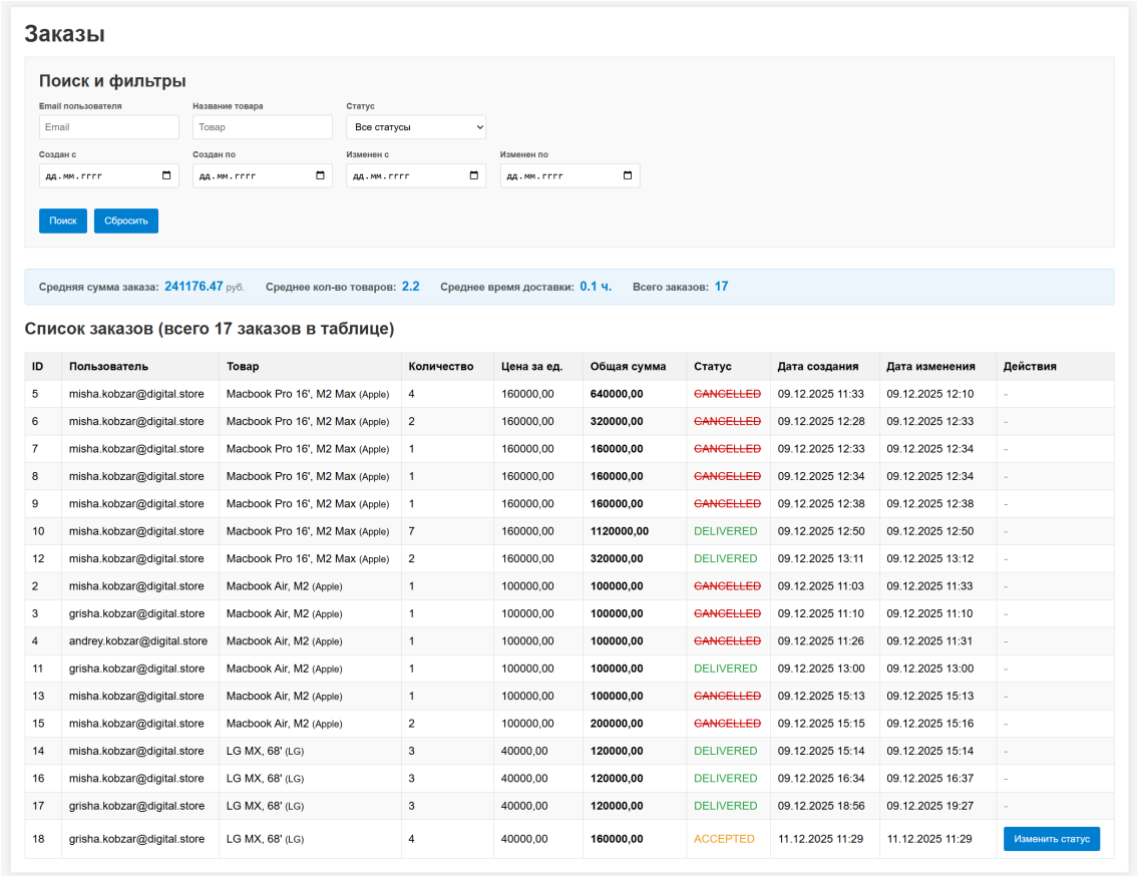


Рисунок 7 – Страница заказов для пользователей с ролями ADMIN или MANAGER

Страница содержит расширенные фильтры для поиска заказов по различным параметрам: email пользователя, названию товара, статусу заказа, датам создания и обновления. Интерфейс фильтров адаптирован под возможности разных ролей: клиенты могут фильтровать только свои заказы, а менеджеры — все заказы в системе.

Аналогично странице товаров, здесь присутствует статистический блок с динамически рассчитываемыми показателями: средняя сумма заказа, среднее количество товаров в заказе, среднее время доставки и общее количество заказов. Эти данные помогают администраторам и менеджерам оперативно оценивать эффективность работы магазина.

Таблица заказов содержит подробную информацию о каждом заказе: ID, пользователя, товар, количество, цену за единицу, общую сумму, текущий статус и даты создания и обновления. Для отмененных

заказов название товара отображается зачеркнутым текстом красного цвета, для доставленных — зеленым цветом с соответствующим статусом, что обеспечивает визуальное восприятие информации. Клиенты могут отменять только свои заказы в статусе NEW, в то время как менеджеры и администраторы имеют возможность изменять статус заказов на всех этапах жизненного цикла.

1.2.4. Управление категориями

Раздел управления категориями товаров предназначен для менеджеров и администраторов системы. Страница категорий (рисунок 8) содержит поиск по названию категории и таблицу со списком всех категорий. Для каждой категории отображается ID, название и описание, а также кнопки редактирования и удаления. При удалении категории система запрашивает подтверждение действия, предотвращая случайное удаление данных.

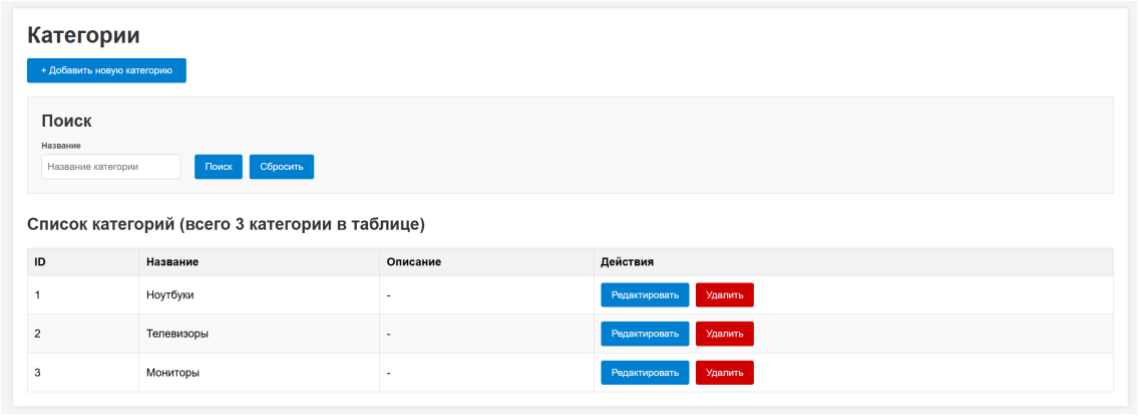


Рисунок 8 – Страница категорий товаров для пользователей с ролями ADMIN или MANAGER

Форма создания и редактирования категории (рисунок 9) содержит поля для ввода названия (обязательное поле) и описания (необязательное). Интерфейс формы унифицирован для обоих операций (создание и редактирование), с изменением только заголовка и адреса отправки данных. При возникновении ошибок (например, при попытке создать категорию с уже существующим названием) система

отображает понятное сообщение об ошибке и сохраняет введенные пользователем данные.

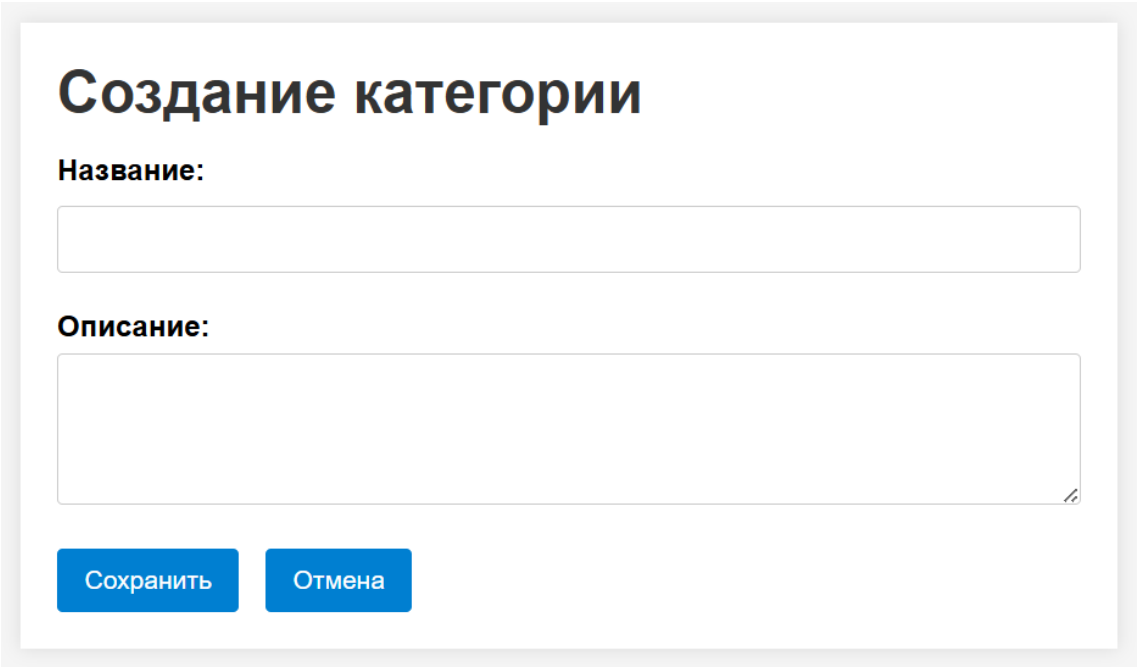


Рисунок 9 – Форма создания категории

1.2.5. Страница профиля и дополнительные разделы

Страница профиля пользователя (рисунок 10) предоставляет возможность просмотра и изменения личных данных: email, имени, фамилии и пароля. При изменении пароля система проверяет его сложность (минимум 6 символов). После обновления профиля отображается уведомление об успешном сохранении данных.

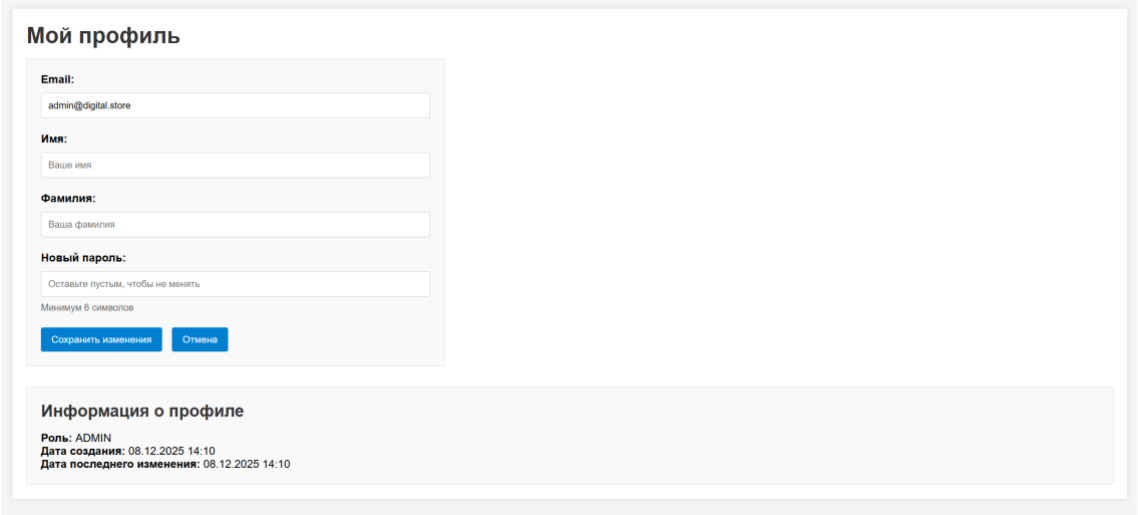


Рисунок 10 – Страница профиля пользователя

Раздел "Об авторе" (рисунок 11) содержит информацию о разработчике системы: ФИО, учебное заведение, контактные данные, используемые технологии и сроки выполнения проекта. Этот раздел доступен всем пользователям системы независимо от их роли.

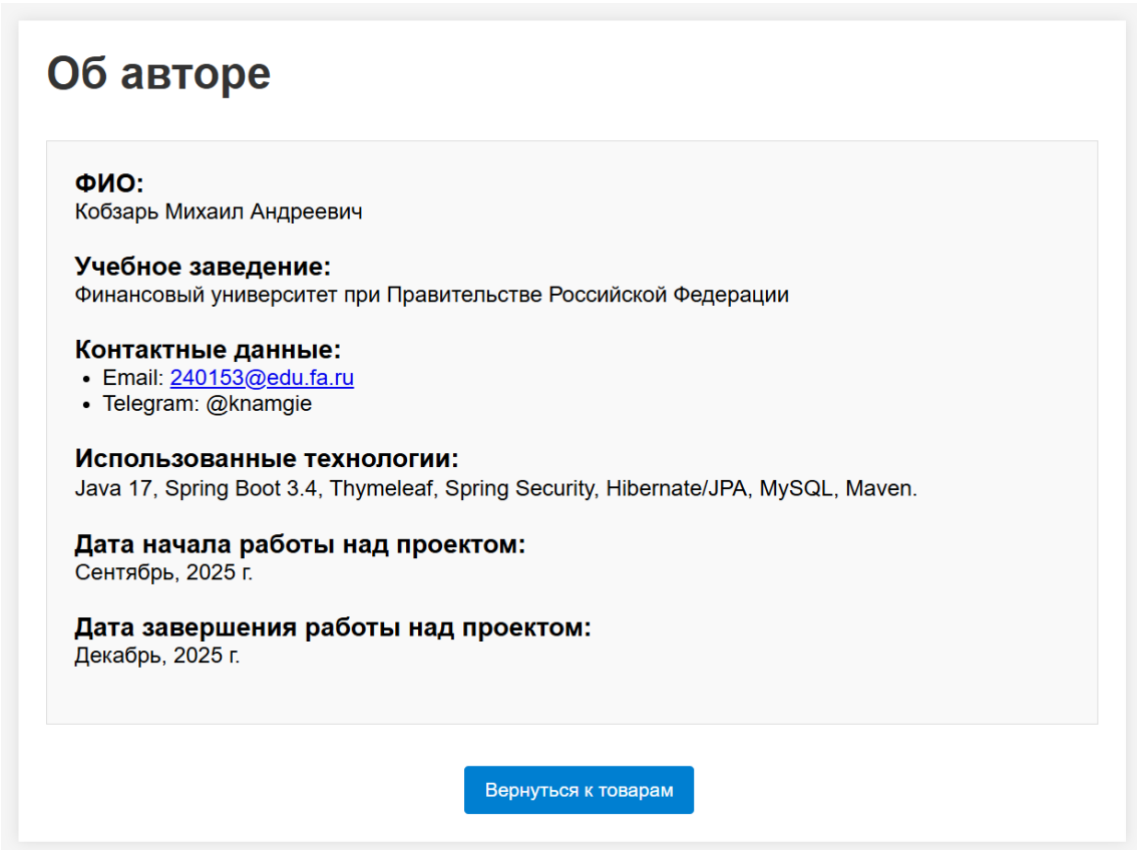


Рисунок 11 – Страница «Об авторе»

Для обработки ошибок в системе реализованы специальные страницы (рисунок 12), отображающие понятные сообщения с указанием кода ошибки и технических деталей. Все страницы ошибок содержат кнопки навигации для возврата на главную страницу или предыдущую страницу, а также возможность входа в систему для неавторизованных пользователей.

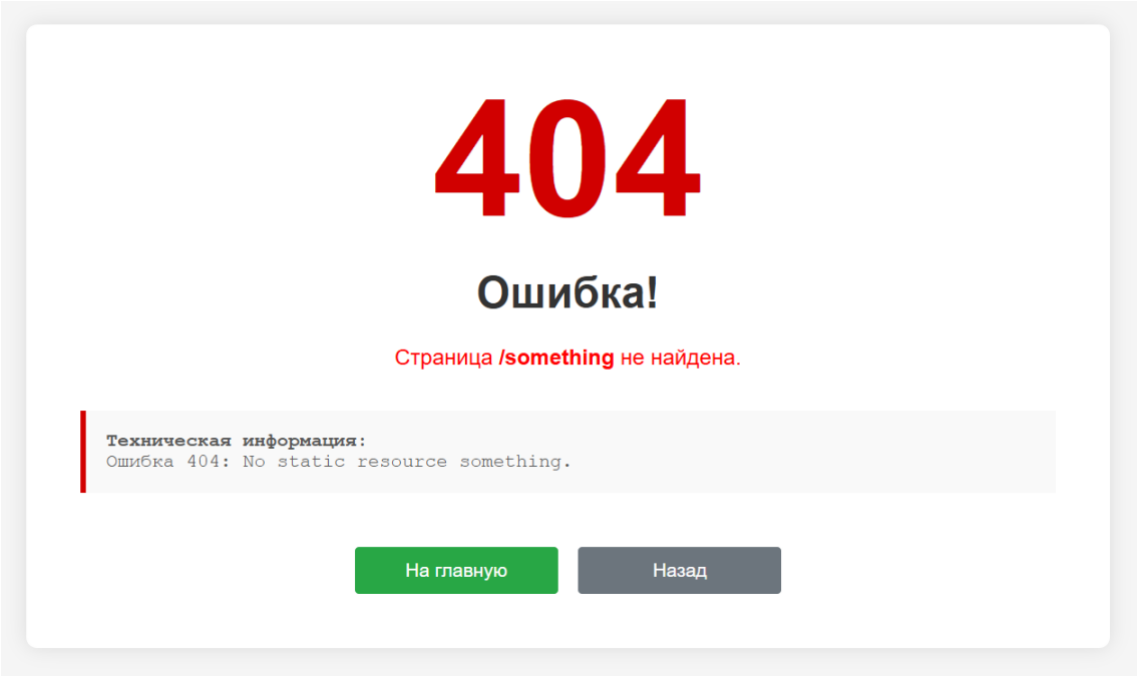


Рисунок 12 – Пример страницы ошибки

Таким образом, интерфейс информационно-справочной системы магазина цифровой техники обеспечивает удобное взаимодействие с приложением для пользователей всех ролей, предоставляя необходимый функционал с интуитивно понятным оформлением и эффективной навигацией.

1.3. Архитектура приложения

1.3.1. Зависимости проекта

Проект реализован с использованием фреймворка Spring Boot 3.4.0 и системы сборки Maven. Зависимости проекта определены в файле pom.xml, который представляет собой основной конфигурационный файл сборки приложения.

В качестве основных зависимостей используются spring-boot-starter-web для создания веб-приложения, spring-boot-starter-data-jpa для работы с базой данных через JPA [5-8], spring-boot-starter-security для организации системы безопасности [12], spring-boot-starter-thymeleaf для генерации HTML-страниц на стороне сервера [10]. Для работы с базой данных MySQL используется драйвер mysql-connector-java

версии 8.0.33. [11]. Для упрощения разработки применяется библиотека Lombok [13], которая автоматически генерирует геттеры, сеттеры, конструкторы и другие шаблонные методы, что значительно сокращает объем кода. Для валидации входных данных используется spring-boot-starter-validation [5-7], обеспечивающий проверку корректности данных на уровне DTO-объектов. В процессе разработки применялись инструменты spring-boot-devtools для ускорения цикла разработки и отладки. Для генерации документации используется плагин maven-javadoc-plugin, что позволяет автоматически создавать JavaDoc на основе комментариев в исходном коде.

1.3.2. Структура приложения

Архитектура приложения построена на основе принципов трехслойной архитектуры (presentation layer, business logic layer, data access layer) с четким разделением ответственности между компонентами (рисунок 13). Слой представления (presentation layer) реализован с использованием Spring MVC, где контроллеры обрабатывают HTTP-запросы, преобразуют данные и возвращают результаты в виде HTML-страниц, генерируемых с помощью шаблонизатора Thymeleaf. Бизнес-логика (business logic layer) инкапсулирована в сервисных классах, которые реализуют основные операции с данными и применяют бизнес-правила приложения. Слой доступа к данным (data access layer) представлен репозиториями, работающими с базой данных через Spring Data JPA и Hibernate в качестве ORM-провайдера. Для организации безопасного взаимодействия с приложением используется Spring Security с реализацией ролевой модели доступа (RBAC), где определены три основные роли: CLIENT, MANAGER и ADMIN, каждая из которых имеет свои права доступа к функциональности системы.

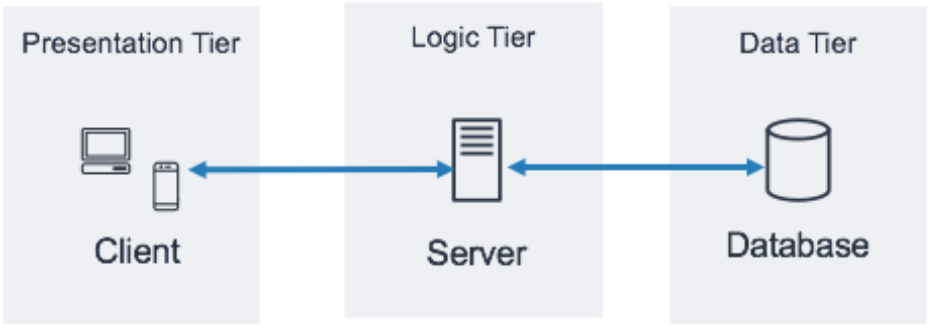


Рисунок 13 – Диаграмма трехуровневой архетиктуры

Клиентская часть приложения реализована с использованием HTML5, CSS3 и JavaScript. Все страницы приложения генерируются на сервере с помощью шаблонизатора Thymeleaf и передаются клиенту в виде готовых HTML-документов. Для улучшения пользовательского опыта и обеспечения интерактивности используются JavaScript-скрипты, обрабатывающие события на стороне клиента (сортировка таблиц, вычисление статистических данных, валидация форм). Стилизация интерфейса выполнена с использованием каскадных таблиц стилей (CSS).

Серверная часть приложения написана на Java 17 с использованием Spring Boot 3.4.0. Все контроллеры приложения обрабатывают HTTP-запросы и возвращают соответствующие представления или перенаправления. Бизнес-логика реализована в сервисных классах, которые инкапсулируют правила работы с данными и применяют валидацию перед сохранением в базу данных. Для работы с базой данных используется паттерн Repository, реализованный через Spring Data JPA, что обеспечивает абстракцию от конкретной СУБД и упрощает написание запросов к данным.

1.3.3. База данных и конфигурация

Система использует реляционную базу данных MySQL 8.0 для хранения информации о товарах, категориях, заказах и пользователях.

Настройки подключения к базе данных и параметры работы JPA определены в файле application.properties (приложение В). В конфигурации указаны URL-адрес базы данных, учетные данные для подключения, диалект Hibernate для работы с MySQL [8, 11], а также параметры автоматического создания/обновления схемы базы данных (ddl-auto=update). Для обеспечения безопасности хранения паролей пользователей применяется алгоритм хеширования BCrypt через BCryptPasswordEncoder [12], что гарантирует защиту паролей даже в случае несанкционированного доступа к базе данных.

Конфигурация безопасности приложения реализована в классе SecurityConfig (приложение Г), где определены правила доступа к различным ресурсам системы в зависимости от роли пользователя. В настройках безопасности определены публичные ресурсы (доступные без аутентификации), ресурсы, доступные только аутентифицированным пользователям, и ресурсы, требующие специальных прав (MANAGER или ADMIN). Для защиты от распространенных веб-атак внедрены механизмы защиты от CSRF-атак, SQL-инъекций (посредством использования параметризованных запросов в Hibernate) и XSS-уязвимостей (посредством экранирования данных в шаблонизаторе Thymeleaf).

Таким образом, архитектура приложения обеспечивает модульность, масштабируемость и безопасность, а разделение на слои упрощает поддержку кода и позволяет гибко изменять отдельные компоненты системы без влияния на другие части приложения.

2. Структура классов и их назначение в рамках проекта

2.1. Сервер

Серверная часть информационно-справочной системы магазина цифровой техники реализована на основе фреймворка Spring Boot 3.4 с использованием Java 17. Архитектура серверного приложения построена по принципу многослойной организации кода с четким разделением ответственности между компонентами. Серверная часть состоит из семи основных пакетов, каждый из которых отвечает за определенную функциональную область системы.

Пакет ru.fa.kobzar.mikhail.digitalstore.config содержит классы конфигурации приложения. Ключевым классом данного пакета является SecurityConfig, который отвечает за настройку безопасности приложения с использованием Spring Security [12]. В этом классе определены правила аутентификации и авторизации, настроены шаблоны доступа для различных ролей пользователей (CLIENT, MANAGER, ADMIN), а также реализованы механизмы шифрования паролей с помощью BCryptPasswordEncoder. Класс SecurityConfig обеспечивает Role-Based Access Control (RBAC) с тремя уровнями доступа, что позволяет гибко управлять правами пользователей в системе.

Пакет ru.fa.kobzar.mikhail.digitalstore.controller реализует слой представления (presentation layer) по архитектуре MVC. В нем содержится восемь контроллеров, каждый из которых отвечает за обработку HTTP-запросов определенной функциональной области:

- CategoryController — управление категориями товаров,
- CustomErrorController — обработка ошибок,
- HomeController — базовые маршруты приложения,
- LoginController — страница аутентификации,

– OrderController — управление заказами,

– ProductController — управление товарами,

– RegisterController — регистрация пользователей,

– UserController — управление учетными записями.

Контроллеры реализуют обработку GET и POST запросов, валидацию входных данных, а также подготовку модели для отображения в представлениях. Особое внимание уделено безопасности: каждый контроллер содержит проверки прав доступа через аннотации Spring Security, такие как @PreAuthorize("hasRole('ADMIN')").

Пакет ru.fa.kobzar.mikhail.digitalstore.entity реализует доменную модель приложения с использованием аннотаций JPA и Hibernate [8]. Здесь определены четыре основные сущности: Category, Order, Product и User. Каждая сущность помечена аннотацией @Entity и содержит поля с соответствующими аннотациями для маппинга в реляционную базу данных. Особое внимание уделено оптимизации связей между сущностями: для отношений один-ко-многим (например, между категорией и товарами) используются аннотации @ManyToOne и @OneToMany. Класс User дополнительно реализует интерфейс UserDetails из Spring Security, что позволяет интегрировать модель пользователей с механизмами аутентификации и авторизации фреймворка.

Пакет ru.fa.kobzar.mikhail.digitalstore.exception содержит глобальный обработчик исключений — класс GlobalExceptionHandler, помеченный аннотацией @ControllerAdvice, который централизованно обрабатывает исключения на уровне приложения. Этот класс перехватывает исключения типа AccessDeniedException и RuntimeException, преобразуя их в понятные пользователю сообщения с соответствующими HTTP-статусами. Обработчик возвращает

специальное представление "error/error", содержащее код ошибки, понятное сообщение и технические детали для отладки.

Пакет ru.fa.kobzar.mikhail.digitalstore.repository реализует шаблон Repository для доступа к данным. Все репозитории расширяют стандартные интерфейсы Spring Data JPA (JpaRepository, JpaSpecificationExecutor), что позволяет автоматически генерировать базовые CRUD-операции. Особое значение имеют методы поиска с фильтрацией:

- CategoryRepository содержит методы findByNameContaining-IgnoreCase и findByName для поиска категорий;
- OrderRepository включает комплексный метод findByFilters с поддержкой множественных параметров (приложение Д);
- ProductRepository реализует метод findByFilters для поиска товаров по различным критериям;
- UserRepository содержит метод findByFilters для поиска пользователей по email, имени, фамилии и другим атрибутам.

Также многие репозитории включают специализированные запросы, определенные с помощью аннотации @Query и JPQL.

Пакет ru.fa.kobzar.mikhail.digitalstore.service реализует бизнес-логику приложения. Каждый сервисный класс (CategoryService, OrderService, ProductService, UserService) инкапсулирует правила работы с соответствующей сущностью и содержит методы для выполнения бизнес-операций. Сервисы используют аннотацию @Transactional для обеспечения целостности данных при операциях записи. Например, OrderService содержит логику создания заказов с проверкой наличия товара на складе и автоматическим расчетом общей стоимости; UserService реализует регистрацию пользователей с автоматическим определением роли (приложение Е). Все сервисы

проектируются как stateless компоненты, что позволяет эффективно использовать в многопользовательской среде.

Структура серверного приложения соответствует принципам SOLID [9] и обеспечивает высокую степень модульности (рисунок 14). Каждый пакет имеет четко определенную ответственность, а зависимости между компонентами строго регламентированы: контроллеры зависят от сервисов, сервисы — от репозиториев и DTO, репозитории — от сущностей. Такая организация кода позволяет легко вносить изменения в отдельные модули без влияния на остальную систему.

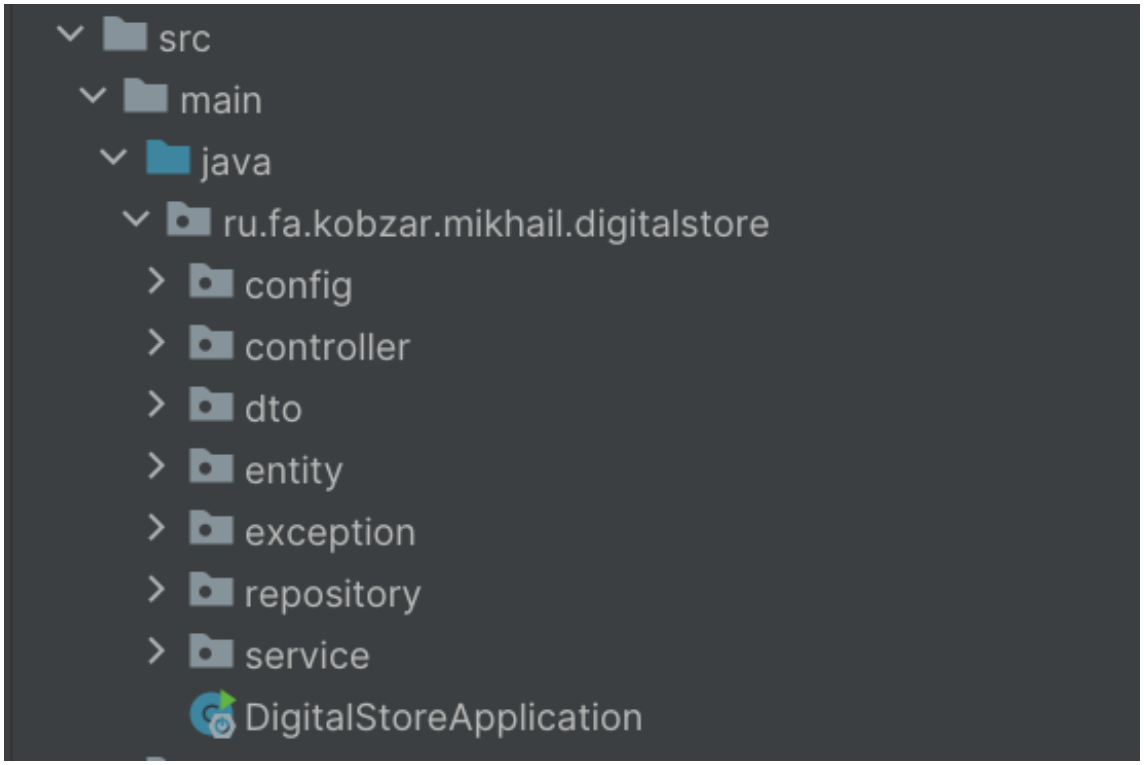


Рисунок 14 – Структура классов

2.2. Клиент

2.2.1. HTML-шаблоны и страничная структура

Клиентская часть приложения использует шаблонизатор Thymeleaf для динамической генерации HTML-страниц на основе данных, полученных от сервера. Все HTML-файлы организованы в

логическую структуру папок, отражающую функциональное назначение страниц (рисунок 15).

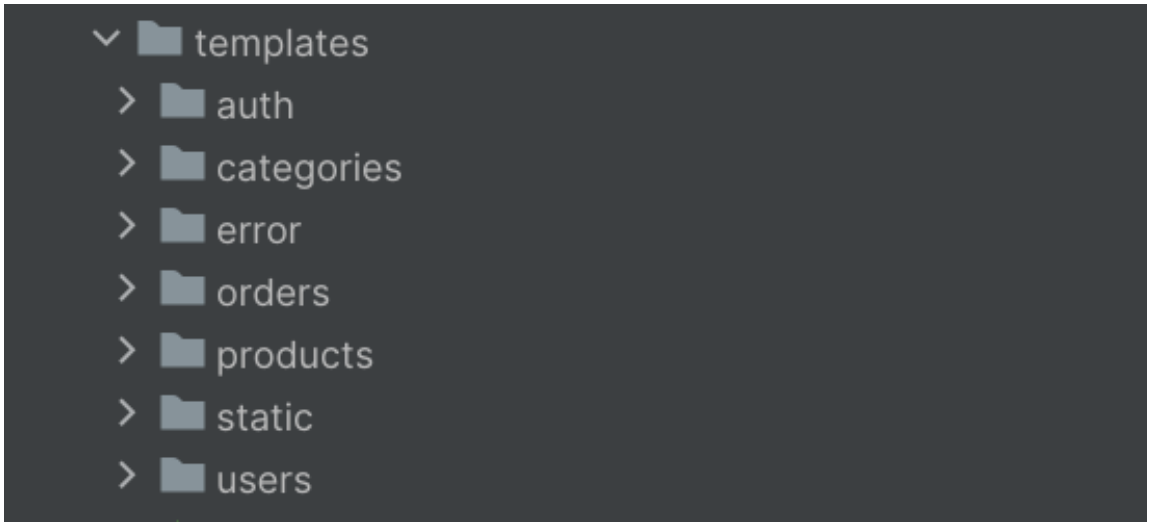


Рисунок 15 – Структура каталога HTML-шаблонов

Каталог auth содержит страницы аутентификации пользователей: login.html и registration.html. Первая страница предоставляет форму для входа в систему с полями email и пароля, вторая — форму регистрации с полями email, имени, фамилии и пароля. Обе страницы содержат валидацию данных на стороне клиента и интеграцию с серверной аутентификацией через Spring Security.

Каталог error включает специализированные страницы для обработки различных типов ошибок, обеспечивающие дружелюбное отображение ошибок вместо стандартных сообщений сервера.

Каталог orders содержит страницы для управления заказами: orders.html для просмотра списка заказов с возможностью фильтрации и сортировки, order-form.html для создания и редактирования заказов, order-status-form.html для изменения статуса заказа. Эти страницы реализуют сложную бизнес-логику взаимодействия с заказами и включают JavaScript-валидацию данных перед отправкой на сервер.

Каталог products включает страницы управления товарами: products.html для отображения каталога товаров с расширенной фильтрацией, product-form.html для создания и редактирования товаров.

Страница каталога товаров содержит динамический подсчет статистических показателей (средней цены, среднего количества товаров на складе) с использованием JavaScript.

Страницы управления категориями (categories.html и category-form.html) расположены в каталоге categories и обеспечивают функционал просмотра, создания и редактирования категорий товаров с валидацией уникальности названия.

Для управления пользователями предназначены страницы из каталога users: users.html для просмотра списка пользователей с возможностью поиска, user-form.html для создания и редактирования пользователей. Страница профиля (profile.html) позволяет аутентифицированным пользователям изменять свои данные.

Страница author.html из каталога static содержит информацию о разработчике системы и доступна из главного меню независимо от роли пользователя.

2.2.2. Статические ресурсы и скрипты

Статические ресурсы приложения включают файлы CSS и JavaScript, организованные в соответствии с их функциональным назначением (рисунок 16).

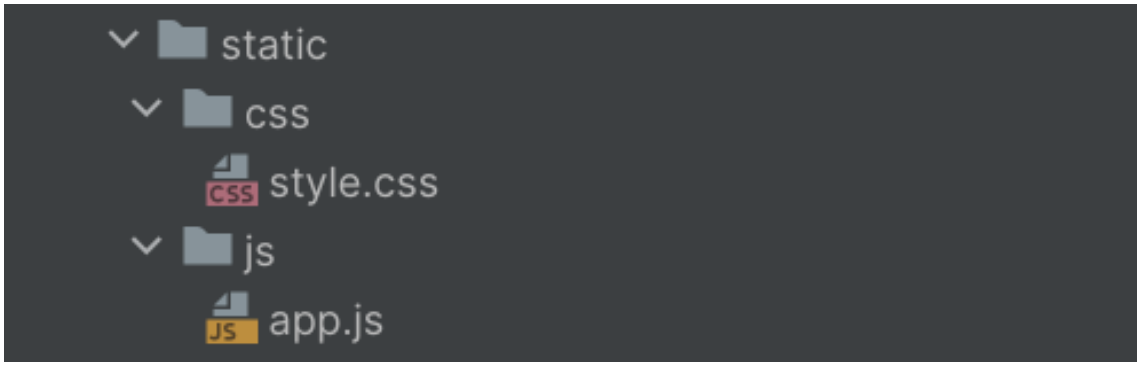


Рисунок 16 – Структура каталога статических ресурсов и скриптов

Файлы CSS реализуют адаптивную верстку интерфейса с использованием медиа-запросов для корректного отображения на устройствах с разным разрешением экрана. Стили обеспечивают

единообразное оформление элементов управления, таблиц и форм по всему приложению, следуя принципам современного веб-дизайна. Файл style.css содержит правила оформления навигационного меню, таблиц данных, форм ввода, кнопок действий и служебных элементов интерфейса.

JavaScript-файлы обеспечивают интерактивность пользовательского интерфейса и реализуют функционал, недоступный на чистом HTML. Файл app.js является основным скриптом приложения и содержит функции валидации форм, динамического расчета статистических показателей, обработки событий сортировки таблиц и управления состоянием интерфейса.

Все JavaScript-функции написаны с использованием современных подходов к веб-разработке: события обрабатываются через addEventListener, для работы с DOM используется метод querySelector, а для обеспечения совместимости с разными браузерами применяются полифилы. Особое внимание уделено обработке ошибок и валидации пользовательского ввода для предотвращения некорректных данных.

Страницы интерфейса используют логику, реализованную в файле app.js, для обеспечения интерактивности, включая динамический подсчет статистических данных о товарах и заказах, валидацию форм на стороне клиента и улучшение пользовательского опыта через немедленную обратную связь.

Таким образом, клиентская часть приложения представляет собой полнофункциональный веб-интерфейс, обеспечивающий удобное взаимодействие пользователей с системой управления магазином цифровой техники. Использование современных веб-технологий позволяет создать адаптивный, интуитивно понятный и

высокопроизводительный интерфейс, полностью интегрированный с серверной частью приложения.

Заключение

В ходе выполнения курсовой работы была разработана полнофункциональная информационно-справочная система магазина цифровой техники, полностью соответствующая требованиям задания и современным стандартам разработки веб-приложений. Система реализована с использованием стека технологий Java/Spring Boot с применением трехуровневой архитектуры, что обеспечивает ее модульность, масштабируемость и удобство поддержки.

Поставленная цель проекта — создание комплексной системы управления магазином цифровой техники — была успешно достигнута. В процессе разработки решены все определенные во введении задачи: реализован клиент-серверный механизм взаимодействия с использованием RESTful API, разработан интуитивно понятный веб-интерфейс с возможностью адаптации под различные роли пользователей, применена модель MVC для разделения ответственности компонентов системы. Особое внимание уделено безопасности приложения: реализована ролевая модель доступа с тремя уровнями привилегий (CLIENT, MANAGER, ADMIN), применено шифрование паролей с использованием алгоритма BCrypt, предусмотрена защита от распространенных веб-уязвимостей.

Разработанная система обладает широким функционалом: позволяет вести учет товаров с возможностью фильтрации и поиска по различным критериям, управлять категориями товаров, обрабатывать заказы на всех этапах их жизненного цикла, осуществлять управление пользователями с гибкими правами доступа. Интерфейс системы обеспечивает удобную работу с данными: реализована сортировка таблиц, динамический расчет статистических показателей (средняя цена товара, среднее время доставки заказов), визуальное представление информации о статусах заказов.

Практическая значимость разработанной системы заключается в возможности ее внедрения в реальные магазины цифровой техники для автоматизации бизнес-процессов, повышения эффективности работы персонала и улучшения качества обслуживания клиентов. Система позволяет сократить время обработки заказов, минимизировать ошибки при управлении складскими запасами, обеспечить оперативное получение аналитической информации для принятия управленческих решений.

В процессе выполнения работы были получены ценные навыки проектирования и разработки полнофункциональных веб-приложений с использованием современных фреймворков и технологий, а также практический опыт применения принципов объектно-ориентированного программирования, проектирования баз данных и обеспечения безопасности веб-приложений.

В перспективе систему можно дополнить следующими возможностями: интеграцией с платежными системами для онлайн-оплаты заказов, реализацией механизмов рекомендаций товаров для клиентов на основе их истории покупок, добавлением мобильного приложения для клиентов и персонала магазина, расширением аналитических функций за счет построения прогнозов спроса на товары. Реализация этих улучшений позволит создать полноценную экосистему для автоматизации управления магазином цифровой техники любого масштаба.

Список использованных источников

1. Рынок цифровой техники России: аналитический отчет за 2024 год / Под ред. С. Г. Кузнецова. – Москва: ИД «Экономика и статистика», 2024. – 156 с.
2. Бирюков А.В. Управление запасами в розничной торговле электроникой: современные подходы и методы / А.В. Бирюков, М.Л. Петров // Вестник торговой науки. – 2023. – № 4. – С. 78-92.
3. Васильева Е.Н. Эффективность внедрения информационных систем в розничной торговле: статистический анализ / Е.Н. Васильева, И.К. Соколова // Информационные технологии в экономике и управлении. – 2024. – Т. 12, № 2. – С. 145-160.
4. Громов И.Т. Проблемы цифровизации малого и среднего бизнеса в сфере розничной торговли / И.Т. Громов // Цифровая экономика. – 2023. – № 3. – С. 211-225.
5. Walls C. Spring в действии / C. Walls. – Москва: ДМК Пресс, 2021. – 616 с.
6. Шарма Р., Чопра К. Spring 5 для профессионалов / Р. Шарма, К. Чопра. – Москва: Питер, 2022. – 624 с.
7. Лонг Д. Spring Boot в действии / Д. Лонг. – Москва: ДМК Пресс, 2020. – 432 с.
8. Бауэр К. Java Persistence с Hibernate / К. Бауэр, Г. Кинг. – Санкт-Петербург: Питер, 2022. – 560 с.
9. Хорстманн К.С. Java. Библиотека профессионала. Том 2 / К.С. Хорстманн. – Москва: ДМК Пресс, 2021. – 784 с.
10. Блэк Д. Thymeleaf: современный шаблонизатор для Java-приложений / Д. Блэк // Программирование. – 2023. – № 5. – С. 89-103.

11. MySQL 8.0 Reference Manual [Электронный ресурс]. – Режим доступа: https://dev.mysql.com/doc/refman/8.0/en/ (дата обращения: 15.11.2025).

12. Spring Security Documentation [Электронный ресурс]. – Режим доступа: https://docs.spring.io/spring-security/reference/index.html (дата обращения: 17.11.2025).

13. Project Lombok: Java annotation library [Электронный ресурс]. – Режим доступа: https://projectlombok.org/features/all (дата обращения: 18.11.2025).

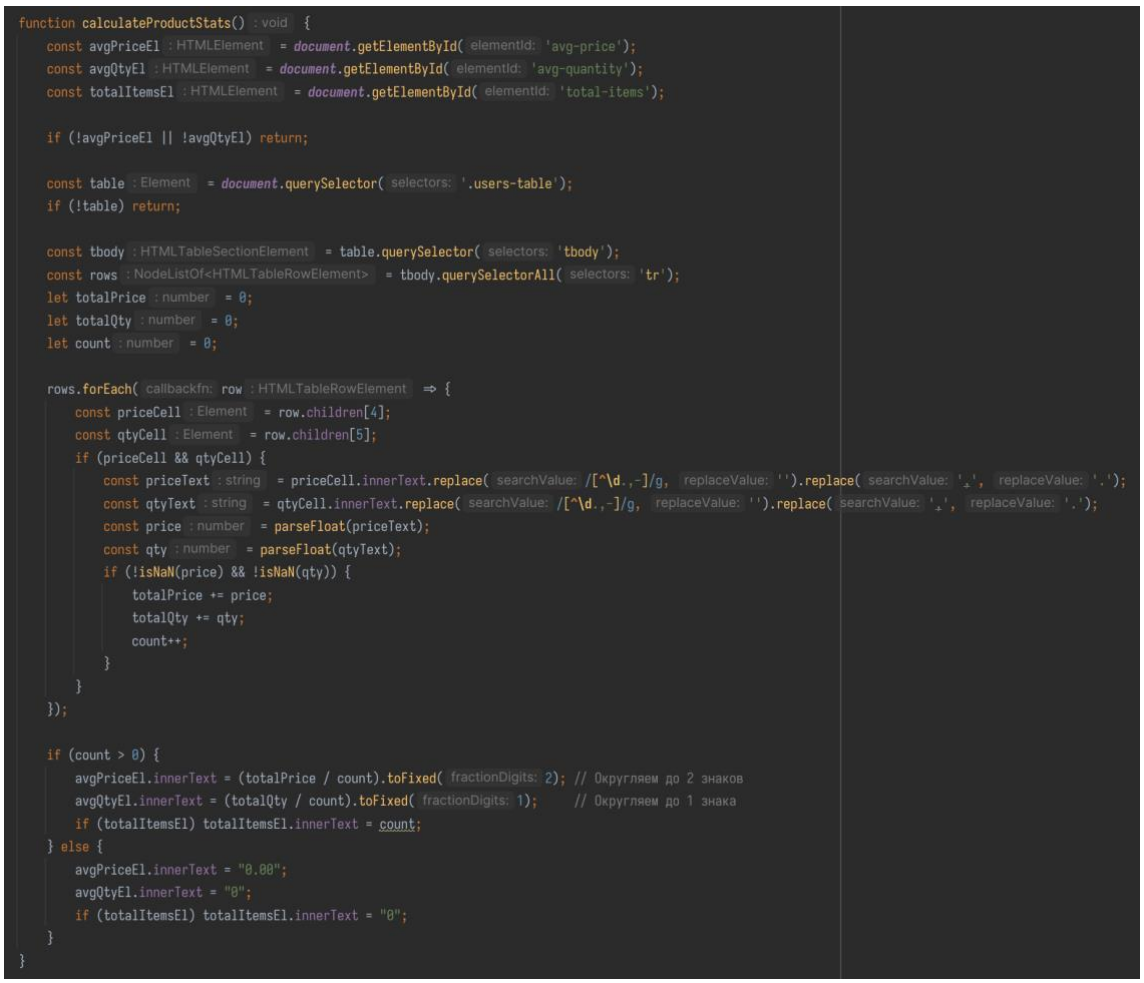
Репозиторий проекта: https://github.com/knamgie/digital-store

Видео-демонстрация проекта: https://youtu.be/Je6j3F5iK8U

Приложения

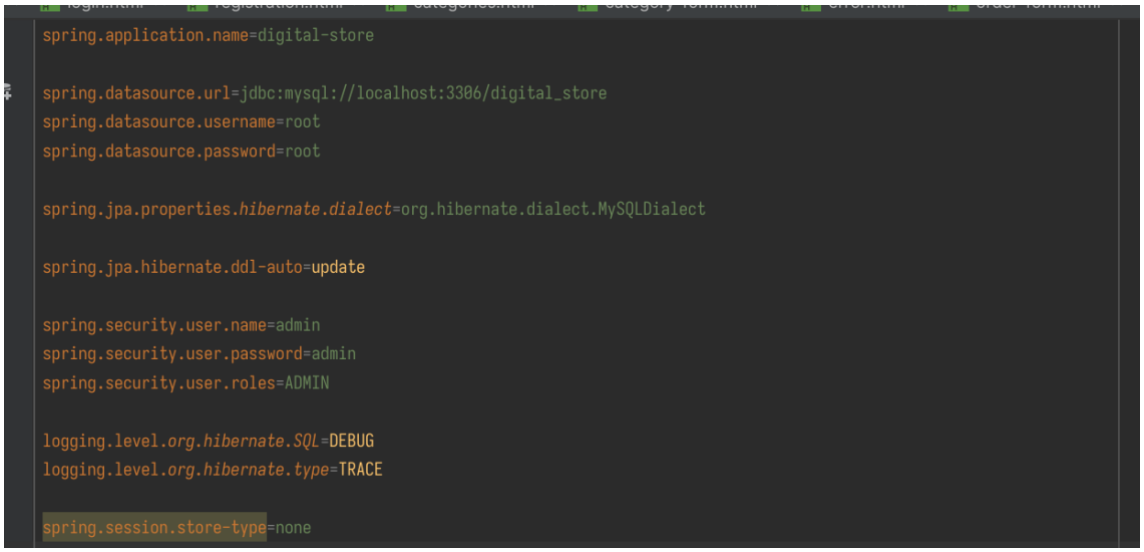
```java
@Transactional
public OrderDto createOrder(OrderDto orderDto) {
    User user = userRepository
        .findById(orderDto.getUserId())
        .orElseThrow(() -> new RuntimeException("Пользователь не найден"));
    Product product = productRepository
        .findById(orderDto.getProductId())
        .orElseThrow(() -> new RuntimeException("Товар не найден"));

    if (product.getQuantity() < orderDto.getQuantity()) {
        throw new RuntimeException("Недостаточно товара в наличии. Доступно: " + product.getQuantity());
    }

    Order order = new Order();
    order.setUser(user);
    order.setProduct(product);
    order.setQuantity(orderDto.getQuantity());
    BigDecimal totalPrice = product.getPrice().multiply(BigDecimal.valueOf(orderDto.getQuantity()));
    order.setTotalPrice(totalPrice);
    order.setStatus(Order.Status.NEW);
    order.setCreatedAt(LocalDateTime.now());
    order.setUpdatedAt(LocalDateTime.now());

    product.setQuantity(product.getQuantity() - orderDto.getQuantity());
    productRepository.save(product);

    Order savedOrder = orderRepository.save(order);
    return convertToDto(savedOrder);
}
```

Приложение А – Метод для создания заказа



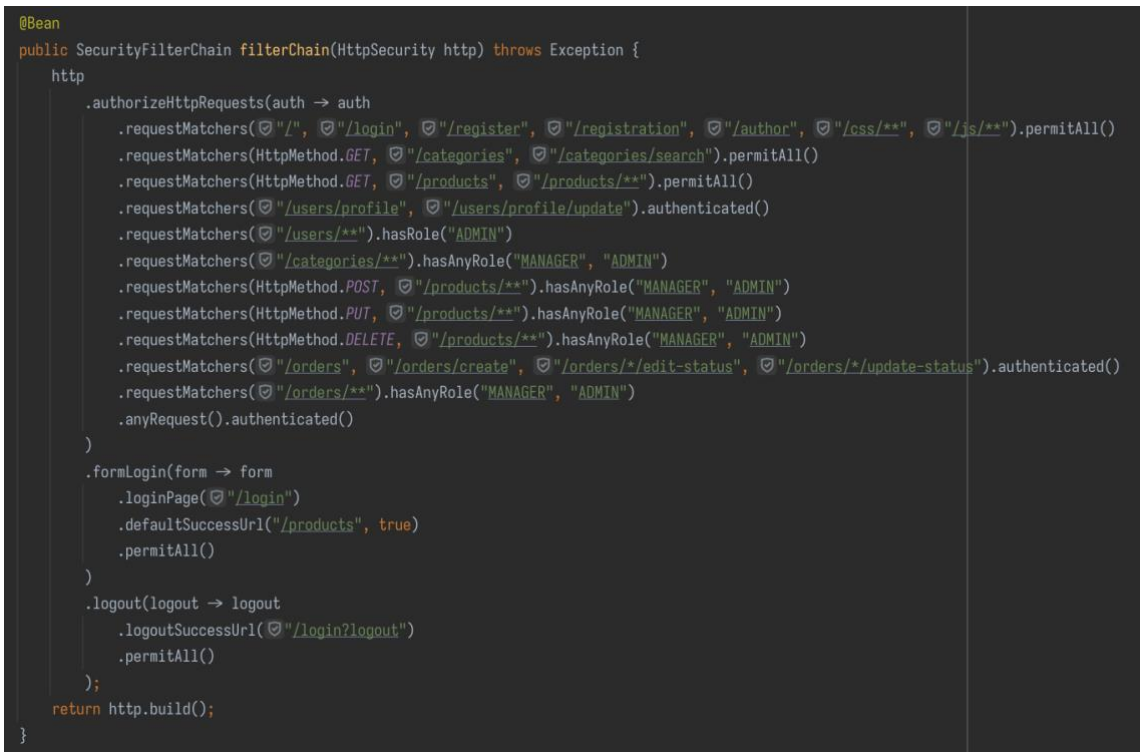
```
function calculateProductStats() : void {
    const avgPriceEl : HTMLElement = document.getElementById( elementId: 'avg-price');
    const avgQtyEl : HTMLElement = document.getElementById( elementId: 'avg-quantity');
    const totalItemsEl : HTMLElement = document.getElementById( elementId: 'total-items');

    if (!avgPriceEl || !avgQtyEl) return;

    const table : Element = document.querySelector( selectors: '.users-table');
    if (!table) return;

    const tbody : HTMLTableSectionElement = table.querySelector( selectors: 'tbody');
    const rows : NodeListOf<HTMLTableRowElement> = tbody.querySelectorAll( selectors: 'tr');
    let totalPrice : number = 0;
    let totalQty : number = 0;
    let count : number = 0;

    rows.forEach( callbackfn: row : HTMLTableRowElement => {
        const priceCell : Element = row.children[4];
        const qtyCell : Element = row.children[5];
        if (priceCell && qtyCell) {
            const priceText : string = priceCell.innerText.replace( searchValue: /[^\d.,-]/g, replaceValue: '').replace( searchValue: ',', replaceValue: '.');
            const qtyText : string = qtyCell.innerText.replace( searchValue: /[^\d.,-]/g, replaceValue: '').replace( searchValue: ',', replaceValue: '.');
            const price : number = parseFloat(priceText);
            const qty : number = parseFloat(qtyText);
            if (!isNaN(price) && !isNaN(qty)) {
                totalPrice += price;
                totalQty += qty;
                count++;
            }
        }
    });

    if (count > 0) {
        avgPriceEl.innerText = (totalPrice / count).toFixed( fractionDigits: 2); // Округляем до 2 знаков
        avgQtyEl.innerText = (totalQty / count).toFixed( fractionDigits: 1);     // Округляем до 1 знака
        if (totalItemsEl) totalItemsEl.innerText = count;
    } else {
        avgPriceEl.innerText = "0.00";
        avgQtyEl.innerText = "0";
        if (totalItemsEl) totalItemsEl.innerText = "0";
    }
}
```

Приложение Б – Метод по подсчёту статистических данных о товаре



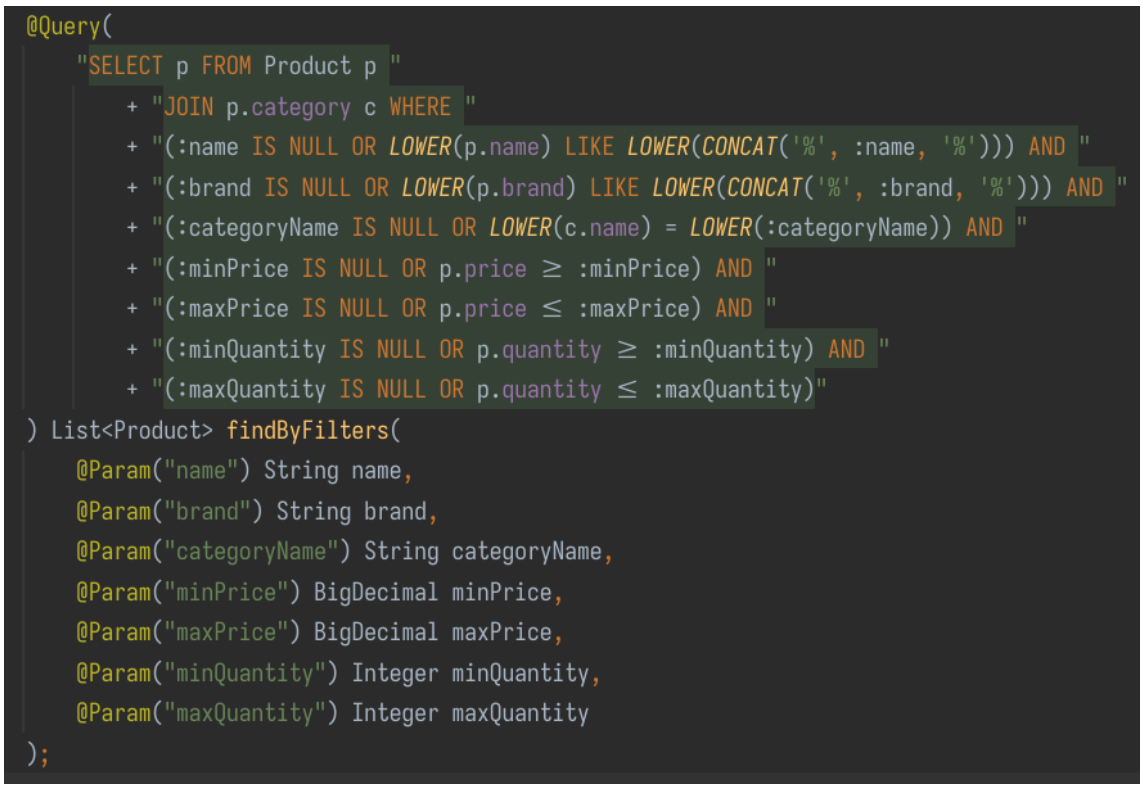
```
spring.application.name=digital-store

spring.datasource.url=jdbc:mysql://localhost:3306/digital_store
spring.datasource.username=root
spring.datasource.password=root

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

spring.jpa.hibernate.ddl-auto=update

spring.security.user.name=admin
spring.security.user.password=admin
spring.security.user.roles=ADMIN

logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type=TRACE

spring.session.store-type=none
```

Приложение В – Файл application.properties



```
@Bean
public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
    http
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/", "/login", "/register", "/registration", "/author", "/css/**", "/js/**").permitAll()
            .requestMatchers(HttpMethod.GET, "/categories", "/categories/search").permitAll()
            .requestMatchers(HttpMethod.GET, "/products", "/products/**").permitAll()
            .requestMatchers("/users/profile", "/users/profile/update").authenticated()
            .requestMatchers("/users/**").hasRole("ADMIN")
            .requestMatchers("/categories/**").hasAnyRole("MANAGER", "ADMIN")
            .requestMatchers(HttpMethod.POST, "/products/**").hasAnyRole("MANAGER", "ADMIN")
            .requestMatchers(HttpMethod.PUT, "/products/**").hasAnyRole("MANAGER", "ADMIN")
            .requestMatchers(HttpMethod.DELETE, "/products/**").hasAnyRole("MANAGER", "ADMIN")
            .requestMatchers("/orders", "/orders/create", "/orders/*/edit-status", "/orders/*/update-status").authenticated()
            .requestMatchers("/orders/**").hasAnyRole("MANAGER", "ADMIN")
            .anyRequest().authenticated()
        )
        .formLogin(form -> form
            .loginPage("/login")
            .defaultSuccessUrl("/products", true)
            .permitAll()
        )
        .logout(logout -> logout
            .logoutSuccessUrl("/login?logout")
            .permitAll()
        );
    return http.build();
}
```

Приложение Г – Конфигурация безопасности приложения



```
@Query(
    "SELECT p FROM Product p "
        + "JOIN p.category c WHERE "
        + "(:name IS NULL OR LOWER(p.name) LIKE LOWER(CONCAT('%', :name, '%'))) AND "
        + "(:brand IS NULL OR LOWER(p.brand) LIKE LOWER(CONCAT('%', :brand, '%'))) AND "
        + "(:categoryName IS NULL OR LOWER(c.name) = LOWER(:categoryName)) AND "
        + "(:minPrice IS NULL OR p.price ≥ :minPrice) AND "
        + "(:maxPrice IS NULL OR p.price ≤ :maxPrice) AND "
        + "(:minQuantity IS NULL OR p.quantity ≥ :minQuantity) AND "
        + "(:maxQuantity IS NULL OR p.quantity ≤ :maxQuantity)"
) List<Product> findByFilters(
    @Param("name") String name,
    @Param("brand") String brand,
    @Param("categoryName") String categoryName,
    @Param("minPrice") BigDecimal minPrice,
    @Param("maxPrice") BigDecimal maxPrice,
    @Param("minQuantity") Integer minQuantity,
    @Param("maxQuantity") Integer maxQuantity
);
```

Приложение Д – Пример реализации репозитория с комплексным поиском (ProductRepository)

```java
@Transactional
public User registerUser(UserDto userDto) {
    if (userRepository.existsByEmail(userDto.getEmail())) {
        throw new RuntimeException("Пользователь с таким Email уже зарегистрирован");
    }

    User user = new User();
    user.setEmail(userDto.getEmail());
    user.setFirstName(userDto.getFirstName());
    user.setLastName(userDto.getLastName());

    if ("admin@digital.store".equalsIgnoreCase(userDto.getEmail())) {
        user.setRole(User.Role.ADMIN);
    } else {
        user.setRole(User.Role.CLIENT);
    }

    user.setPassword(passwordEncoder.encode(userDto.getPassword()));
    user.setCreatedAt(LocalDateTime.now());
    user.setUpdatedAt(LocalDateTime.now());

    return userRepository.save(user);
}
```

Приложение Е – Фрагмент реализации сервисного слоя (UserService)