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

Департамент анализа данных и машинного обучения

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

«Информационно справочная система издательского дома»

Выполнил:
Студент группы ИД23-1
Маслов А. Н.

Научный руководитель:
к.э.н., доцент
Гринева Н. В.

Москва – 2025

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Стремительное развитие информационных технологий и активная цифровизация всех сфер деятельности привели к существенному изменению процессов взаимодействия организаций с пользователями. Особенно это заметно в книжной индустрии, где традиционные механизмы распространения и предоставления информации о литературе замещаются современными веб-ориентированными решениями.

Издательские дома сталкиваются с необходимостью своевременно информировать читателей о выходе новых книг, упорядочивать данные о каталогах, обеспечивать доступ к электронным версиям описаний, расширять взаимодействие с клиентами и автоматизировать процессы внутреннего документооборота. Отсутствие удобных инструментов для поиска, фильтрации, уведомления о выходе новых изданий приводит к снижению вовлечённости аудитории и потерям потенциальных продаж.

Исследования последних лет подтверждают: цифровизация библиотечных сервисов значительно повышает удобство пользователей и эффективность самого информационного обслуживания. Так, в статье Recommendation Systems in Libraries: an Application with Heterogeneous Data Sources в 2023 году [12] описан подход к реализации рекомендательной системы на основе анализа пользовательских данных – что показывает: при грамотной организации цифровой библиотеки можно улучшить опыт пользователей и повысить качество рекомендаций.

Другое исследование – Evaluating Digital Library Search Systems by using Formal Process Modelling тоже 2023 год [13] – рассматривает эффективность систем поиска в цифровых библиотеках, показывая, что правильно спроектированная цифровая система действительно повышает успешность и скорость нахождения нужной информации.

В этих условиях информационно-справочные системы становятся ключевым элементом цифровой инфраструктуры издательств. Они обеспечивают:

- оперативный доступ к базе данных книг, авторов, жанров и серий;
- удобный интерфейс для читателей и сотрудников;
- автоматизированную рассылку уведомлений о новинках;
- хранение и структуризацию данных;
- механизмы регистрации и персонализации взаимодействия.

Актуальность разработки подобной системы обусловлена необходимостью сокращения временных затрат пользователей на поиск информации, улучшения сервиса и повышения конкурентоспособности издательских домов на рынке.

Объектом исследования является информационное обеспечение деятельности издательского дома.

Предметом исследования является веб-ориентированная информационно-справочная система издательского дома.

В связи с этим, целью курсового проекта является разработка программного обеспечения, обеспечивающего удобный доступ к данным каталога и автоматизирующего взаимодействие издательства с пользователями.

Для достижения цели были поставлены следующие задачи:

1. Проанализировать предметную область и определить требования к системе;
2. Исследовать существующие решения и провести сравнительный анализ;
3. Спроектировать архитектуру и структуру базы данных;
4. Реализовать клиентскую, серверную и системную логику;
5. Обеспечить безопасность и обработку ошибок;
6. Настроить автоматическое оповещение пользователей;
7. Провести тестирование и отладку системы;

ГЛАВА 1 ПРЕДПРОЕКТНОЕ ИСЛЕДОВАНИЕ

1.1 Описание предметной области

Деятельность издательского дома включает широкий спектр процессов: от обработки литературных произведений до их публикации и распространения. Одной из важных задач является предоставление читателям своевременной и структурированной информации о книгах, авторах, жанрах и сериях.

В традиционном виде информационное взаимодействие происходило через печатные каталоги, буклеты, стенды в магазинах, рассылку физических писем и сайты без сложной структурной части. Это приводило к следующим проблемам:

- сложность поиска интересующих книг;
- отсутствие своевременной информации о новых релизах;
- невозможность персонализации;
- отсутствие обратной связи с пользователями;
- дублирование данных и ошибки при ручной обработке.

В современных условиях цифровая трансформация является необходимостью для любого издательства. Современная информационно-справочная система должна обеспечивать:

- централизованное хранение данных о книгах;
- формирование структурированного каталога;
- предоставление удобного поиска и фильтрации;
- авторизацию пользователей и ведение их профилей;
- механизм формирования списка желаний;
- автоматическую рассылку уведомлений о выходе интересующих книг.

Таким образом, разработка веб-системы для издательского дома способствует оптимизации процессов и повышению качества взаимодействия с читателями.

1.2 Сравнительный анализ аналогов

В процессе анализа предметной области и определения требований к будущей системе важно рассмотреть существующие цифровые решения, применяемые в сфере книжной индустрии.

ЛитРес [14] – крупнейшая в России платформа по продаже и распространению электронных и аудиокниг. Он предоставляет обширный каталог литературы, охватывающий множество жанров, и предлагает удобные фильтры, подборки, работу с текстовыми и аудиоформатами. Это делает его популярным ресурсом для читателей.

Однако, хотя ЛитРес хорошо подходит для массового распространения и чтения книг, он не предназначен как специализированная справочная система для конкретного издательства. То есть, функциональность ЛитРес ориентирована на продажу и чтение, а не на административное управление и персонализированную справочно-информационную систему.

АСТ [15] – одно из крупнейших российских издательств. На сайте АСТ представлен официальный каталог книг: доступны категории “новинки”, “бестселлеры”, подборки, есть структурированное представление книг и возможность просмотра информации о названиях, жанрах, и др.

Тем не менее, сайт АСТ представляет собой ресурс больше для продажи книг, а не информационную систему со всем набором необходимых функций, как например возможность добавить книгу в список желаемого.

Издательский дом Мещерякова [16] – издательство, специализирующееся на детской литературе, переизданиях классики и современных изданиях. На их сайте представлен каталог книг, разнообразные жанры, и при этом есть форма подписки на рассылку новостей, акций и специальных предложений.

Проанализировав особенности имеющихся решений, можно заметить, что каждый из рассмотренных ресурсов (ЛитРес, АСТ и Издательский дом

Мещерякова) выполняет свою задачу и предоставляет пользователям определённый набор функций. Однако ни одна из платформ не содержит в полной мере тех возможностей, которые необходимы для построения гибкой информационно-справочной системы, ориентированной именно на персонализированное взаимодействие читателя и издательства.

Исходя из выявленных особенностей, можно сделать вывод, что разрабатываемая система должна включать: удобное визуальное оформление, расширенные инструменты поиска и фильтрации книг, возможность формирования персонального списка интересов, а также средства своевременного уведомления пользователей о новых поступлениях. Такой функционал позволит обеспечить более тесную связь между читателем и издательством и создать удобный сервис, учитывающий индивидуальные потребности пользователей.

1.3 Постановка задачи

Входной информацией для системы будут являтся запросы пользователя в виде переходов по страницам, поисковых запросов, фильтров и действий в интерфейсе.

Выходной информацией являются отображаемые данные каталога, результаты поиска, карточки книг, автора, жанра, серии, уведомления и действия системы (например, отправка email-сообщений).

Функциональные требования к приложению:

- отображение каталога книг, доступных в системе;
- возможность поиска книг по названию, описанию, автору или ISBN и другим параметрам;
- возможность фильтрации книг по жанрам, сериям, языку и статусу выхода;
- вывод подробной информации о книге (описание, дата выхода, авторы, жанры и др.), авторе, жанре и серии;
- регистрация и авторизация пользователей;
- добавление книг в список желаний и его просмотр;

- автоматическая отправка уведомлений о выходе книг из списка желаний;
- возможность администратора добавлять, редактировать и удалять книги, авторов, жанры и серии.

Эксплуатационные требования к приложению:

- однопользовательская веб-сессия (каждый пользователь работает со своей учётной записью);
- простой и интуитивно понятный интерфейс;
- корректная работа в современных браузерах (Chrome, Firefox, Edge, Safari);
- работа сервера на ОС Windows или Linux;
- процессор: 2.0 GHz+,
- оперативная память минимум 1 GB RAM и выше,
- место на диске 200 MB для приложения и базы данных;
- использование стандартных веб-технологий, не требующих установки клиентских программ;
- корректная работа при стабильном интернет-подключении.

1.4 Характеристика инструментальных средств разработки

Для реализации программного обеспечения был выбран язык программирования Java [24] – современный объектно-ориентированный язык, широко применяемый для разработки веб- и серверных систем. Java отличается стабильностью, безопасностью, кроссплатформенностью и поддержкой широкого спектра библиотек, что делает её удобным выбором для разработки информационно-справочной системы.

Основой серверной части является Spring Boot [17 - 18] – фреймворк, предназначенный для быстрой разработки веб-приложений благодаря автоматической конфигурации и модульной структуре. Он включает Spring MVC для построения веб-слоя, Spring Data JPA и Hibernate для работы с базой данных, а также Spring Security для реализации авторизации и защиты приложения. В качестве альтернатив рассматривались JDBC [28] и чистая

работа с SQL, однако использование ORM (Hibernate) было выбрано как более безопасный и удобный путь, защищающий от SQL-инъекций и уменьшающий объём ручного кода.

В качестве системы управления базами данных была выбрана PostgreSQL[23], однако в процессе анализа рассматривалась и MySQL[30], которая также широко используется в веб-разработке. PostgreSQL был выбран благодаря большей гибкости, поддержке расширенных типов данных, устойчивости к нагрузкам и лучшей совместимости с современными ORM-библиотеками. MySQL остаётся популярным решением, но PostgreSQL лучше подходит для сложных структур данных, используемых в информационных системах.

Для описания сущностей и сокращения объёма кода применяется библиотека Lombok, автоматически генерирующая геттеры, сеттеры и другие служебные методы. Для реализации автоматических фоновых задач используется модуль Spring Scheduler, который отвечает за периодические проверки дат релизов и рассылку уведомлений.

Пользовательский интерфейс создаётся с использованием стандартных веб-технологий – HTML, CSS и JavaScript. Для стилизации и адаптивной верстки используется фреймворк Bootstrap [20], обеспечивающий удобную сетку и набор готовых UI-компонентов. Также в процессе анализа рассматривался CSS-фреймворк Foundationn [29]. Foundation обладает широкими возможностями кастомизации и гибкости, однако Bootstrap был выбран благодаря большему сообществу, лучшей документации и типичности для проектов подобного масштаба. Отображение данных в интерфейсе реализуется через шаблонизатор Thymeleaf [19], который позволяет напрямую связывать данные Java-приложения с HTML-структурой страниц. Для визуализации статистики и представления данных в графическом виде применяется библиотека Chart.js [21] – лёгкий JavaScript-инструмент, позволяющий создавать интерактивные графики и диаграммы прямо в браузере.

Для отправки автоматических уведомлений используется модуль Spring Boot Mail, обеспечивающий формирование сообщений и взаимодействие со стандартными SMTP-серверами. Этот инструмент позволяет без дополнительных библиотек реализовать корректную и надёжную email-рассылку.

Управление зависимостями и сборкой проекта осуществляется с помощью Apache Maven [25], который автоматизирует подключение библиотек и поддержку структуры проекта. Maven был выбран благодаря высокой совместимости со Spring Boot и стандартам индустрии.

Для построения диаграмм, схем и архитектурных моделей используется draw.io [22] – удобный веб-редактор, позволяющий создавать блок-схемы, диаграммы классов, ER-диаграммы и другие визуальные элементы.

Документация проекта создаётся в текстовом процессоре Microsoft Word 2023 [27], обеспечивающем удобное форматирование и структурирование пояснительной записки.

В качестве основной среды разработки для реализации программного приложения была выбрана IntelliJ IDEA [26]. Данный инструмент является одной из наиболее мощных и удобных IDE для разработки на Java и широко применяется в профессиональной практике. IntelliJ IDEA обеспечивает высокую производительность, глубокую интеграцию с экосистемой Java и предоставляет широкий набор инструментов, упрощающих процесс разработки и сопровождения программного продукта.

ГЛАВА 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ

2.1 Анализ требований и определение спецификаций программного обеспечения

Разработка информационно-справочной системы издательского дома требует всестороннего анализа потребностей пользователей, определения границ системы, выявления ключевых сценариев использования, а также формализации требований и ограничений. Данный этап является одним из наиболее важных, поскольку от качества сформулированных требований зависит корректность архитектуры, функциональность программы и последующая успешность её эксплуатации.

Пользователи системы делятся на четрые категории:

1. Обычные пользователи (читатели) – просматривают каталог, ищут книги, формируют избранное, получают уведомления.
2. Редакторы (сотрудники издательства) – управляют данными: добавляют книги, обновляют описания, загружают обложки, редактируют жанры, серии и информацию об авторах.
3. Менеджеры (сотрудники издательства) – могут просматривать статистику информационной системы;
4. Администраторы (сотрудники издательства) – следят за пользователями и общим состоянием информационно-справочной системы. Имеют доступ ко всем данным.

Предметная область включает информацию, отражающую структуру издательского каталога:

- книга (название, ISBN, описание, обложка, формат, дата выхода);
- автор;
- жанр;
- серия;
- статус релиза (вышла / ожидается).

Каждая сущность связана с остальными, что требует использования реляционной модели данных. Кроме того, информация должна быть

структурированной, актуальной и полной, что накладывает требования на методы её хранения и обработки.

Система должна обеспечить корректную работу с каталогом, поддерживая операции поиска, фильтрации и отображения взаимосвязей между сущностями.

Основные функциональные требования:

- отображение общего каталога книг;
- поиск книг по названию, автору, жанру, серии, ключевым словам, ISBN;
- фильтрация каталога по жанрам, сериям, формату, статусу выхода;
- просмотр карточки книги с полной информацией;
- регистрация, авторизация и управление учётной записью;
- добавление книг в список желаний и удаление из него;
- рассылка email-уведомлений о выходе интересующих книг;
- отображение статистических данных в виде диаграмм;
- корректная обработка ошибок и логирование событий сервера;
- административная панель с возможностью управления книгами, жанрами, авторами и сериями;
- загрузка и обновление изображений обложек.

Нефункциональные требования определяют характеристики системы, условия её исполнения и эксплуатации.

Основные нефункциональные требования:

- удобство: интерфейс должен быть интуитивно понятным.
- производительность: время ответа сервера — не более 2 секунд при штатной нагрузке.
- совместимость: поддержка всех современных браузеров.
- надёжность: сохранение целостности данных при сбоях.
- безопасность: защита от SQL-инъекций, XSS, надёжное хэширование паролей.

- масштабируемость: возможность подключения к внешним базам данных, поддержка контейнеризации в будущем.
- поддерживаемость: код должен быть структурирован по слоям (MVC, сервисы, репозитории).
- доступность: круглосуточная работа с минимальными простоями.

Система подлежит следующим ограничениям:

- зависимость от стабильности интернет-соединения;
- ограничения SMTP-серверов на массовую отправку уведомлений;
- зависимость производительности от конфигурации оборудования сервера;

Данные каталога должны соответствовать следующим критериям:

- корректность форматов дат, ISBN, текстовых описаний;
- уникальность идентификаторов;
- отсутствие дублирующихся записей;
- хранение изображений обложек в корректируемом формате;
- поддержка кодировки UTF-8.

На основании проведённого анализа сформулированы следующие спецификации программного обеспечения:

Программное обеспечение должно:

- функционировать по трёхзвенной архитектуре «клиент – сервер – база данных»;
- использовать Spring Boot как основу серверной части;
- применять PostgreSQL для хранения данных;
- работать с ORM-моделью на Hibernate;
- формировать интерфейс с помощью HTML, CSS, Bootstrap и шаблонизатора Thymeleaf;
- обеспечивать визуализацию данных средствами Chart.js;
- отправлять уведомления через модуль Spring Boot Mail;
- выполнять фоновые задачи через Spring Scheduler;

- соответствовать эксплуатационным требованиям и стандартам безопасности.

2.2 Проектирование программного обеспечения

Для реализации системы была выбрана трёхзвенная архитектура, включающая следующие уровни:

- Клиентский слой — интерфейс пользователя (HTML, CSS, Bootstrap, JavaScript);
- Серверный слой — логика обработки запросов, сервисы, контроллеры (Spring Boot);
- Слой данных — взаимодействие с БД через ORM (JPA/Hibernate) и PostgreSQL.

Такой подход обеспечивает:

- разделение ответственности между уровнями;
- упрощённое тестирование и отладку;
- возможность масштабирования;
- защиту от прямого доступа клиента к базе данных;
- возможность расширения функциональности без изменения всей системы.

Выбор архитектуры обоснован её распространённостью, надёжностью, удобством для веб-систем и высокой совместимостью с используемыми инструментами Java-экосистемы.

Логическая структура системы отражает деление приложения на функциональные модули. Основные логические блоки:

- модуль управления пользователями
- регистрация и авторизация;
- хранение данных профиля;
- управление списком желаний.

- модуль каталога книг
- вывод списка книг;
- поиск и фильтрация;
- отображение карточки книги.
- модуль работы с уведомлениями
- проверка статуса релиза;
- отправка email-уведомлений через Spring Boot Mail;
- периодическое выполнение задач через Spring Scheduler.
- административный модуль
- управление книгами;
- редактирование жанров, авторов и серий;
- загрузка обложек.
- модуль обработки ошибок
- отображение кастомных страниц ошибок (404, 500);
- централизованная обработка исключений.

Каждый модуль реализован в виде отдельного набора сервисов, контроллеров и репозиториев.

База данных разрабатывается на основе требований предметной области. При проектировании используется реляционная модель, а взаимодействие с ней выполняется через ORM-модель Hibernate.

PK – первичный ключ.

FK – внешний ключ.

На рисунке 1 представлена ER-диаграмма базы данных;

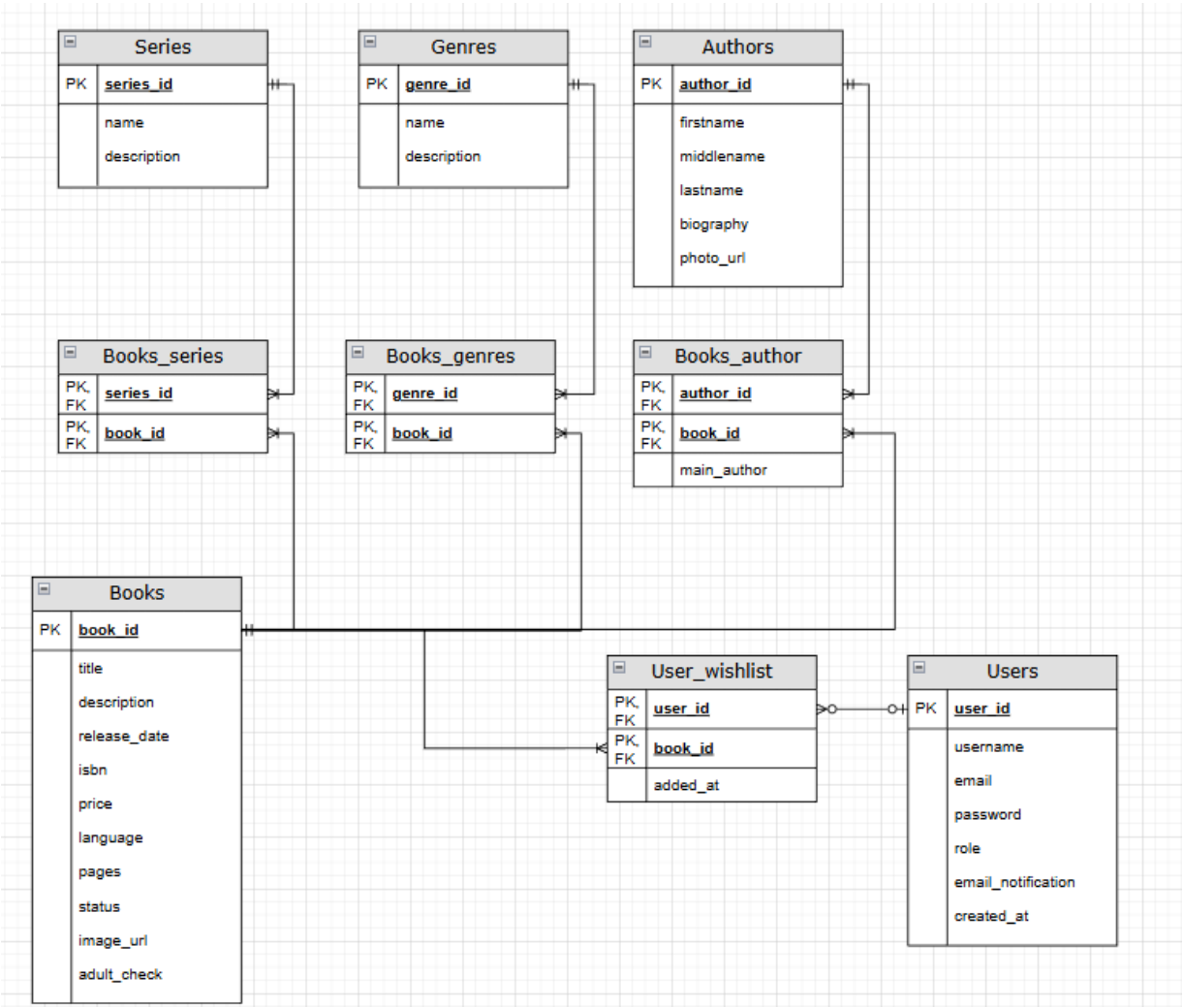


Рисунок 1. ER-диграмма информационно-справочной системы

Основные сущности:

- Books (Книги)

book_id (PK), title, description, release_date, isbn, price, language, pages, status image_url, adult_check;

- Authors (Авторы)

author_id (PK), firstname, middlename, lastname, biography, photo_url;

- Genres (Жанры)

genre_id (PK), name, description;

- Series (Серии)

series_id (PK), name, description;

- Users (Пользователь)

user_id (PK), username, email, password, role, email_notification, created_at, updated_at, log_in_at, logout_at;

Многие-ко-многим связи между сущностями были реализованы через дополнительные сущности:

- Books_series (Книги_серии)

series_id (PK, FK), book_id (PK, FK);

- Books_genres (Книги_жанры)

genre_id (PK, FK), book_id (PK, FK);

- Books_authors (Книги_авторы)

author_id (PK, FK), book_id (PK, FK), main_author;

- User_wishlist (Пользователь_список желаний)

user_id (PK, FK), book_id (PK, FK), added_at;

Для обеспечения целостности используются внешние ключи и каскадные операции.

Приложение разделено на компоненты согласно концепции MVC и традициям Spring-экосистемы.

Основные компоненты:

- Controller-слой (рисунок 2)
    - принимает запросы пользователей;
    - вызывает соответствующие сервисы;
    - передаёт данные в шаблоны Thymeleaf.

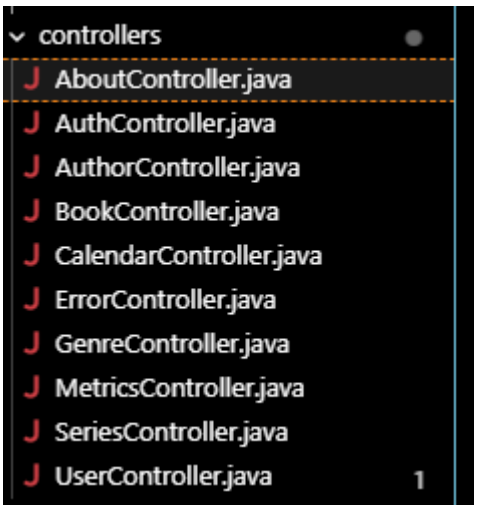


Рисунок 2. Controller-слой

- Service-слой (рисунок 3)
    - содержит бизнес-логику приложения;
    - обрабатывает данные, работает с репозиториями, вызывает email-модуль.

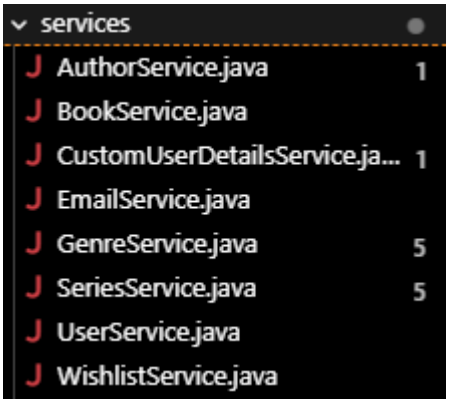


Рисунок 3. Service-слой

- Repository-слой (рисунок 4)
    - взаимодействует с базой данных через Spring Data JPA;
    - выполняет CRUD-операции.

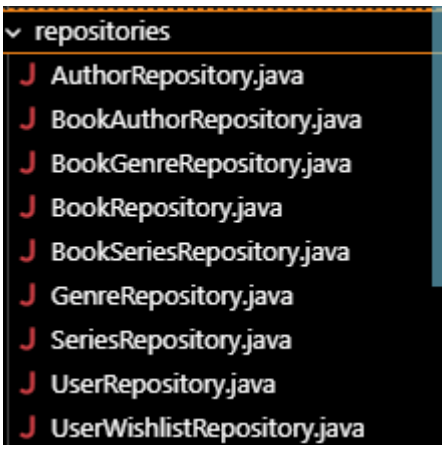


Рисунок 4. Repository-слой

- Model-слой (рисунок 5)
  - содержит описания сущностей и связь их с таблицами БД.

Рисунок 5. Repository-слой

- Presentation-слой (рисунок 6)
  - HTML-шаблоны Thymeleaf (templates) ;
  - CSS-стили (style.css);
  - JS скрипты (script.js);
  - Bootstrap-компоненты;

- Chart.js для визуализации статистических данных.

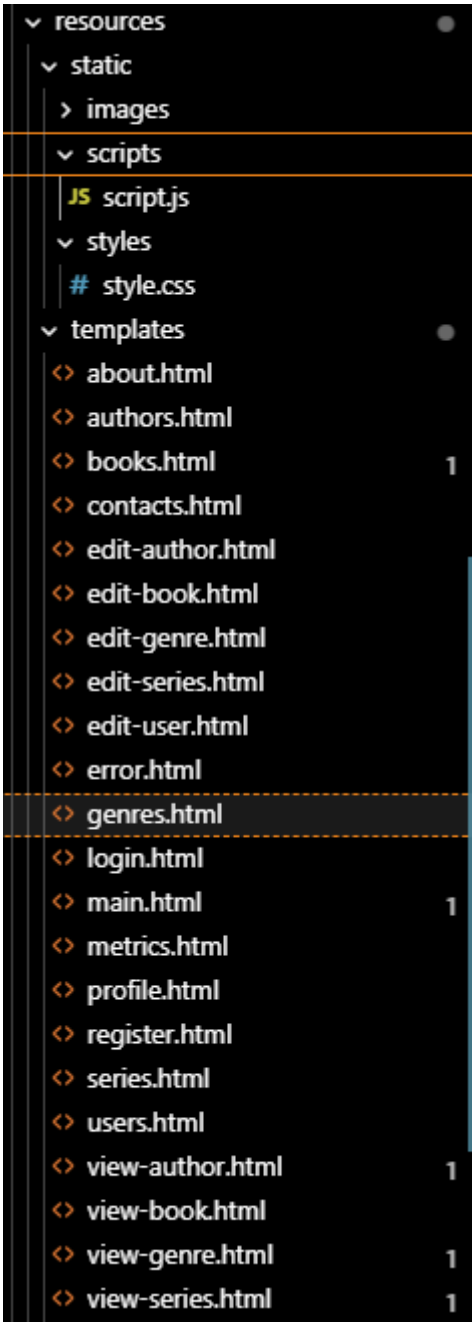


Рисунок 6. Presentation-слой

Интерфейс основывается на принципах простоты, понятности и минимального количества вложенных действий.

Пользовательский интерфейс разрабатывается с использованием:

- Bootstrap — для адаптивной гибкой верстки;
- Thymeleaf — для динамической загрузки данных;

- JavaScript — для взаимодействия с пользовательскими элементами;
- Chart.js — для отображения диаграмм и статистики.

Основные интерфейсные страницы:

- главная страница с будущими релизами;
- каталог книг, авторов, жанров, серий;
- карточка книги, автора, жанра, серии;
- личный профиль пользователя (список желаний);
- страница метрик;
- административная страница;
- форма добавления и редактирования книг, авторов, жанров, серий;
- страница ошибок;
- страница «О себе»;
- страница «Контакты».

Взаимодействие клиента и сервера:

1. Пользователь отправляет HTTP-запрос (GET или POST).
2. Контроллер принимает запрос и вызывает соответствующий сервис.
3. Сервис обрабатывает данные, взаимодействует с базой через репозиторий.
4. Сформированный ответ направляется в шаблон Thymeleaf.
5. Клиент получает готовую HTML-страницу.

Такой подход обеспечивает безопасность данных, поскольку клиент не имеет прямого доступа к базе.

Для защиты данных применяется:

- Spring Security — защита маршрутов, ролевая модель (USER/EDITOR/MANAGER/ADMIN);

- BCrypt — хэширование паролей;
- ORM-уровень защиты — предотвращение SQL-инъекций;
- Экранирование данных в шаблонизаторе Thymeleaf — защита от XSS;
- Валидация форм — предотвращение некорректного ввода.

Обработка ошибок реализуется через:

- обработчики исключений;
- пользовательскую страницу ошибок (404, 403, 500) – error.html;

Это обеспечивает устойчивость системы и отсутствие аварийных завершений.

2.3 Разработка информационной-системы

Разработка информационно-справочной системы издательского дома осуществлялась на основе проектных решений, сформированных на предыдущем этапе. В ходе работы были реализованы база данных, серверная логика, пользовательский интерфейс, модуль уведомлений, а также административные инструменты для управления каталогом. Разработка велась последовательно: от настройки среды и создания базовой структуры проекта до реализации функциональных модулей и тестирования готового решения.

На начальном этапе была выполнена настройка рабочей среды разработки. В проекте использовались:

- IntelliJ IDEA для разработки и отладки кода;
- Java 25 как язык разработки;
- Spring Boot 4 как основной фреймворк;
- Maven для управления зависимостями и сборкой;
- PostgreSQL для хранения данных;

После создания проекта структура директорий была организована по принципу MVC с разделением логики на уровни (рисунок 7):

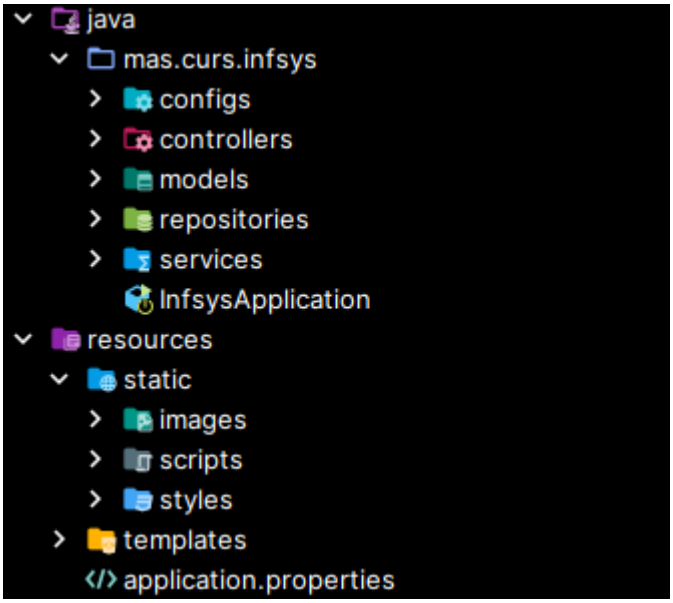


Рисунок 7. Структура директорий проекта

Такой подход позволил обеспечить гибкость, унифицированность и удобство расширения на последующих этапах разработки.

На основе разработанной ER-диаграммы была создана физическая модель базы данных (рисунок 8). Реализованы следующие таблицы:

books — информация о книгах;

authors, genres, series — справочные таблицы;

books_authors, books_genres, books_series — связывающие таблицы;

users — данные пользователей;

user_wishlist — связи между пользователем и книгами.

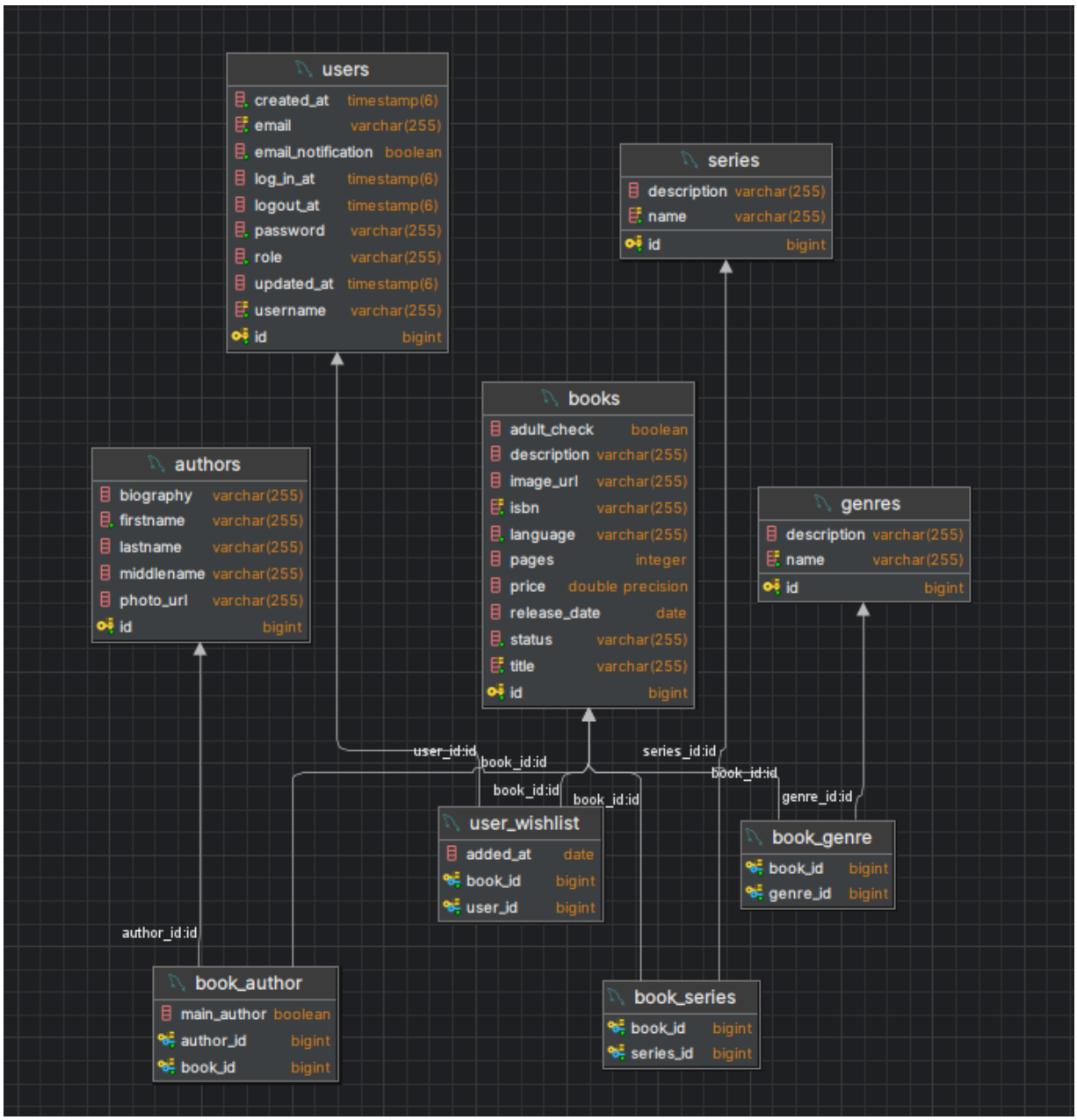


Рисунок 8. Физическая модель базы данных

Каждая сущность была реализована в виде Java-класса с аннотациями (рисунок 9):

@Entity — указание на хранение в БД;

@Table — связь с конкретной таблицей;

@Id, @GeneratedValue — первичные ключи;

@OneToMany — связи между таблицами.

Использование Hibernate позволило автоматизировать преобразование данных в объекты Java и обратно, а также исключить ручное написание SQL-запросов при типовых операциях.

```java
@Entity  AknasMacefg
@Table (name = "books")

public class Book {

    @Id
    @GeneratedValue (strategy = GenerationType.IDENTITY)
    private Long id;

    @OneToMany(mappedBy = "book", cascade = CascadeType.ALL, orphanRemoval = true)  2 usages
    private Set<UserWishlist> UserWishlist = new HashSet<>();

    @OneToMany(mappedBy = "book", cascade = CascadeType.ALL, orphanRemoval = true)  2 usages
    private Set<BookAuthor> BookAuthor = new HashSet<>();

    @OneToMany(mappedBy = "book", cascade = CascadeType.ALL, orphanRemoval = true)  2 usages
    private Set<BookGenre> BookGenre = new HashSet<>();

    @OneToMany(mappedBy = "book", cascade = CascadeType.ALL, orphanRemoval = true)  2 usages
    private Set<BookSeries> BookSeries = new HashSet<>();


    @Column (nullable = false, unique = true)  3 usages
    private String title;

    @Column  3 usages
    private String description;

    @Column  3 usages
    private LocalDate release_date;
```

Рисунок 9. Реализация сущности на примере Book

Серверная часть реализована в соответствии с трёхзвенной архитектурой и включает контроллеры, сервисы и репозитории.

Модуль каталога книг является основой системы и включает (рисунок 10):

- загрузку списка всех книг из базы;
- сортировку данных по различным полям;
- поиск по названию, ISBN, автору, жанру;
- фильтрацию по сериям, статусам и другим параметрам.

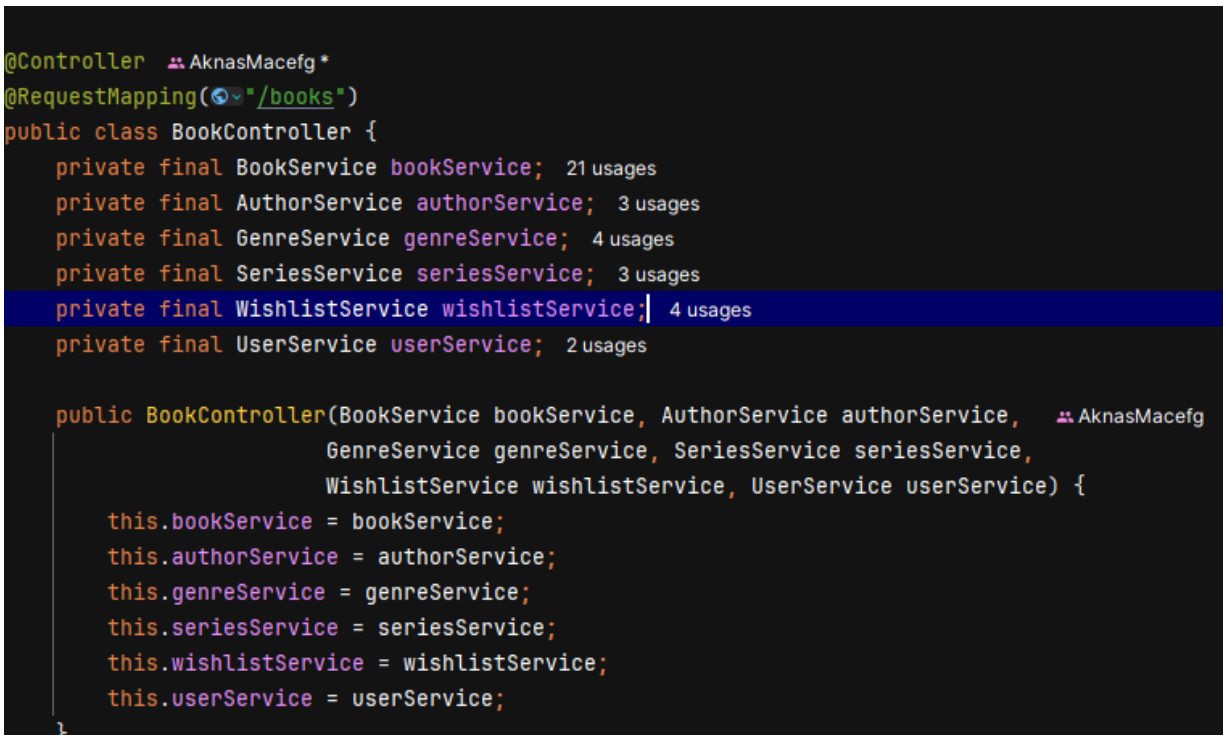


Рисунок 10. BookController загружает все книги при переходе на /books

Поиск и фильтрация (рисунок 11) был реализован через репозитории Spring Data JPA, что позволяет автоматически формировать SQL-запросы на основе имени метода.

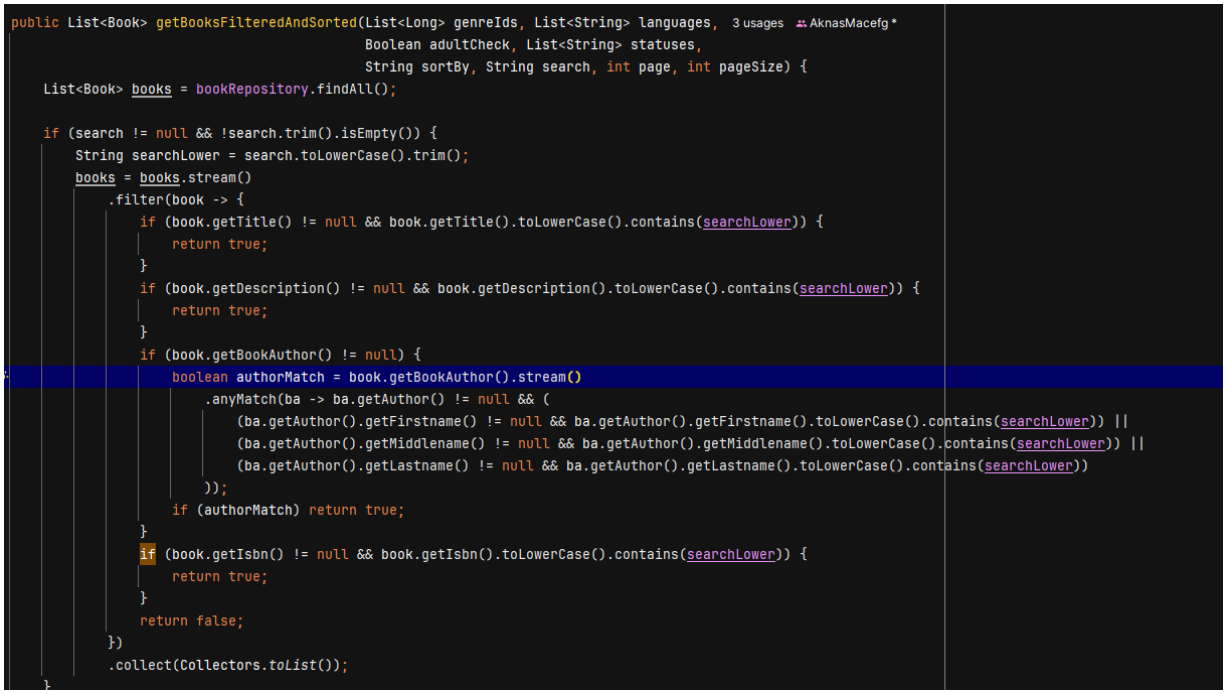


Рисунок 11. Часть кода функции фильтрации и сортировки в BookService

Фильтрация выполнялась через комбинированные параметры запроса, а сортировка — через функцию sort().

При переходе на страницу книги система (рисунок 12):

- принимает id книги через URL;
- извлекает данные книги через сервис;
- загружает связанные данные:
  - авторов,
  - жанры,
  - серию,
  - статус релиза.

```
@GetMapping("/view/{id}")  AknasMacefg *
public String showViewForm(@PathVariable("id") Long id, Model model) {
    Book book = bookService.getBookById(id);
    model.addAttribute( attributeName: "books", book);

    User currentUser = getCurrentUser();
    boolean inWishlist = false;
    if (currentUser != null) {
        inWishlist = wishlistService.isInWishlist(currentUser.getId(), id);
    }
    model.addAttribute( attributeName: "inWishlist", inWishlist);
    model.addAttribute( attributeName: "currentUser", currentUser);

    return "view-book";
}
```

Рисунок 12. BookController загружает всю информацию о книге при переходе на /books/view/{id} (id – id книги в БД)

Данные передаются в Thymeleaf для отображения в виде карточки.

Реализованы (рисунок 13-14):

- регистрация нового пользователя;
- авторизация через Spring Security;
- хеширование паролей через BCrypt;
- хранение профиля пользователя.

При регистрации выполняются:

- проверка корректности email,
- проверка уникальности адреса,
- защита от слабых паролей.

Отсутствие авторизации ограничивает доступ к личному профилю и некоторому функционалу веб-приложения (например добавить в список в желаемого)

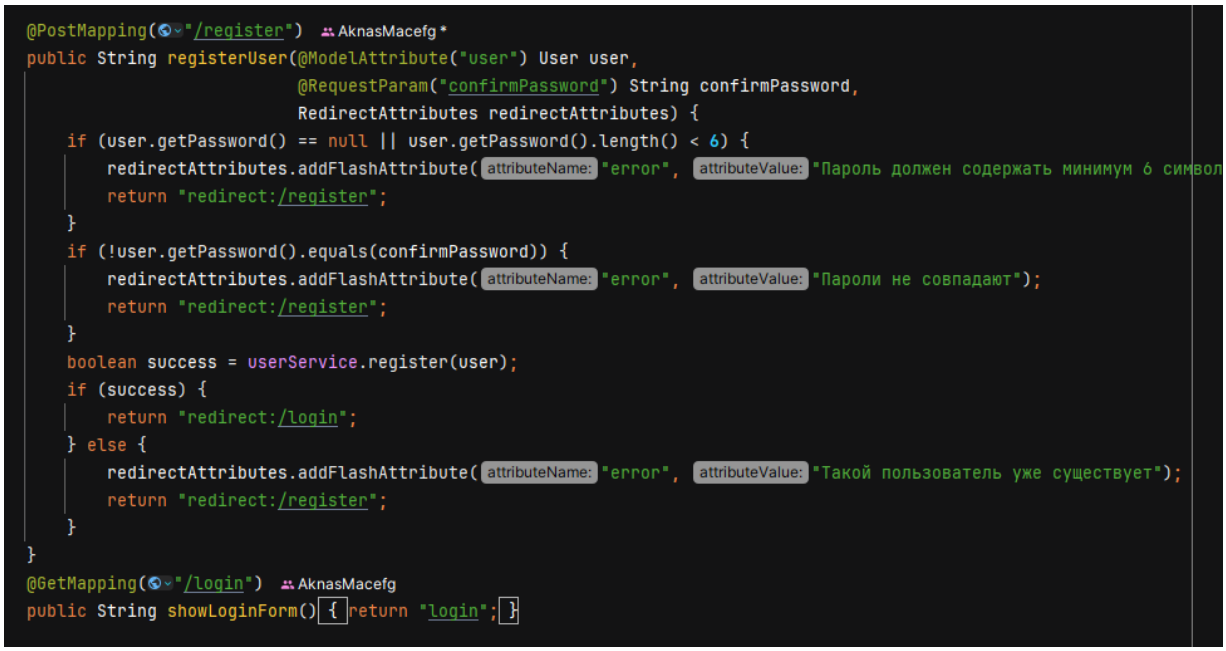


Рисунок 13. AuthController загрузка форм для логина и регистрации

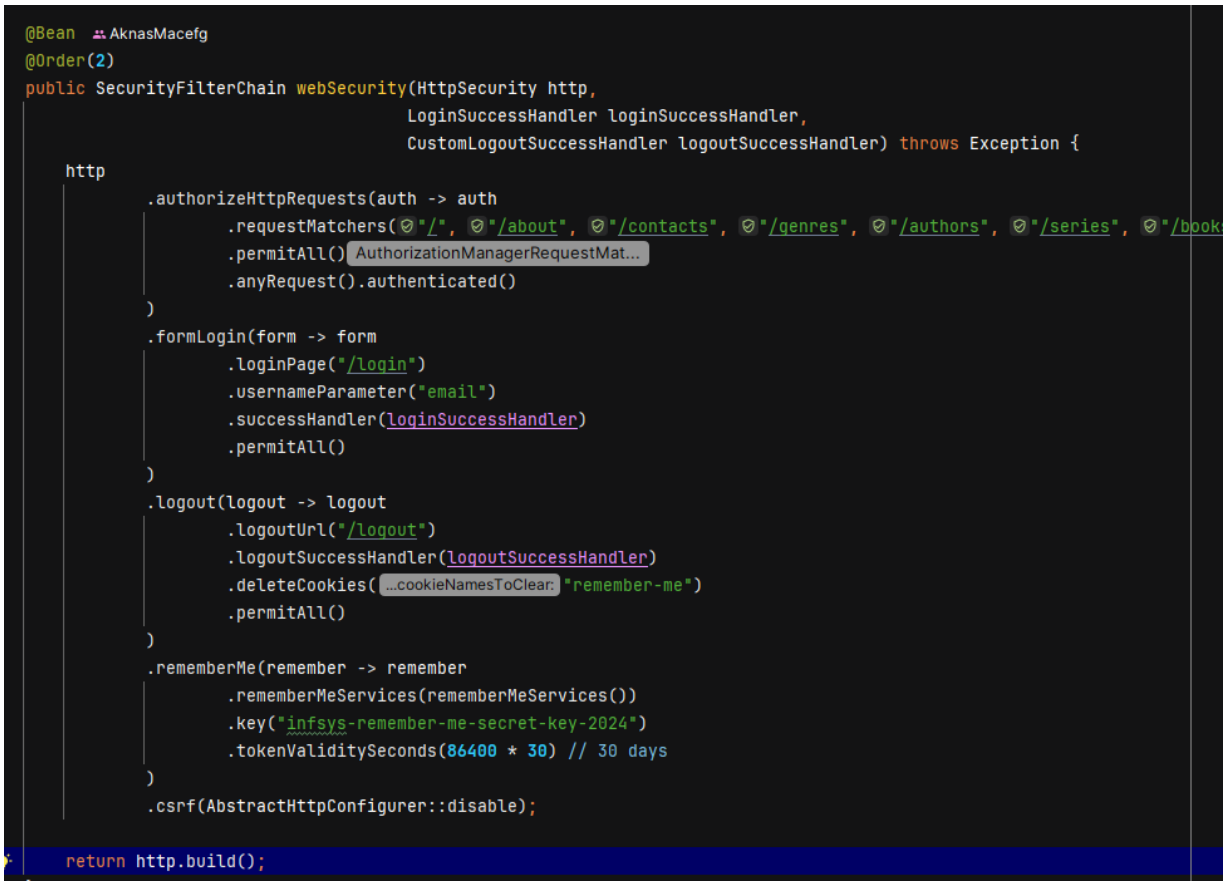


Рисунок 14. SecurityConfig логика обработки доступа и перенаправления

Список желаемых книг реализован через сервис, который:

- добавляет книгу в избранное;
- удаляет из избранного;
- отображает все элементы пользователя.

Проверяется:

- уникальность добавления;
- корректность прав пользователя.

```
@Autowired
private UserRepository userRepository;

@Autowired
private BookRepository bookRepository;

@Transactional  1 usage  AknasMacefg *
public boolean addToWishlist(Long userId, Long bookId) {
    User user = userRepository.findById(userId).orElse( other: null);
    Book book = bookRepository.findById(bookId).orElse( other: null);

    if (user == null || book == null) {
        return false;
    }
    boolean alreadyExists = user.getUserWishlist().stream()
        .anyMatch(wl -> wl.getBook().getId().equals(bookId));

    if (alreadyExists) {
        return false;
    }

    UserWishlist wishlistItem = new UserWishlist(user, book, LocalDate.now());
    UserWishlistId id = new UserWishlistId(userId, bookId);
    wishlistItem.setId(id);
    user.getUserWishlist().add(wishlistItem);
    userRepository.save(user);
    return true;
}

@Transactional  2 usages  AknasMacefg
public boolean removeFromWishlist(Long userId, Long bookId) {
    User user = userRepository.findById(userId).orElse( other: null);
    if (user == null) {
        return false;
    }

    boolean removed = user.getUserWishlist().removeIf(
        wl -> wl.getBook().getId().equals(bookId)
```

Рисунок 15. WishlistSerivce часть кода добавления и удаления книг в список желаемого

Для сотрудников издательства создан набор функций:

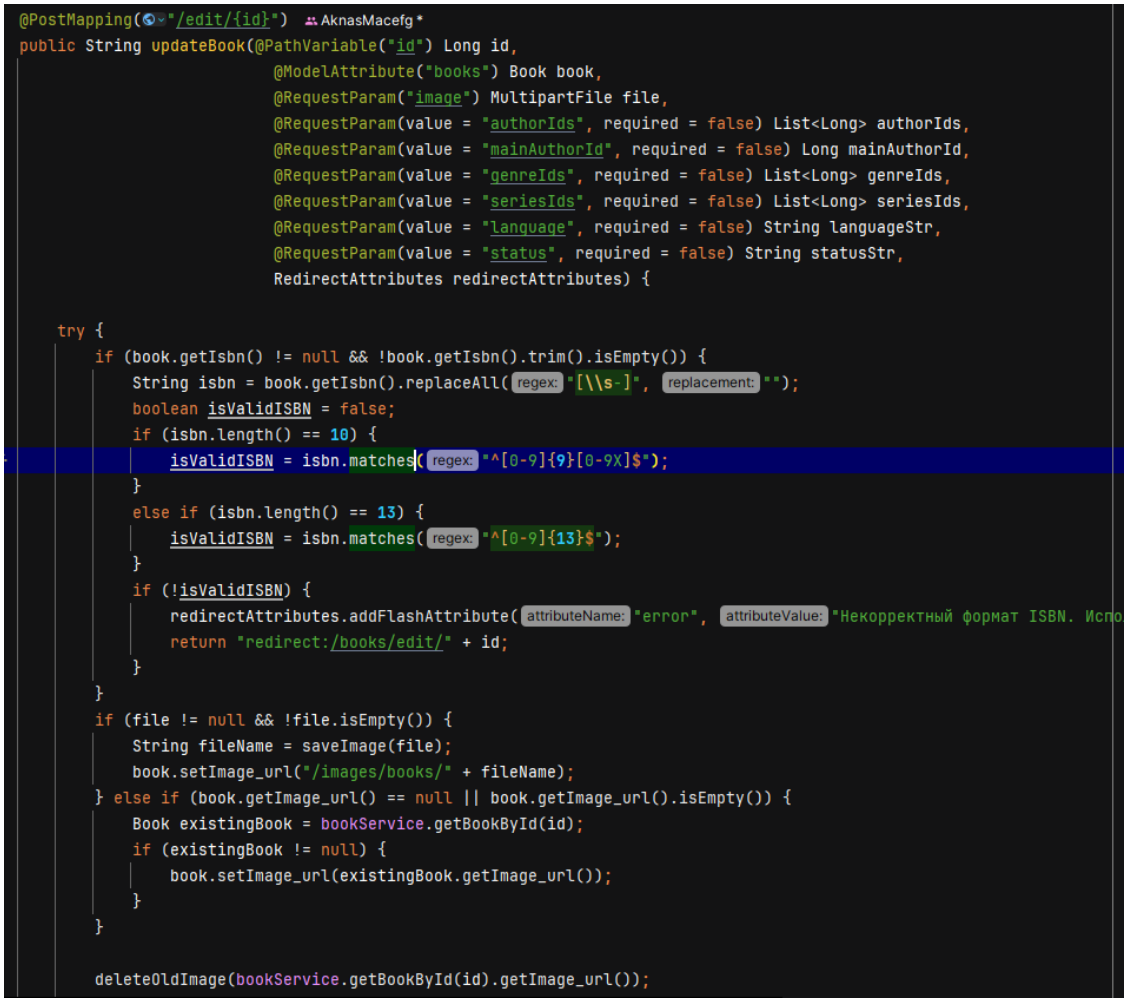
- добавление новой книги;

- редактирование существующей;
- управление авторами, жанрами, сериями;
- загрузка изображений обложек с проверкой формата;
- удаление записей.

При добавлении книги выполняется:

- заполнение всех обязательных полей,
- проверка уникальности ISBN,
- загрузка обложки,
- создание связей с авторами и жанрами.

Страницы редактирования имеют отдельные маршруты, доступные только EDITOR и ADMIN-пользователям.



```
@PostMapping("/edit/{id}")  AknasMacefg *
public String updateBook(@PathVariable("id") Long id,
                         @ModelAttribute("books") Book book,
                         @RequestParam("image") MultipartFile file,
                         @RequestParam(value = "authorIds", required = false) List<Long> authorIds,
                         @RequestParam(value = "mainAuthorId", required = false) Long mainAuthorId,
                         @RequestParam(value = "genreIds", required = false) List<Long> genreIds,
                         @RequestParam(value = "seriesIds", required = false) List<Long> seriesIds,
                         @RequestParam(value = "language", required = false) String languageStr,
                         @RequestParam(value = "status", required = false) String statusStr,
                         RedirectAttributes redirectAttributes) {

    try {
        if (book.getIsbn() != null && !book.getIsbn().trim().isEmpty()) {
            String isbn = book.getIsbn().replaceAll( regex: "[\\s-]",  replacement: "");
            boolean isValidISBN = false;
            if (isbn.length() == 10) {
                isValidISBN = isbn.matches( regex: "^[0-9]{9}[0-9X]$");
            }
            else if (isbn.length() == 13) {
                isValidISBN = isbn.matches( regex: "^[0-9]{13}$");
            }
            if (!isValidISBN) {
                redirectAttributes.addFlashAttribute( attributeName: "error",  attributeValue: "Некорректный формат ISBN. Испо
                return "redirect:/books/edit/" + id;
            }
        }
        if (file != null && !file.isEmpty()) {
            String fileName = saveImage(file);
            book.setImage_url("/images/books/" + fileName);
        } else if (book.getImage_url() == null || book.getImage_url().isEmpty()) {
            Book existingBook = bookService.getBookById(id);
            if (existingBook != null) {
                book.setImage_url(existingBook.getImage_url());
            }
        }

        deleteOldImage(bookService.getBookById(id).getImage_url());
```

Рисунок 16. BookController часть кода обработки изменения книги.

Модуль уведомлений был разработан для автоматической рассылки писем пользователям, когда книга из их списка желаний выходит в продажу.

Работа модуля строится на следующих шагах (рисунок 17):

- Планировщик Spring Scheduler запускается по расписанию (каждый день).
- Планировщик проверяет таблицу книг на наличие изменений в статусе.
- Если дата выхода сопоставлена со списком пользователя, формируется уведомление.

С помощью Spring Boot Mail отправляется email (рисунок 18).

Этот модуль работает полностью автономно и не требует участия пользователя.

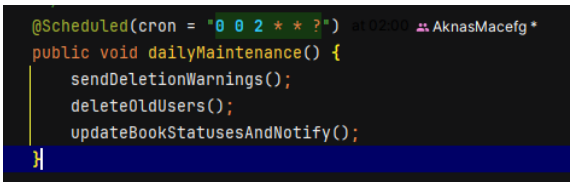


Рисунок 17. ScheduledTasks функции которые запускаются по расписанию в 2:00

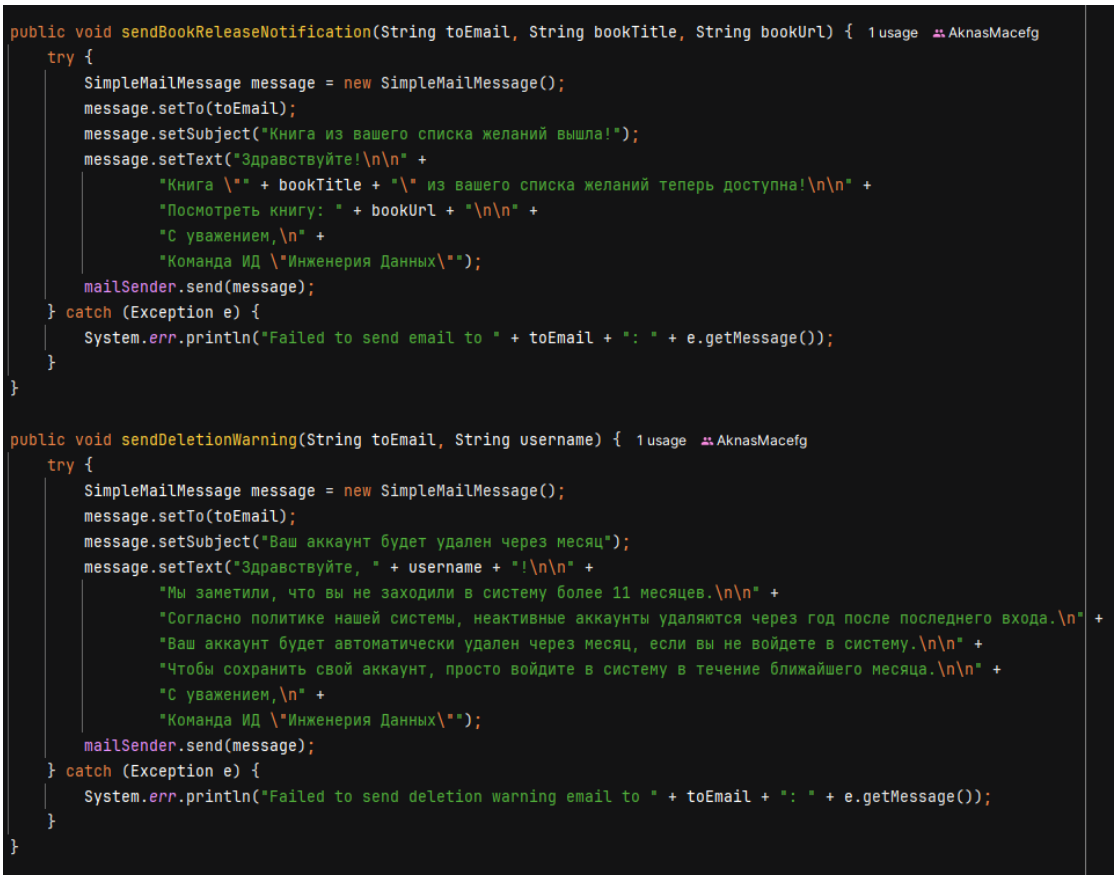


Рисунок 18. EmailService функции отправки писем

Основные элементы UI:

- главная страница со списком ближайших релизов – список карточек по датам (main.html);
- страница с карточками по книгам, авторам, жанрам, сериям – сетка карточек (books.html, authors.html, genres.html, series.html, рисунок 19);
- панель поиска, сортировки и фильтров – жанры, серии, статус релиза (books.html, authors.html, genres.html, series.html, рисунок 19);

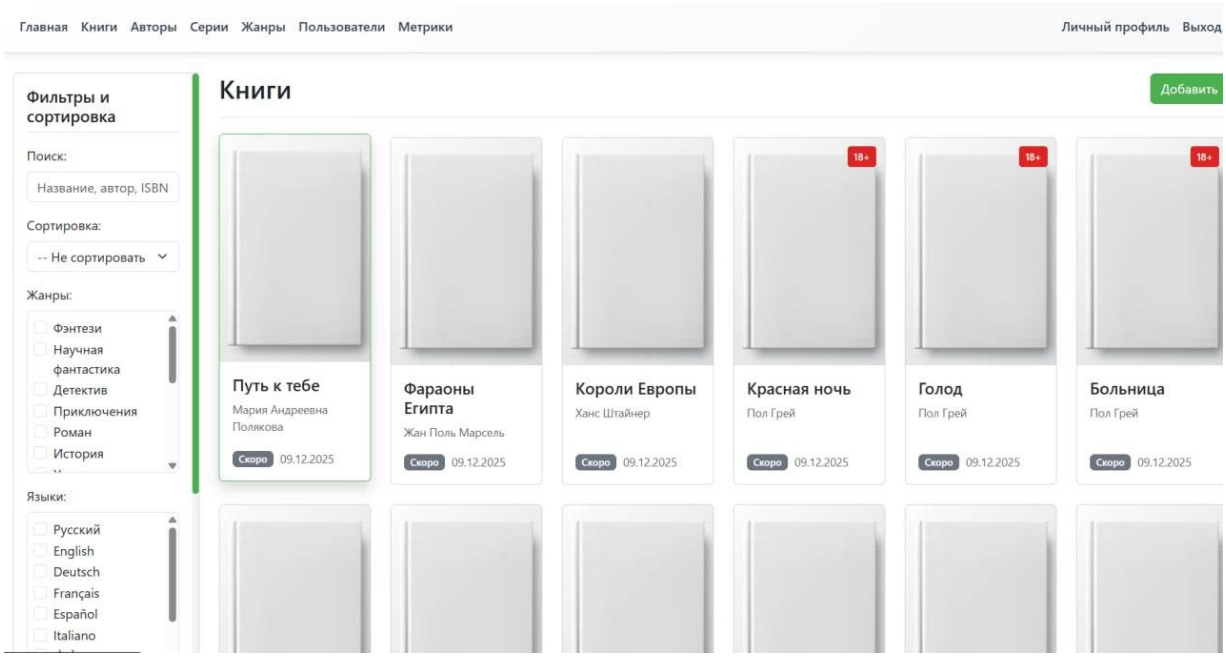


Рисунок 19. books.html

- карточка книги, автора, жанра, серии – описание, обложка (если есть), информация (view-book.html, view-author.html, view-genre.html, view-series.html, рисунок 20);
- страница изменения книги, автора, жанра, серии – поля для изменения информации (edit-book.html, edit-author.html, edit-genre.html, edit-series.html)

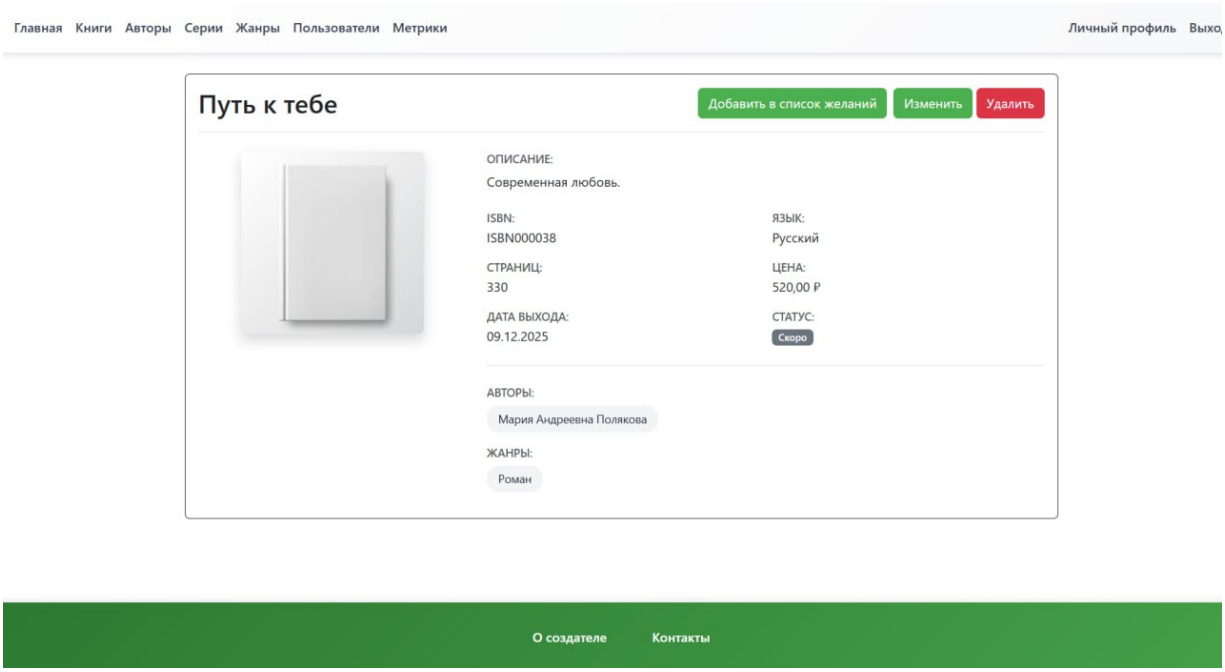


Рисунок 20. view-book.html

- страницы входа и регистрации (login.html, register.html)
- страницы связанные с пользователями – список пользователей, изменение пользователей (users.html, edit-user.html)
- личный кабинет – email, смена email и пароля (profile.html);
- список желаний – карточки избранных книг (profile.html, рисунок 21);

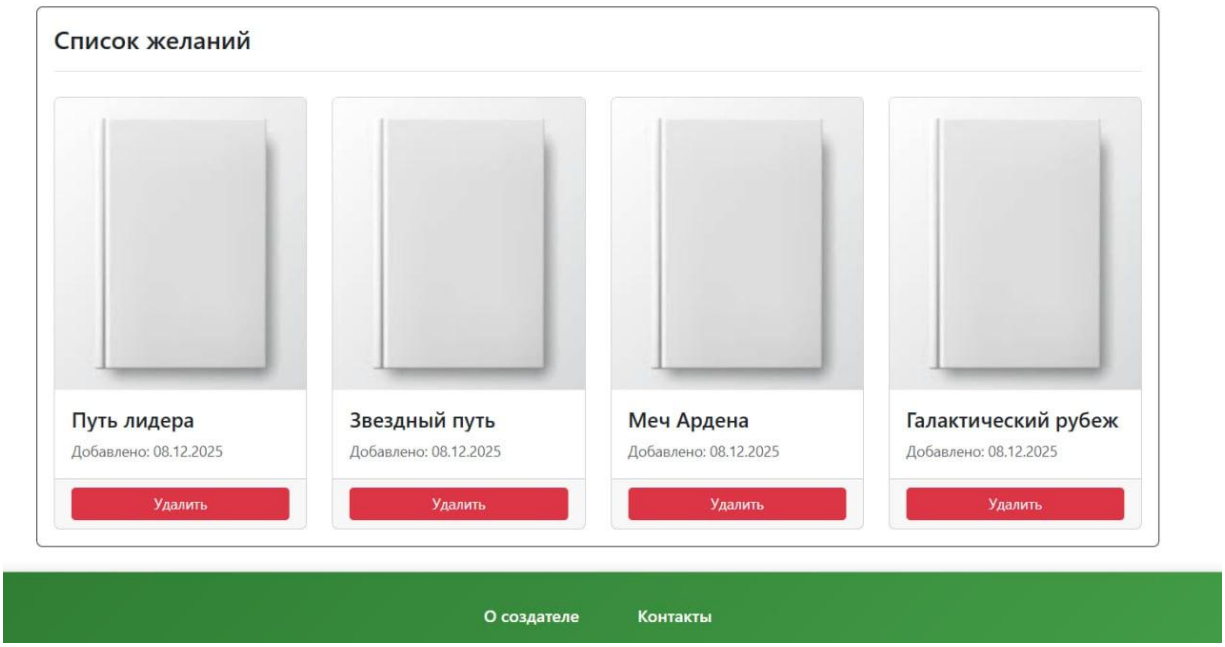


Рисунок 21. Часть profile.html

- страница с метриками – аналитические данные (metrics.html).

Для отображения аналитических данных используется Chart.js, доступ к странице доступ только для MANAGER и ADMIN-пользователей (рисунок 22).

Диаграммы могут показывать:

- количество авторов, добавленных в список желаемого по книге;
- количество жанров, добавленных в список желаемого по книге;
- распределение дат добавления в список желаемого.

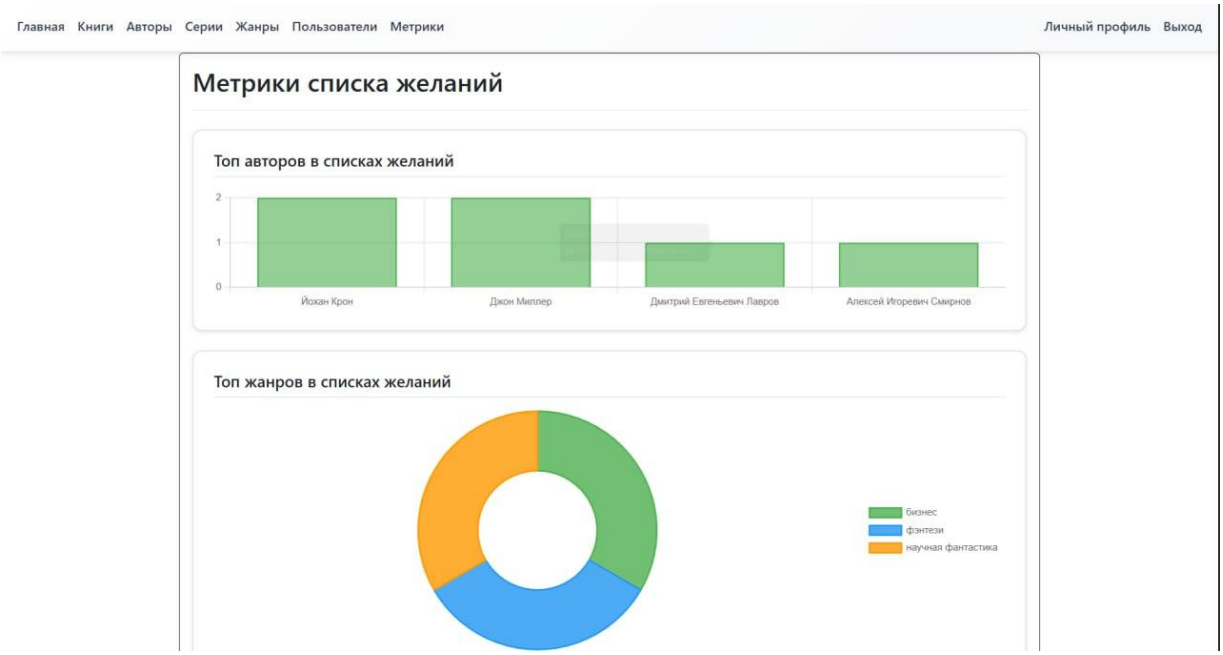


Рисунок 22. metrics.html

Обработка ошибок включает:

- глобальный перехват исключений @ErrorController (рисунок 23);
- кастомная HTML-страница ошибок (error.html):
  - 404 — страница не найдена;
  - 403 — доступ запрещён;
  - 500 — внутренняя ошибка сервера.

Такой подход исключает аварийные завершения работы программы.

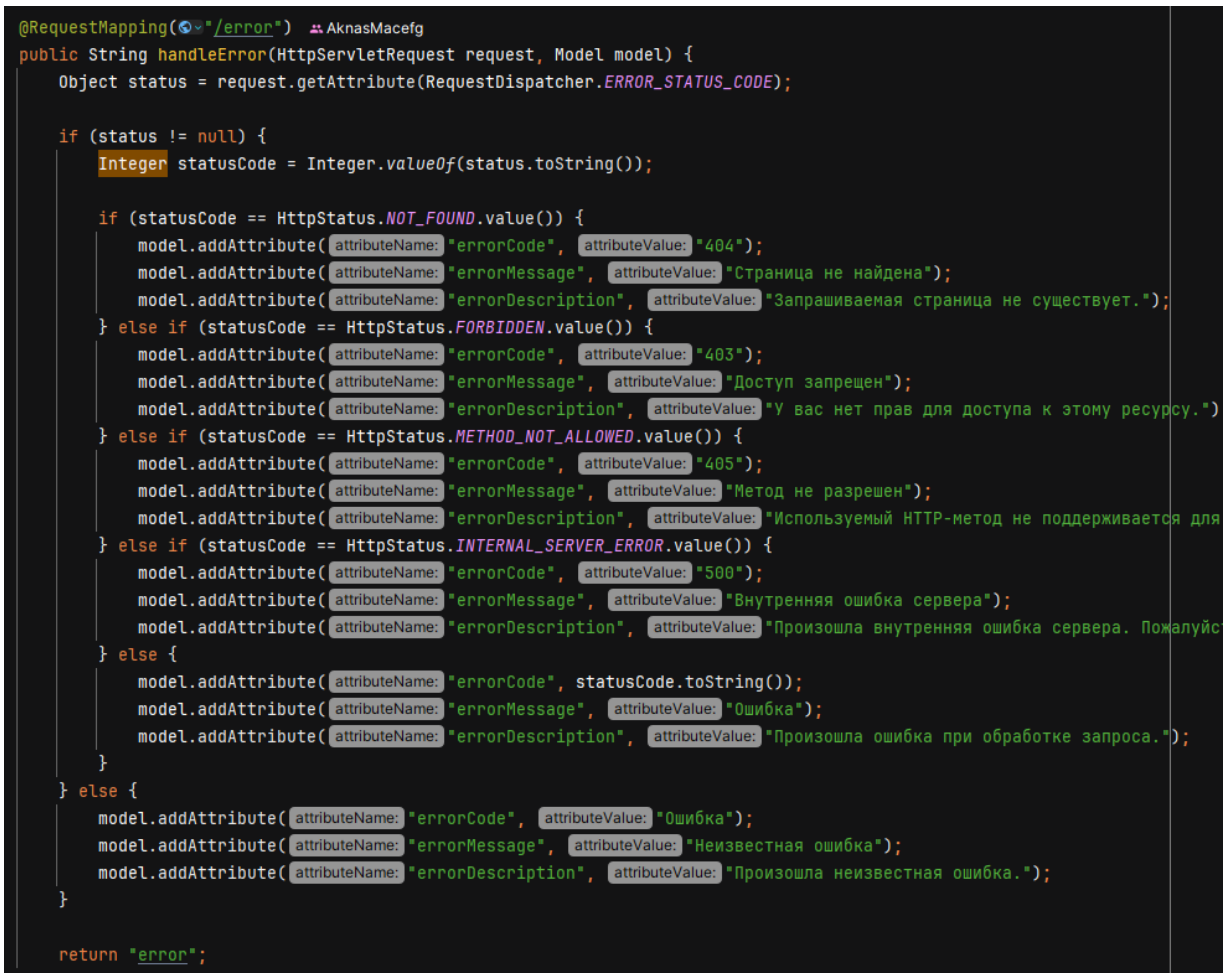


Рисунок 23. ErrorController обработка ошибок

2.4 Отладка и тестирование программы

В качестве метода для тестирования программного обеспечения был использован метод белого ящика. Тестирование производилось ручным типом, без использования инструментов автоматизации. В таблице 1 представлены результаты отладки и тестирования программы.

Таблица 1. Результаты отладки и тестирования веб-приложения

№ теста	Входные данные	Вводимое значение	Ожидаемая реакция программы	Реакция программы	Ошибка выявлена
1	Текст	“”	Сообщение «ISBN должен быть в правильном формате»	На рисунке 25	Нет

2	Путь к странице	Путь к несуществующей странице (/books)	Ошибка 404	На рисунке 26	Нет

В результате тестирования и отладки программы ошибок в поведении программы не было выявлено. Программа обрабатывает вводимые пользователем данные и реагирует при выявлении некорректных значениях.

Критических ошибок при выполнении программы не обнаружено.

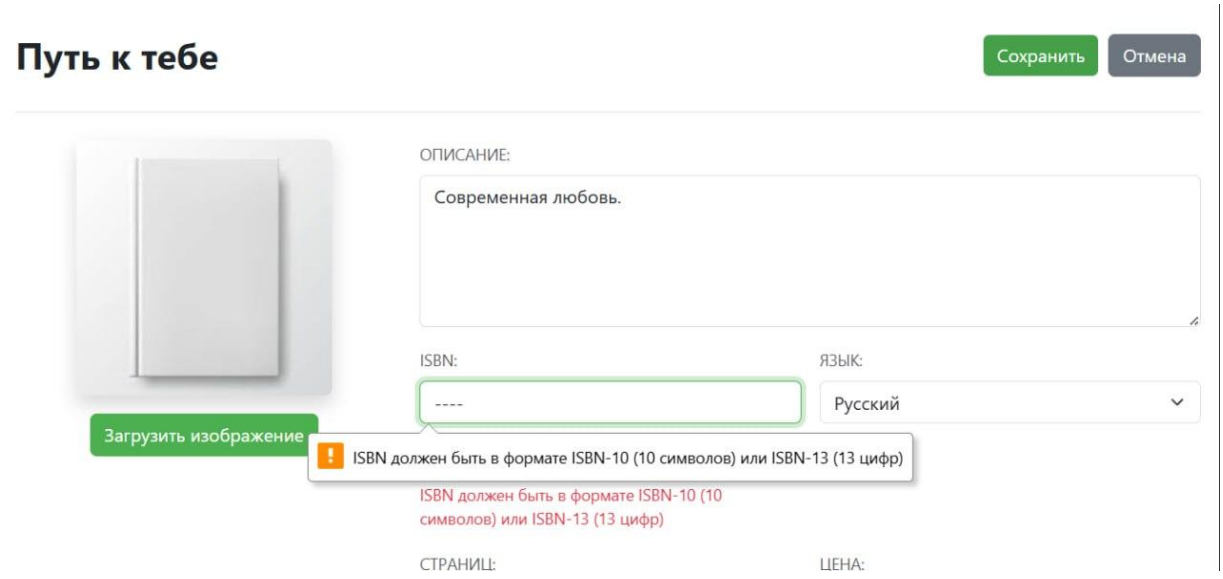


Рисунок 26. Ввод неправильного текста

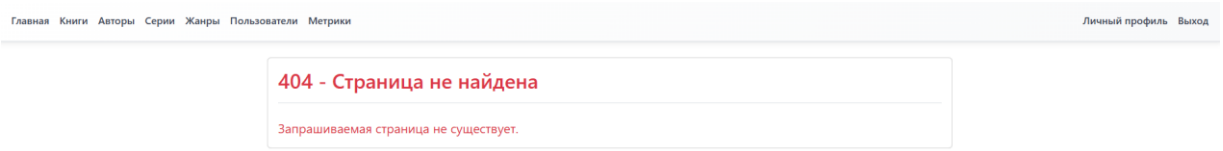


Рисунок 27. Переход на несуществующую страницу

Программа работает в соответствии с заявленными функциональными и не функциональными задачами.

ЗАКЛЮЧЕНИЕ

В результате выполнения курсовой работы была разработана информационно-справочная система издательского дома, предназначенная для поиска, просмотра и структурирования сведений о книгах, а также для взаимодействия пользователей с каталогом издательства.

В ходе разработки курсового проекта были выполнены все поставленные задачи:

- реализован веб-интерфейс для работы с каталогом книг;
- разработана серверная часть с поддержкой поиска, фильтрации и просмотра информации;
- создан личный профиль пользователя и система списка желаний;
- реализован модуль автоматической рассылки уведомлений о выходе новых книг;
- создан административный интерфейс для управления каталогом.

Во время выполнения курсовой работы были решены следующие трудности:

- разработана реляционная структура базы данных с поддержкой сложных связей между сущностями;
- реализована корректная обработка ошибок и исключений, обеспечивающая стабильность системы;
- внедрены механизмы безопасности, включая аутентификацию, авторизацию и защиту от уязвимостей;
- оптимизировано взаимодействие между слоями приложения в рамках трёхзвенной архитектуры.

В ходе реализации проекта были изучены и применены следующие подходы и технологии:

- архитектурный паттерн MVC, разделяющий представление, бизнес-логику и доступ к данным;
- ORM-подход на основе JPA/Hibernate, упрощающий управление данными;

- шаблонизация с использованием Thymeleaf;
- технология Spring Security для обеспечения безопасной работы системы.

Достоинствами разработанной системы можно считать:

- удобный и интуитивно понятный пользовательский интерфейс;
- возможность расширения функциональности без изменения основной архитектуры;
- автоматическая рассылка уведомлений пользователям;
- гибкость и структурированность базы данных;
- кроссплатформенная работа серверной части.

Недостатками приложения являются:

- отсутствие мобильной версии интерфейса;
- зависимость от интернет-подключения;
- необходимость ручного заполнения данных в административной панели при большом объёме информации;
- отсутствие автоматических рекомендаций и интеллектуального поиска.

На данном этапе разработки система является логически завершённой и полностью работоспособной. В случае продолжения развития проекта в сторону расширения функционала можно добавить:

- интеллектуальный поиск и рекомендации на основе пользовательской активности;
- адаптацию под мобильные приложения;
- расширение модуля статистики и визуализации данных;
- автоматизированный импорт и экспорт данных из внешних источников;
- систему рецензий, отзывов и рейтингов книг.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Законодательные и нормативные акты:

1. ГОСТ Р 7.0.12–2011. Библиографическая запись. Сокращение слов и словосочетаний на русском языке. Общие требования и правила. — М.: Стандартинформ, 2012. — 61 с.

2. ГОСТ 7.1–2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления. — М.: Стандартинформ, 2010. — 92 с.

3. ГОСТ 7.32–2017. Отчёт о научно-исследовательской работе. Структура и правила оформления. — М.: Стандартинформ, 2017. — 47 с.

4. ГОСТ 7.82–2001. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. — М.: ИПК Изд-во стандартов, 2001. — 39 с.

5. ГОСТ Р 7.0.100–2018. Библиографическая запись. Общие требования и правила составления. — М.: Стандартинформ, 2018. — 122 с.

6. ГОСТ Р 7.0.5–2008. Библиографическая ссылка. Общие требования и правила составления. — М.: Стандартинформ, 2008. — 32 с.

7. Единая система программной документации. Общие положения. — М.: Стандартинформ, 2005. — 128 с.

Учебная и научная литература:

8. Урванов, Ф. В. Spring и Spring Boot. Разработка облачных приложений на Java : учебное пособие. — М.: ДМК Пресс, 2021. — 480 с.

9. Курбатова, И. В., Печкуров, А. В. Основы программирования на языке Java : учебное пособие для вузов. — М.: ИНФРА-М, 2020. — 368 с.

10. Пруцков, А. В. Программирование на языке Java : введение в курс с примерами и практическими заданиями. — М.: ФОРУМ, 2019. — 352 с.

11. Иванова, Г. С. Технология программирования : учебник для вузов. — 3-е изд. — М.: Кнорус, 2018. — 333 с.

Интернет-документы:

12. Recommendation Systems in Libraries: an Application with Heterogeneous Data Sources [Электронный ресурс] — URL: https://arxiv.org/abs/2303.11746 (дата обращения: 09.12.2025)

13. Evaluating Digital Library Search Systems by Using Formal Process Modelling [Электронный ресурс] — URL: https://arxiv.org/abs/2304.11651 (дата обращения: 09.12.2025)

14. Литрес [Электронный ресурс] — URL: https://www.litres.ru/ (дата обращения: 25.11.2025)

15. АСТ [Электронный ресурс] — URL: https://ast.ru/ (дата обращения: 25.11.2025)

16. Издательский дом Мещерякова [Электронный ресурс] — URL: https://www.idmkniga.ru/ (дата обращения: 25.11.2025)

17. Spring Boot Documentation [Электронный ресурс] — URL: https://docs.spring.io/spring-boot/docs/current/reference/ (дата обращения: 28.11.2025)

18. Spring Framework Documentation [Электронный ресурс] — URL: https://docs.spring.io/spring-framework/docs/current/reference/ (дата обращения: 28.11.2025)

19. Thymeleaf — официальный сайт шаблонизатора [Электронный ресурс] — URL: https://www.thymeleaf.org (дата обращения: 28.11.2025)

20. Bootstrap — CSS-фреймворк адаптивной вёрстки [Электронный ресурс] — URL: https://getbootstrap.com (дата обращения: 28.11.2025)

21. Chart.js — библиотека визуализации данных [Электронный ресурс] — URL: https://www.chartjs.org (дата обращения: 05.12.2025)

22. Draw.io — онлайн-система построения диаграмм [Электронный ресурс] — URL: https://draw.io (дата обращения: 29.11.2025)

23. PostgreSQL Documentation [Электронный ресурс] — URL: https://www.postgresql.org/docs (дата обращения: 28.11.2025)

24. Oracle Java SE Documentation [Электронный ресурс] — URL: https://docs.oracle.com/javase (дата обращения: 28.11.2025)

25. Apache Maven Documentation [Электронный ресурс] — URL: https://maven.apache.org (дата обращения: 29.11.2025)

26. IntelliJ IDEA Documentation [Электронный ресурс] — URL: https://www.jetbrains.com/help/idea (дата обращения: 27.11.2025)

27. Microsoft Word [Электронный ресурс] — URL: https://www.microsoft.com/ru-ru/microsoft-365/word (дата обращения: 26.11.2025)

28. JDBC — официальная документация Oracle [Электронный ресурс] — URL: https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/ (дата обращения: 28.11.2025)

29. Foundation — официальный сайт CSS-фреймворка [Электронный ресурс] — URL: https://get.foundation (дата обращения: 28.11.2025)

30. MySQL Documentation — официальная документация СУБД MySQL [Электронный ресурс] — URL: https://dev.mysql.com/doc/ (дата обращения: 28.11.2025)

ПРИЛОЖЕНИЕ

Приложение А

Ссылка на репозиторий проекта

https://github.com/AknasMacefg/ID23-1_MaslovAN_STP_Kursovaya