

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

Кафедра информационных технологий

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 .....	3
ГЛАВА 1 ПРЕДПРОЕКТНОЕ ИСЛЕДОВАНИЕ .....	5
1.1 Описание предметной области .....	5
1.2 Анализ литературы .....	6
1.3 Сравнительный анализ аналогов .....	7
1.4 Постановка задачи .....	8
1.5 Характеристика инструментальных средств разработки .....	10
ГЛАВА 2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРОГРАММЫ .....	13
2.1 Анализ требований и определение спецификаций программного обеспечения .....	13
2.2 Проектирование программного обеспечения .....	16
2.3 Разработка информационной-системы .....	22
2.4 Отладка и тестирование программы .....	35
ЗАКЛЮЧЕНИЕ .....	37
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	39
ПРИЛОЖЕНИЕ .....	42

## 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.

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.

В связи с этим, целью курсового проекта является решение проблемы эффективного доступа к актуальной информации о текущих и будущих книгах издательского дома путем разработки веб-ориентированной информационно-справочной системы издательского дома, обеспечивающей

структурированное хранение данных, удобный доступ к каталогу книг, средства поиска и фильтрации информации, а также автоматизацию взаимодействия с пользователями за счёт системы уведомлений и персонализированных функций.

Для достижения цели были поставлены следующие задачи:

1. Проанализировать предметную область и определить требования к системе;
2. Провести анализ существующих решений;
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;

Объектом исследования является информационное обеспечение деятельности издательского дома, связанные с хранением, обновлением и предоставлением данных о книжной продукции.

Предметом исследования является предназначенная для управления каталогом издательства и взаимодействия пользователей с информационными ресурсами.

В процессе курсового проектирования будут применяться следующие теоретические методы:

- теоретический анализ нормативно-технической документации;
- моделирование структуры системы;
- системный и структурный анализ;

и эмпирические методы:

- наблюдение;
- разработка программного прототипа;
- тестирование.

Источниковой базой исследования является информация, полученная в ходе анализа интернет-сайтов.

## 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 Анализ литературы

Современные тенденции цифровизации библиотечных сервисов и издательских решений описаны в ряде исследований. Анализ этих источников показывает, что информационно-справочные системы повышают удобство пользователей, эффективность поиска и качество сервиса в информационной среде.

В работе *Recommendation Systems in Libraries: an Application with Heterogeneous Data Sources* (2023) рассматривается подход к построению рекомендательной системы на основе анализа пользовательских данных, что демонстрирует возможность улучшения пользовательского опыта через интеллектуальные механизмы обработки запросов и рекомендаций [12]. Это подтверждает, что современные системы классификации и рекомендации могут способствовать повышению вовлечённости читателей.

В исследовании *Evaluating Digital Library Search Systems by using Formal Process Modelling* (2023) анализируется эффективность поисковых систем в цифровых библиотеках, показывая, что корректная архитектура поиска улучшает точность выдачи и скорость нахождения нужной информации [13]. Это актуально для проектируемой системы, поскольку одним из её функциональных требований является расширенный поиск.

Кроме того, в исследовании отмечается, что цифровые каталоги и справочные платформы играют критическую роль в формировании современной инфраструктуры доступа к знаниям, что повышает общую цифровую грамотность пользователей и удовлетворяет растущие информационные потребности общества.

Таким образом, выделенные источники подтверждают актуальность темы, и помогают определить ключевые направления проектирования будущей системы – в частности, необходимость реализации расширенного поиска, персонализации и анализа пользовательских запросов.

### 1.3 Сравнительный анализ аналогов

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

РОСМЭН [14] – крупное российское издательство, основанное в 1992 году, специализирующееся на выпуске детской и подростковой литературы, включая художественные книги, энциклопедии, развивающие издания и популярные серии. На сайте издательства представлен каталог книг с возможностью просмотра названий, жанров и описаний, а также информация о новинках и сериях.

Тем не менее, ресурс в первую очередь ориентирован на презентацию и распространение продукции, и не только книжной, а не на предоставление расширенной справочно-информационной системы. На нём отсутствует возможность отфильтровать по жанрам, посмотреть ближайшие релизы, а также настроить получение уведомлений, а также, графический интерфейс не интуитивен.

Издательство «Просвещение» [15] – одно из крупнейших российских издательств, ориентированное на образовательную литературу, включая учебники для школ и методические пособия. На официальном сайте публикуются каталоги учебных материалов, описания издательской деятельности и цифровые образовательные продукты.

Несмотря на обширность каталога, сайт «Просвещения» функционирует преимущественно как ресурс для доступа к учебным материалам и представления продукции, причем не только книжной, но не предоставляет развитых механизмов структурированного справочного поиска, также отсутствует механизм оповещения о новых интересующих книгах, как и сам каталог ближайших релизов.

АСТ [16] – 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 [17] – 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.

#### 1.4 Постановка задачи

Входной информацией для системы будут являться 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.

## 1.5 Характеристика инструментальных средств разработки

Для реализации программного обеспечения был выбран язык программирования Java [25] – 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 [18 - 19] – 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 [29] 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[24], однако в процессе анализа рассматривалась и MySQL[31], 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 [21], обеспечивающий удобную сетку и набор готовых UI-компонентов. Также в процессе анализа рассматривался CSS-фреймворк Foundationn [30]. Foundation обладает широкими возможностями кастомизации и гибкости, однако Bootstrap был выбран благодаря большому сообществу, лучшей документации и типичности для проектов подобного масштаба. Отображение данных в интерфейсе реализуется через шаблонизатор Thymeleaf [20], который позволяет напрямую связывать данные Java-приложения с HTML-структурой страниц. Для визуализации статистики и представления данных в графическом виде применяется библиотека Chart.js [22] – 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 e-mail-рассылку.

Управление зависимостями и сборкой проекта осуществляется с помощью Apache Maven [26], 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 [23] – удобный веб-редактор, позволяющий создавать блок-схемы, диаграммы классов, ER-диаграммы и другие визуальные элементы.

Документация проекта создаётся в текстовом процессоре Microsoft Word 2023 [28], обеспечивающем удобное форматирование и структурирование пояснительной записки.

В качестве основной среды разработки для реализации программного приложения была выбрана IntelliJ IDEA [27]. 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.

Разделение пользователей системы на роли обусловлено необходимостью обеспечить безопасность, структурированность и удобство работы с информационно-справочной системой, а также поддерживать бизнес-процессы издательского дома. Каждая категория пользователей выполняет собственный набор задач, требующих разных уровней доступа к данным и функционалу системы.

Пользователи системы делятся на четыре 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 e-mail-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 редактируемом 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 e-mail-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 репозиторийев.

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

РК – первичный ключ.

FK – внешний ключ.

На рисунке 1 представлена ER-диаграмма базы данных;

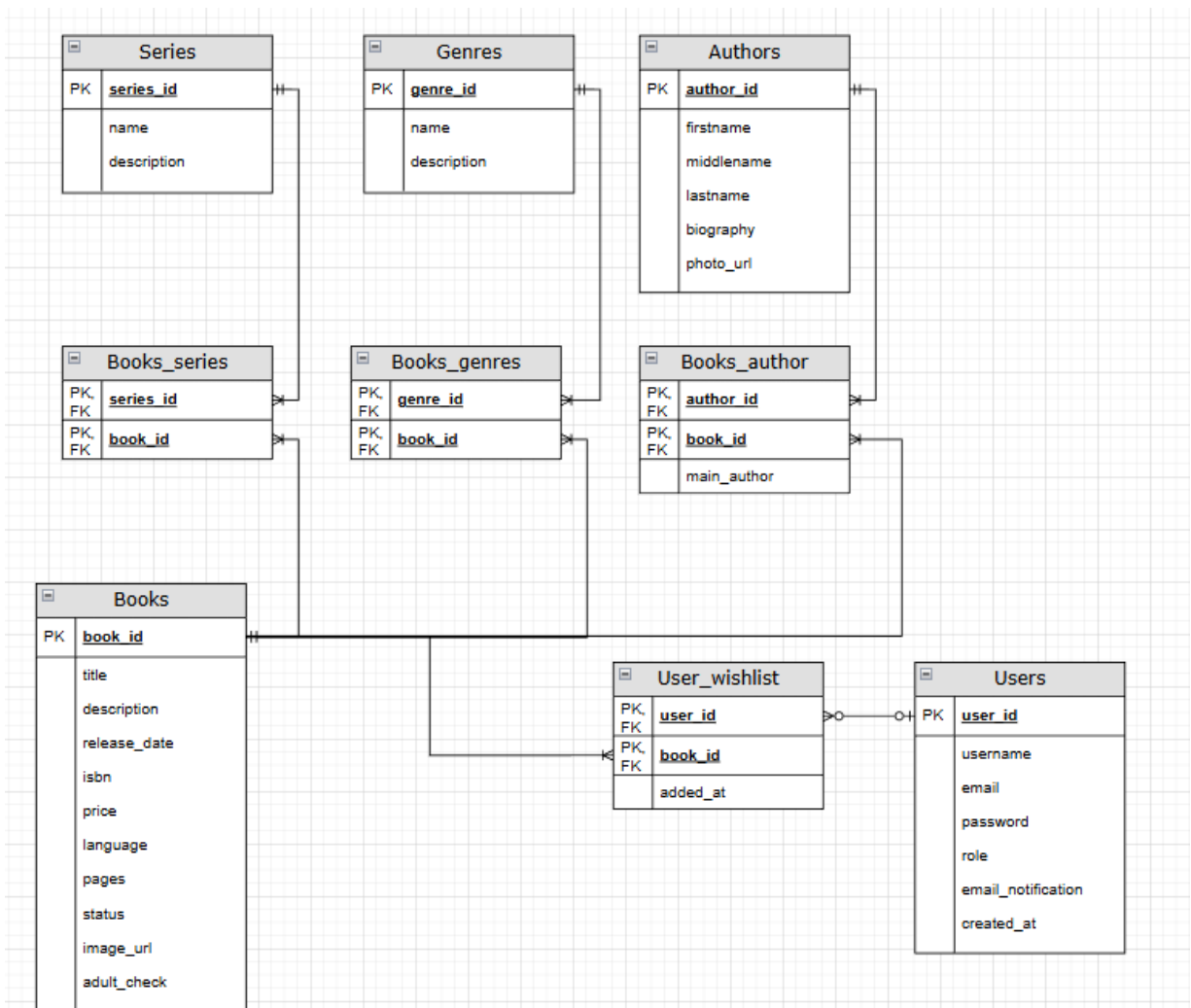


Рисунок 1. ER-диграмма информационно-справочной системы

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 (рисунок 2).

Основные компоненты:

- Controller-слой
  - принимает запросы пользователей;
  - вызывает соответствующие сервисы;
  - передаёт данные в шаблоны Thymeleaf.
- Service-слой
  - содержит бизнес-логику приложения;
  - обрабатывает данные, работает с репозиториями, вызывает e-mail-модуль.
- Repository-слой
  - взаимодействует с базой данных через Spring Data JPA;
  - выполняет CRUD-операции.
- Model-слой
  - содержит описания сущностей и связь их с таблицами БД.

- Presentation-слой
  - HTML-шаблоны Thymeleaf (templates) ;
  - CSS-стили (style.css);
  - JS скрипты (script.js);
  - Bootstrap-компоненты;
  - Chart.js для визуализации статистических данных.

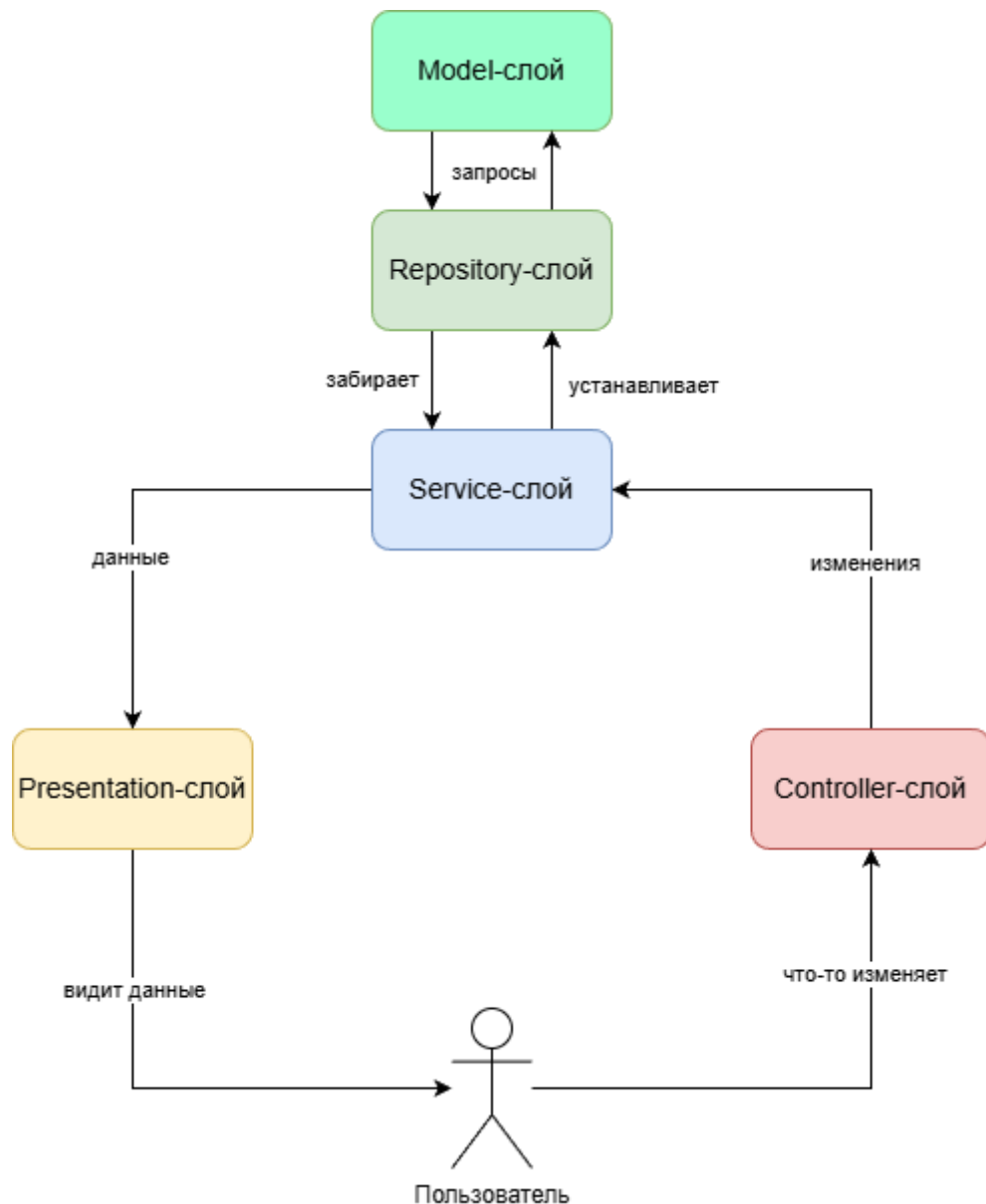


Рисунок 2. Концепция MVC + конфигурация Spring

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 3):

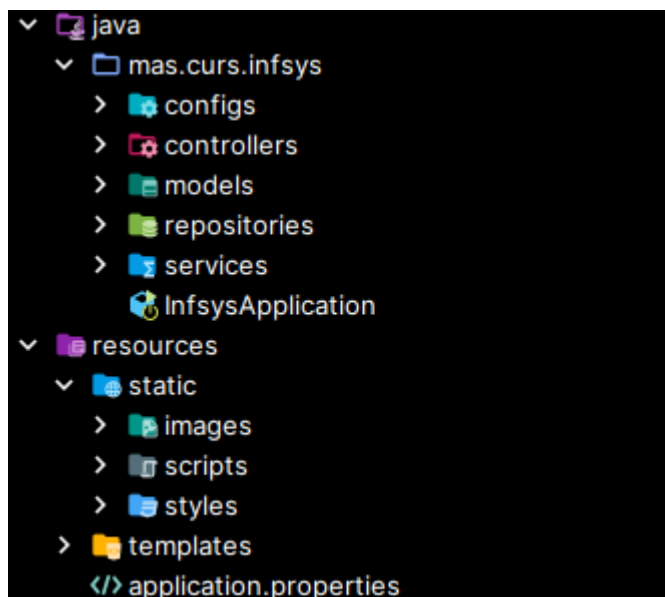


Рисунок 3. Структура директорий проекта

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 4). 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.

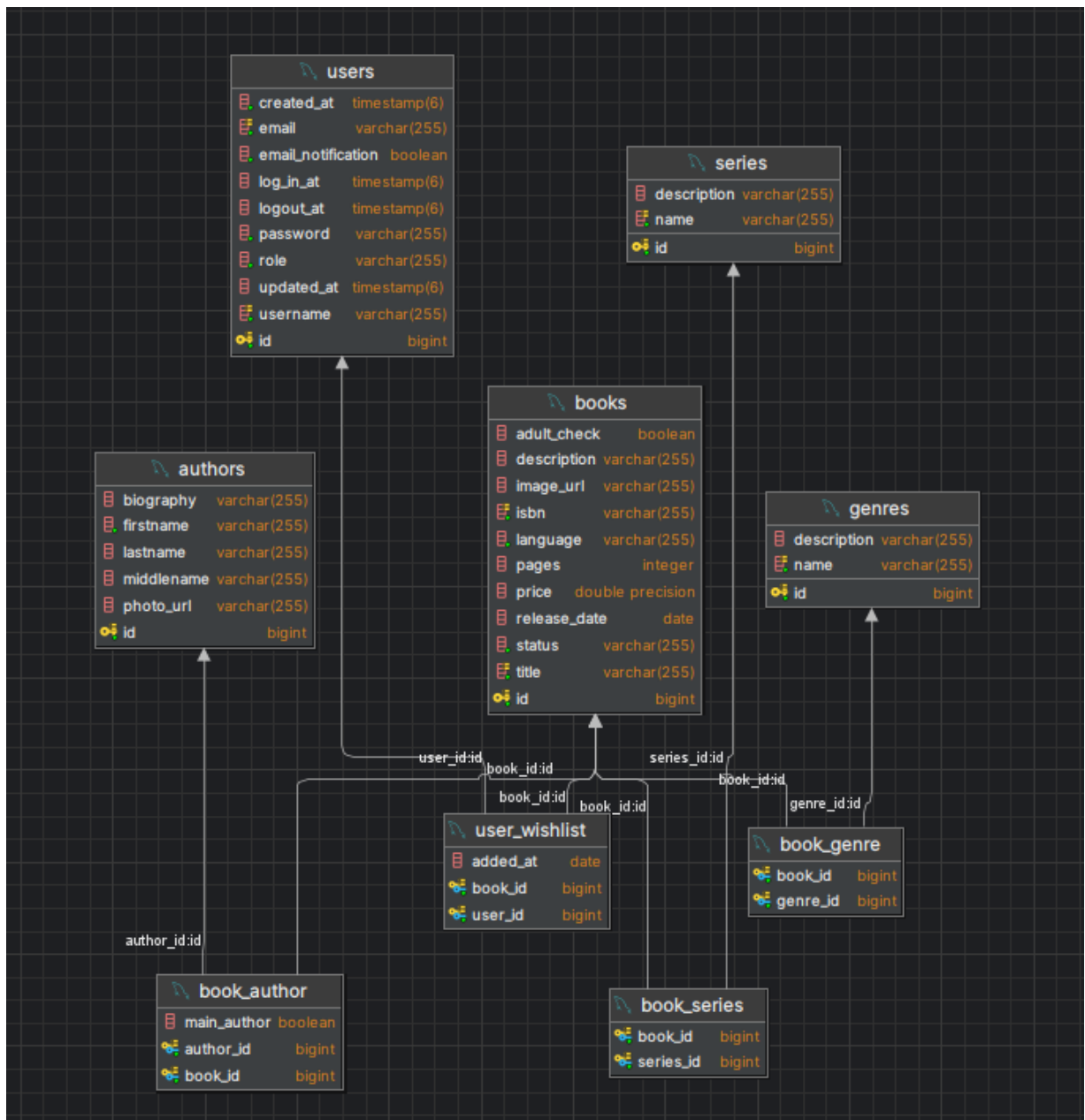


Рисунок 4. Физическая модель базы данных

Каждая сущность была реализована в виде Java-класса с аннотациями (рисунок 5):

- @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.

```
@Entity
@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 5. 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 6):

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

```

@Controller  AknasMacefg *
@RequestMapping("/books")
public class BookController {
    private final BookService bookService; 21 usages
    private final AuthorService authorService; 3 usages
    private final GenreService genreService; 4 usages
    private final SeriesService seriesService; 3 usages
    private final WishlistService wishlistService; 4 usages
    private final UserService userService; 2 usages

    public BookController(BookService bookService, AuthorService authorService, AknasMacefg
        GenreService genreService, SeriesService seriesService,
        WishlistService wishlistService, UserService userService) {
        this.bookService = bookService;
        this.authorService = authorService;
        this.genreService = genreService;
        this.seriesService = seriesService;
        this.wishlistService = wishlistService;
        this.userService = userService;
    }
}

```

Рисунок 6. BookController загружает все книги при переходе на /books

Поиск и фильтрация (рисунок 7) был реализован через репозитории Spring Data JPA, что позволяет автоматически формировать SQL-запросы на основе имени метода.

```

public List<Book> getBooksFilteredAndSorted(List<Long> genreIds, List<String> languages, 3 usages AknasMacefg *
    Boolean adultCheck, List<String> statuses,
    String sortBy, String search, int page, int pageSize) {
    List<Book> books = bookRepository.findAll();

    if (search != null && !search.trim().isEmpty()) {
        String searchLower = search.toLowerCase().trim();
        books = books.stream()
            .filter(book -> {
                if (book.getTitle() != null && book.getTitle().toLowerCase().contains(searchLower)) {
                    return true;
                }
                if (book.getDescription() != null && book.getDescription().toLowerCase().contains(searchLower)) {
                    return true;
                }
                if (book.getBookAuthor() != null) {
                    boolean authorMatch = book.getBookAuthor().stream()
                        .anyMatch(ba -> ba.getAuthor() != null && (
                            (ba.getAuthor().getFirstname() != null && ba.getAuthor().getFirstname().toLowerCase().contains(searchLower)) ||
                            (ba.getAuthor().getMiddlename() != null && ba.getAuthor().getMiddlename().toLowerCase().contains(searchLower)) ||
                            (ba.getAuthor().getLastname() != null && ba.getAuthor().getLastname().toLowerCase().contains(searchLower))
                        ));
                    if (authorMatch) return true;
                }
                if (book.getIsbn() != null && book.getIsbn().toLowerCase().contains(searchLower)) {
                    return true;
                }
                return false;
            })
            .collect(Collectors.toList());
    }
}

```

Рисунок 7. 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 8):

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

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

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

    return "view-book";
}
```

Рисунок 8. 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 9-10):

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

При регистрации выполняются:

- проверка корректности e-mail,
- 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)

```
@PostMapping("/register")
public String registerUser(@ModelAttribute("user") User user,
                           @RequestParam("confirmPassword") String confirmPassword,
                           RedirectAttributes redirectAttributes) {
    if (user.getPassword() == null || user.getPassword().length() < 6) {
        redirectAttributes.addFlashAttribute("error", "Пароль должен содержать минимум 6 символ");
        return "redirect:/register";
    }
    if (!user.getPassword().equals(confirmPassword)) {
        redirectAttributes.addFlashAttribute("error", "Пароли не совпадают");
        return "redirect:/register";
    }
    boolean success = userService.register(user);
    if (success) {
        return "redirect:/login";
    } else {
        redirectAttributes.addFlashAttribute("error", "Такой пользователь уже существует");
        return "redirect:/register";
    }
}

@GetMapping("/login")
public String showLoginForm() { return "login"; }
```

Рисунок 9. AuthController загрузка форм для логина и регистрации

```
@Bean
@Order(2)
public SecurityFilterChain webSecurity(HttpSecurity http,
                                       LoginSuccessHandler loginSuccessHandler,
                                       CustomLogoutSuccessHandler logoutSuccessHandler) throws Exception {

    http
        .authorizeHttpRequests(auth -> auth
            .requestMatchers("/", "/about", "/contacts", "/genres", "/authors", "/series", "/book")
            .permitAll()
            .anyRequest().authenticated()
        )
        .formLogin(form -> form
            .loginPage("/login")
            .usernameParameter("email")
            .successHandler(loginSuccessHandler)
            .permitAll()
        )
        .logout(logout -> logout
            .logoutUrl("/logout")
            .logoutSuccessHandler(logoutSuccessHandler)
            .deleteCookies("remember-me")
            .permitAll()
        )
        .rememberMe(remember -> remember
            .rememberMeServices(rememberMeServices())
            .key("infsys-remember-me-secret-key-2024")
            .tokenValiditySeconds(86400 * 30) // 30 days
        )
        .csrf(AbstractHttpConfigurer::disable);

    return http.build();
}
```

Рисунок 10. SecurityConfig логика обработки доступа и перенаправления

Список желаемых книг реализован через сервис (рисунок 11), 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 11. WishlistService часть кода добавления и удаления книг в список  
желаемого

Для сотрудников издательства создан набор функций (рисунок 12):

- 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}")
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. Исправьте и повторите попытку.");
                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 12. 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 13):

- 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 e-mail (рисунок 14).

Этот модуль работает полностью автономно и не требует участия пользователя.

```
@Scheduled(cron = "0 0 2 * * ?") @AknasMacefg *
public void dailyMaintenance() {
    sendDeletionWarnings();
    deleteOldUsers();
    updateBookStatusesAndNotify();
}
```

Рисунок 13. ScheduledTasks функции которые запускаются по расписанию в

2:00

```
public void sendBookReleaseNotification(String toEmail, String bookTitle, String bookUrl) { 1 usage @AknasMacefg
    try {
        SimpleMailMessage message = new SimpleMailMessage();
        message.setTo(toEmail);
        message.setSubject("Книга из вашего списка желаний вышла!");
        message.setText("Здравствуйте!\n\n" +
            "Книга \"" + bookTitle + "\" из вашего списка желаний теперь доступна!\n\n" +
            "Посмотреть книгу: " + bookUrl + "\n\n" +
            "С уважением,\n" +
            "Команда ИД \"Инженерия Данных\"");
        mailSender.send(message);
    } catch (Exception e) {
        System.err.println("Failed to send email to " + toEmail + ": " + e.getMessage());
    }
}

public void sendDeletionWarning(String toEmail, String username) { 1 usage @AknasMacefg
    try {
        SimpleMailMessage message = new SimpleMailMessage();
        message.setTo(toEmail);
        message.setSubject("Ваш аккаунт будет удален через месяц");
        message.setText("Здравствуйте, " + username + "!\n\n" +
            "Мы заметили, что вы не заходили в систему более 11 месяцев.\n\n" +
            "Согласно политике нашей системы, неактивные аккаунты удаляются через год после последнего входа.\n\n" +
            "Ваш аккаунт будет автоматически удален через месяц, если вы не войдете в систему.\n\n" +
            "Чтобы сохранить свой аккаунт, просто войдите в систему в течение ближайшего месяца.\n\n" +
            "С уважением,\n" +
            "Команда ИД \"Инженерия Данных\"");
        mailSender.send(message);
    } catch (Exception e) {
        System.err.println("Failed to send deletion warning email to " + toEmail + ": " + e.getMessage());
    }
}
```

Рисунок 14. 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 15);
- панель поиска, сортировки и фильтров – жанры, серии, статус релиза (books.html, authors.html, genres.html, series.html, рисунок 19);

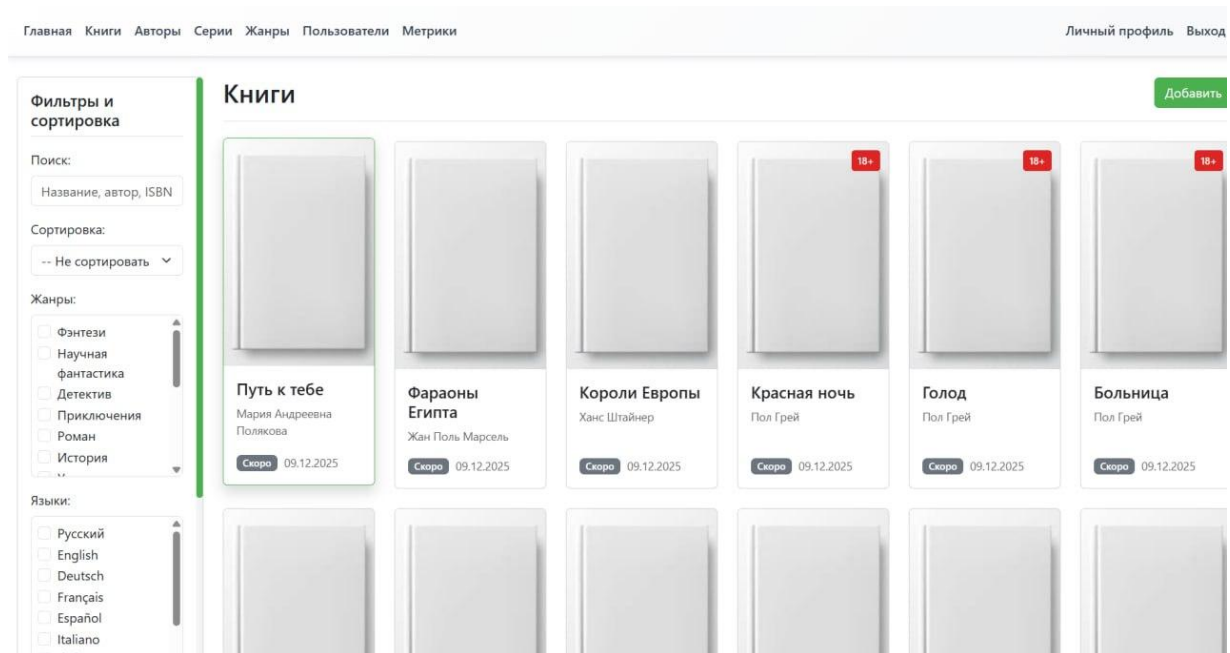


Рисунок 15. books.html

- карточка книги, автора, жанра, серии – описание, обложка (если есть), информация (view-book.html, view-author.html, view-genre.html, view-series.html, рисунок 16);
- страница изменения книги, автора, жанра, серии – поля для изменения информации (edit-book.html, edit-author.html, edit-genre.html, edit-series.html)



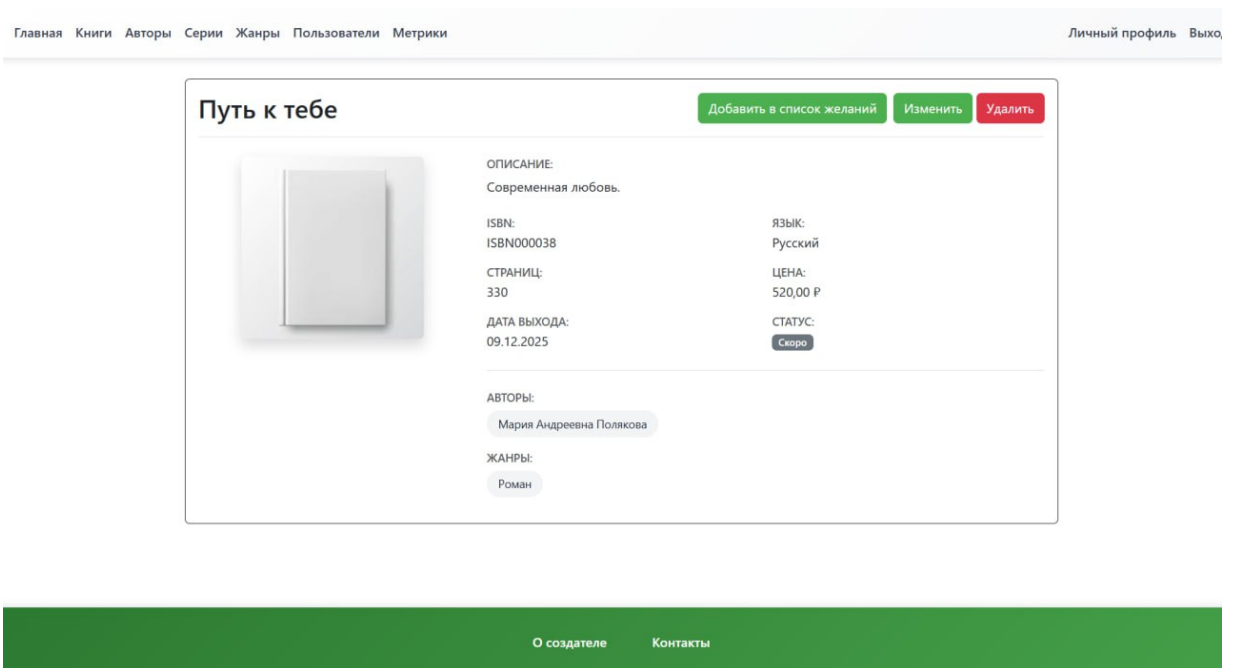


Рисунок 16. view-book.html

- страницы входа и регистрации (login.html, register.html)
- страницы связанные с пользователями – список пользователей, изменение пользователей (users.html, edit-user.html)
- личный кабинет – e-mail, смена e-mail и пароля (profile.html);
- список желаний – карточки избранных книг (profile.html, рисунок 17);

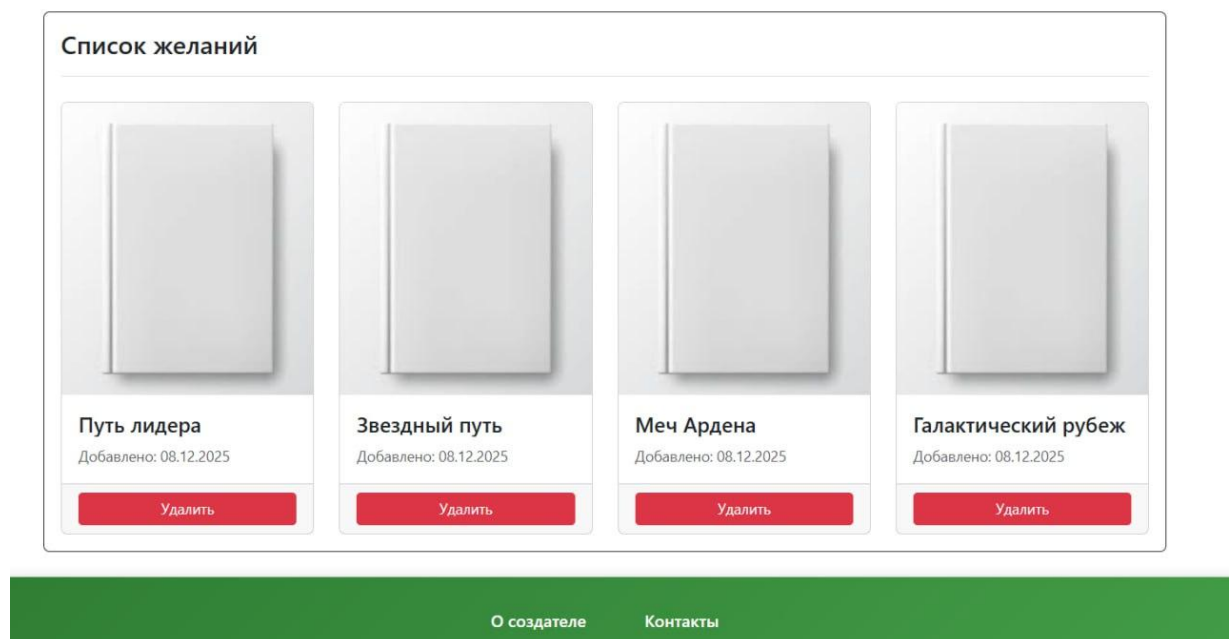


Рисунок 17. Часть profile.html

- 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 18).

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.

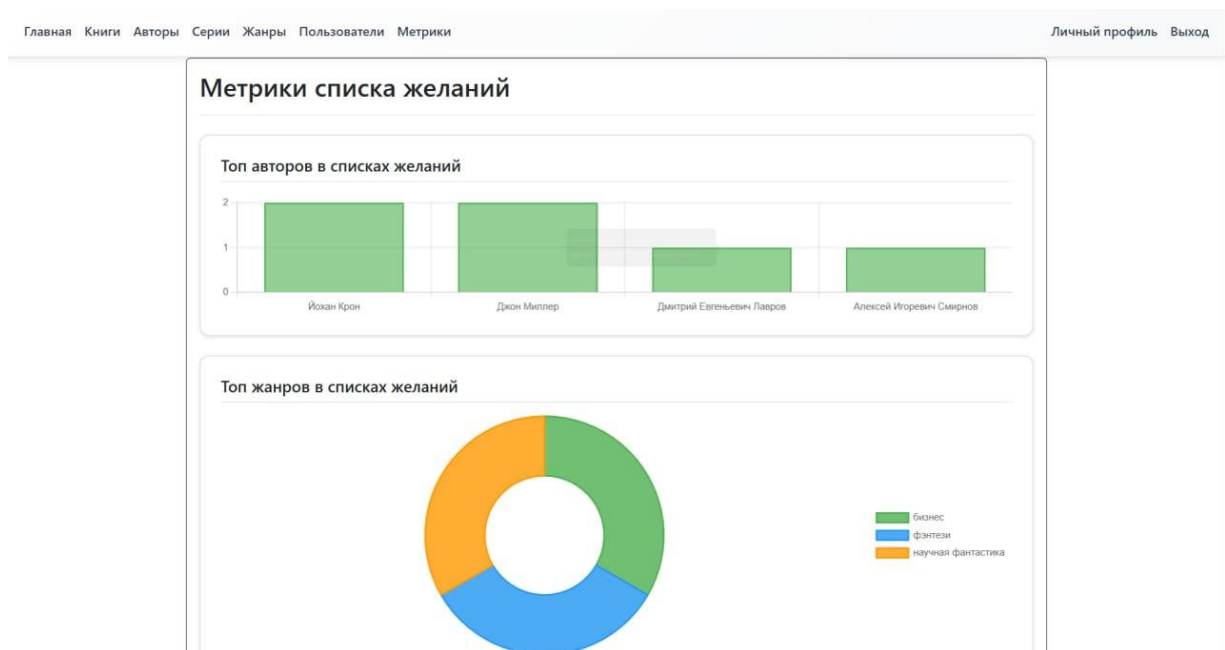


Рисунок 18. metrics.html

Обработка ошибок включает:

- глобальный перехват исключений @ExceptionHandler (рисунок 19);
- кастомная HTML-страница ошибок (error.html):
  - 404 – страница не найдена;
  - 403 – доступ запрещён;
  - 500 – внутренняя ошибка сервера.

Такой подход исключает аварийные завершения работы программы.

```

@RequestMapping("/error")
public String handleError(HttpServletRequest request, Model model) {
    Object status = request.getAttribute(RequestDispatcher.ERROR_STATUS_CODE);

    if (status != null) {
        Integer statusCode = Integer.valueOf(status.toString());

        if (statusCode == HttpStatus.NOT_FOUND.value()) {
            model.addAttribute("errorCode", "404");
            model.addAttribute("errorMessage", "Страница не найдена");
            model.addAttribute("errorDescription", "Запрашиваемая страница не существует.");
        } else if (statusCode == HttpStatus.FORBIDDEN.value()) {
            model.addAttribute("errorCode", "403");
            model.addAttribute("errorMessage", "Доступ запрещен");
            model.addAttribute("errorDescription", "У вас нет прав для доступа к этому ресурсу.");
        } else if (statusCode == HttpStatus.METHOD_NOT_ALLOWED.value()) {
            model.addAttribute("errorCode", "405");
            model.addAttribute("errorMessage", "Метод не разрешен");
            model.addAttribute("errorDescription", "Используемый HTTP-метод не поддерживается для");
        } else if (statusCode == HttpStatus.INTERNAL_SERVER_ERROR.value()) {
            model.addAttribute("errorCode", "500");
            model.addAttribute("errorMessage", "Внутренняя ошибка сервера");
            model.addAttribute("errorDescription", "Произошла внутренняя ошибка сервера. Пожалуйс");
        } else {
            model.addAttribute("errorCode", statusCode.toString());
            model.addAttribute("errorMessage", "Ошибка");
            model.addAttribute("errorDescription", "Произошла ошибка при обработке запроса.");
        }
    } else {
        model.addAttribute("errorCode", "Ошибка");
        model.addAttribute("errorMessage", "Неизвестная ошибка");
        model.addAttribute("errorDescription", "Произошла неизвестная ошибка.");
    }
}

return "error";

```

Рисунок 19. 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 20	Нет

2	Путь к странице	Путь к несуществующей странице (/bks)	Ошибка 404	На рисунке 21	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.

## Путь к тебе

Сохранить

Отмена



Загрузить изображение

ОПИСАНИЕ:

Современная любовь.

ISBN:

----

ЯЗЫК:

Русский

ISBN должен быть в формате ISBN-10 (10 символов) или ISBN-13 (13 цифр)

ISBN должен быть в формате ISBN-10 (10 символов) или ISBN-13 (13 цифр)

СТРАНИЦ:

ЦЕНА:

Рисунок 20. Ввод неправильного текста

Главная Книги Авторы Серии Жанры Пользователи Метрики

Личный профиль Выход

404 - Страница не найдена

Запрашиваемая страница не существует.

Рисунок 21. 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. РОСМЭН [Электронный ресурс] — URL: <https://rosman.ru/> (дата обращения: 25.11.2025)

15. Просвещение [Электронный ресурс] — URL: <https://prosv.ru/> (дата обращения: 25.11.2025)

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

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

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

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

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

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

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

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



24. PostgreSQL Documentation [Электронный ресурс] — URL: <https://www.postgresql.org/docs> (дата обращения: 28.11.2025)
25. Oracle Java SE Documentation [Электронный ресурс] — URL: <https://docs.oracle.com/javase> (дата обращения: 28.11.2025)
26. Apache Maven Documentation [Электронный ресурс] — URL: <https://maven.apache.org> (дата обращения: 29.11.2025)
27. IntelliJ IDEA Documentation [Электронный ресурс] — URL: <https://www.jetbrains.com/help/idea> (дата обращения: 27.11.2025)
28. Microsoft Word [Электронный ресурс] — URL: <https://www.microsoft.com/ru-ru/microsoft-365/word> (дата обращения: 26.11.2025)
29. JDBC — официальная документация Oracle [Электронный ресурс] — URL: <https://docs.oracle.com/javase/8/docs/technotes/guides/jdbc/> (дата обращения: 28.11.2025)
30. Foundation — официальный сайт CSS-фреймворка [Электронный ресурс] — URL: <https://get.foundation> (дата обращения: 28.11.2025)
31. 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\\_Kurovaya](https://github.com/AknasMacefg/ID23-1_MaslovAN_STP_Kurovaya)