Министерство науки и высшего образования Российской Федерации

ФГАОУ ВО «Уральский федеральный университет им. первого Президента России Б.Н. Ельцина»

Институт радиоэлектроники и информационных технологий – РТФ

Департамент/кафедра школа бакалавриата

Отчет по проектному практикуму

«Разработка веб-приложения Spring Boot»

Руководитель Разработчик

Н.А. Архипов М.А. Бабенко

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г. «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

2025, Екатеринбург

### Содержание

[Определения 4](#_Toc212336434)

[Введение 5](#_Toc212336435)

[1 Описание используемых технологий 6](#_Toc212336436)

[1.1 Существующие фреймворки Java 6](#_Toc212336437)

[1.2 Spring Boot 6](#_Toc212336438)

[1.3 H2 Database 6](#_Toc212336439)

[1.4 Spring MVC 7](#_Toc212336440)

[1.5 Lombok 7](#_Toc212336441)

[1.6 JPA 7](#_Toc212336442)

[2 Описание задачи 9](#_Toc212336443)

[2.1 Описание задачи согласно заданию 9](#_Toc212336444)

[2.2 Описание структуры базы данных 9](#_Toc212336445)

[2.2 Описание классов (диаграмма классов и описание словами) 11](#_Toc212336446)

[2.3 Востребованность фреймворка Spring Boot 13](#_Toc212336447)

[3 Разработка веб-приложения 14](#_Toc212336448)

[3.1 Описание структуры проекта 14](#_Toc212336449)

[3.1.1 Разбор пакета «entity» 14](#_Toc212336450)

[3.1.2 Разбор пакета «repository» 15](#_Toc212336451)

[3.1.3 Разбор пакета «service» 15](#_Toc212336452)

[3.1.4 Разбор пакета «controller» 16](#_Toc212336453)

[3.1.5 Разбор пакета «config» 17](#_Toc212336454)

[3.1.6 Разбор пакета «health» 18](#_Toc212336455)

[3.1.7 Разбор пакета «templates» 18](#_Toc212336456)

[3.1.8 Разбор «application.properties» 19](#_Toc212336457)

[3.2 Описание решения поставленной задачи 20](#_Toc212336458)

[3.3 Описание реализации веб-приложения 22](#_Toc212336459)

[3.4 Результат работы работы-приложения (вставить скриншоты работы UI) 23](#_Toc212336460)

[Заключение 38](#_Toc212336461)

[Список используемых источников 39](#_Toc212336462)

Приложение на 2 листах

# Определения

Веб-приложение (Web Application) – клиент-серверное приложение, в котором клиент взаимодействует с сервером через веб-браузер. В контексте данной работы – приложение, построенное на архитектуре Spring MVC.

Spring Boot – фреймворк для упрощенной разработки приложений на основе Spring, предоставляющий автоконфигурацию и готовые к использованию компоненты.

H2 Database – встраиваемая реляционная база данных, написанная на Java. Используется на этапе разработки для быстрого старта без необходимости развертывания внешней СУБД.

Spring Security – мощный и настраиваемый фреймворк для проверки подлинности пользователя (аутентификации) и определения его прав доступа (авторизации) в приложениях Spring.

Spring Boot Actuator – модуль Spring Boot, предоставляющий готовые HTTP-эндпоинты для мониторинга состояния работающего приложения (например, проверка здоровья, просмотр метрик).

Фреймворк (Framework) – готовый каркас для разработки программного обеспечения, который содержит набор библиотек и стандартов для упрощения создания приложений.

Автоконфигурация (Auto-configuration) – механизм в Spring Boot, который автоматически настраивает компоненты приложения на основе подключенных библиотек, избавляя разработчика от ручной настройки.

Starter-зависимости – заранее сконфигурированные наборы библиотек в Spring Boot для решения конкретных задач (например, создание веб-приложения или работа с базой данных), которые нужно просто подключить к проекту.

In-Memory Database – база данных, которая хранит всю информацию в оперативной памяти компьютера, что обеспечивает очень высокую скорость работы, но данные обычно теряются после остановки приложения.

JDBC (Java Database Connectivity) – низкоуровневый стандартный API языка Java для выполнения SQL-запросов к реляционным базам данных.

JPA (Java Persistence API) – стандарт Java, который описывает систему для управления реляционными данными с помощью объектно-реляционного отображения (ORM).

Spring Data JPA – надстройка над JPA в экосистеме Spring, которая значительно сокращает объем кода, необходимого для реализации слоя работы с данными, позволяя создавать репозитории просто объявляя интерфейсы.

Spring MVC – модуль фреймворка Spring для построения веб-приложений, который реализует архитектурный паттерн «Модель-Представление-Контроллер» (Model-View-Controller).

Паттерн «Модель-Представление-Контроллер» (MVC) – архитектурный шаблон, разделяющий приложение на три компонента: Модель (данные и бизнес-логика), Представление (пользовательский интерфейс) и Контроллер (обрабатывает пользовательский ввод).

DispatcherServlet – центральный диспетчер (Front Controller) в Spring MVC, который принимает все входящие запросы и направляет их для обработки соответствующим контроллерам.

Thymeleaf – современный шаблонизатор для генерации HTML на стороне сервера, легко интегрируемый с Spring и позволяющий создавать динамические веб-страницы.

Project Lombok – библиотека, которая подключается к среде разработки и компилятору, позволяя генерировать шаблонный код (геттеры, сеттеры, конструкторы) с помощью аннотаций, делая исходный код чище и короче.

ORM (Object-Relational Mapping) – технология программирования, которая позволяет преобразовывать данные между реляционной базой данных и объектной моделью в коде на Java, позволяя работать с объектами вместо прямых SQL-запросов.

Hibernate – самая популярная реализация стандарта JPA, предоставляющая мощный и гибкий механизм для объектно-реляционного отображения.Основные сущности предметной области:

а) пользователь (User) – сущность, представляющая учетную запись в системе с атрибутами username (логин), password (пароль);

б) книга (Book) – основная бизнес-сущность, представляющая собой издание с атрибутами: id (идентификатор), title (название), author (автор), isbn (международный стандартный книжный номер), price (стоимость) и другие, необходимые для учета;

в) книжный магазин (Bookstore) – сущность, представляющая точку продаж с атрибутами id (идентификатор), name (название магазина), address (адрес);

г) роль (Role) – сущность, определяющая уровень доступа пользователя к функционалу приложения;

д) READ\_ONLY – роль, предоставляющая право только на просмотр данных без возможности их изменения;

е) USER – роль, позволяющая пользователю создавать, просматривать и редактировать только те записи, которые были созданы им самим;

ж) ADMIN – роль, предоставляющая полный доступ ко всем данным и функциям системы, включая управление ролями других пользователей;

з) лог (Log) – запись о событии, произошедшем в системе (вход пользователя, операции с данными и т.д.); записывается в отдельный файл для последующего аудита.

# Введение

Актуальность разработки веб-приложения для управления каталогом книг и книжными магазинами обусловлена необходимостью эффективного учёта таких активов, как наличие и стоимость книг, для малого бизнеса и библиотек. Целью данной работы является создание полнофункционального веб-приложения на основе фреймворка Spring Boot, которое автоматизирует процессы учёта и предоставляет интуитивно понятный интерфейс для работы с данными. Для достижения этой цели были поставлены задачи по проектированию базы данных с использованием H2, реализации системы аутентификации и ролевой авторизации (READ\_ONLY, USER, ADMIN) средствами Spring Security, а также разработке комплекса веб-форм для выполнения операций CRUD.

Дополнительно в приложение интегрирован Spring Boot Actuator для мониторинга состояния системы. В ходе работы применялись следующие технологии: Java, Spring Boot (MVC, Data JPA, Security), H2 Database и Thymeleaf для создания веб-интерфейсов, а среда разработки IntelliJ IDEA использовалась в качестве основного инструмента.

## 1 Описание используемых технологий

1.1 Существующие фреймворки Java

Для разработки на Java применяются несколько фреймворков:

а) Spring Boot – предназначен для создания автономных готовых приложений, является открытым и использует лицензию Apache 2.0;

б) Jakarta EE – ориентирован на крупные корпоративные системы, развертываемые на серверах приложений, и также имеет открытый код.

в) Micronaut и Quarkus – сфокусированы на микросервисах, оба с открытым исходным кодом под лицензией Apache 2.0, причем Quarkus оптимизирован для GraalVM.

Для данного проекта выбран Spring Boot благодаря автоконфигурации, быстроте разработки и обширному сообществу.

1.2 Spring Boot

Spring Boot – это фреймворк, значительно упрощающий создание приложений на Spring. Его ключевые особенности включают:

а) автоконфигурацию на основе зависимостей проекта;

б) использование встроенных серверов (Tomcat, Jetty) для создания приложений;

в) предоставление стартовых зависимостей для минимизации ручной настройки;

г) интеграцию Spring Boot Actuator для мониторинга.

1.3 H2 Database

H2 – это быстрая база данных, которая полностью работает в памяти компьютера или в одном файле, что идеально подходит для этапа разработки. Среди похожих баз данных можно выделить: H2 (написана на Java, работает в памяти или как сервер), SQLite (хранит все в одном файле) и Apache Derby (тоже на Java).

В Spring есть несколько способов работы с базой данных: прямой низкоуровневый способ (JDBC), более удобный способ через объекты (JPA), который позволяет не писать простые запросы вручную (Spring Data JPA), и упрощенный низкоуровневый способ (JDBC Template). В этом проекте используется Spring Data JPA, как самый современный и удобный.

1.4 Spring MVC

Spring MVC – это часть Spring, отвечающая за работу веб-страниц. Она построена по классической схеме «модель-представление-контроллер». Работа происходит так: браузер отправляет запрос на главный сервер Spring; тот находит нужный класс-обработчик (контроллер); контроллер выполняет логику, готовит данные (модель) и сообщает, какую страницу показать; система находит нужный файл-шаблон (представление), подставляет в него данные из модели; и в браузер возвращается готовая HTML-страница.

1.5 Lombok

Lombok – это библиотека, которая помогает писать меньше повторяющегося кода. Она позволяет автоматически создавать стандартные методы для классов (например, получение и установка значений, сравнение объектов) простым добавлением специальных команд. Плюсы её использования:

а) код становится короче и чище;

б) разработчик тратит меньше времени на рутину и делает меньше ошибок.

Минусы использования Lombok:

а) для её работы нужно установить дополнительный модуль в среду разработки;

б) новичкам может быть непонятно, откуда берутся автоматически созданные методы.

1.6 JPA

JPA – это стандартный способ работы с реляционными базами данных из Java, при котором таблицы в базе данных отображаются на обычные Java-объекты. Основные программы, которые реализуют этот стандарт: Hibernate (самый популярный), EclipseLink и Apache OpenJPA. Плюсы JPA:

а) не нужно привязываться к конкретной базе данных;

б) код становится намного короче;

в) работа ведется с удобными объектами, а не с таблицами.

Минусы JPA:

а) нужно время, чтобы освоить технологию;

б) при неумелом использовании программа может работать медленнее.

В данном проекте используется Hibernate.

## 2 Описание задачи

2.1 Описание задачи согласно заданию

Задача – разработка веб-приложения для управления каталогом книг и книжных магазинов с использованием стека технологий Spring Boot и H2 Database. Приложение должно предоставлять функциональность для учета стоимости книг и их наличия в магазинах.

Реализация базы данных со следующими сущностями:

а) таблица Users для хранения учетных записей (логин и пароль);

б) таблица Book для учета книг (с идентификатором, названием, автором, стоимостью);

в) таблица Bookstore для хранения данных о книжных магазинах (с идентификатором, названием, адресом).

Система аутентификации и авторизации с ролевой моделью доступа:

а) READ\_ONLY - возможность только просматривать данные (роль по умолчанию при регистрации);

б) USER - возможность управлять только своими записями;

в) ADMIN - полный доступ ко всем данным и функциям системы;

Набор веб-форм для взаимодействия с пользователем:

а) форма входа в приложение;

б) формы списков для таблиц книг и магазинов;

в) формы ввода и редактирования данных;

г) главная страница;

д) страница "О приложении" с информацией о разработчике;

е) форма расчета характеристик (например, общей стоимости книг);

ж) форма управления ролями пользователей (для администратора).

2.2 Описание структуры базы данных

Создаем базу данных – bookstore\_db. База данных приложения состоит из основных четырех таблиц, которые образуют систему управления книгами. Все таблицы связаны через внешние ключи – база данных целостна. Ее схема представлена на рисунке 1.

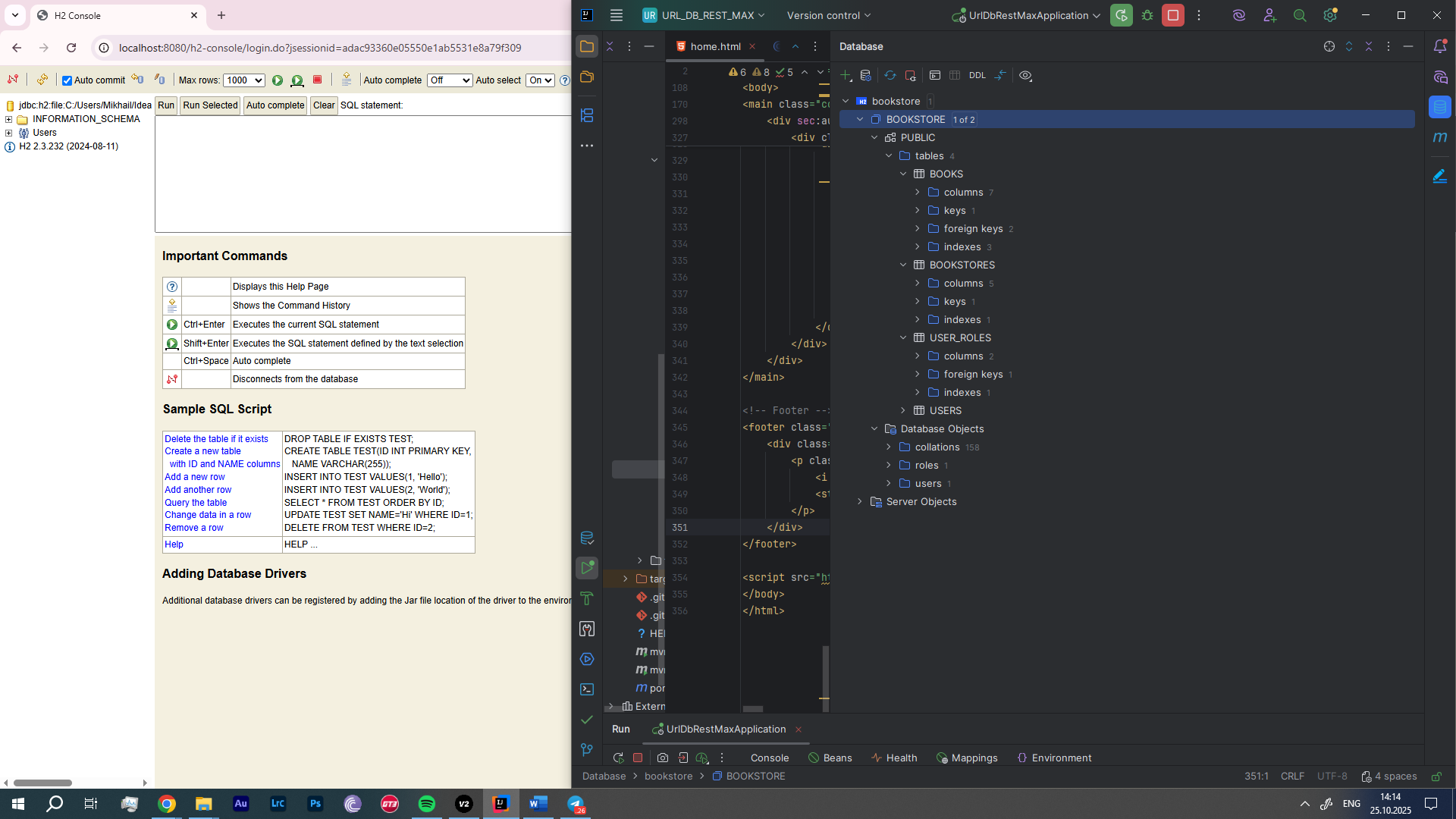


Рисунок 1 – Схема базы данных bookstore\_db

Теперь детально опишем таблицы. Начнем с таблицы USERS (пользователи):

а) id - уникальный идентификатор;

б) username - имя пользователя;

в) password – зашифрованный пароль.

Для таблицы USER\_ROLES (права пользователей):

а) user\_id - ссылка на пользователя;

б) role – название роли (READ\_ONLY – только чтение, USER – обычный пользователь (право на создание и редактирование), ADMIN – администратор (полный доступ)).

Для таблицы BOOKSTORES (книжные магазины):

а) id - уникальный идентификатор;

б) name – название магазина;

в) address - адрес;

г) city – город;

д) phone – телефон.

Для таблицы BOOKS (книги):

а) id - уникальный идентификатор;

б) title – название книги;

в) author - автор;

г) isbn – ISBN код (международный стандартный книжный номер, уникальный идентификатор книжного издания);

д) price – цена;

е) bookstore\_id – ссылка на магазин;

ж) user\_id – пользователь, создавший запись.

2.2 Описание классов (диаграмма классов и описание словами)

Изучим структуру созданного приложения. Все пакеты и классы внутри него показаны на рисунке 2.

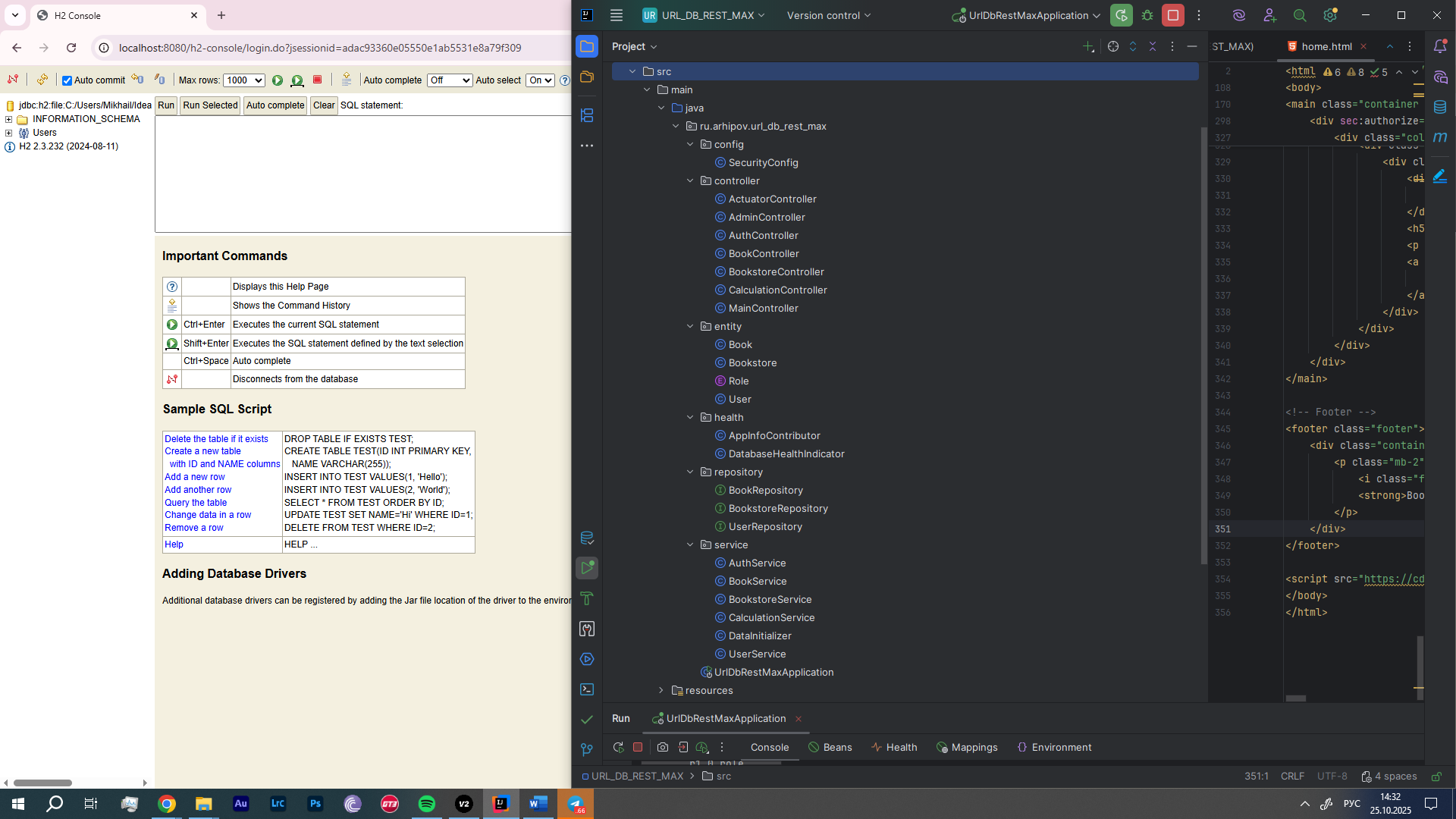


Рисунок 2 – Все пакеты и классы внутри приложения

Разберем поочередно все созданные пакеты. Начнем с пакета «entity» - он представляет собой сущности для баз данных. В нем содержатся сущности книг (связь многие-к-одному с книжным магазином и многие-к-одному с пользователем), ролей (определяет возможные роли пользователей), пользователей (связь один-ко-многим), а также книжных магазинов (связь один-ко-многим с книжным магазином).

Далее, пакет «repository» – содержит классы: UserRepository.java (обеспечивает доступ к данным пользователей), BookRepository.java (обеспечивает доступ и поиск по книгам), BookstoreRepository.java (управляет данными книжных магазинов).

Следующий пакет – «service», является пакетом бизнес-логики: UserService.java (содержит бизнес-логику работы с пользователями), BookService.java (логика работы с книгами и расчетами), BookstoreService.java (управление книжными магазинами и статистикой), AuthService.java (логика аутентификации и регистрации), CalculationService.java (расчеты и аналитика), DataInitializer.java (инициализация данных при старте приложения).

Пакет «controller» представляет собой веб-слой, содержит: MainController.java (обработка основных страниц приложения), BookController (операции записями для книг), AuthController.java (управление аутентификацией и регистрацией), AdminController.java (функционал администратора), CalculationController (отображение статистики и расчетов), ActuatorController.java (информация о SB Actuator).

Пакет «config» отвечает за конфигурацию и содержит один класс – SecurityConfig.java, он нужен для конфигурации Spring Security.

Пакет «health» содержит два класса: DatabaseHealthIndicator.java (проверка доступности базы) и DatabaseHealthIndicator.java (добавление информации в /actuator/info).

Также создаем диаграмму классов, она представлена на рисунке 3.

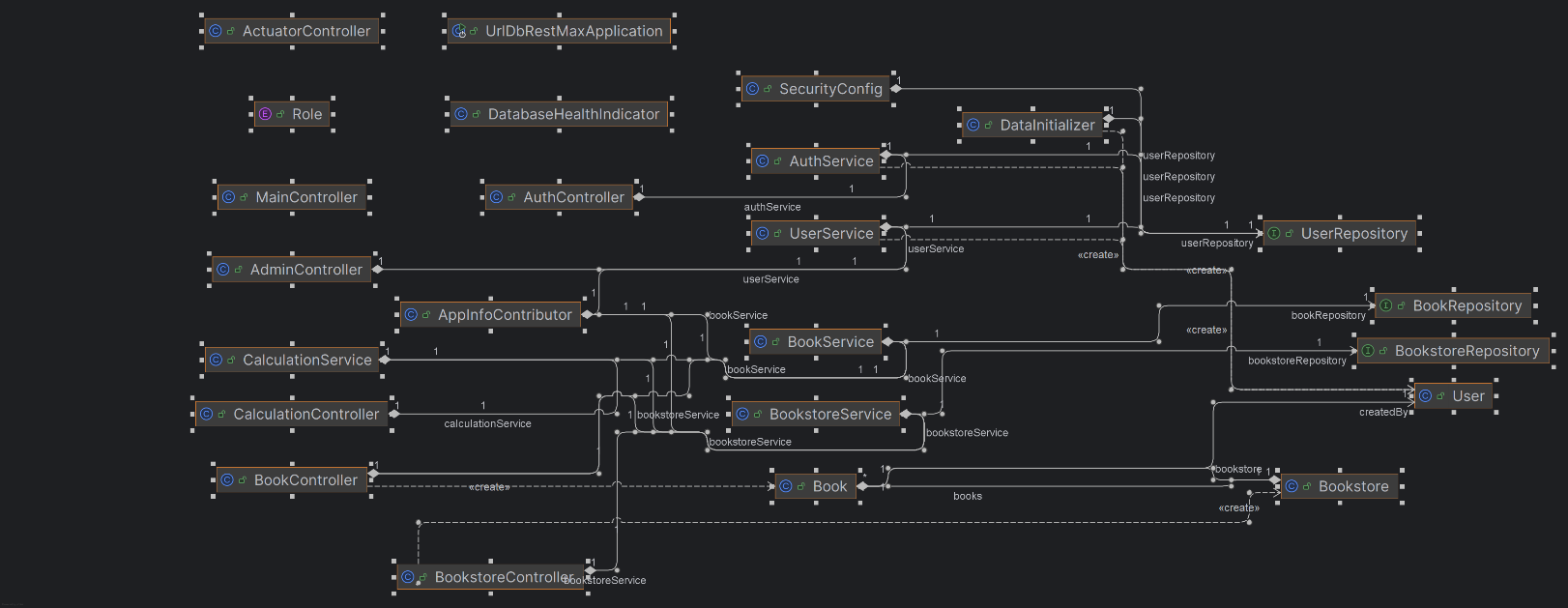


Рисунок 3 – Диаграмма классов приложения

2.3 Востребованность фреймворка Spring Boot

Фреймворк Sring Boot очень востребован в разработке приложений, он является промышленным стандартом в enterprise-разработке. Используется многими крупными компаниями: Яндекс, ВК, Озон; большое количество стартапов также использует Spring Boot.

В этой работе мной должны быть освоены:

а) многоуровневая архитектура SB приложения (Spring Boot Layered Application Architecture);

б) стандартный стек технологий Spring Boot (Standard Spring Boot Tech Stack).

## 3 Разработка веб-приложения

3.1 Описание структуры проекта

Созданное приложение содержит множество пакетов: . Рассмотрим их подробно по отдельности.

3.1.1 Разбор пакета «entity»

Пакет содержит следующие классы:

1 User.java – класс представляет собой сущность пользователей системы. Он хранит учетные данные (логин и пароль), систему ролей доступа. Каждый пользователь может иметь несколько ролей, которые хранятся в таблице через связь один-ко-многим. Ключевые моменты:

а) аннотация «@Entity» указывает, что класс является сущностью;

б) «@Table(name = "users")» задает имя таблицы в базе данных;

в) «@ElementCollection» используется для хранения набора ролей в отдельной таблице «user\_roles»;

г) «FetchType.EAGER» гарантирует, что роли загружаются сразу при загрузке пользователя.

2 Book.java – класс представляет собой сущность книги в системе. Содержит библиографическую информацию о книге, а также связи с магазином и пользователем-создателем записи. Ключевые моменты:

а) тип «BigDecimal» используется для точного хранения цены книги;

б) связь «@ManyToOne» с магазином Bookstore означает, что многие книги могут принадлежать одному магазину;

в) связь «@ManyToOne» с пользователем User осуществляет привязку с создателем записи.

3 Bookstore.java – класс представляет собой сущность книжного магазина. Содержит контактную информацию и связь с книгами, участвует в операциях при управлении книгами. Ключевые моменты:

а) связь «@OneToMany» обеспечивает связь книг в каскадных операциях;

б) «orphanRemoval = true» отвечает за автоматическое удаление дочерних записей, которые потеряли связь с родительской.

4 Role.java – определяет возможные роли в системе, не является сущностью.

3.1.2 Разбор пакета «repository»

Пакет содержит интерфейсы доступа к данным, содержит следующие классы:

1 UserRepository.java – класс обеспечивает все операции с данными пользователей. Автоматически реализует стандартные методы управления и добавляет дополнительные запросы. Ключевые моменты:

а) «findByUsername()» – осуществляет поиск по имени пользователя для аутентификации;

б) «existsByUsername()» – осуществляет проверку уникальности имени при регистрации.

2 BookRepository.java – класс предоставляет сложные запросы для работы с книгами (например, поиск по критерям или оптимизированная загрузка). Ключевые моменты:

а) содержит методы поиска с «ContainingIgnoreCase» для реализации более «умного» поиска (например, по части слова или без учета регистра);

б) «@Query» с «JOIN FETCH» – данный запрос позволяет за один «большой» запрос получить данные, аналогичные нескольким «маленьким» запросам.

3 BookstoreRepository.java – класс позволяет управлять данными магазинов и предоставляет запросы для статистики.

3.1.3 Разбор пакета «service»

Пакет содержит бизнес-логику приложения. Сервисы работают как посредники между контроллерами и репозиториями, обеспечивают целостность данных и сложную логику. Пакет содержит следующие классы:

1 UserService.java – сервис содержит логику работы с пользователем: управление ролями, валидация, проверка прав доступа. Ключевые моменты:

а) назначение и отзыв ролей с проверкой бизнес-правил;

б) метод «hasRole()» осуществляет проверку прав доступа.

2 BookService.java – сервис содержит логику работы с книгами, включая операции поиска, фильтрации и финансовые расчеты.

3 BookstoreService.java – сервис, управляющий информацией о книжных магазинах, представляет данные для отчетности. Ключевые моменты:

а) «BookController» используется при создании или редактировании книги для магазина;

б) «CalculationController» используется для отображения статистики по магазинам.

4 AuthService.java – класс отвечает за процессы регистрации и аутентификации, обеспечивает безопасное хранение паролей.

а) используется интеграция с Spring Security для аутентификации;

б) «AuthController» нужен для обработки регистрации и входа.

5 CalculationService.java – выполняет сложные аналитические расчеты и статистический анализ данных. «CalculationController» используется для отображения всей статистики на веб-странице.

6 DataInitializer.java – запускается при старте приложения и создает первичные данные для демонстрации функционала приложения. Ключевые моменты:

а) проверка «count() == 0» используется, чтобы не перезаписывать существующие данные;

б) «passwordEncoder.encode()» – для хранения шифрованных паролей;

в) создаются пользователи с разными ролями для тестирования системы безопасности.

3.1.4 Разбор пакета «controller»

Пакет обрабатывает запросы и управляет взаимодействием между клиентом и сервером. Использует сервисы для выполнения бизнес-логики, и возвращают данные. Пакет содержит следующие классы;

1 AuthController.java – класс выполняет управления всеми аспектами аутентификации, интегрируется со Spring Security. Ключевые моменты:

а) «@GetMapping("/login")» – форма входа в приложения;

б) «@GetMapping("/register")» – форма регистрации нового пользователя;

в) «@PostMapping("/register")» – обработка данных регистрации;

г) «AuthService» используется для бизнес-логики регистрации;

д) сообщения об успехе или ошибках возвращаются.

2 AdminController.java – предоставляет функции администратора, доступные только пользователям с ролью «ADMIN». Ключевые моменты:

а) «@PreAuthorize("hasRole('ADMIN')")» – осуществляется защита на уровне класса;

б) «@GetMapping("/users")» – используется список всех пользователей и их ролей;

в) «@PostMapping» – используется для добавления или удаления ролей.

3 CalculationController.java – отвечает за аналитические отчеты, статистику и результаты расчетов. Ключевые моменты:

а) «@PreAuthorize("isAuthenticated()")» – доступ для всех аутентицицированных пользователей;

б) для вычислений используется «CalculationService»;

в) данные организуется по категориям (основная статистика, по магазинам, по авторам).

4 ActuatorController.java – простой контроллер для отображения информации об pring Boot Actuator.

3.1.5 Разбор пакета «config»

Пакет содержит классы конфигурации Spring, настраивающие безопасность, бины, компоненты и интеграции. Пакет содержит только один класс.

SecurityConfig.java – центральный класс для настройки безопасности приложения; определяет политику доступа, аутентификации и авторизации. Ключевые моменты:

а) «@EnableWebSecurity» – активируется Spring Security;

б) «PasswordEncoder» – бин для хеширования паролей;

в) настройка правил доступа через «authorizeHttpRequests()»;

г) страница логина реализуется через «formLogin()»;

д) интеграция с H2 Console (для удобства работы с БД);

е) ролевая модель доступа с «hasRole()» и «authenticated()».

3.1.6 Разбор пакета «health»

Содержит «health checks» для Spring Boot Actuator, который представляет информацию о состоянии компонентов приложения. Содержит следующие классы:

1 DatabaseHealthIndicator.java – класс отвечает проверку «здоровья» базы данных (проверка работоспособности). Ключевые моменты:

а) реализуется интерфейс «HealthIndicator»;

б) соединение с базой данных проверяется через «DataSource».

2 AppInfoContributor.java – поставляет информацию о приложении в «/actuator/info». Реализует интерфейс «InfoContributor», статистика приложения собирается из сервисов.

3.1.7 Разбор пакета «templates»

Пакет содержит HTML шаблоны, которые представляют собой веб-интерфейс приложения. Для «красоты» дизайна используем Thymeleaf – современный серверный Java шаблонизатор, который позволяет создавать современно выглядящие HTML шаблоны. Используемые шаблоны:

1 layout.html – основной макет приложеия, служит базовым для всех страниц приложения. Определяет общую структуру – меню навигации, заголовки, нижний слой страницы, области для контента. Для адаптивного дизайна используется Bootstrap 5.

2 home.html – главная страница приложения, предоставляет пользователю навигацию по основным разделам. Содержит информацию о текущем пользователе и его ролях (правах).

3 login.html – страница аутентификации, используется для входа в систему, интегрируется со Spring Security. Имеет форму входа с полями «username» и «password» (имя пользователя и пароль), показывает ошибки аутентификации, есть ссылка на регистрацию новых пользователей, выведены тестовые учетные записи.

4 register.html – страница регистрации новых пользователей, помогает создать новую учетную запись в системе. Есть вывод сообщений об успешной регистрации или ошибках, возникшей во время нее. Есть автоматическое назначение ролей (прав).

5 books/list.html – отображает таблицу со всеми книгами в системе, есть возможности для операций над ними.

6 books/create.html – форма для добавления новой книги в систему, имеет обязательные поля (автор, название, цена), выпадающий список для выбора книжного магазина, кнопки отправки и отмены.

7 books/edit.html – форма для редактирования информации о существующей книги, есть предзаполненные поля.

8 bookstores/list.html – список магазинов, аналогичен списку книг. Отображается название магазина, адрес и контактные данные, кнопки управления.

9 bookstores/create.html – форма для создания нового магазина.

10 bookstores/edit.html – форма для редактирования информации о магазинах.

11 admin/users.html – страница администратора для управления ролями пользователей. Выводится таблица пользователей с текущими ролями, имеются динамические формы для добавления или удаления ролей.

12 calculations/statistics.html – комплексная страница с аналитикой: финансовая статистика, наибольшие и наименьшие значения стоимости, статистика по магазинам, статистика по авторам, показатели оборота и рентабельности. Для группировки используются карточки, для структурированных данных – таблицы.

3.1.8 Разбор «application.properties»

Это главный файл конфигурации Spring Boot приложения, содержащий все настройки баз данных, безопасности и мониторинга:

1 Настройки баз данных H2

а) «spring.datasource.url» – путь к файлу базы данных;

б) «spring.jpa.hibernate.ddl-auto=update» – автоматическое обновление схемы базы данных;

в) «spring.h2.console.enabled=true» – активация веб-консоли H2 (используем ее для удобства работы с базой данных).

2 Настройки JPA и Hibernate

а) «spring.jpa.show-sql=true» – разрешаем логгирование SQL запросов;

б) «hibernate.format\_sql=true» – разрешаем форматирование SQL для читаемости.

3 Настройки Spring Boot Actuator

а) «management.endpoints.web.exposure.include» – определяем, какие страницы с информацией о приложении открыты для просмотра;

б) «management.endpoint.health.show-details=always» – разрешим показ подробностей о работе каждого компонента.

4 Настройки получаемой информации – «info.app.\*» – метаданные приложения для «/actuator/info».

5 Настройки логгирования – logging.file.name – файл для записи логгов.

3.2 Описание решения поставленной задачи

В результате курсового проектирования, было создано полнофункциональное веб-приложение для управления библиотекой книг и сетью книжных магазинов. Оно представляет собой сложную программную систему с многоуровневой архитектурой, ролевой моделью доступа и комплексным веб-интерфейсом. Согласно заданию, во время разработки должен быть четко определен стек технологий; в веб-приложении должны быть реализованы его безопасность, функциональные возможности.

Безопасность приложения реализована следующим образом:

1 Ролевая модель доступа:

а) для «READ\_ONLY» (только чтение) – просмотр списков книг и магазинов, доступ к статистике и аналитике; невозможность создания, редактирования и удаления в списках;

б) для «USER» (обычный пользователь) – все возможности «READ\_ONLY», а также возможности создания и редактирования собственных записей, а также управление связанными данными;

в) для «ADMIN» (администратор) – полный доступ ко всем данным системы, управление ролями других пользователей, административные функции.

2 Процесс аутентификации:

а) имеется возможность регистрации новых пользователей и авторизации для уже существующих;

б) при неудачной авторизации – пользователю выводится соответствующее сообщение;

в) при регистрации нового пользователя – ему дается доступ «READ\_ONLY».

Функциональные возможности приложения реализованы следующим образом:

1 Управление книгами:

а) просмотр каталога книг в табличном формате;

б) добавление новых книг;

в) редактирование существующих записей о книгах;

г) удаление существующих записей о книгах;

д) привязка книг к книжным магазинам;

е) поиск книг по параметрам (автор, цена, и др.).

2 Управление магазинами:

а) отображение списка книжных магазинов;

б) создание новых магазинов с контактной информацией;

в) редактирование информации о существующих магазинах;

г) удаление информации о существующих магазинах.

3 Аналитика и расчеты:

а) финансовая статистика по книгам;

б) наибольшие и наименьше значения;

в) распределение книг по магазинам и авторам;

г) финансовые показатели по обороту.

4 Административные функции:

а) просмотр списка пользователей системы;

б) управление ролями;

в) мониторинг системы через Spring Boot Actuator;

г) контроль над данными приложения.

Задача решалась при помощи следующих технологий:

1 Серверные технологии:

а) Spring Boot 3.5.7 – основной «фреймворк»;

б) Spring Security – аутентификация и авторизация;

в) Spring Data JPA – работа с базой данных;

г) Spring MVC – веб-слой и обработка запросов;

д) Hibernate – связь Java объектов с таблицами базы данных;

е) Spring Boot Actuator – система мониторинга приложения.

2 Клиентские технологии:

а) Thymeleaf – серверный шаблонизатор;

б) Bootstrap 5 – готовый набор стилей для быстрой верстки сайтов;

в) HTML5 – разметка веб-страниц.

3 Инструменты и база данных:

а) IntelliJ IDEA – среда разработки приложения;

б) Maven – система сборки и управления зависимостями;

в) H2 Database – встроенная реляционная база данных;

г) H2 Console – веб-интерфейс для работы с базой данных.

Бизнес-логика приложения реализована следующим образом:

1 Финансовые расчеты.

2 Статистические отчеты:

а) распределение книг по авторам и магазинам;

б) анализ цен;

в) прогнозы для бизнес-планирования.

3.3 Описание реализации веб-приложения

Приложение построено по классической трехуровневой архитектуре Spring Boot (Controller > Service > Repository). Разберем эти уровни:

1 Контроллеры (Controller) – верхний уровень; обрабатывают HTTP запросы (например, «BookController»).

2 Сервисы (Service) – средний уровень; содержат бизнес-логику, используют репозитории для доступа к данным (например, «BookService»).

3 Репозитории (Repository) – нижний уровень; Spring Data JPA автоматически создает запросы к базе данных по названиям методов в репозитории, все отличающиеся запросы реализованы через «@Query».

В приложении также была реализована система безопасности – через Spring Security Framework. Правила настроены в классе SecurityConfig, имеются правила для доступа к различным URL в зависимости от ролей (прав) пользователей:

1 Для публичных страниц (главная, авторизация, регистрация) – разрешен доступ без аутентификации.

2 Для защищенных страниц (книги, магазины, статистика) – требуется аутентификация, пользователь должен войти в систему.

3 Для административных функций (управление ролями, доступ к Actuator) – требуется роль «ADMIN».

Реализована настраиваемая страница входа вместо стандартной, а хэширование паролей осуществляется при помощи алгоритма BCrypt; он обеспечивает надежное хранение учетных данных.

Также в приложении реализована работа с базой данных при помощи JPA. Все сущности помечены аннотациями, которые определяют их отображение в таблицах баз данных. Были реализованы различные типы связи между сущностями:

1 Один-ко-многим – между User и UserRole (один пользователь может иметь несколько ролей);

2 Один-ко-многим – между Bookstore и Book (один магазин может содержать много книг);

3 Многие-к-одному – между Book и User (много книг могут быть созданы одним пользователем).

Также были реализованы мониторинг и диагностика приложения с помощью Actuator:

а) «/actuator/health» – общее состояние здоровья приложения;

б) «/actuator/info» – информация о приложении и его метрики;

в) «/actuator/metrics» – различные метрики производительности;

г) «/actuator/env» – переменные окружения и конфигурации.

3.4 Результат работы работы-приложения (вставить скриншоты работы UI)

Сначала запустим приложение. Лог приложения при успешном запуске показан на рисунке 4.

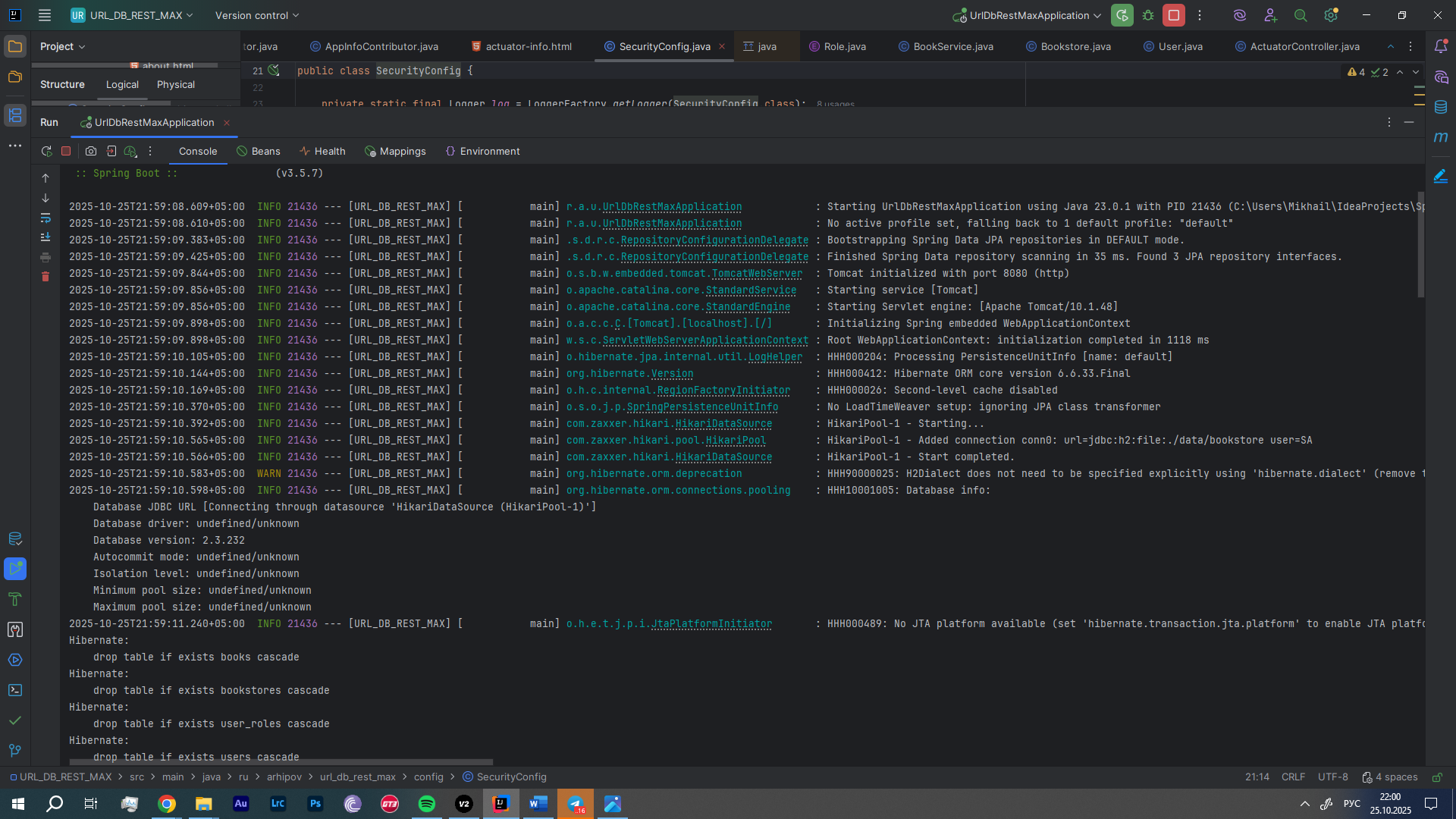


Рисунок 4 – Лог приложения при успешном запуске

Переходим теперь на страницу <http://localhost:8080/>. Результат выполнения запроса показан на рисунке 5.

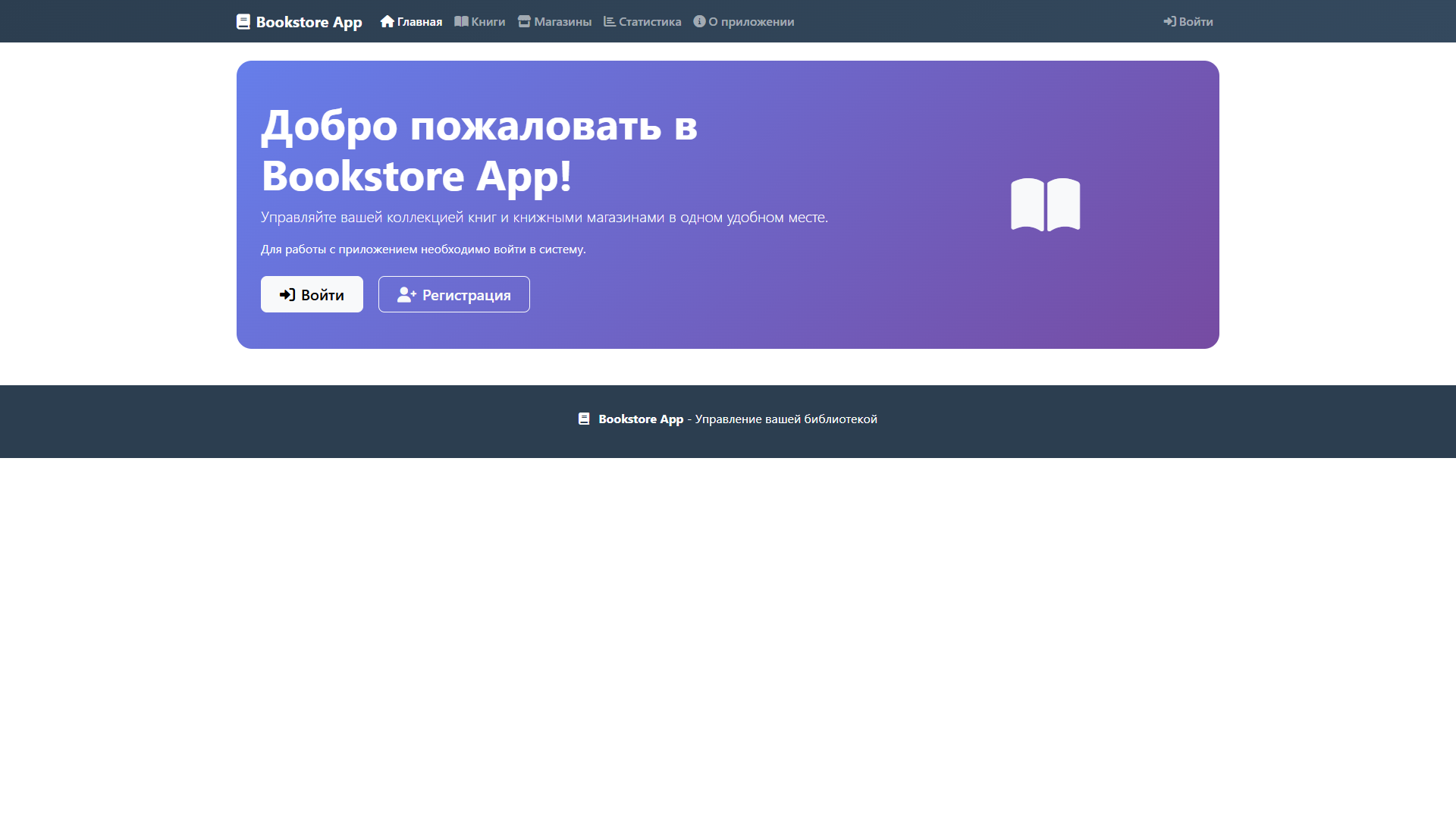


Рисунок 5 – Результат выполнения запроса

При попытке войти в один из разделов в верхнем меню – нас переносит на страницу авторизации пользователя. Теперь пробуем войти под ранее созданной учетной записью администратора, меню авторизации показано на рисунке 6.

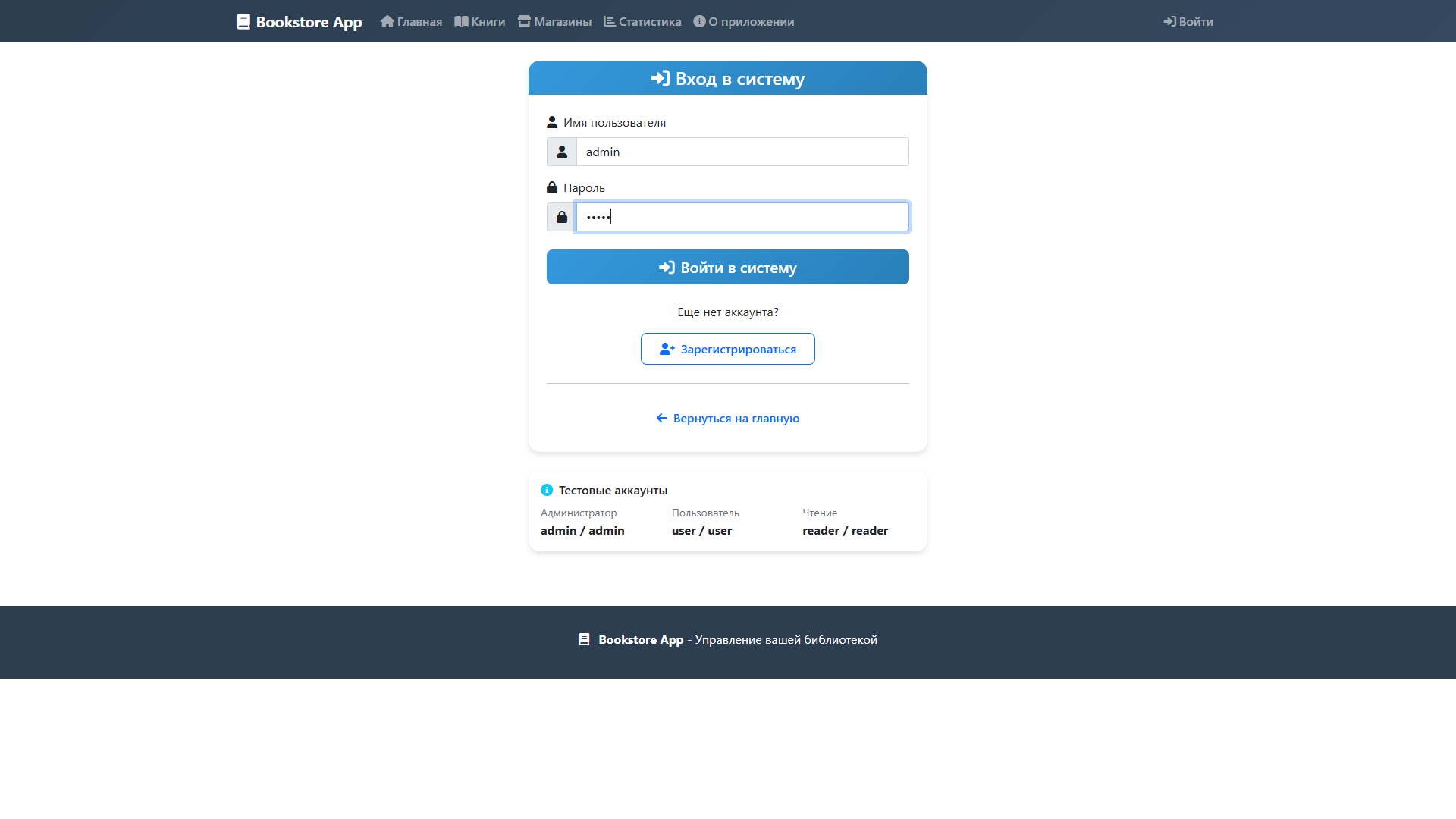


Рисунок 6 – Меню авторизации

Получилось успешно войти в учетную запись, главная страница показана на рисунке 7.

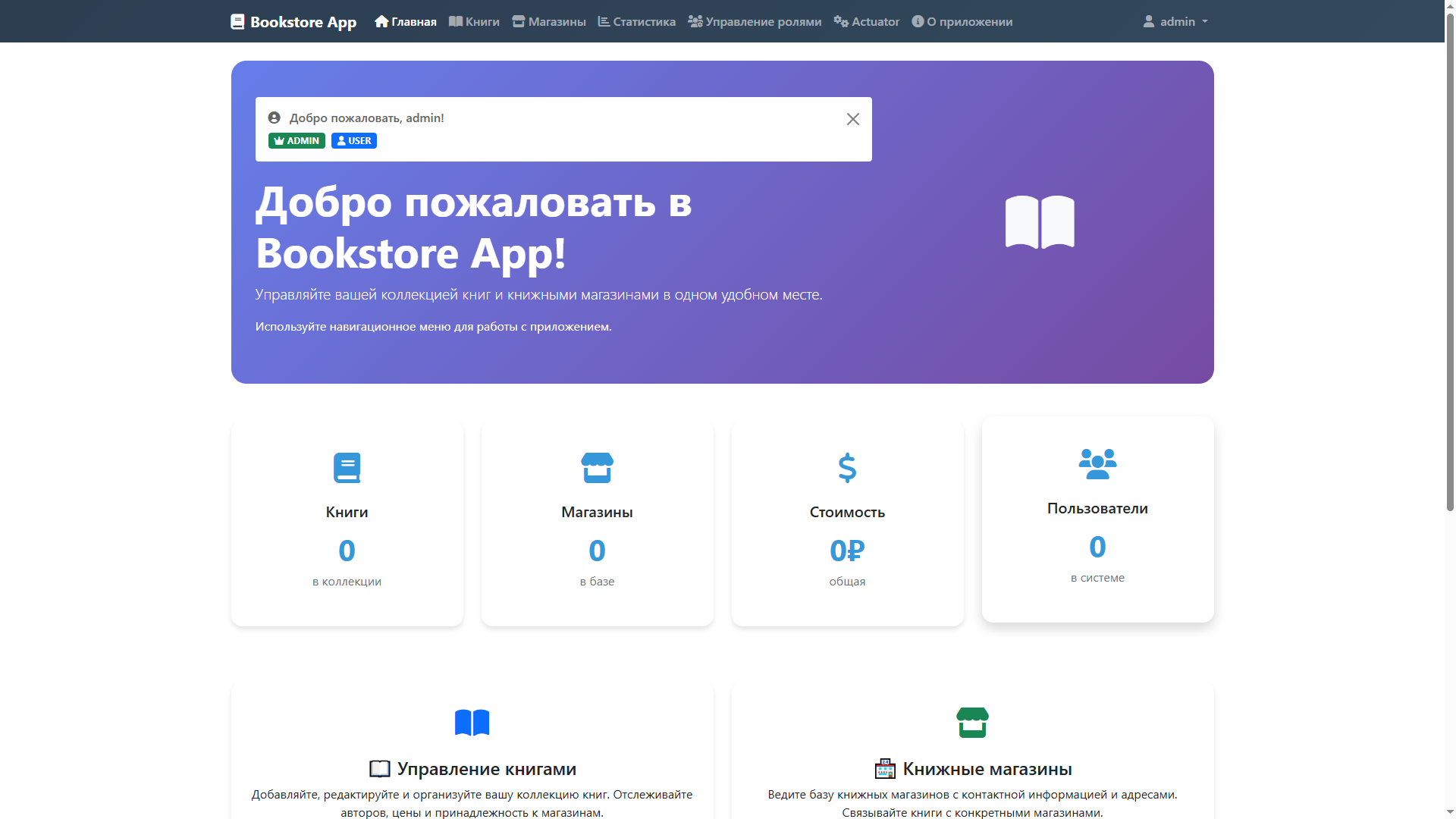


Рисунок 7 – Главная страница приложения

Теперь входим в учетную запись под ником «user» с меньшим уровнем прав доступа, главная страница показана на рисунке 8.

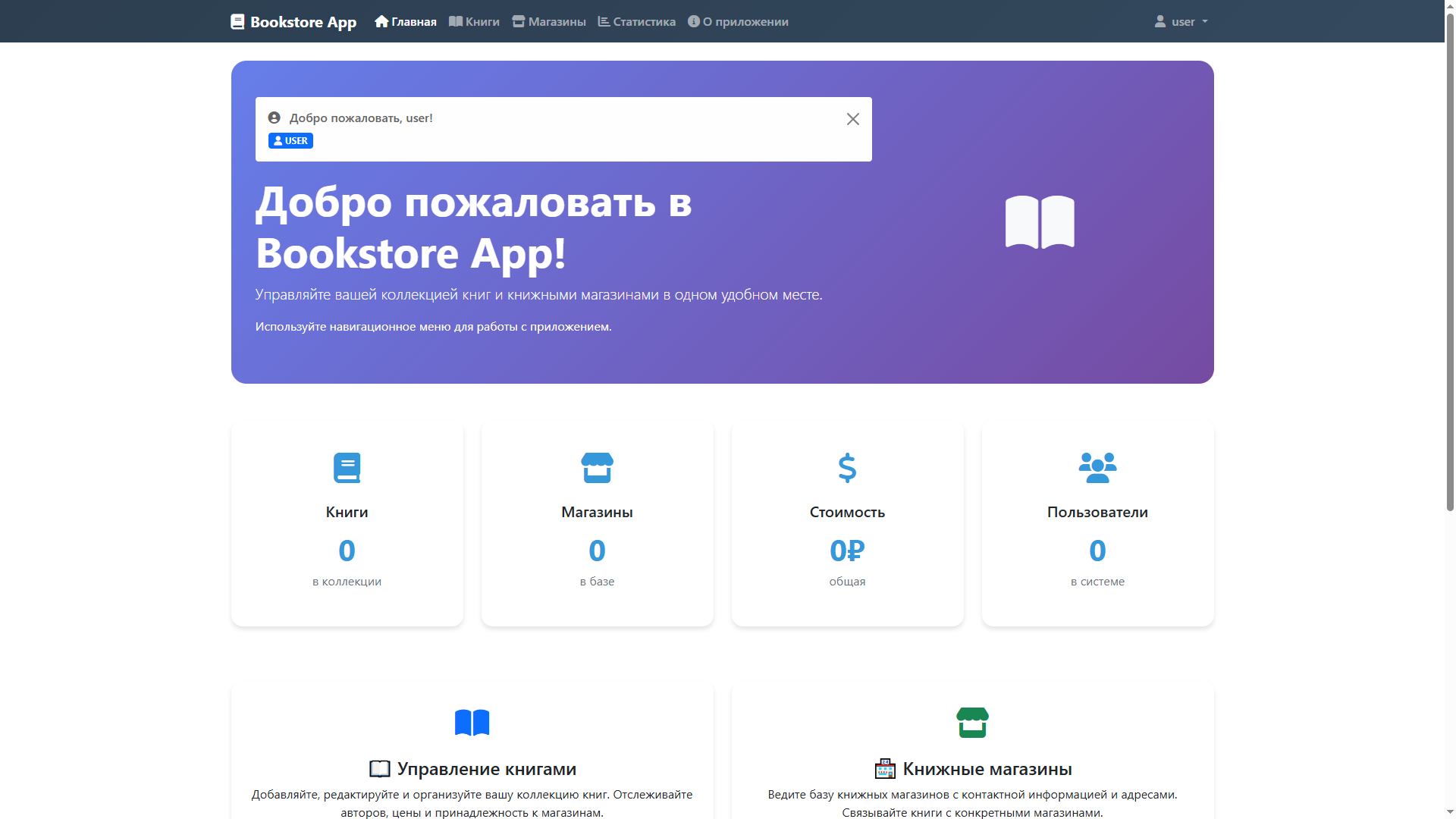


Рисунок 8 – Главная страница приложения при входе в учетную запись «user»

Теперь входим в учетную запись под ником «reader» с меньшим уровнем прав доступа, главная страница показана на рисунке 9.

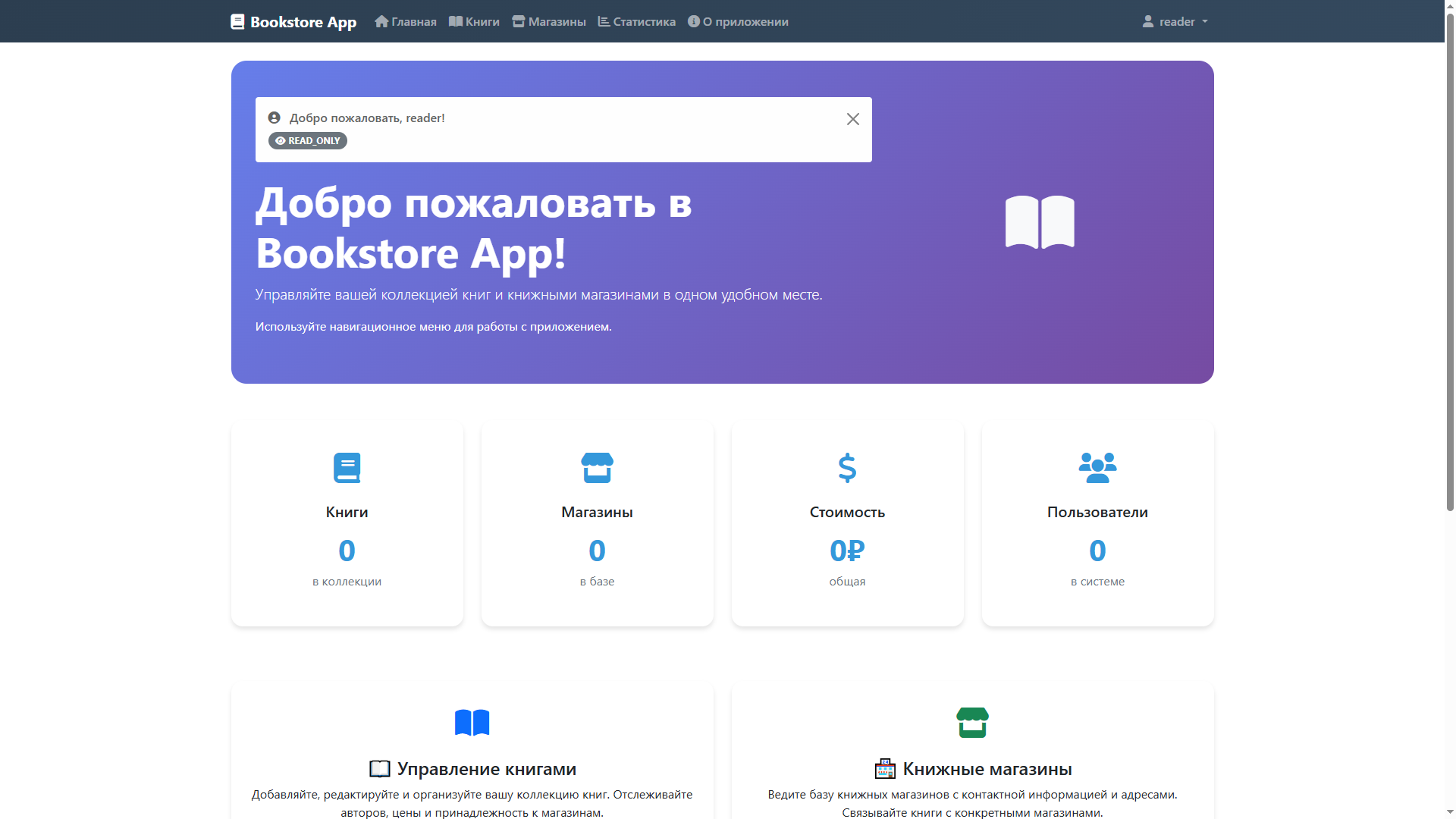


Рисунок 9 – Главная страница приложения при входе в учетную запись «reader»

Входим в «Список книжных магазинов» под ником «reader» – видим, что мы можем только просмотреть данный раздел, а кнопка создания – отсутствует; показано на рисунке 10.

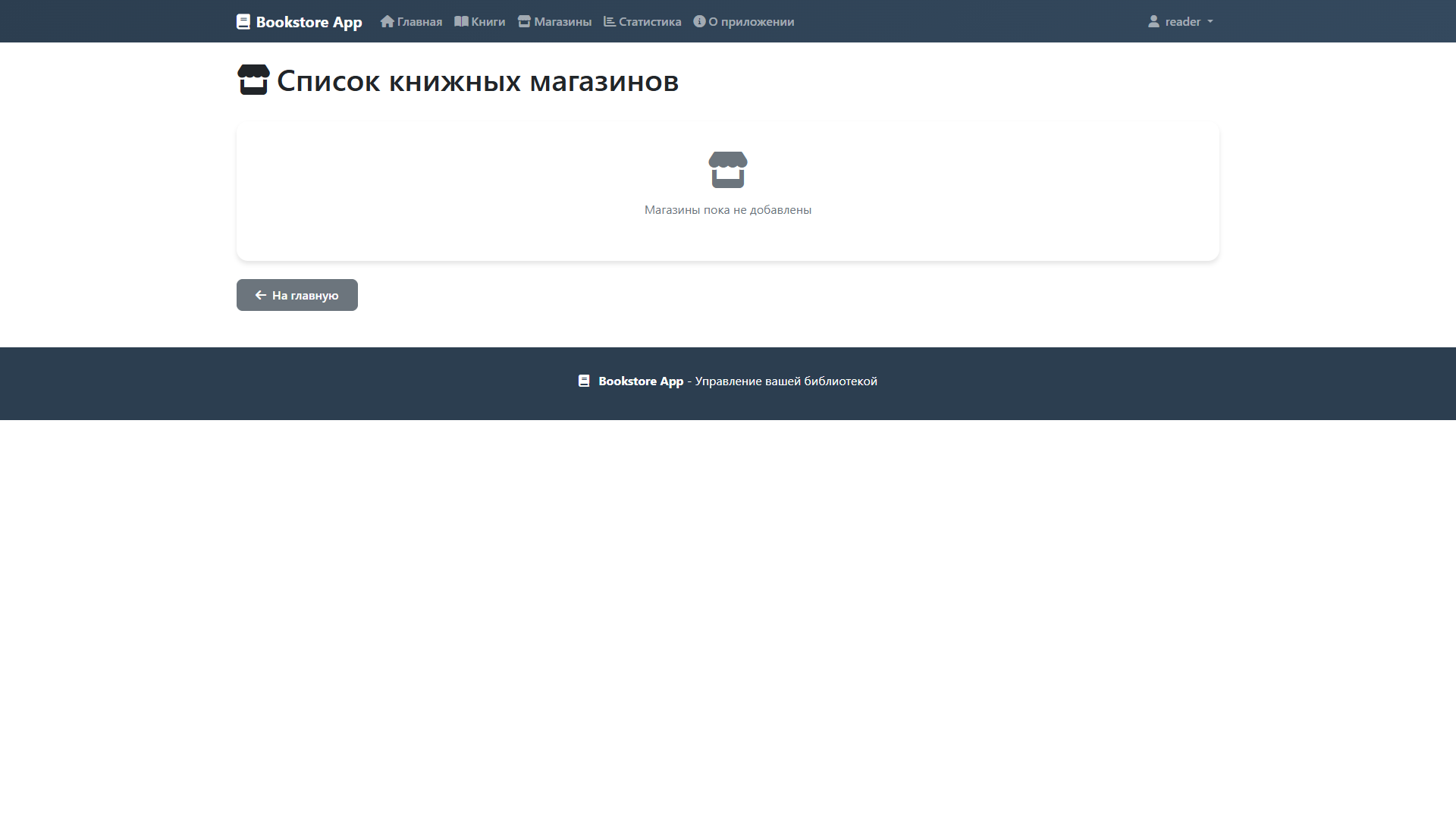


Рисунок 10 – «Список книжных магазинов» под ником «reader»

При выходе из учетной записи показывается соответствующее сообщение – показано на рисунке 11.

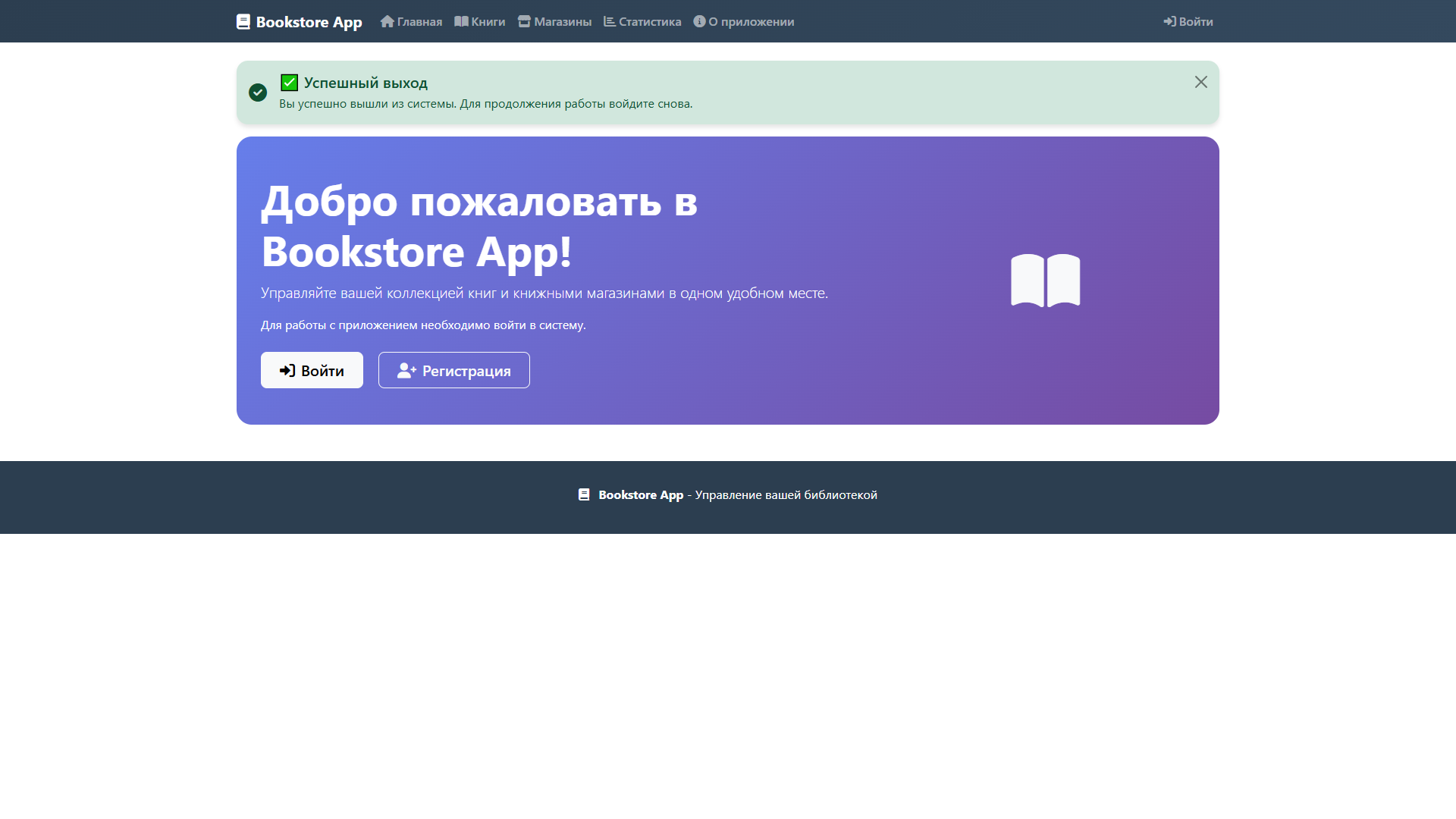


Рисунок 11 – Сообщение при выходе из учетной записи

Теперь регистрируем нового пользователя – результат показан на рисунке 12.

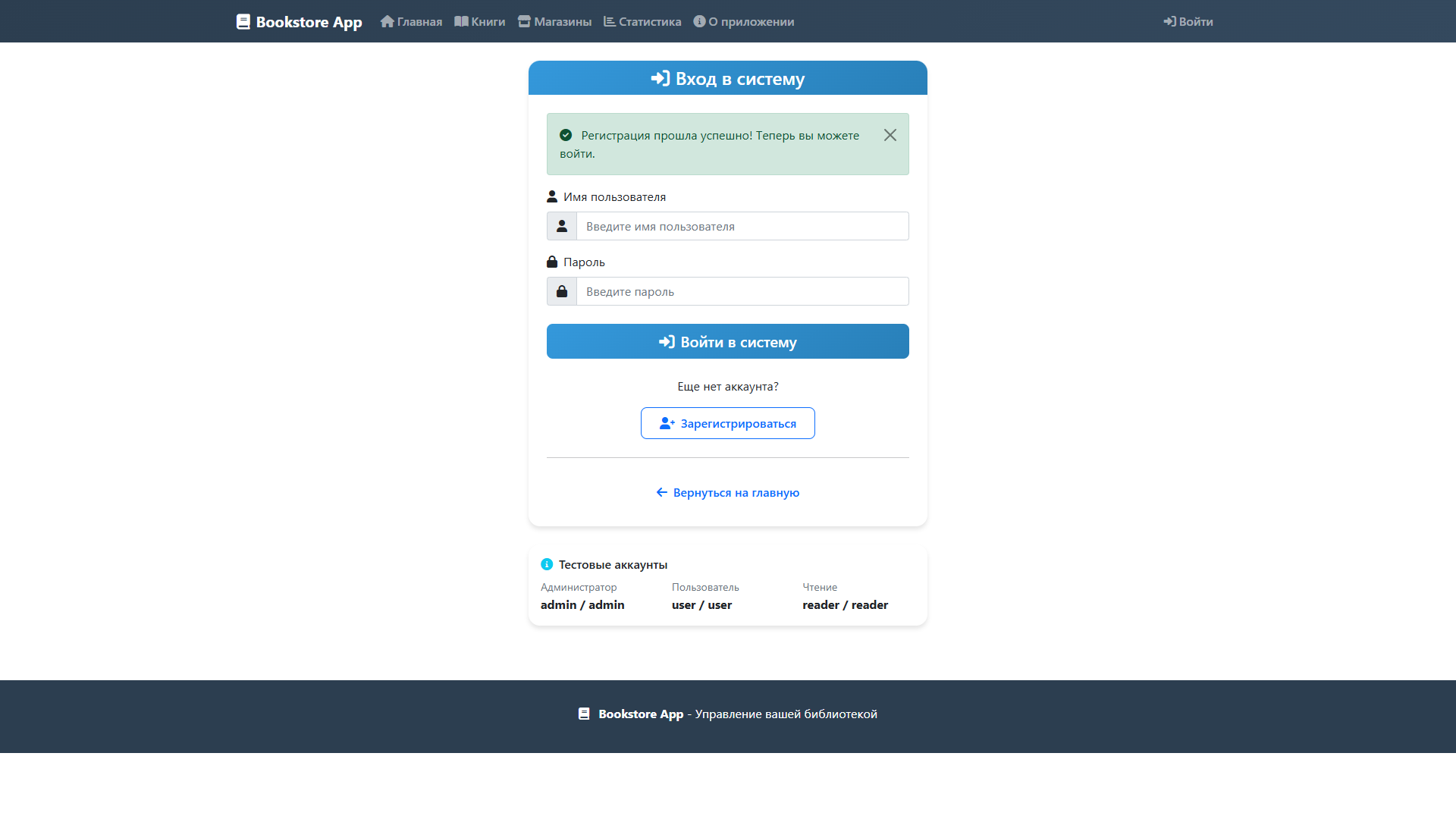


Рисунок 12 – Регистрация нового пользователя

Входим под учетной записью только что созданного пользователя – видим уровень доступа «READ\_ONLY», показано на рисунке 13.

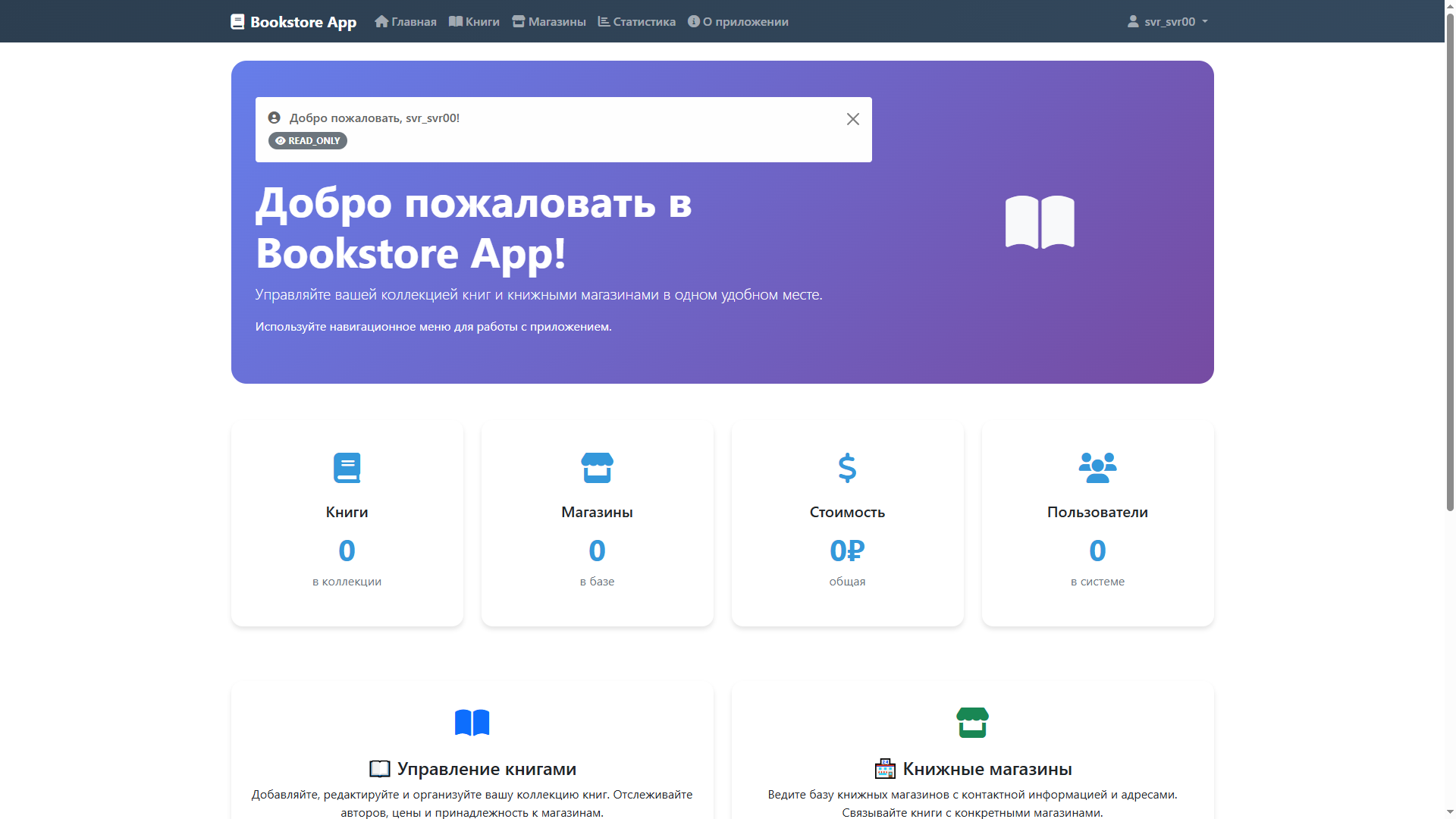


Рисунок 13 – Успешный вход под зарегестрированной учетной записью

Заходим под учетной записью администратора, переходим в раздел управления ролями; показан на рисунке 14.

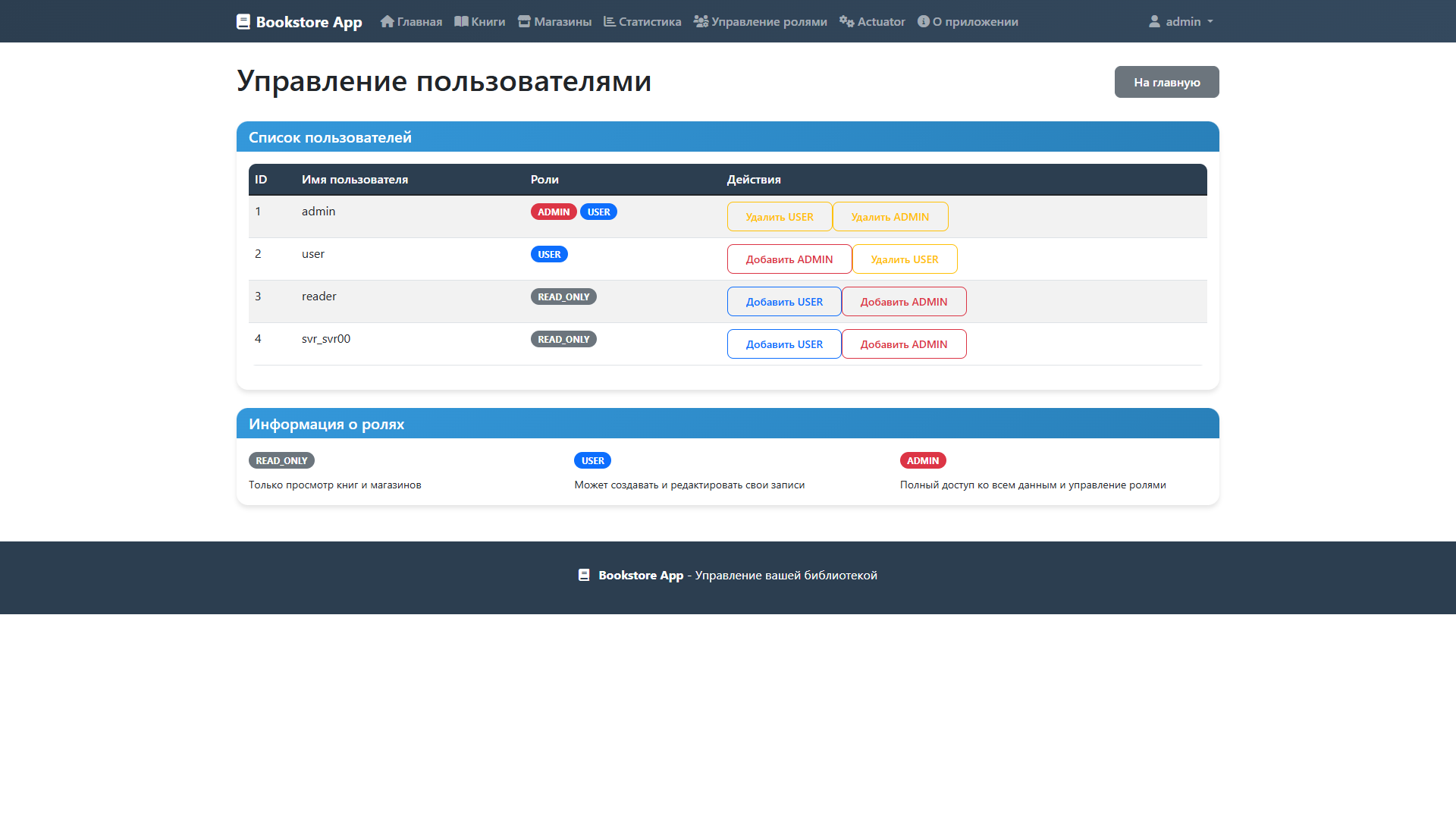


Рисунок 14 – Раздел управления ролями

Добавляем право USER для пользователя; показано на рисунке 15.

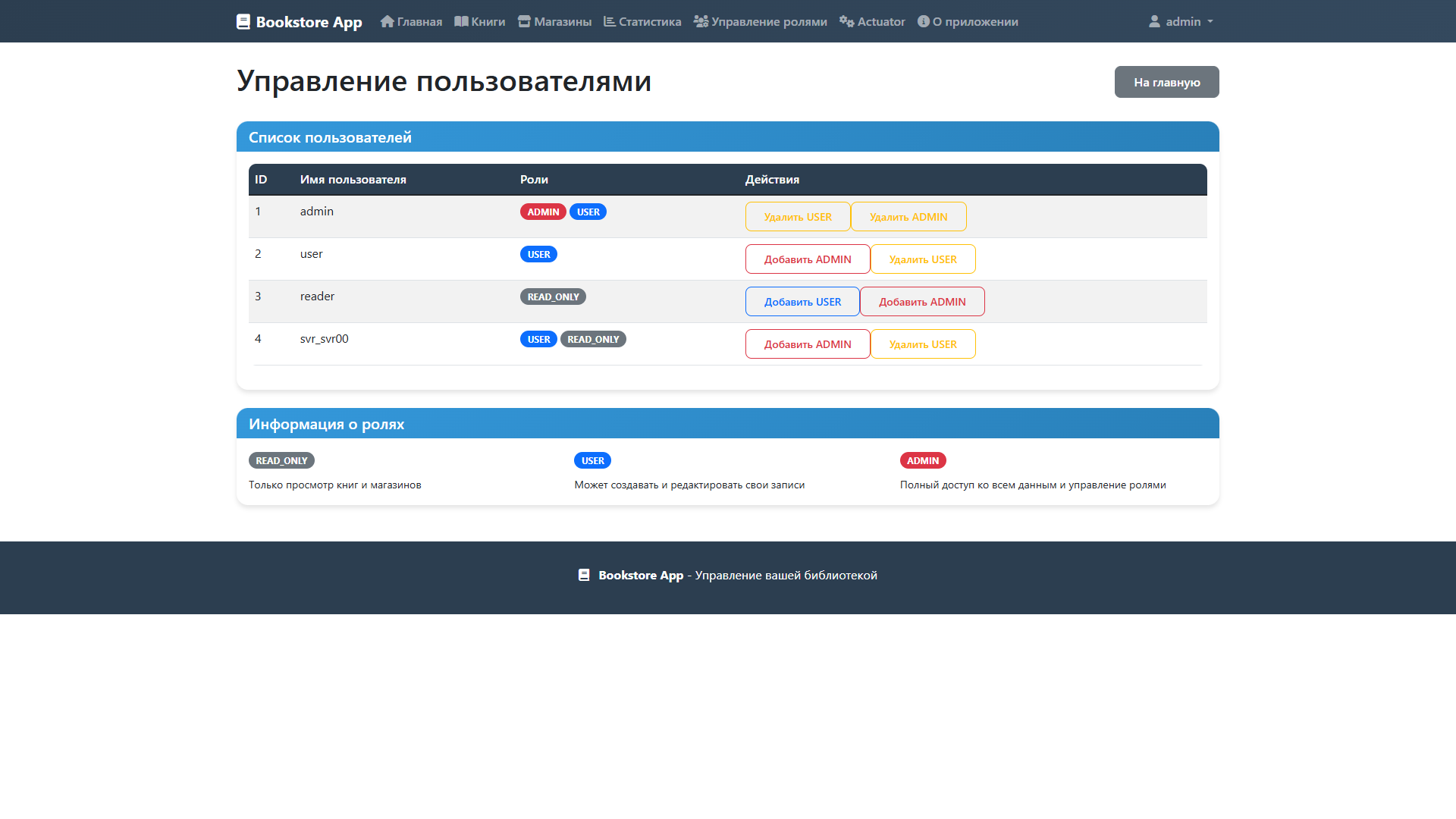


Рисунок 15 – Раздел управления ролями после добавления права

Входим снова под созданной недавно учетной записью, создаем новый книжный магазин и новую книгу; показано на рисунках 16…21.

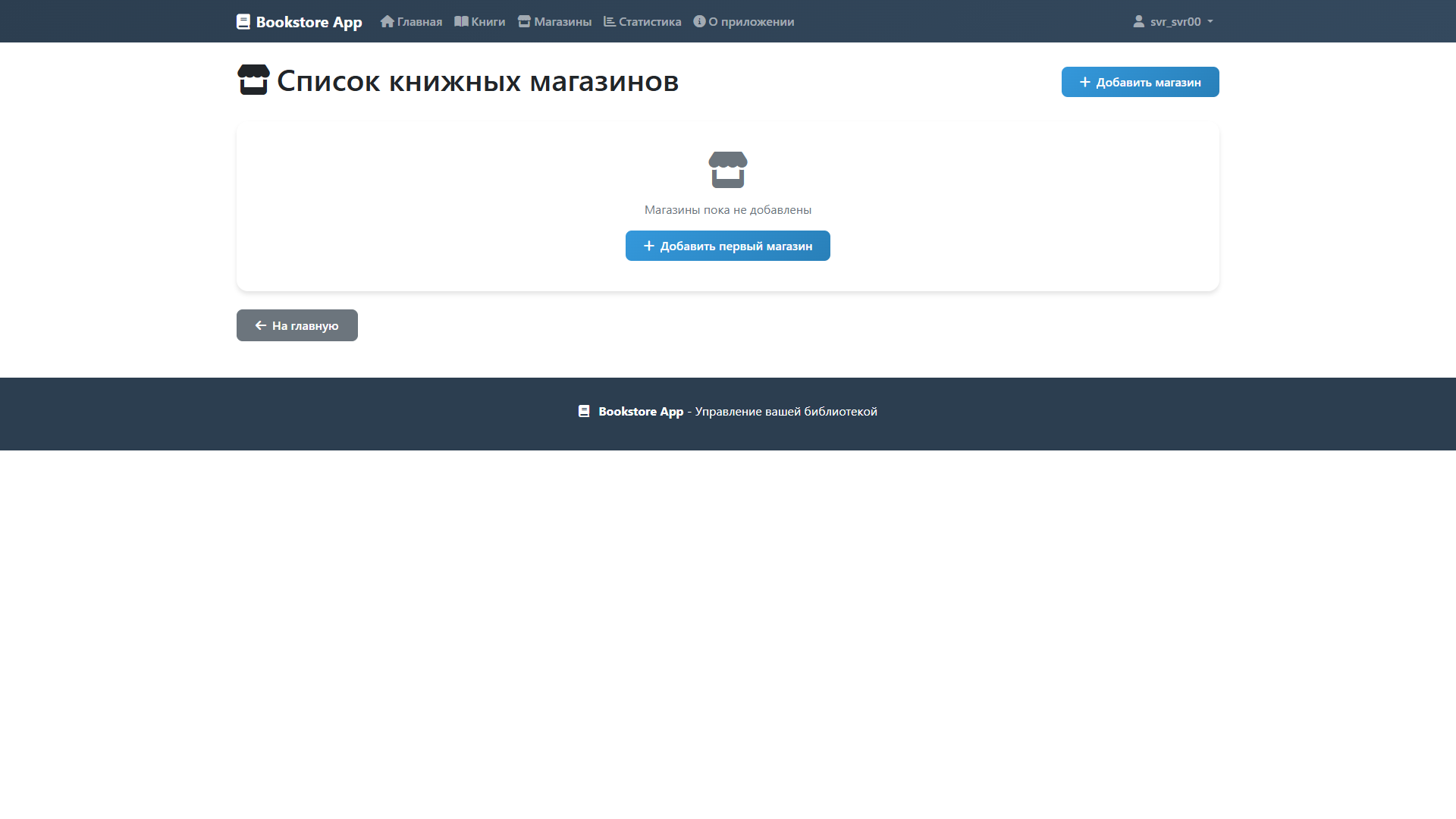


Рисунок 16 – Пустой список книжных магазинов



Рисунок 17 – Добавление нового магазина

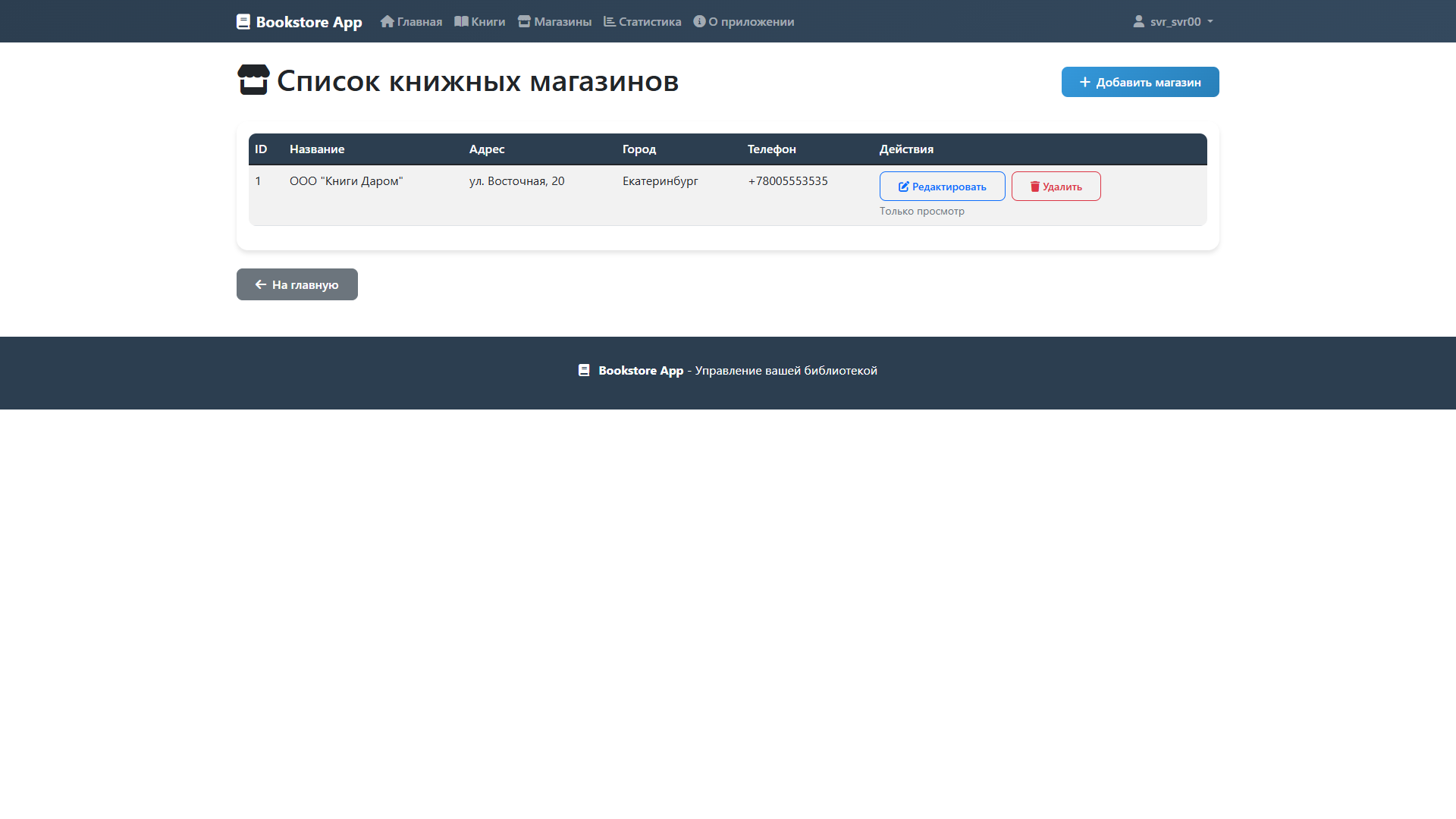


Рисунок 18 – Обновленный список магазинов

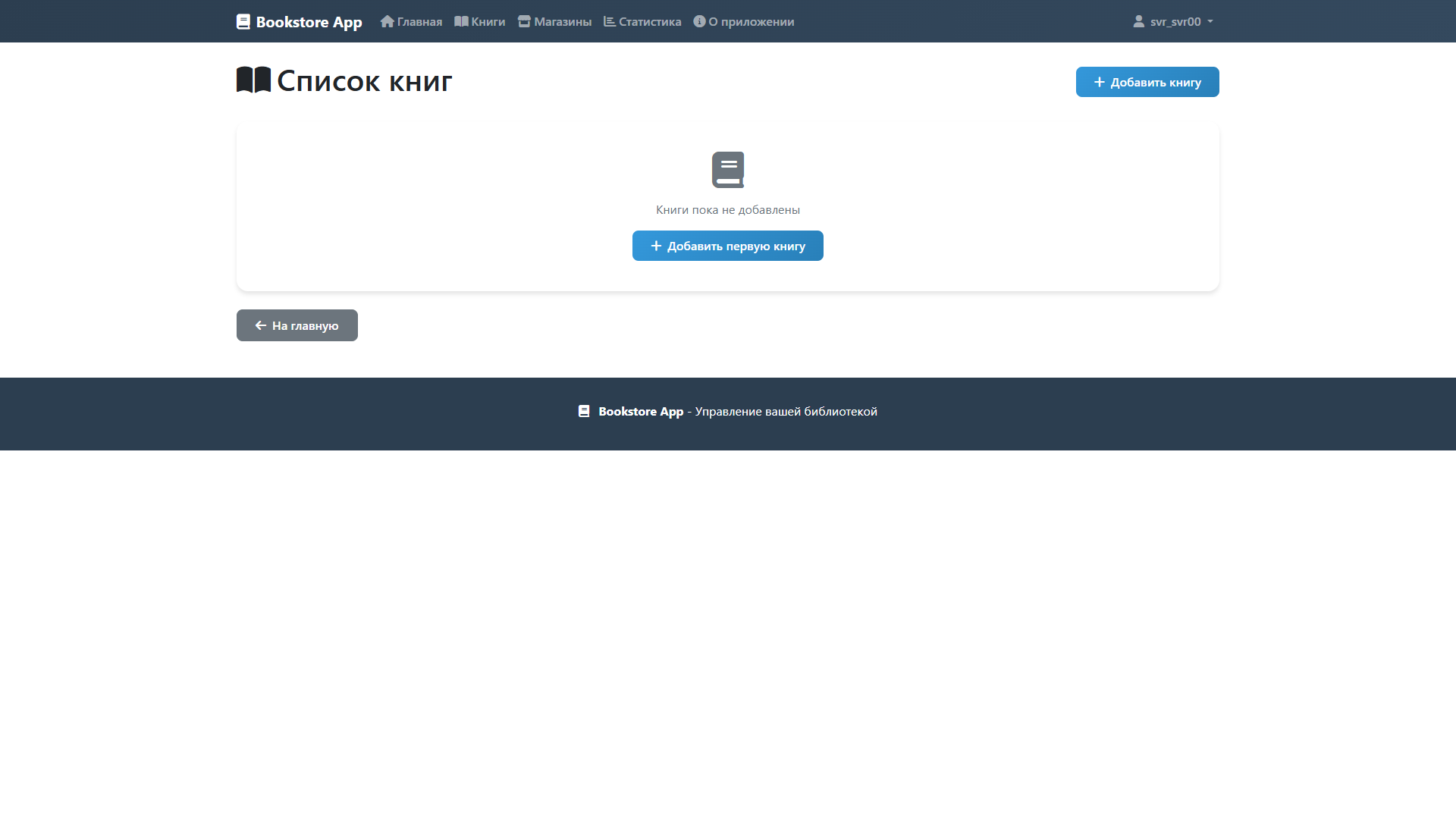


Рисунок 19 – Пустой список книг

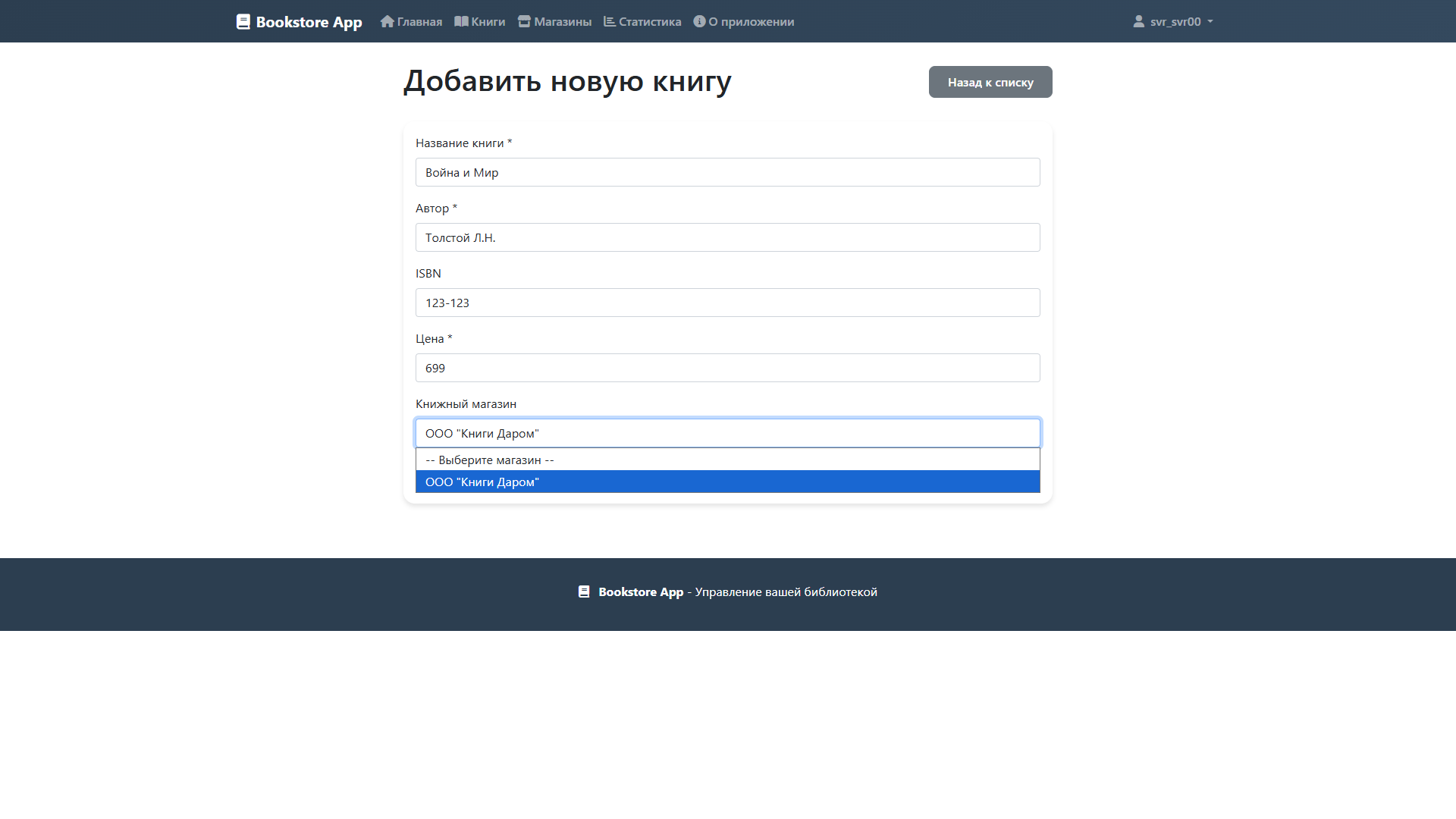


Рисунок 20 – Добавление новой книги

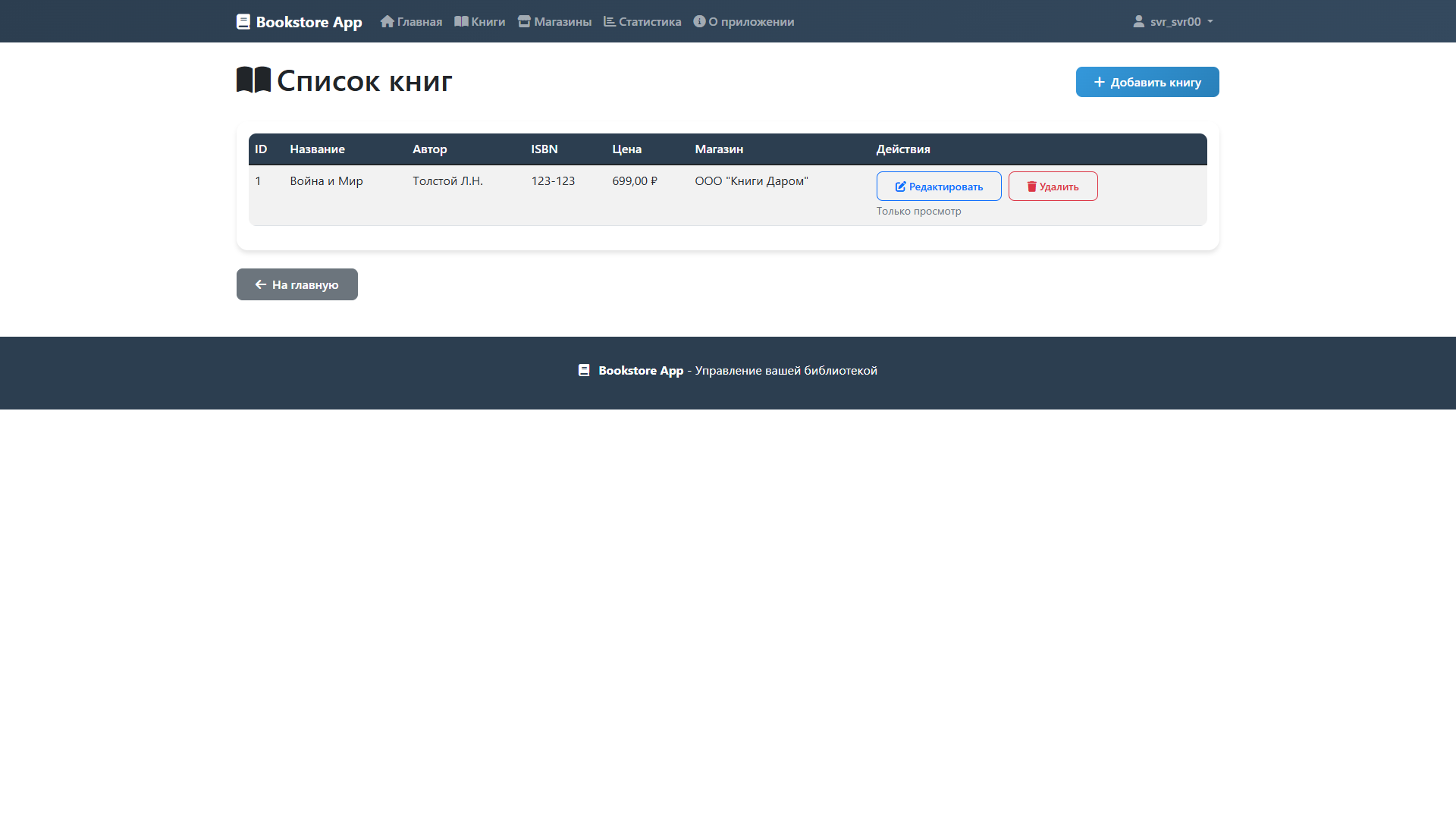


Рисунок 21 – Обновленный список книг

Теперь пробуем редактировать записи; показано на рисунках 22…25.

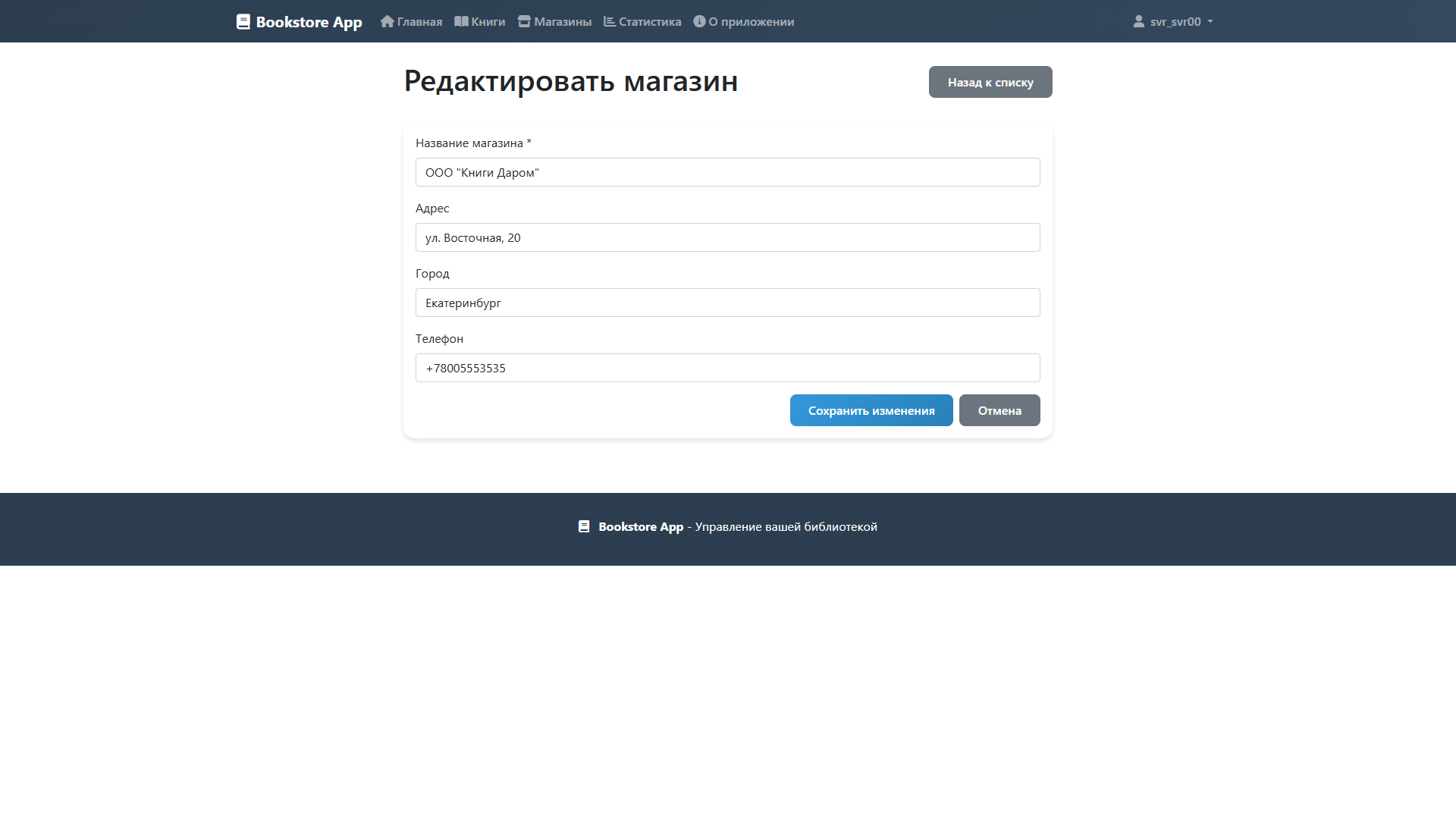


Рисунок 22 – Редактирование созданного магазина

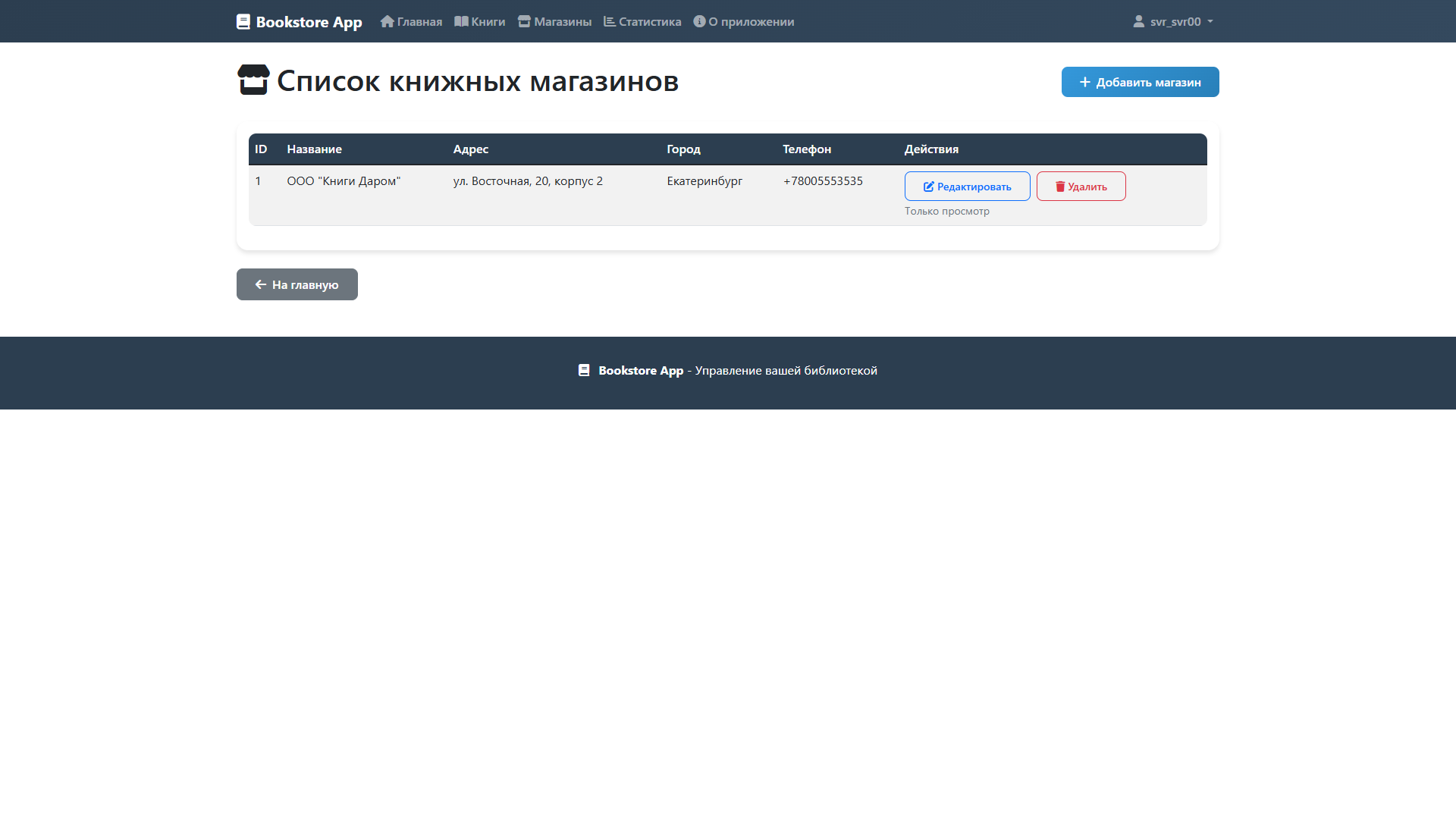


Рисунок 23 – Обновленный список магазинов

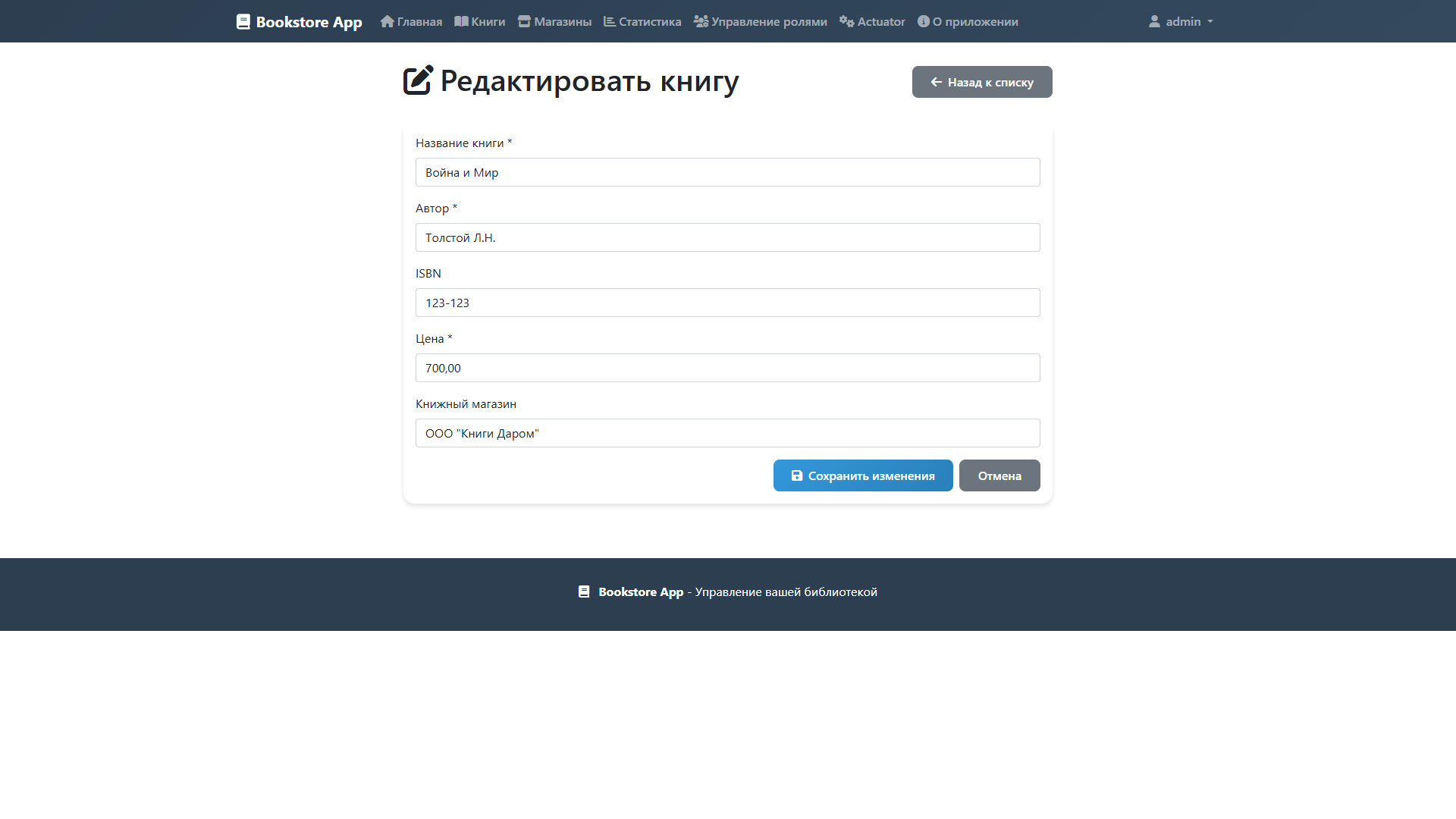


Рисунок 24 – Редактирование созданной книги

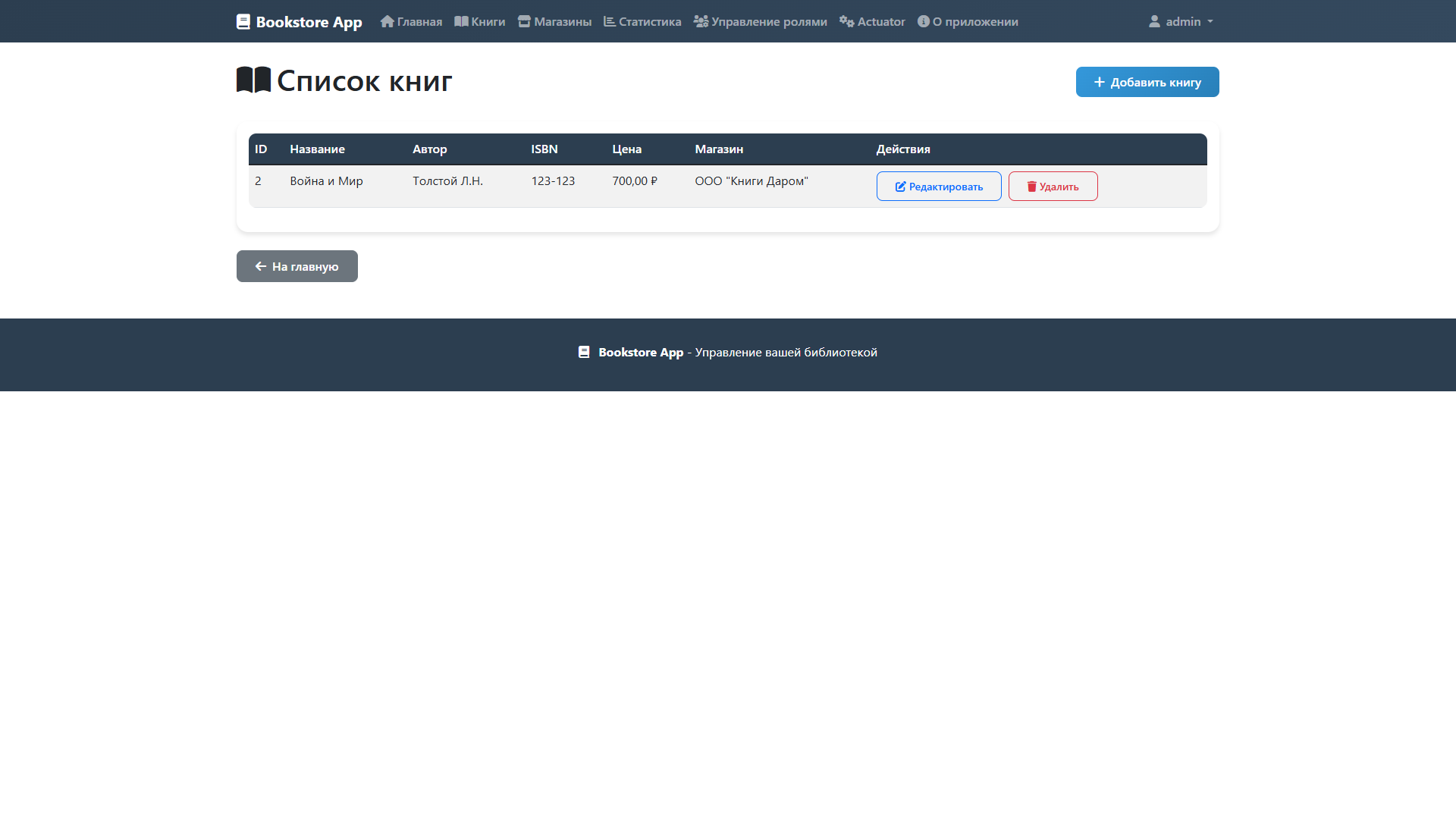


Рисунок 25 – Обновленный список книг

Теперь удалить книгу; показано на рисунках 26 и 27.

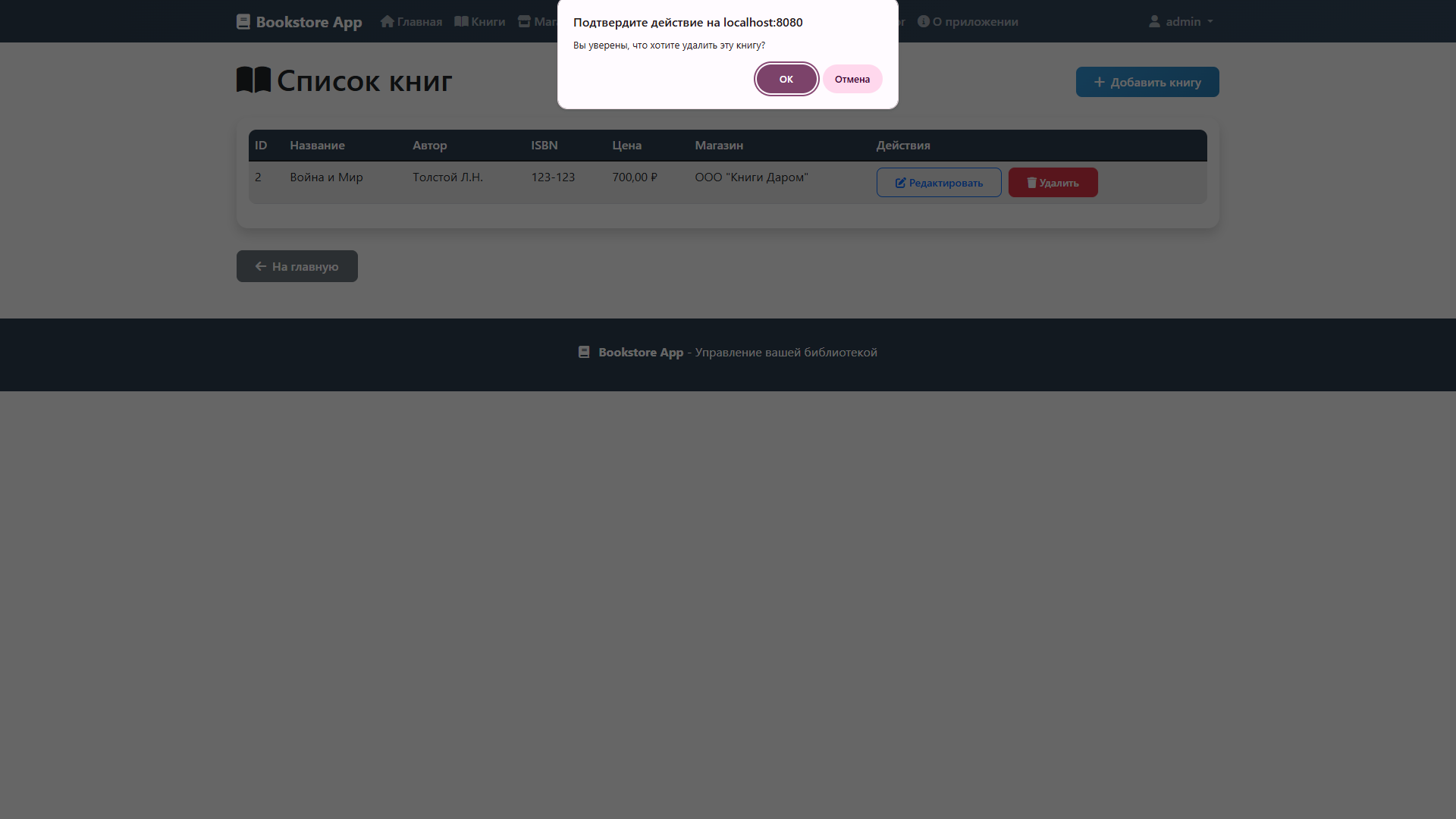


Рисунок 26 – Подтверждение удаления книги

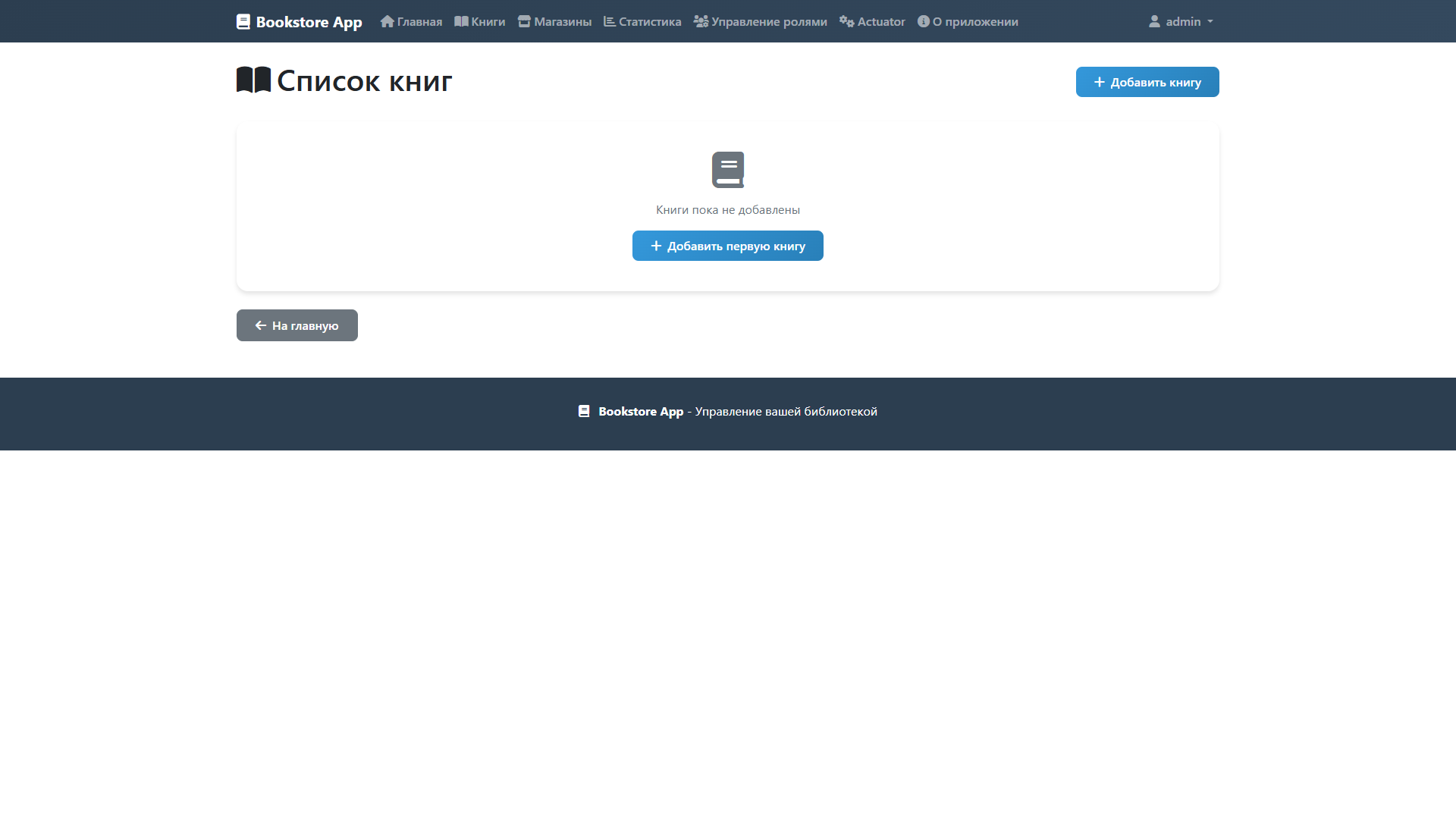


Рисунок 27 – Книга удалена

Создаем снова две книги; показано на рисунке 28.

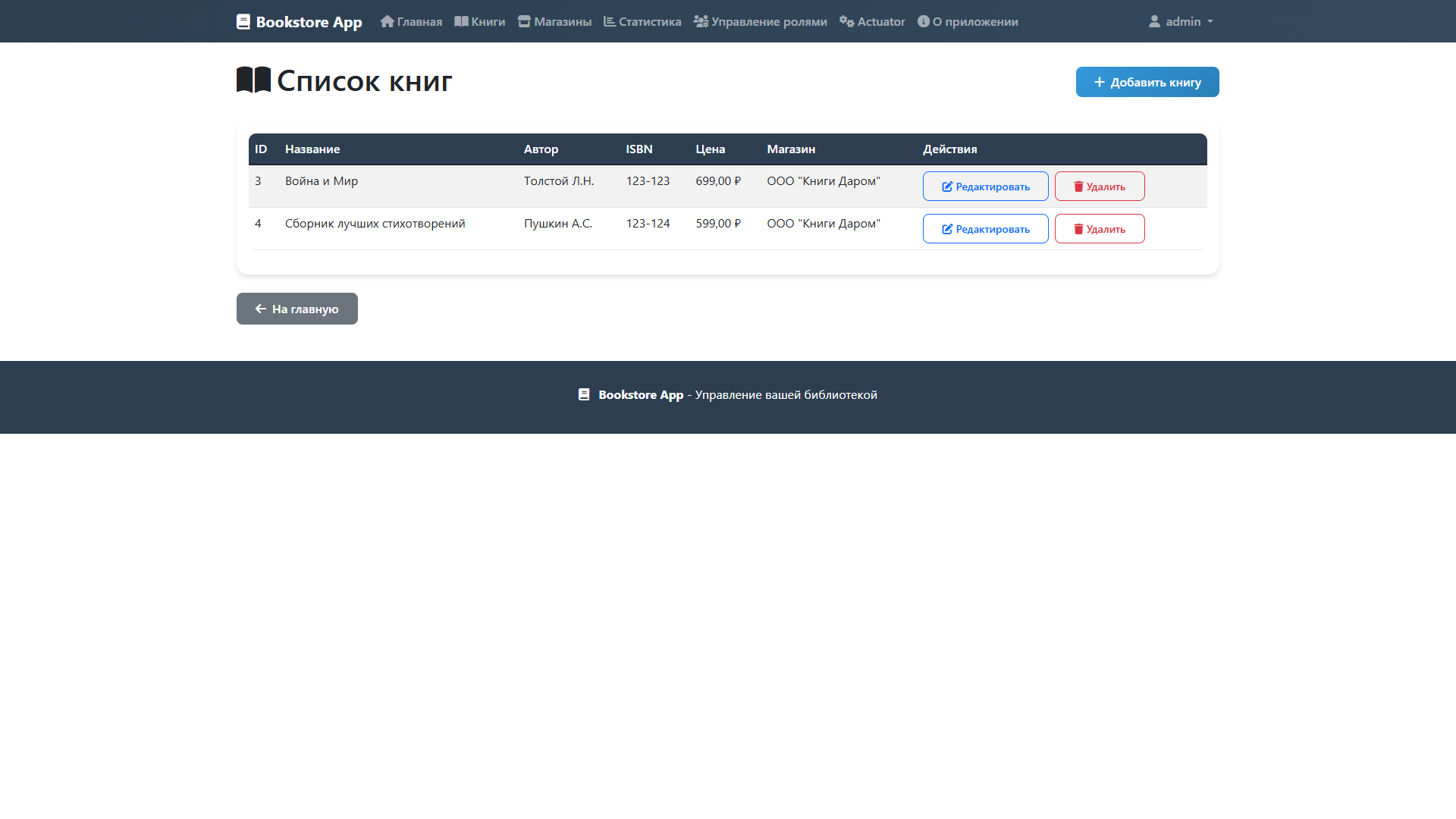


Рисунок 28 – Созданные две книги

Также смотрим на работу счетчиков на главной странице – показано на рисунке 29.

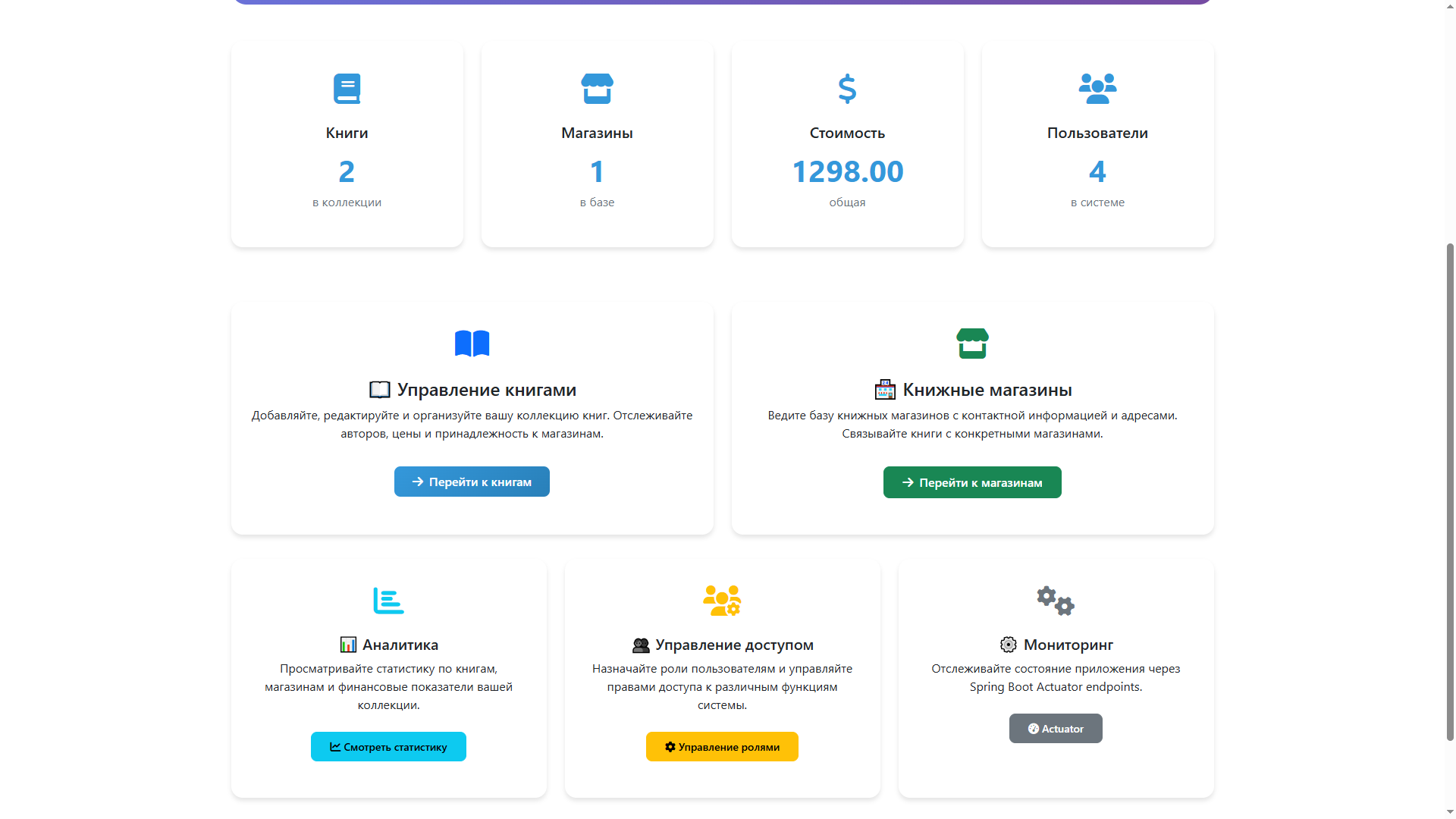


Рисунок 29 – Демонстрация работы счетчиков

Теперь перейдем на раздел «Статистика», его интерфейс и расчеты показаны на рисунке 30.

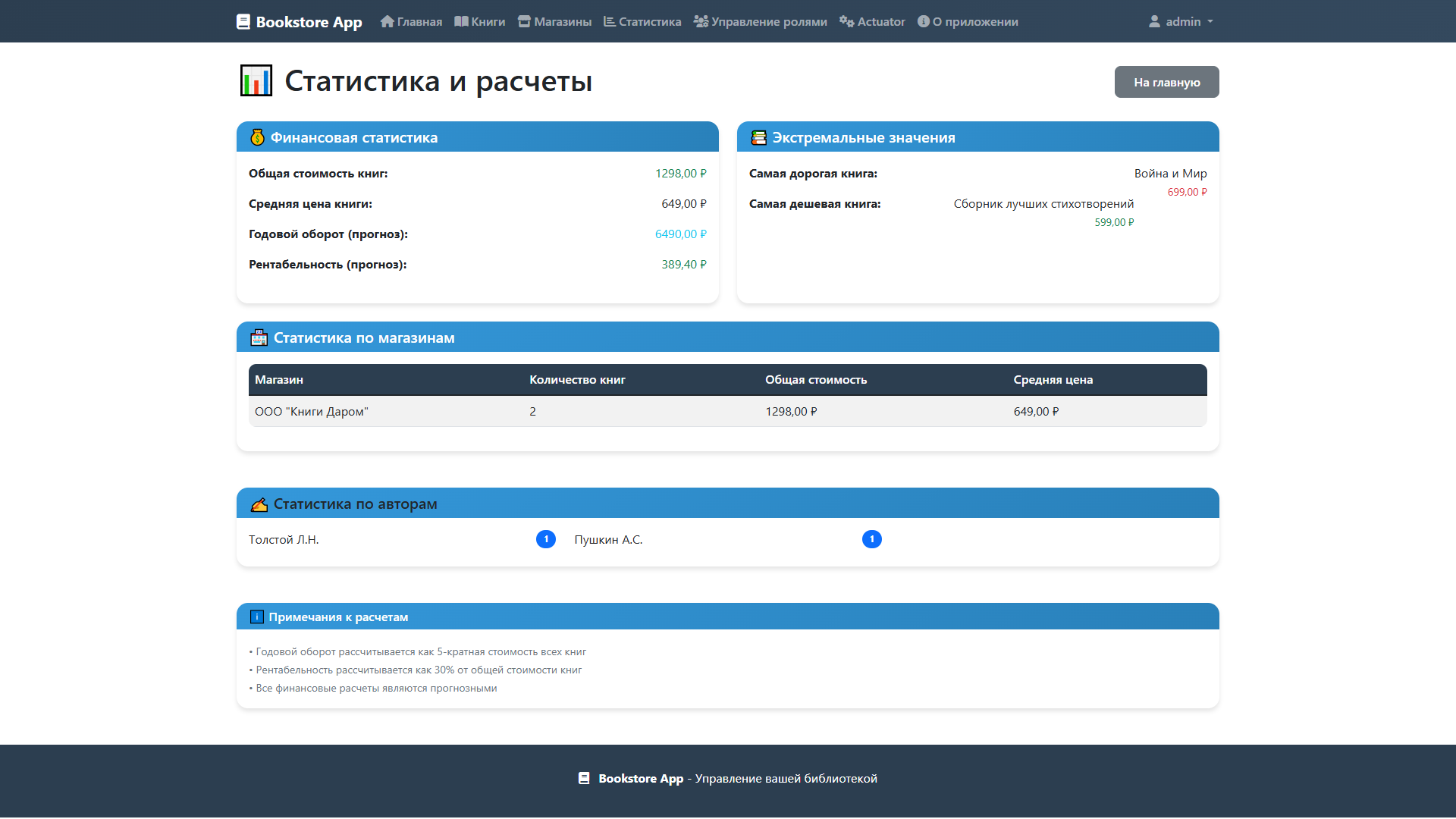


Рисунок 30 – Интерфейс и расчеты раздела «Статистика»

Выбираем раздел «Мониторинг», его интерфейс показан на рисунке 31.

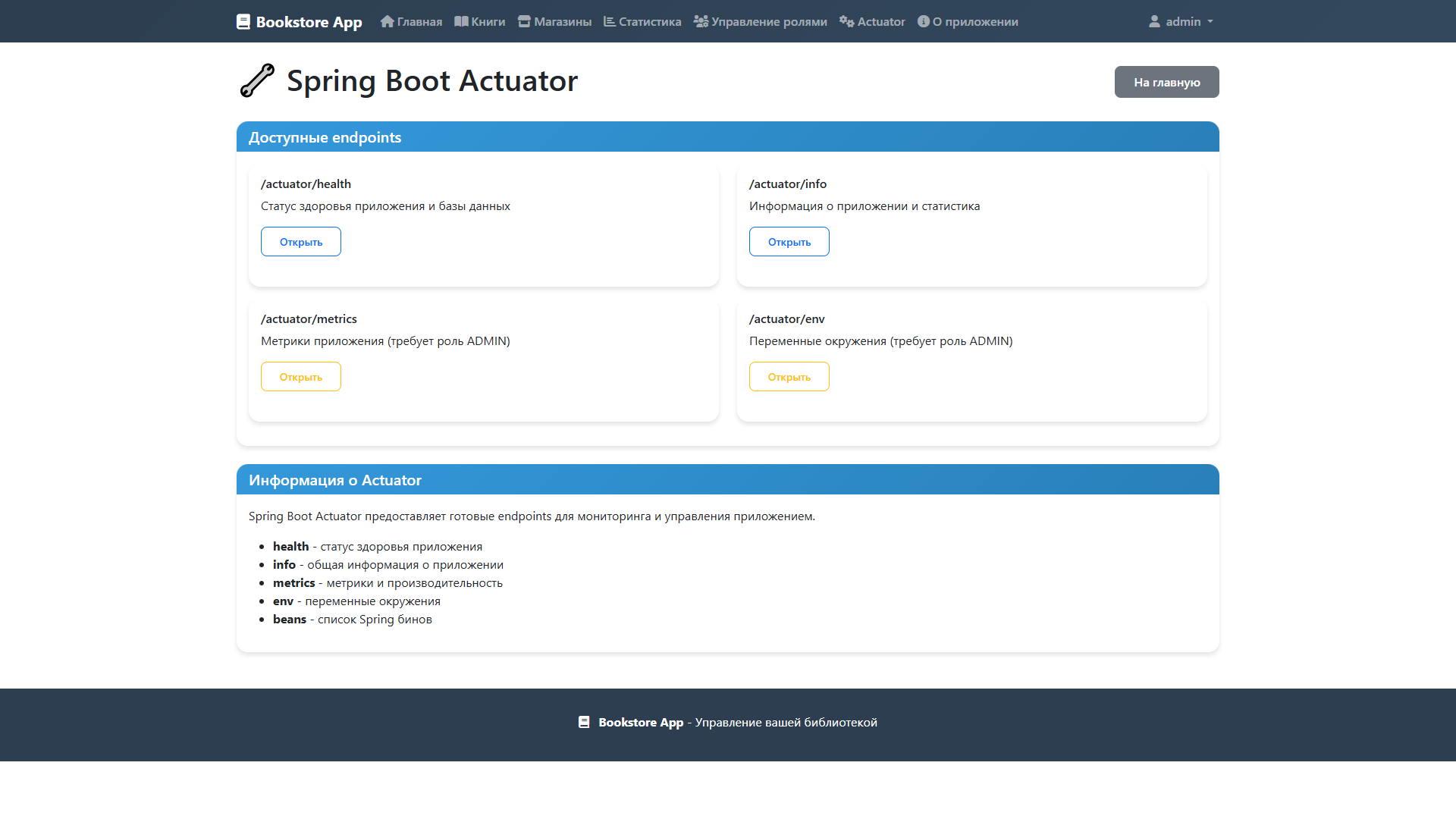


Рисунок 31 – Интерфейс раздела «Мониторинг»

Демонстрация раздела «Мониторинг» показана на рисунках 32…35.

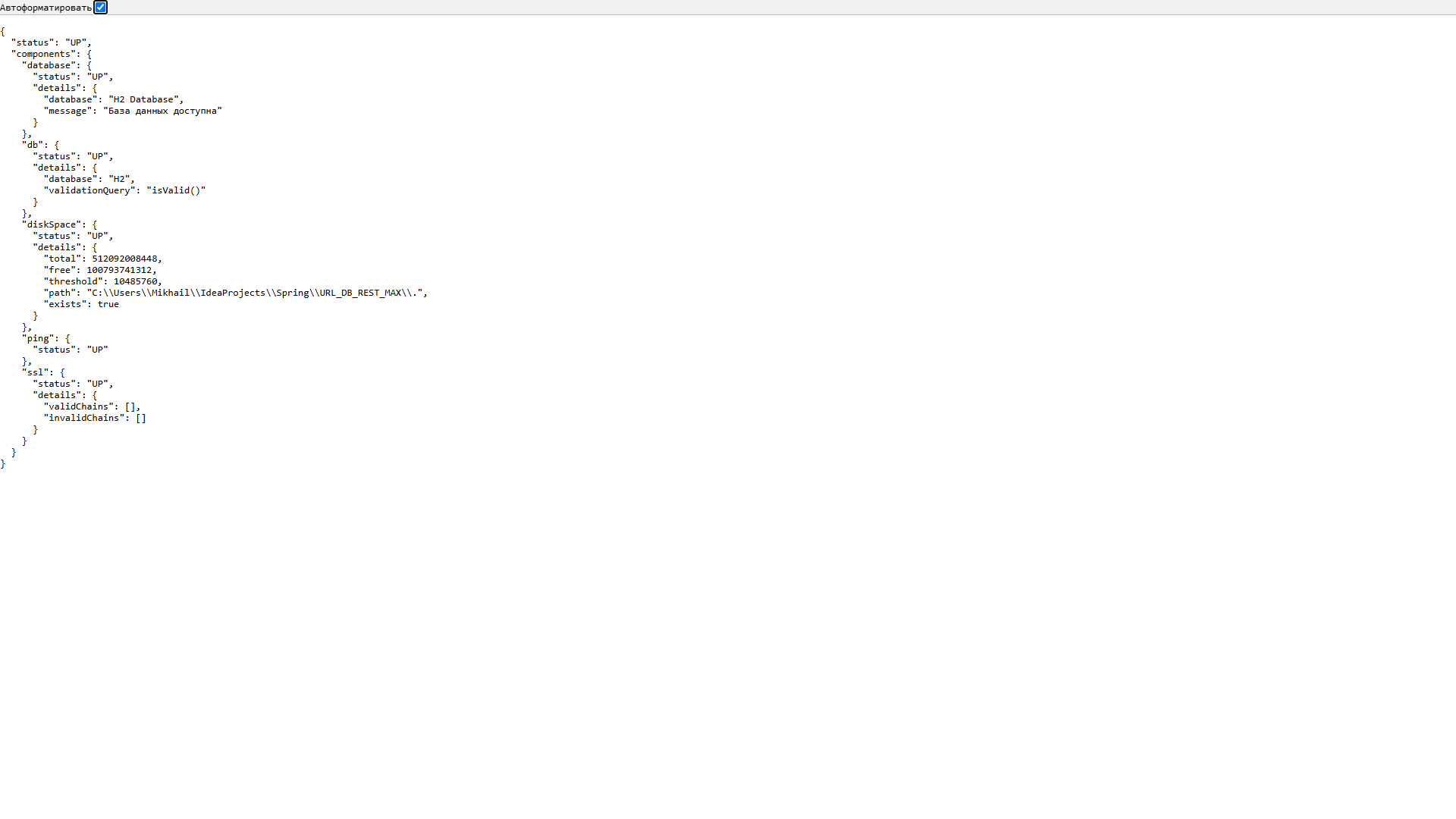


Рисунок 32 – Работа раздела «/actuator/health»

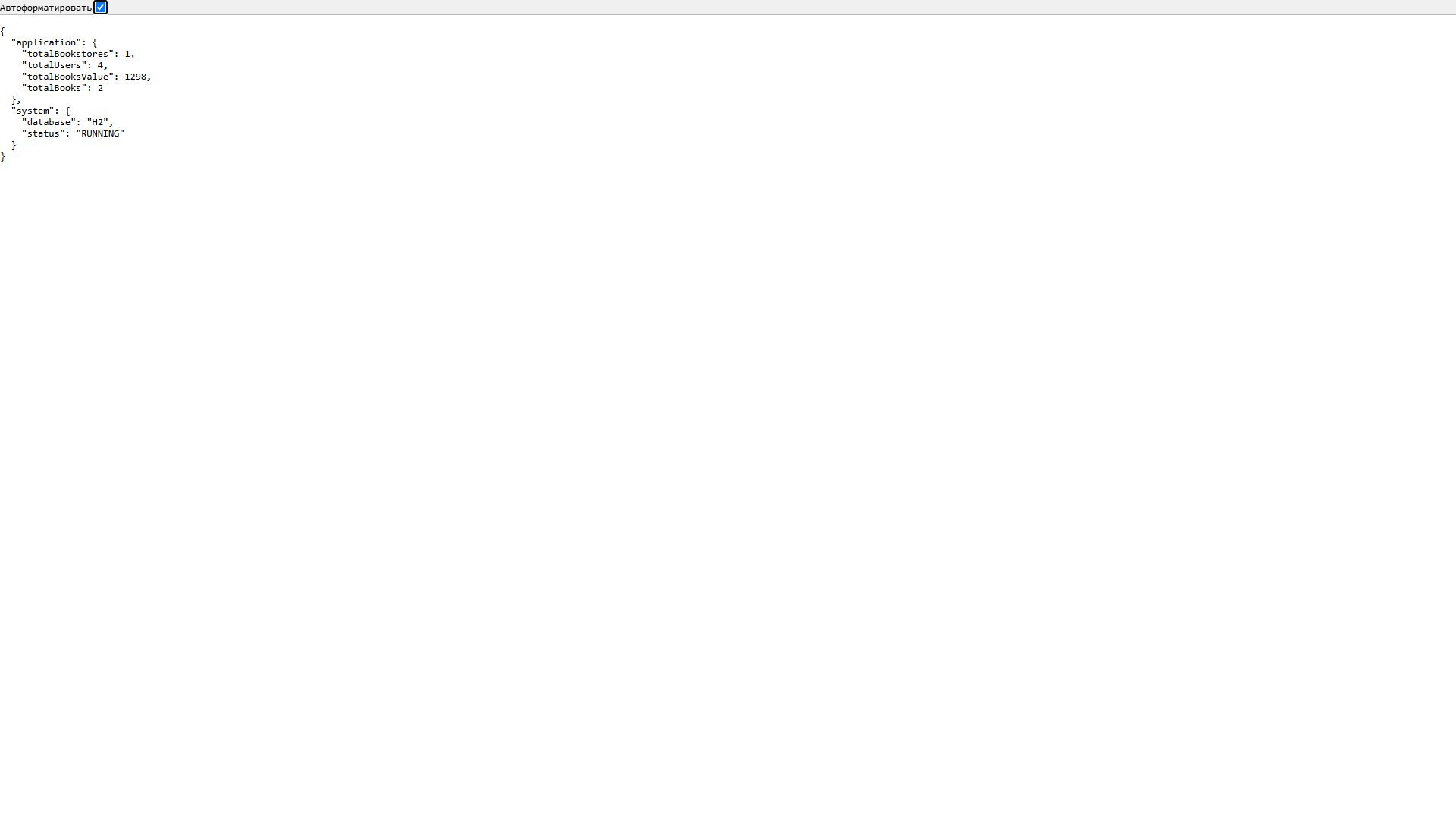


Рисунок 33 – Работа раздела «/actuator/info»



Рисунок 34 – Работа раздела «/actuator/metrics» (неформатированный вид)

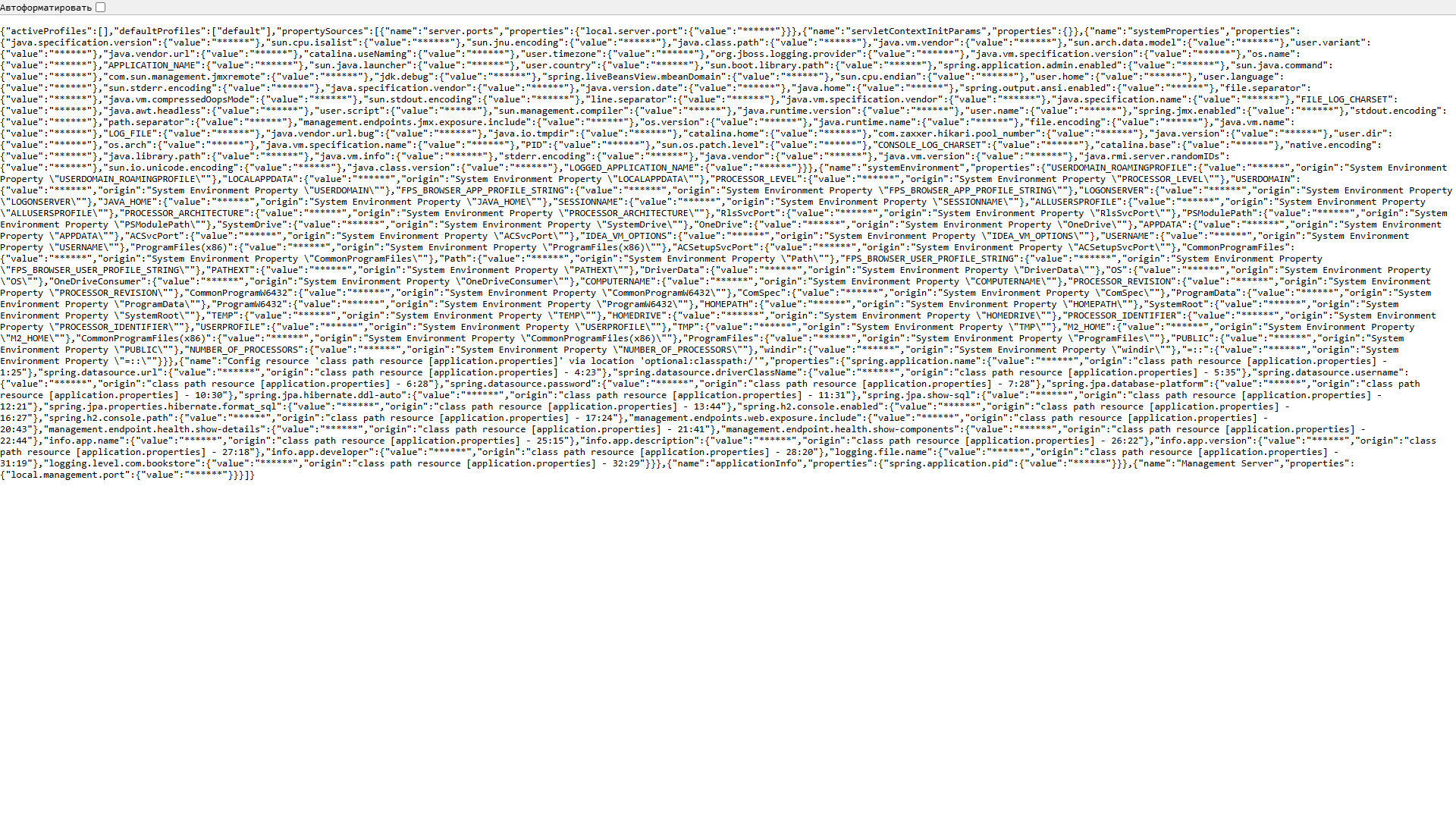


Рисунок 35 – Работа раздела «/actuator/env» (неформатированный вид)

Выбираем раздел «О приложении», его интерфейс и содержание показан на рисунке 36.

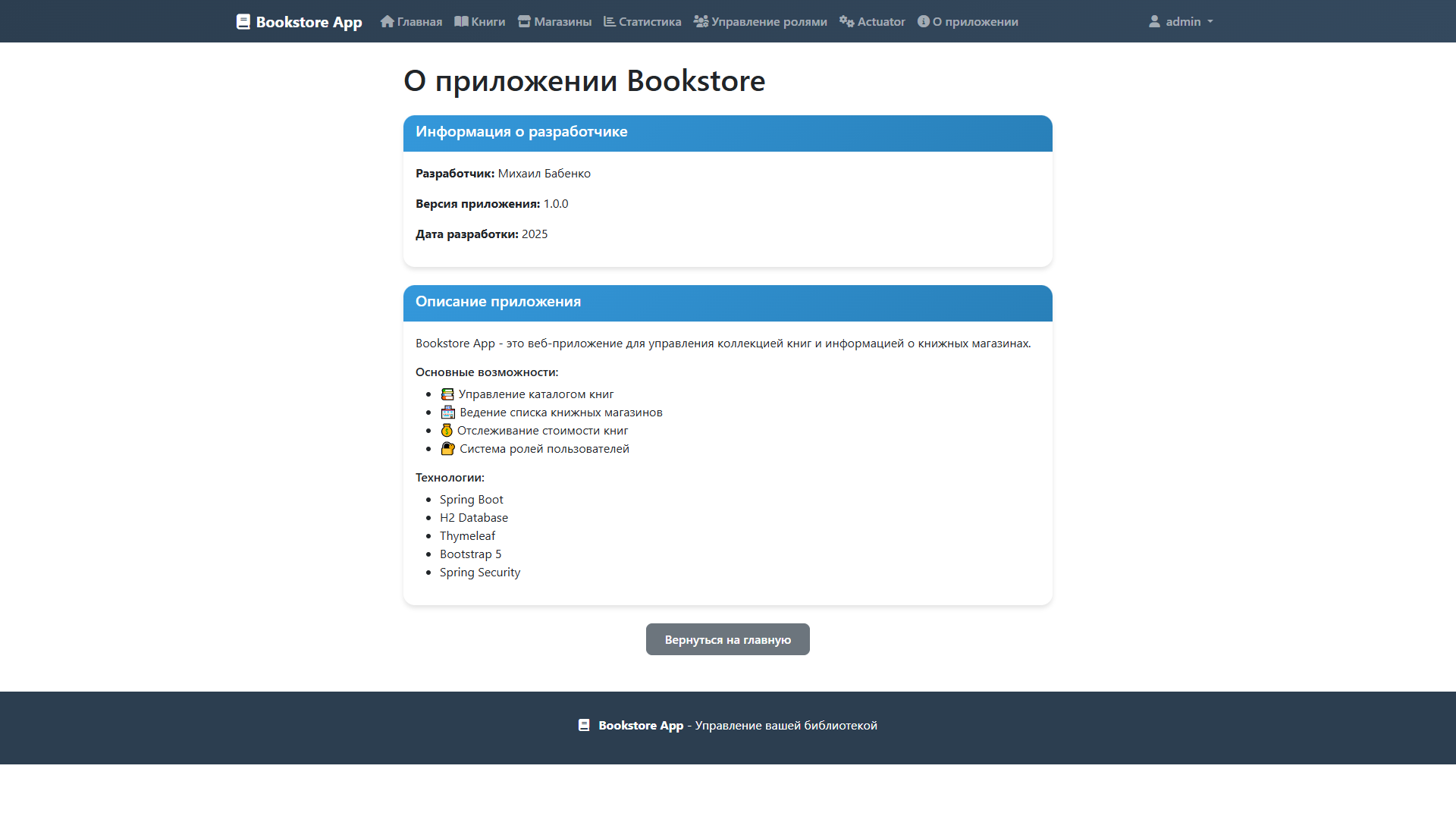


Рисунок 36 – Интерфейс и содержание раздела «О приложении»

Все элементы сайта работают корректно! Теперь нам нужно убедиться в правильной работе базы данных. Для этого перейдем в <http://localhost:8080/h2-console/>. Вводим: «User Name» – sa; «Password» – password. Мы успешно зашли в консоль H2! Теперь, для вывода содержимого базы данных, выполним следующий запрос: «SELECT \* FROM USERS; SELECT \* FROM BOOKS; SELECT \* FROM BOOKSTORES;». Были выведены созданные нами записи – значит, база данных работает корреткно! Результат выполнения запроса показан на рисунке 37.

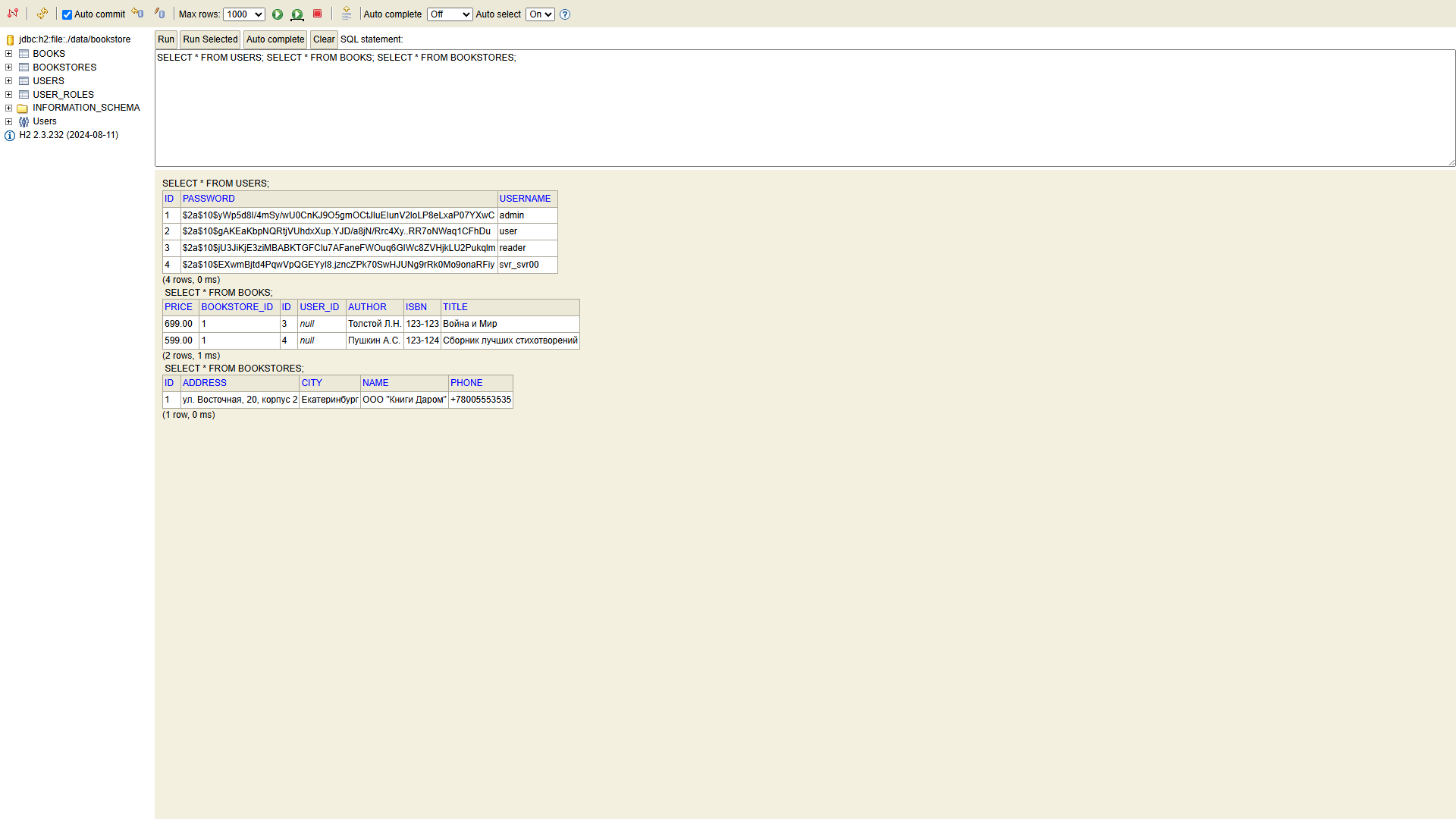


Рисунок 37 – Результат выполнения запроса

# Заключение

В рамках курсового проекта было разработано полнофункциональное веб-приложение «Bookstore App» для управления коллекцией книг и книжными магазинами. Приложение реализовано на основе современного стека технологий Java и предоставляет комплексное решение для учета книг, магазинов и пользователей с разграничением прав доступа.

Основные достижения и реализованные функции:

а) полноценная система аутентификации и авторизации;

б) система ролевого доступа;

в) управление данными (полное управление данными книг и магазинов);

г) расширенная функциональность (статистика и аналитика, финансовые отчеты, группировка данных, мониторинг через SB Actuator);

д) пользовательский интерфейс с использованием Bootstrap 5.

Все обязательные веб-формы реализованы, система ролей функционирует корректно, база данных работает исправно, реализовано логгирование в файл.

# Список используемых источников

1 Официальная документация Spring Boot. [Электронный ресурс] – Электрон. дан. – Режим доступа: https://spring.io/projects/spring-boot/

2 Документация по настройке аутентификации и авторизации. [Электронный ресурс] – Электрон. дан. – Режим доступа: https://spring.io/projects/spring-security/

3 Руководство по использованию Thymeleaf шаблонизатора. [Электронный ресурс] – Электрон. дан. – Режим доступа: https://www.thymeleaf.org/doc/tutorials/3.1/usingthymeleaf.html

4 Документация по фреймворку Bootstrap 5. [Электронный ресурс] – Электрон. дан. – Режим доступа: https://getbootstrap.com/docs/5.1/getting-started/introduction/

5 Документация по встроенной базе данных H2. [Электронный ресурс] – Электрон. дан. – Режим доступа: http://www.h2database.com/html/main.html

6 Практические руководства по Spring Boot. [Электронный ресурс] – Электрон. дан. – Режим доступа: https://www.baeldung.com/spring-boot/

7 Статья по настройке и использованию Actuator. [Электронный ресурс] – Электрон. дан. – Режим доступа: https://www.baeldung.com/spring-boot-actuators

8 Материалы по работе с JPA и Hibernate. [Электронный ресурс] – Электрон. дан. – Режим доступа: https://www.baeldung.com/learn-jpa-hibernate/

9 Документация Java Standard Edition. [Электронный ресурс] – Электрон. дан. – Режим доступа: https://docs.oracle.com/en/java/javase/17/

10 Руководство по сборке проектов с помощью Maven. [Электронный ресурс] – Электрон. дан. – Режим доступа: https://maven.apache.org/guides/

### Приложение 1 Ссылка на репозиторий GitHub

<https://github.com/sverdlovsky00/SpringB_LW>

### Приложение 2 Заключение онлайн проверки на антиплагиат

