СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc192596096)

[1 ОБЗОР ОРГАНИЗАЦИИ И ЕЕ СТРУКТУРЫ 3](#_Toc192596097)

[1.1 Организация и ее деятельность 3](#_Toc192596098)

[1.2 Структура организации 4](#_Toc192596099)

[2 ОБЗОР СОСТОЯНИЯ ВОПРОСОВ 5](#_Toc192596100)

[2.1 Описание и анализ предметной области 5](#_Toc192596101)

[2.2 Обзор аналога программного обеспечения 7](#_Toc192596102)

[2.3 Постановка задачи 12](#_Toc192596103)

[2.4 Выбор и обоснование средств программирования 12](#_Toc192596104)

[2.5 Выбор и обоснование модели данных 14](#_Toc192596105)

[2.6 Функциональная модель 19](#_Toc192596106)

[3 ПРОЕКТИРОВАНИЕ СИСТЕМЫ 21](#_Toc192596107)

[3.1 Структурная и функциональная схема разрабатываемого программного обеспечения 21](#_Toc192596108)

[3.2 Проектирование технологии разработки приложения 22](#_Toc192596109)

[3.3 Диаграмма вариантов использования 23](#_Toc192596110)

[3.4 Инфологическое моделирование предметной области 24](#_Toc192596111)

[3.5 Даталогическая модель предметной области 27](#_Toc192596112)

[ЗАКЛЮЧЕНИЕ 30](#_Toc192596113)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 31](#_Toc192596114)

# ВВЕДЕНИЕ

Основной целью данной преддипломной практики является закрепление и применение на практике теоретических знаний, полученных в ходе обучения. Кроме того, практика направлена на развитие профессиональных умений и навыков, необходимых для будущей самостоятельной работы по выбранной специальности. Важными задачами также являются изучение реальной деятельности профильного предприятия, анализ его работы, а также подготовка к более глубокому освоению общепрофессиональных и специальных дисциплин учебной программы.

Студент проходит практику в Новогрудском районном исполнительном комитете по местонахождению ул. Мицкевича 11, Новогрудок, Беларусь.

Производственная практика длится с 09.02.2025 до 08.03.2025. Студент работает на должности инженера-программиста. Таким образом, он выполняет следующий спектр работ: изучение внутренних регламентов, политик и процедур, связанных с информационными системами, информационной безопасностью и IT-поддержкой; анализ документооборота и процессов, связанных с созданием, хранением, обработкой и передачей данных в организации; изучение существующих информационных систем, их архитектуры и функциональности; участие в проектировании, разработке, тестировании и документировании программного обеспечения; внесение изменений и доработок в существующие информационные системы в соответствии с потребностями пользователей.

# 

# 1 ОБЗОР ОРГАНИЗАЦИИ И ЕЕ СТРУКТУРЫ

## Организация и ее деятельность

Исполнительным и распорядительным органом на территории области, района, города, поселка, сельсовета является исполнительный комитет с правами юридического лица (исполком).

Исполкомы в пределах своей компетенции в порядке, установленном законодательством:

* представляют интересы соответствующей административно-территориальной единицы во взаимоотношениях с другими государственными органами, иными организациями и гражданами;
* разрабатывают и вносят для утверждения в местные Советы проекты инвестиционных программ, программ социально-экономического развития соответствующей административно-территориальной единицы, обеспечивают выполнение этих программ;
* разрабатывают и вносят для утверждения в местные Советы депутатов проекты прогнозов социально-экономического развития соответствующей административно-территориальной единицы, обеспечивают выполнение этих программ;
* формируют и вносят для утверждения в местные Советы депутатов региональные комплексы мероприятий, обеспечивающие реализацию государственных программ, предусматривающих финансирование за счет средств местных бюджетов (исполкомы областного и базового уровней), проекты концепций (исполкомы первичного уровня – проекты планов мероприятий) по вопросам жилищного строительства, благоустройства соответствующей территории, дорожного строительства, коммунально-бытового и социального обслуживания граждан, социальной поддержки детей, молодежи, семей, воспитывающих детей (в том числе оказание помощи к учебному году), ветеранов, инвалидов, пожилых людей, а также иных категорий граждан, определяемых законодательными актами, поддержки малого и среднего предпринимательства, здравоохранения, образования, развития физической культуры и спорта, охраны окружающей среды и рационального использования природных ресурсов, улучшения условий и охраны труда, обеспечения радиационной безопасности, охраны историко-культурного наследия и по другим вопросам местного значения и обеспечивают их реализацию;
* осуществляют составление проекта местного бюджета и готовят отчет о его исполнении;
* обеспечивают исполнение местного бюджета, поступление предусмотренных доходов и расходование бюджетных средств в соответствии с их целевым назначением;

## Структура организации

Исполкомы подразделяются на три уровня:

* первичный (сельские, поселковые, городские (городов районного подчинения);
* базовый (городские (городов областного подчинения), районные);
* областной.

В состав исполкома входят:

* председатель;
* заместители (заместитель) председателя;
* управляющий делами;
* члены исполкома.

Исполнительный комитет в пределах своих полномочий принимает решения. Решения исполкома принимаются на его заседании простым большинством голосов от полного состава тайным или открытым голосованием, в том числе поименным. Форма голосования определяется в порядке, установленном регламентом исполкома.

# ОБЗОР СОСТОЯНИЯ ВОПРОСОВ

## Описание и анализ предметной области

Электронная платформа представляет собой цифровую систему, которая упрощает взаимодействие между пользователями, производителями и потребителями. Она объединяет все стороны и способствует эффективному обмену информацией и услугами.

Система аренды товаров ориентирована на снижение затрат, связанных с их использованием. Пользователи получают возможность временного пользования необходимыми предметами без необходимости их приобретения, что значительно сокращает финансовые расходы. Вместо того чтобы покупать, пользователи могут арендовать устройства, одежду, инструменты или автомобили, что позволяет сэкономить значительные средства. Аренда также способствует более рациональному потреблению ресурсов, так как позволяет не производить лишние товары, тем самым снижая нагрузку на окружающую среду.

Кроме того, аренда исключает необходимость длительного хранения товаров, что особенно актуально для редко используемых предметов. Это помогает избежать излишних затрат на их хранение, обслуживание и транспортировку. Также аренда устройств и оборудования на краткосрочные периоды дает возможность пользователю избежать накопления ненужных предметов, которые могут занимать много места в доме или офисе.

Аренда литературных произведений способствует многократному использованию контента, обеспечивая доступ к книгам без их приобретения, что делает процесс получения литературы более экономически и экологически эффективным. Помимо этого, система аренды снижает уровень перепроизводства и потребления ресурсов, способствуя уменьшению количества отходов. Повторное использование товаров и книг снижает нагрузку на окружающую среду, поддерживая устойчивое потребление и минимизируя объемы выбрасываемых материалов.

Платформа для аренды литературных произведений представляет собой цифровую систему, обеспечивающую пользователям удобный доступ к большому количеству современных книг отечественных и зарубежных авторов. Основная цель платформы — предоставить возможность временного пользования литературными произведениями за небольшую плату, обеспечивая доступность контента при минимальных финансовых затратах для читателей.

Аренда литературного произведения осуществляется при наличии свободного экземпляра в указанную пользователем дату аренды. Срок пользования литературой определяется пользователем при оформлении заявки на аренду произведения и может составлять от 1 до 12 недель. Число книг, выдаваемых в аренду, не ограничивается.

Каталог содержит популярные литературные произведения различных жанров отечественных и зарубежных авторов, каждое из которых имеет уникальный идентификатор.

Для каждого произведения представлена следующая информация:

* название;
* уникальный идентификатор;
* имя автора;
* жанр;
* год издания;
* издательство;
* описание.

Система должна обеспечивать пользователю быстрый и удобный поиск произведений по различным критериям, таким как название, имя автора, жанр, язык, цена и рейтинг. Это позволяет значительно ускорить процесс нахождения нужного произведения и сэкономить время, что особенно важно при большом количестве контента в каталоге. Важно, чтобы интерфейс поиска был интуитивно понятен и удобен, а результаты поиска выдавались максимально точно и соответствовали запросу пользователя. Фильтрация и сортировка по таким параметрам, как жанр, автор, язык, цена и рейтинг, предоставляет пользователю дополнительные возможности для более детального выбора книги, учитывая его предпочтения и потребности. Это позволяет создать индивидуализированный опыт для каждого читателя, облегчая процесс выбора и сделав его более комфортным.

Электронная платформа аренды литературных произведений должна предоставить пользователю доступ к широкому каталогу произведений, что позволит ознакомиться с предложениями в различных жанрах, от художественной литературы до научно-популярных изданий. Важной частью платформы является персонализированный профиль пользователя, который позволяет отслеживать арендованные книги, управлять сроками аренды, а также получать рекомендации на основе предыдущих выборов и предпочтений. Это повышает удобство использования платформы, обеспечивая динамичную настройку под каждого пользователя.

Система аренды должна обеспечивать автоматическое прекращение доступа к произведению по истечении арендного срока, что гарантирует эффективное использование контента и позволяет другим пользователям воспользоваться книгой. Также важной частью функционала является система уведомлений, которая информирует пользователей о статусе аренды, сроках возврата и возможных новинках в каталоге. Наличие системы рейтингов и отзывов помогает создать обратную связь между пользователями, улучшая качество обслуживания и помогая другим пользователям принимать решения о выборе произведений.

Программа должна также обеспечивать функционал для администрирования хранилища книг, включая возможность добавления и удаления произведений из каталога. Это важно для поддержания актуальности и полноты ассортимента, а также для быстрого обновления базы данных с учётом новых поступлений или изменений в доступности произведений. Администраторы платформы должны иметь возможность редактировать информацию о произведениях, таких как название, автор, жанр, описание и другие параметры, что позволяет поддерживать точность и актуальность данных в каталоге. Возможность удаления книг из базы данных необходима для исключения недоступных или устаревших произведений, что способствует улучшению качества обслуживания пользователей и повышению эффективности работы платформы в целом.

## Обзор аналога программного обеспечения

Для разработки программного средства необходимо изучить существующие аналоги, определить их преимущества и недостатки, а также проанализировать современные тенденции в данной сфере. Это позволит выявить ключевые аспекты, которые необходимо учесть при создании собственной платформы, а также определить возможные пути её усовершенствования. Анализ аналогичных решений поможет избежать распространённых ошибок, учесть передовой опыт и разработать более эффективное и удобное программное средство.

Приложения для аренды товаров обладают рядом особенностей, обеспечивающих их функциональность. Помимо стандартных возможностей, таких как просмотр каталога, изучение информации о товаре и его стоимости, в приложении должна быть реализована возможность размещения товара на площадку, добавления к ним описания и условия аренды. Наличие личного кабинета позволяет каждому пользователю управлять своими заказами, и просматривать историю аренд. Такой подход повышает удобство работы с системой, делая процесс аренды более прозрачным и доступным.

В качестве объектов исследования были выбраны актуальные программные продукты, связанные с арендой различных товаров, включая книги. Основными критериями выбора послужили востребованность решений, представленная функциональность и соответствие современным требованиям пользователей. В процессе анализа были рассмотрены два программных продукта, предоставляющих услуги аренды. Особое внимание было уделено удобству интерфейса, возможностям поиска, условиям аренды, механизму взаимодействия пользователей с системой, а также поддержке обратной связи и рейтинговой системе.

Веб-приложение «RenBy» предоставляет собой платформу для аренды различных категорий товаров, таких как спортивные товары, товары для туризма, садовые инструменты, детсткие товары и т.д. Интерфейс приложения представлен на рисунке 2.1.

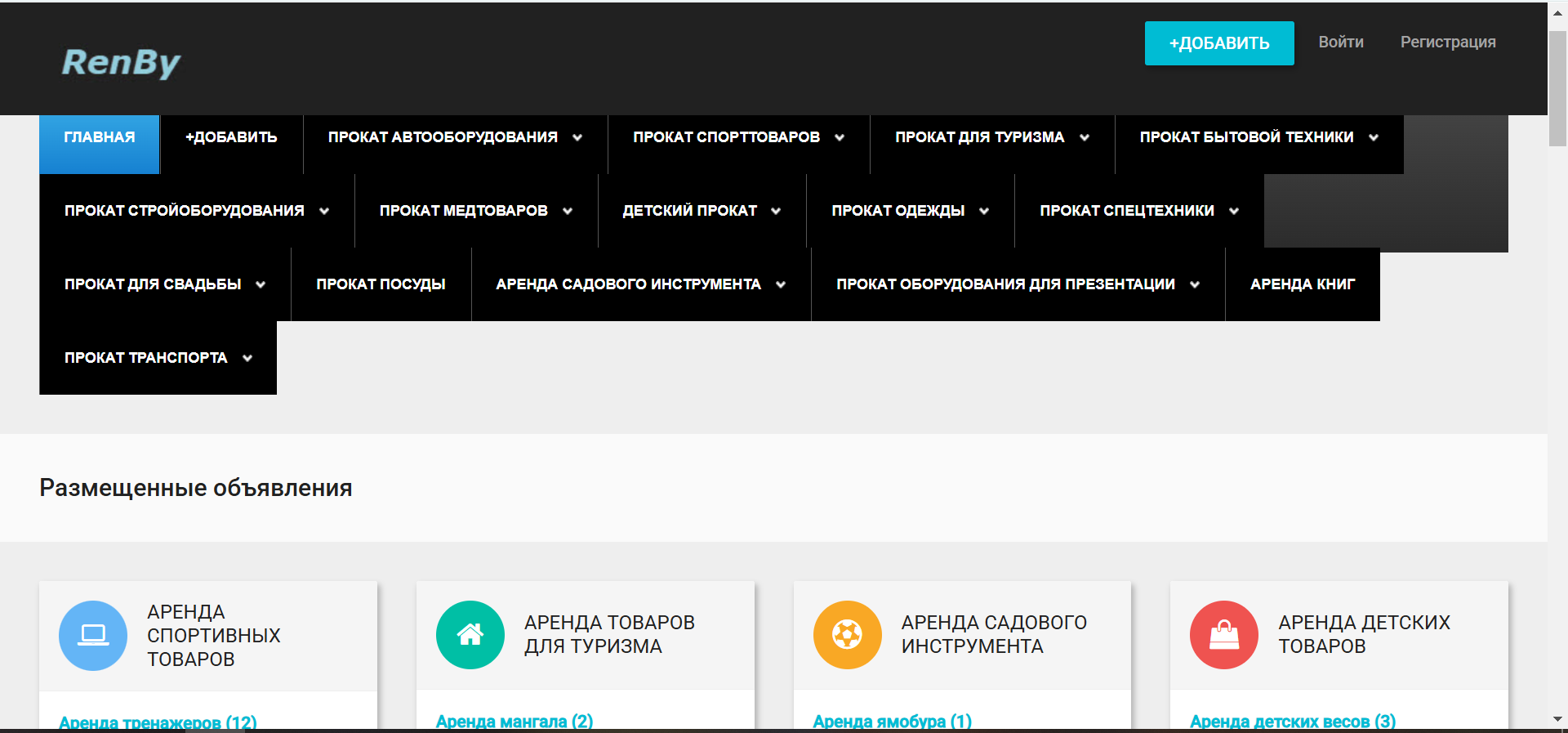


Рисунок 2.1 – Интерфейс программного средства «RenBy»

Недостаток данного приложения – это отсутствие поисковой строки. Для поиска нужного товара необходимо перейти в определенный раздел, представленный в верхней части главной страницы. Интерфейс раздела «Аренда книг» представлен на рисунке 2.2.

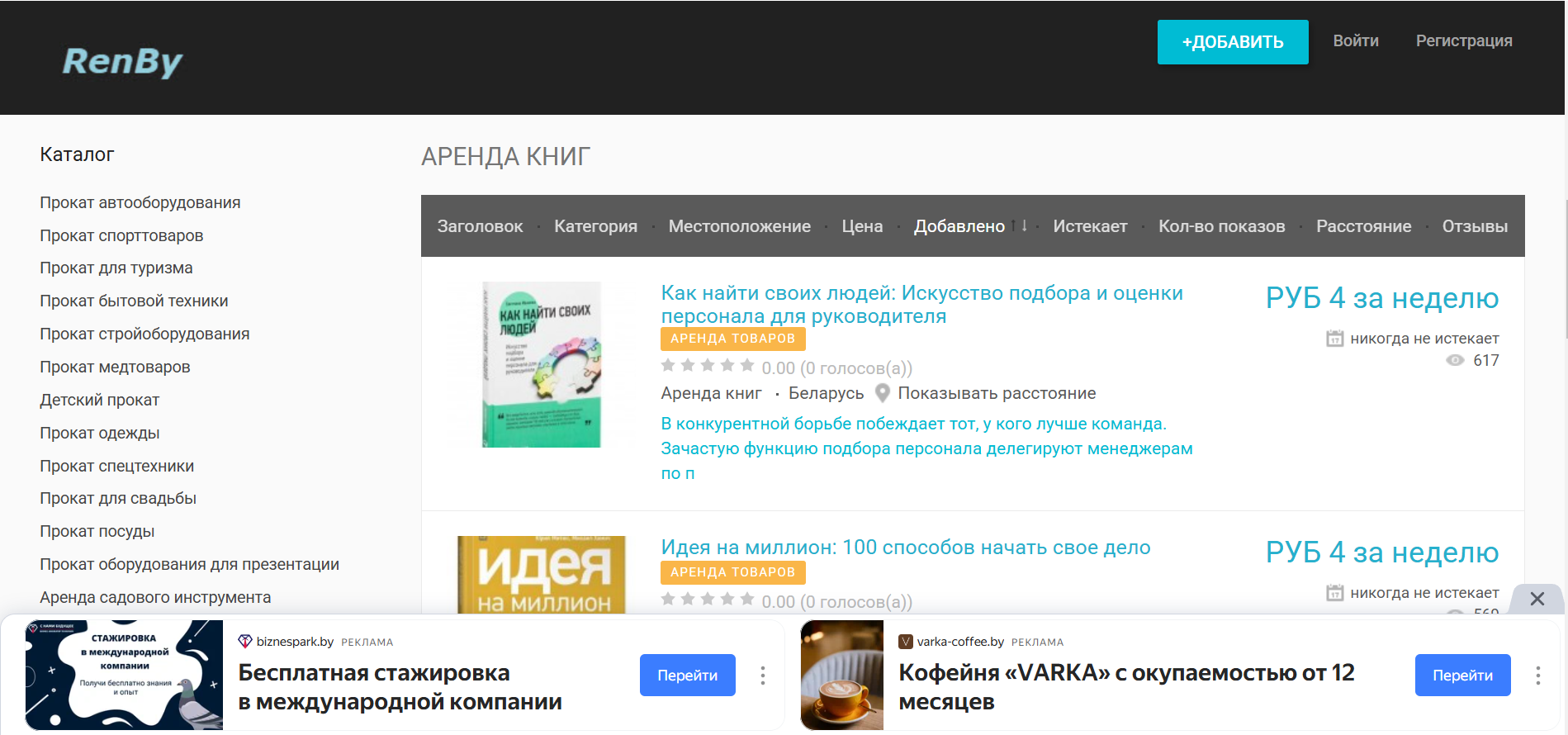


Рисунок 2.2 – Интерфейс раздела «Аренда книг»

К преимуществам данной платформы можно отнести широкий выбор параметров фильтрации, а также наличие визуального отображения товара с кратким описанием, что упрощает процесс выбора для пользователей. Однако избыточное количество рекламных объявлений затрудняет восприятие основной информации, создавая визуальную перегруженность интерфейса. Кроме того, дизайн веб-сайта выглядит устаревшим.

При переходе на страницу выбранного товара (рис. 2.3 и 2.4) выявлены существенные недостатки, влияющие на удобство использования платформы. В частности, сайт не обладает корректной адаптивностью, что может затруднять его использование на различных устройствах, особенно на мобильных. Отсутствует форма для выбора периода аренды, что ограничивает возможности пользователя в управлении условиями временного пользования литературным произведением.

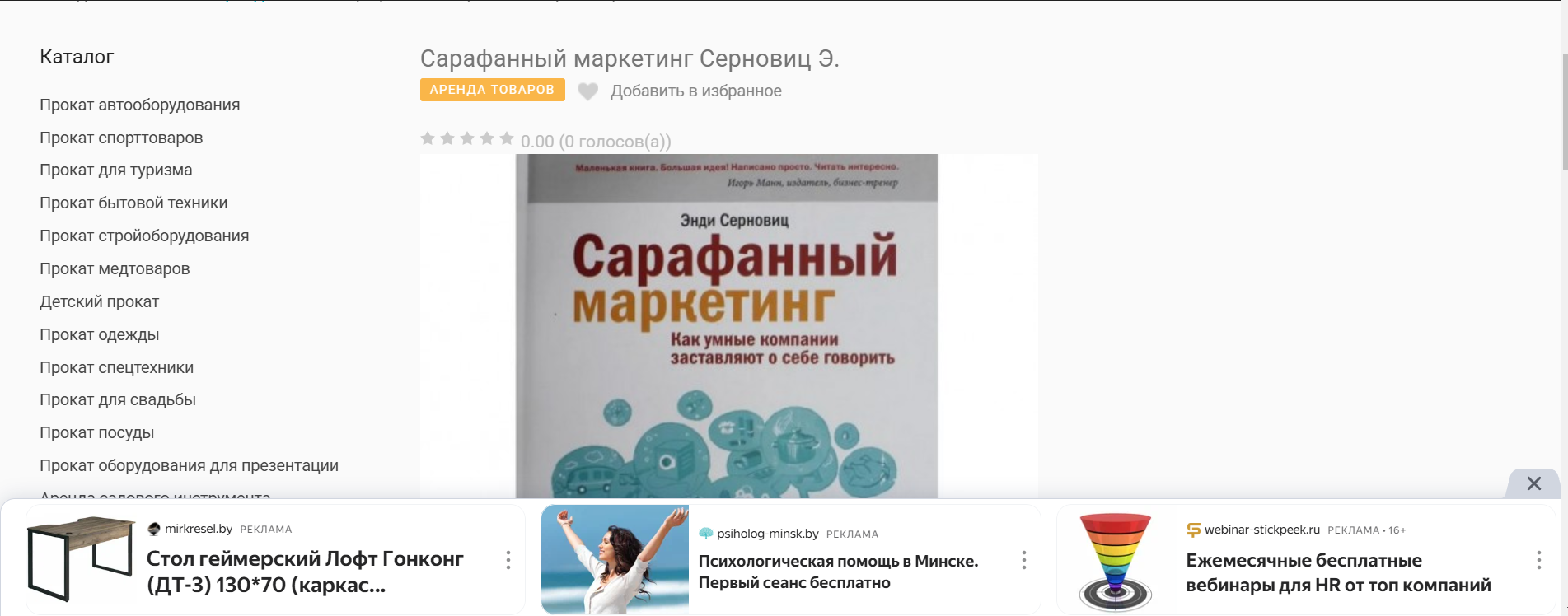


Рисунок 2.3 – Страница отображения товара

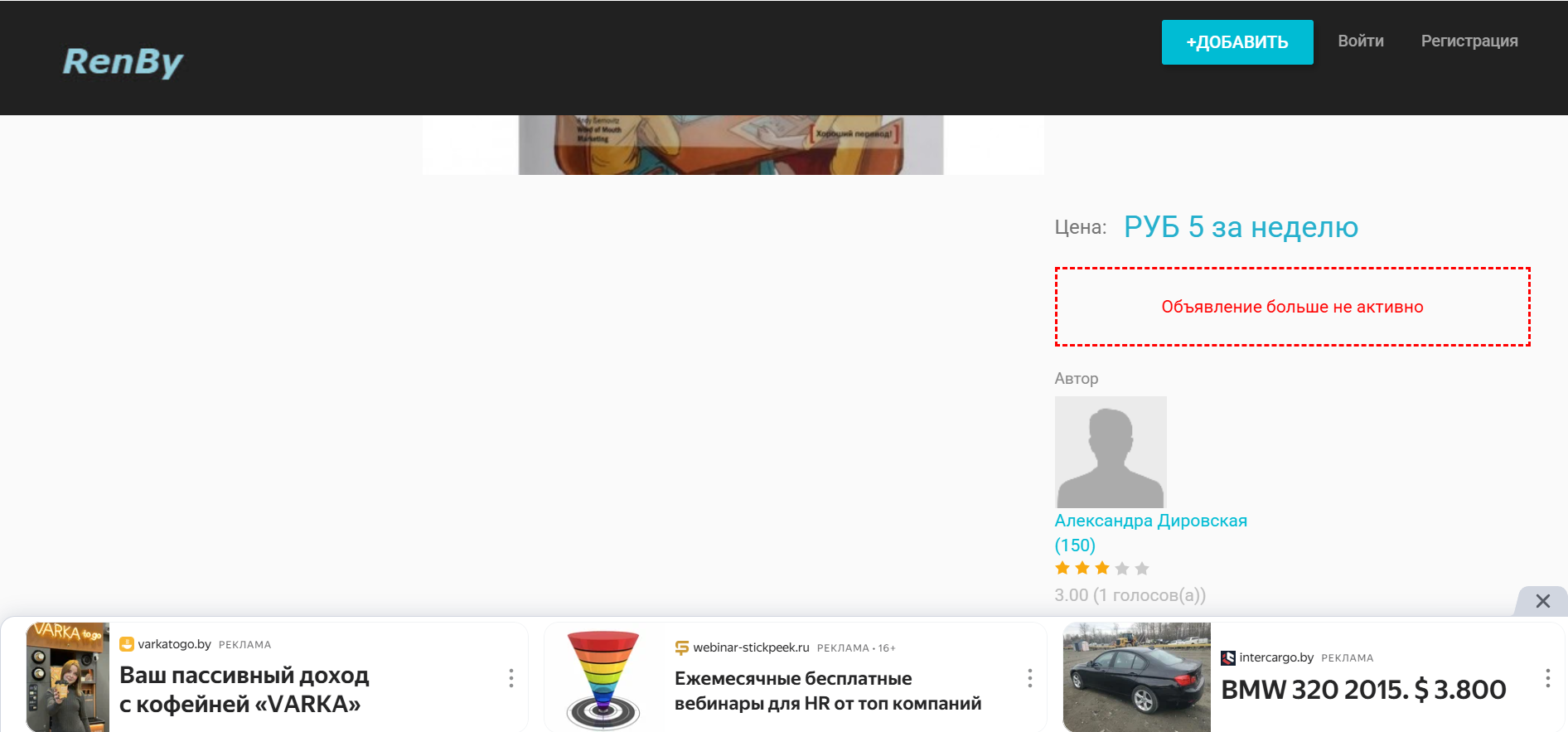


Рисунок 2.4 – Страница отображения товара

Кроме того, на странице товара не представлены важные характеристики книги, такие как жанр, язык, год издания и издательство, что затрудняет пользователям процесс выбора произведения.

Дополнительным недостатком является отсутствие системы уведомлений, информирующих пользователя о сроках аренды и приближающейся дате её завершения. Такая функция позволила бы избежать непреднамеренного истечения срока доступа к произведению и повысить уровень комфорта при пользовании сервисом.

Из преимуществ платформы «RenBy» можно выделить возможность оставить отзыв и рейтинг произвдению (рис. 2.5).

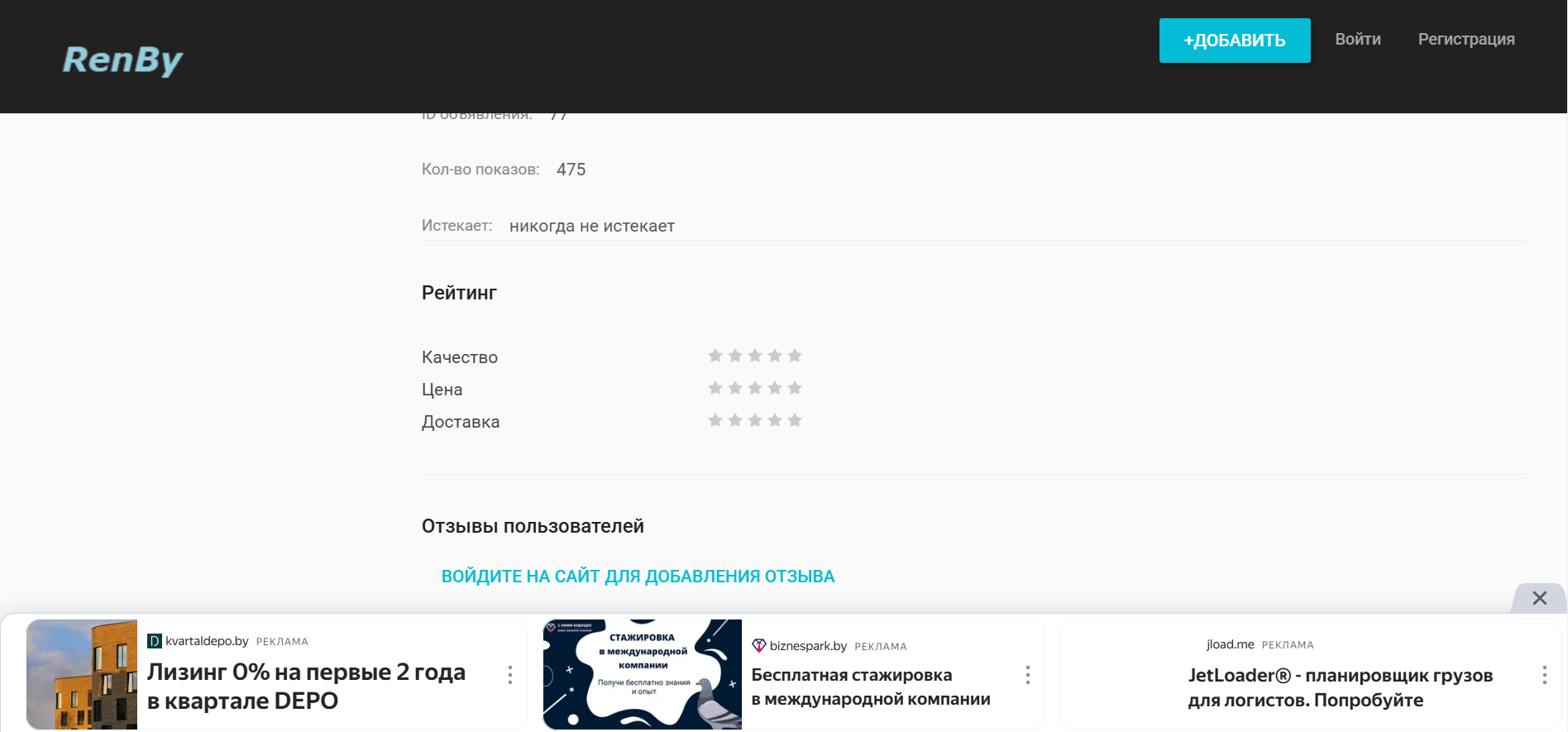


Рисунок 2.5 – Форма для заполнения информации о рейтинге

Приложение «Прокат Минск» имеет стильный дизайн, который направляет внимание пользователя на основнойй функционал сайта. На главной странице расположен номер телефона и адрес организации. Также на сайте представлен баннер с наименованием и уникальным торговым предложением организации, что упрощает понимае основных услуг, предоставляемых компанией. Интерфейс представлен на рисунке 2.6.



Рисунок 2.6 – Интерфейс приложения «Прокат Минск»

К преимуществам веб-приложения можно отнести наличие поисковой строки (рис. 2.7), позволяющей быстро находить необходимые товары. Однако среди недостатков следует отметить несогласованность размеров информационных блоков о товарах, что влияет на визуальное восприятие интерфейса. Также отсутствует указание периода, за который рассчитывается стоимость аренды, что может вызывать неудобства для пользователей при оформлении заказа.

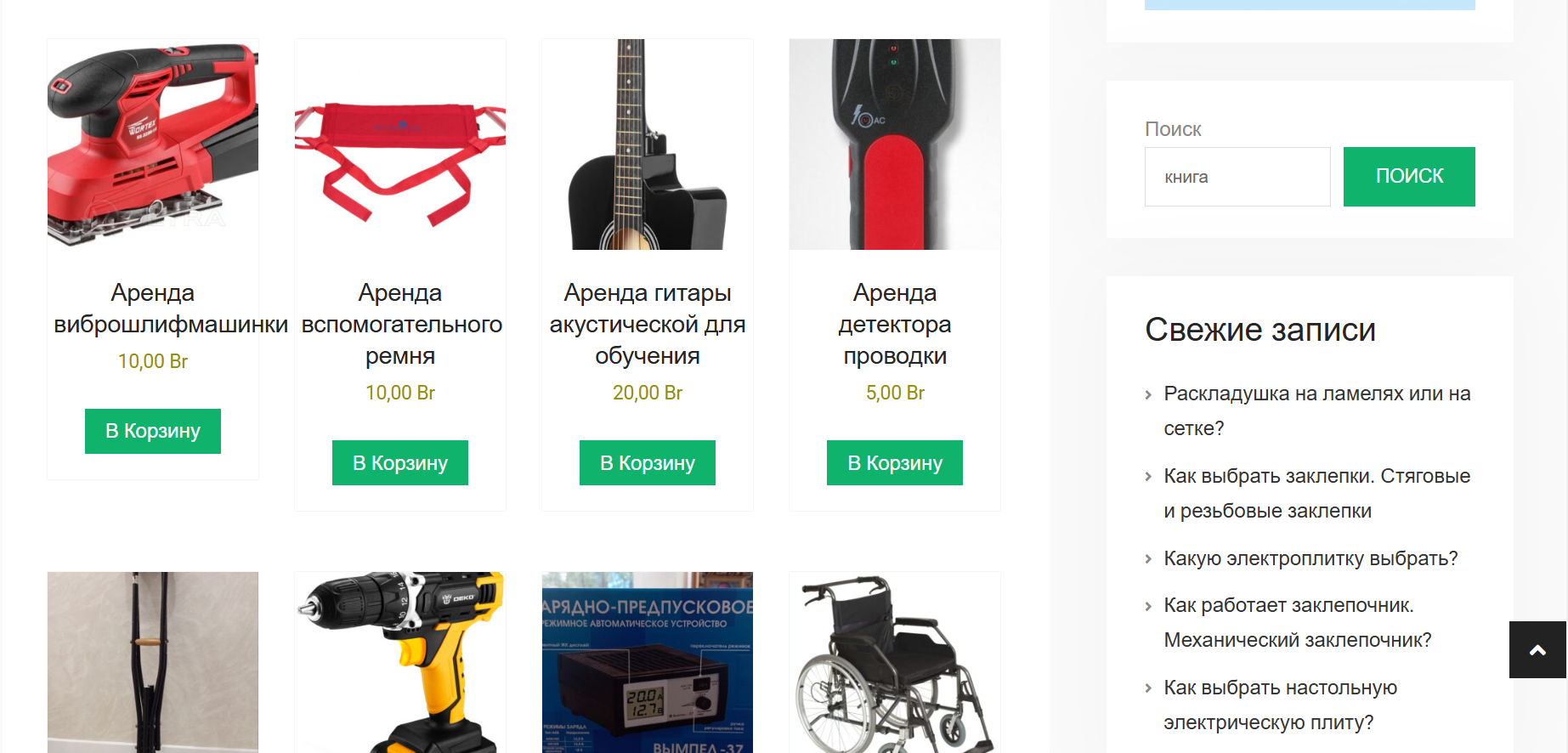


Рисунок 2.7 – Каталог товаров

На странице описания товара (рис. 2.8) отсутствует форма для выбора даты аренды, что ограничивает возможности пользователя в управлении сроками проката. Однако представлена информация об условиях аренды, а также раздел «Как забрать товар», содержащий инструкции по получению товара.

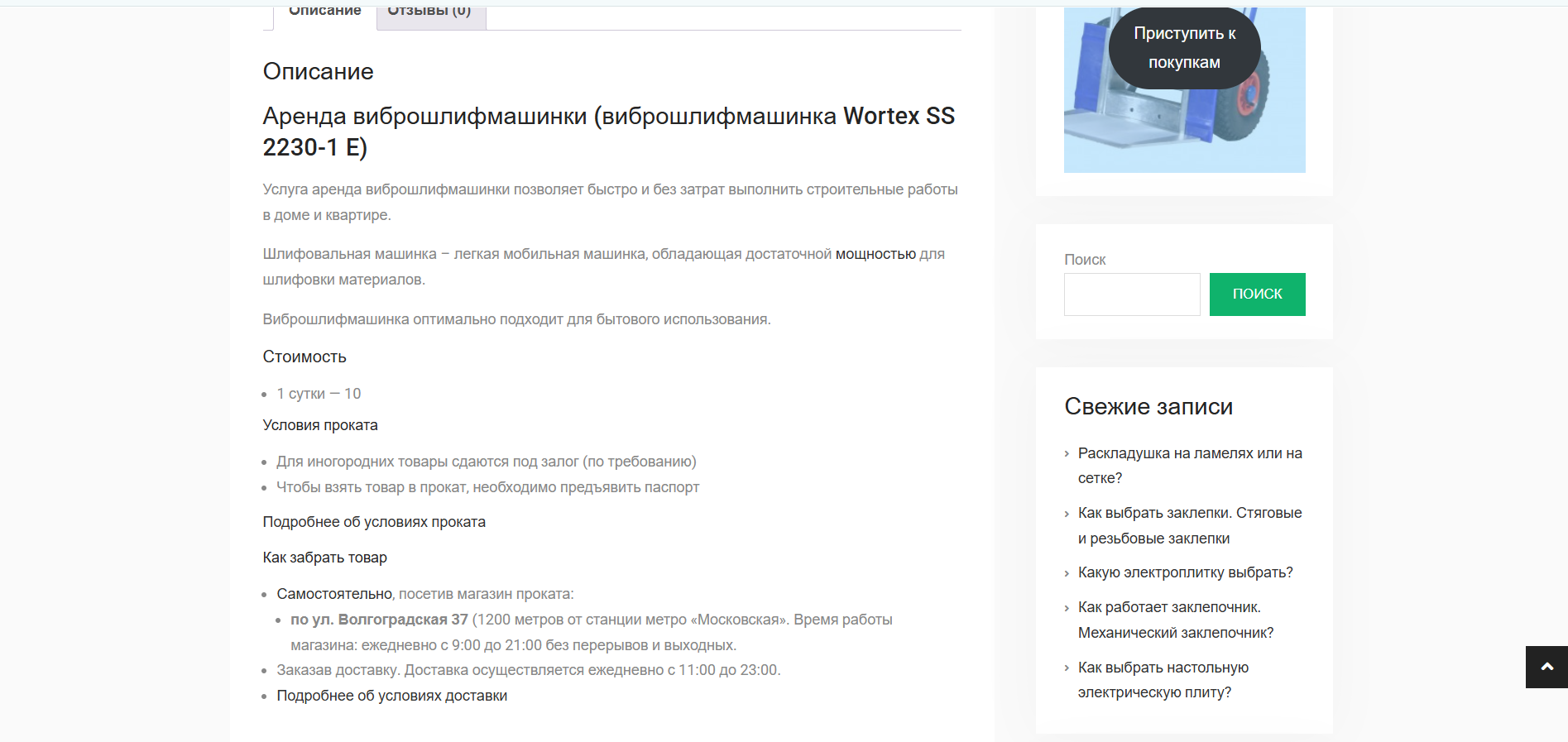


Рисунок 2.8 – Описание товара

## Постановка задачи

Электронная платформа для аренды литературных произведений должна обеспечивать удобный и доступный способ получения книг без необходимости их приобретения. Основной задачей системы является организация цифрового каталога с возможностью поиска и фильтрации произведений по различным параметрам, таким как жанр, автор, год издания и рейтинг.

Платформа должна обеспечивать простую и прозрачную процедуру аренды, позволяя пользователям оформлять временный доступ к книгам на определённый срок с последующим автоматическим прекращением права использования. Такой подход снижает финансовые затраты читателей, предоставляя им возможность пользоваться литературными произведениями по мере необходимости, без обязательного приобретения экземпляров.

Веб-приложение должно решать следующие задачи:

* реализация механизма регистрации и авторизации пользователей;
* предоставление каталога книг – формирование обширного перечня литературных произведений, включая художественную, научно-популярную, учебную и профессиональную литературу, с возможностью удобной фильтрации и поиска;
* реализация системы аренды – предоставление пользователям возможности временного доступа к книгам на установленный срок с автоматическим прекращением аренды по его истечении;
* настройка персонализированного профиля пользователя – создание индивидуального кабинета, позволяющего вносить личные данные, отслеживать арендованные книги и управлять сроками их использования;
* отправка уведомлений пользователю – автоматическое информирование о начале аренды, сроках возврата товара и подтверждении оплаты заказа;
* реализация системы рейтинга и отзывов – предоставление возможности оценить произведение и оставить отзыв после прочтения.

## Выбор и обоснование средств программирования

В рамках разработки электронной платформы предусмотрена реализация веб-приложения, доступного через сеть Интернет. Для обеспечения корректной работы системы необходимо определить программные средства, которые позволят реализовать основные функции платформы. Разрабатываемое приложение должно обеспечивать стабильное отображение в веб-среде, корректную обработку пользовательских запросов и выполнение предусмотренного функционала.

Система управления базами данных PostgreSQL представляет собой мощное и гибкое средство для хранения, обработки и управления данными. Это реляционная СУБД с расширенной поддержкой объектно-ориентированных возможностей, обеспечивающая высокий уровень надежности, масштабируемости и соответствия требованиям ACID (атомарность, согласованность, изолированность, долговечность). Доступ к данным осуществляется с использованием языка SQL, а также расширенных возможностей работы с хранимыми процедурами и пользовательскими функциями. PostgreSQL является свободно распространяемой системой с открытым исходным кодом, что делает её доступной для широкого круга разработчиков и предприятий.

Работать с PostgreSQL можно как в текстовом режиме через командную строку, так и в графическом интерфейсе. Одним из наиболее популярных инструментов для администрирования является pgAdmin, который предоставляет удобные средства для управления базами данных, их настройки, мониторинга и выполнения SQL-запросов. pgAdmin обладает интуитивно понятным интерфейсом, поддерживает визуализацию структуры базы данных и позволяет автоматизировать многие рутинные процессы администрирования.

**IntelliJ IDEA** — это интегрированная среда разработки (IDE), предназначенная для создания программного обеспечения на **Java** и других языках программирования. Данная среда обеспечивает мощные инструменты для написания, отладки и тестирования кода. IntelliJ IDEA поддерживает автоматическое дополнение кода, статический анализ, рефакторинг и широкий спектр инструментов для повышения продуктивности разработчика.

**Java** — это объектно-ориентированный язык программирования, отличающийся высокой кроссплатформенностью, безопасностью и производительностью. Приложения, написанные на Java, могут выполняться на любых устройствах, поддерживающих **Java Virtual Machine (JVM).** Этот язык широко используется в разработке веб-приложений, мобильных программ, корпоративных систем и облачных сервисов.

**Для разработки веб**-приложений на языке Java широко используется Spring Framework. Он предоставляет комплексную инфраструктуру для создания масштабируемых, гибких и поддерживаемых программных решений, упрощая реализацию сложных бизнес-логик. Spring поддерживает принципы **инверсии управления (IoC)** и **внедрения зависимостей (DI),** что способствует повышению модульности и облегчает тестирование компонентов приложения.

Преимущества Spring Framework:

* **гибкость и модульность** – разработчик может использовать только необходимые модули, что позволяет оптимизировать производительность приложения;
* **облегчение тестирования** – благодаря DI и IoC можно легко подменять зависимости и использовать мок-объекты при тестировании;
* **высокая масштабируемость** – поддержка микросервисной архитектуры позволяет эффективно разрабатывать и управлять сложными распределёнными системами;
* **интеграция с различными технологиями** – Spring легко взаимодействует с базами данных, сторонними API, инструментами мониторинга и аналитики;
* **автоматическая конфигурация (Spring Boot)** – сокращает время на настройку приложения, позволяя сосредоточиться на бизнес-логике;
* **безопасность** – встроенные механизмы защиты обеспечивают надёжную работу приложения в условиях современных угроз;
* интегрированная среда разработки Visual Studio и язык программирования JavaScript.

Visual Studio — это мощная интегрированная среда разработки (IDE), предназначенная для создания программного обеспечения на различных языках программирования, включая JavaScript. Данная среда предоставляет широкий набор инструментов для написания, отладки и тестирования кода, а также интеграцию с системами контроля версий. Visual Studio поддерживает интеллектуальное автодополнение кода, встроенные инструменты для работы с веб-технологиями и расширенные возможности для разработки кроссплатформенных приложений. Благодаря интеграции с различными плагинами и сервисами, разработчики могут эффективно разрабатывать веб-приложения, API и серверные решения.

JavaScript — это высокоуровневый, динамический и интерпретируемый язык программирования, который широко применяется в веб-разработке. Он позволяет создавать интерактивные пользовательские интерфейсы, динамически обновлять содержимое веб-страниц и взаимодействовать с сервером без необходимости перезагрузки страницы. JavaScript является ключевым компонентом веб-технологий наряду с HTML и CSS и поддерживается всеми современными браузерами.

## Выбор и обоснование модели данных

Модель данных – это абстрактное представление данных и взаимосвязей между ними, используемое для организации, хранения и обработки информации в базах данных. Она определяет, как данные структурируются, какие отношения между ними устанавливаются и какие операции можно выполнять. Выбор модели данных играет ключевую роль в проектировании информационных систем, так как от него зависит эффективность хранения, обработки и извлечения информации.

Существует несколько основных типов моделей данных:

* иерархическая модель;
* сетевая модель;
* реляционная модель;
* документ-ориентированная модель.

Чтобы разработать модель данных для информационной системы, первым шагом является выбор ее типа из представленных выше. Затем необходимо определить систему управления базами данных (СУБД), поскольку структура модели формируется с учетом особенностей выбранной СУБД. Далее рассмотрим основные разновидности моделей.

Иерархическая модель данных (рис. 2.9) представляет собой древовидную структуру, в которой информация организована по принципу «родитель-потомок». Каждый родительский узел может иметь несколько дочерних элементов, но каждый дочерний элемент связан только с одним родителем. Доступ к данным осуществляется сверху вниз, что обеспечивает высокую скорость поиска информации при четко структурированной организации. Основное преимущество такой модели — простота и высокая производительность при обработке вложенных данных, но ее недостатком является жесткая структура, усложняющая внесение изменений.



Рисунок 2.9 – Пример иерархической модели данных

Иерархическая модель используется в системах, где данные имеют естественную иерархическую организацию. Например, она применяется в файловых системах, каталогах товаров, системах управления производственными процессами и банковских базах данных для учета счетов клиентов.

Сетевая модель данных (рис. 2.10) является расширенной версией иерархической, но в отличие от нее позволяет каждому узлу иметь несколько родителей. Данные организуются в виде графа, где связи между объектами могут быть множественными, что значительно повышает гибкость работы с информацией. Такая модель позволяет быстро находить нужные данные, особенно если существует сложная система связей между элементами. Главный недостаток сетевой модели — сложность в управлении и разработке, так как требуется заранее продумывать структуру всех связей. Сетевые базы данных активно применяются в сложных корпоративных системах, таких как системы управления ресурсами предприятий (ERP), банковские системы и крупные научные базы данных.

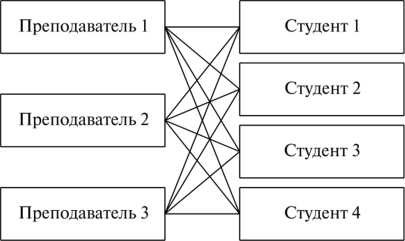


Рисунок 2.10 – Пример сетевой модели данных

Документо-ориентированная модель данных (рис. 2.11) относится к NoSQL-подходу и основана на представлении информации в виде документов, обычно хранящихся в форматах JSON, BSON или XML. В отличие от реляционной модели, данные в такой системе могут иметь гибкую структуру, что позволяет хранить вложенные объекты, массивы и неограниченное количество атрибутов без необходимости жесткого определения схемы. Основное преимущество — высокая производительность при работе с большими объемами данных и возможность горизонтального масштабирования. Однако недостатком является сложность выполнения сложных аналитических запросов и отсутствие строгих связей между объектами. Документо-ориентированные базы данных широко используются в веб-разработке, системах управления контентом, облачных приложениях, интернет-магазинах, мобильных приложениях и платформах для анализа данных в реальном времени.

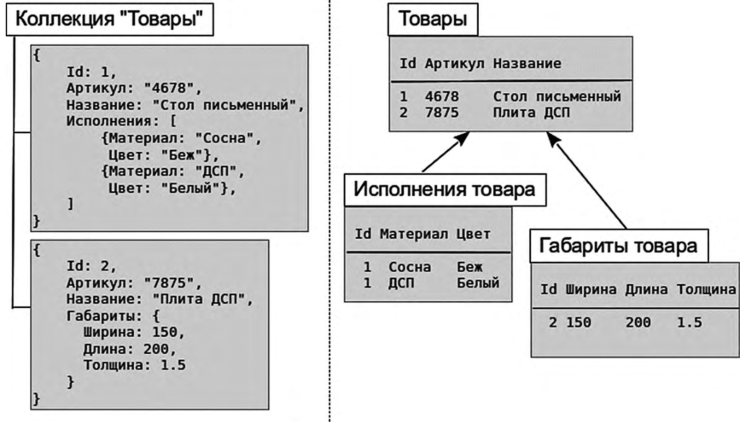


Рисунок 2.11 – Документ-ориентированная модель данных

Реляционная модель данных представляет информацию в виде таблиц (отношений), где строки (записи) содержат экземпляры данных, а столбцы (атрибуты) описывают их свойства. Важной особенностью является наличие уникального идентификатора (первичного ключа) в каждой таблице, который позволяет устанавливать связи между данными. В реляционной модели различают три основных типа связей между сущностями: «один к одному» (1:1), «один ко многим» (1:M) и «многие ко многим» (M:N). Основное преимущество реляционной модели — строгая организация информации, поддержка целостности данных и возможность выполнения сложных запросов с помощью языка SQL. Пример реляционной модели данных представлен на рис. 2.12.

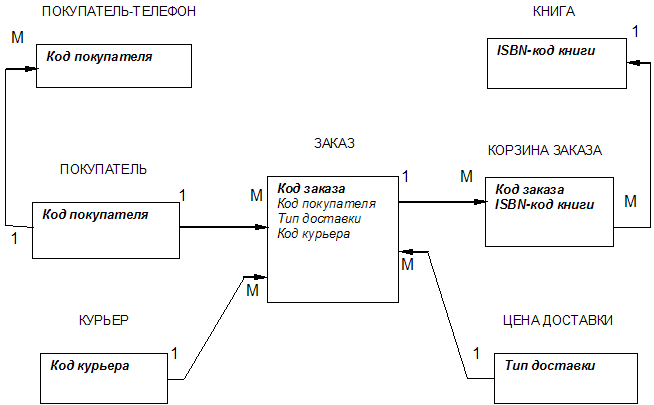


Рисунок 2.12 – Реляционная модель данных

Реляционные базы данных являются стандартом для большинства бизнес-приложений, так как они обеспечивают надежность, предсказуемость и удобные механизмы обработки данных. Благодаря своей универсальности они широко применяются в банковских системах, бухгалтерии, интернет-магазинах, системах управления персоналом и других областях, где важна структурированность информации, поддержка сложных аналитических запросов и надежность хранения.

После анализа различных типов моделей данных было принято решение использовать реляционную модель для проектирования веб-приложения, поскольку она наиболее соответствует требованиям, предъявляемым к разрабатываемой информационной системе. Это обусловлено рядом преимуществ:

* высокая безопасность – встроенные механизмы контроля доступа и разграничение прав пользователей обеспечивают надежную защиту данных;
* легкость интеграции – реляционные базы данных легко взаимодействуют с различными приложениями благодаря поддержке стандартизированных интерфейсов и API;
* поддержка нормализации – предотвращает избыточность данных, способствует оптимизации структуры хранения и улучшает производительность;
* гарантия целостности данных – механизм транзакций обеспечивает сохранность информации даже в случае сбоев или одновременного выполнения множества операций.

## Функциональная модель

Функциональная модель представляет собой систему, состоящую из диаграмм, текстовых описаний и глоссария, которые связаны между собой. Основным элементом модели являются диаграммы, на которых функции и их взаимодействие с окружающей средой изображены в виде блоков и соединяющих их линий. Тип взаимодействия определяется тем, с какой стороны блок соединяется с дугой.

Диаграммы функциональной модели строятся из блоков, где каждый блок описывает отдельную операцию или процесс. У каждого блока есть четыре стороны, обозначающие разные аспекты выполнения функции:

* Слева располагаются входные данные — ресурсы, необходимые для выполнения операции (исходная информация, материалы);
* Справа отображаются выходные данные — результат выполнения функции (готовые данные, обработанные материалы);
* Сверху указываются элементы управления — факторы, влияющие на выполнение функции и конечный результат (правила, регламенты, управляющие механизмы).
* Снизу обозначены механизмы — ресурсы, с помощью которых выполняется операция (оборудование, программное обеспечение, специалисты).

Функциональная модель механизма аренды литературного произведения (книги) представлена на рис. 2.13.

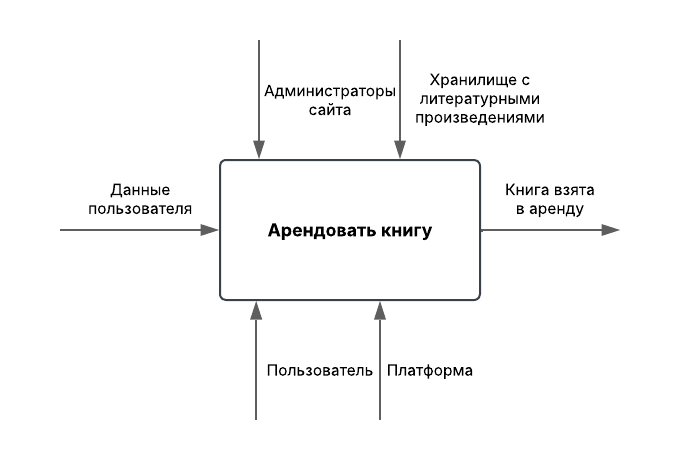


Рисунок 2.13 – Блок «Арендовать книгу»

Декомпозированная модель блока «Арендовать книгу» представлена на рисунке 2.14.

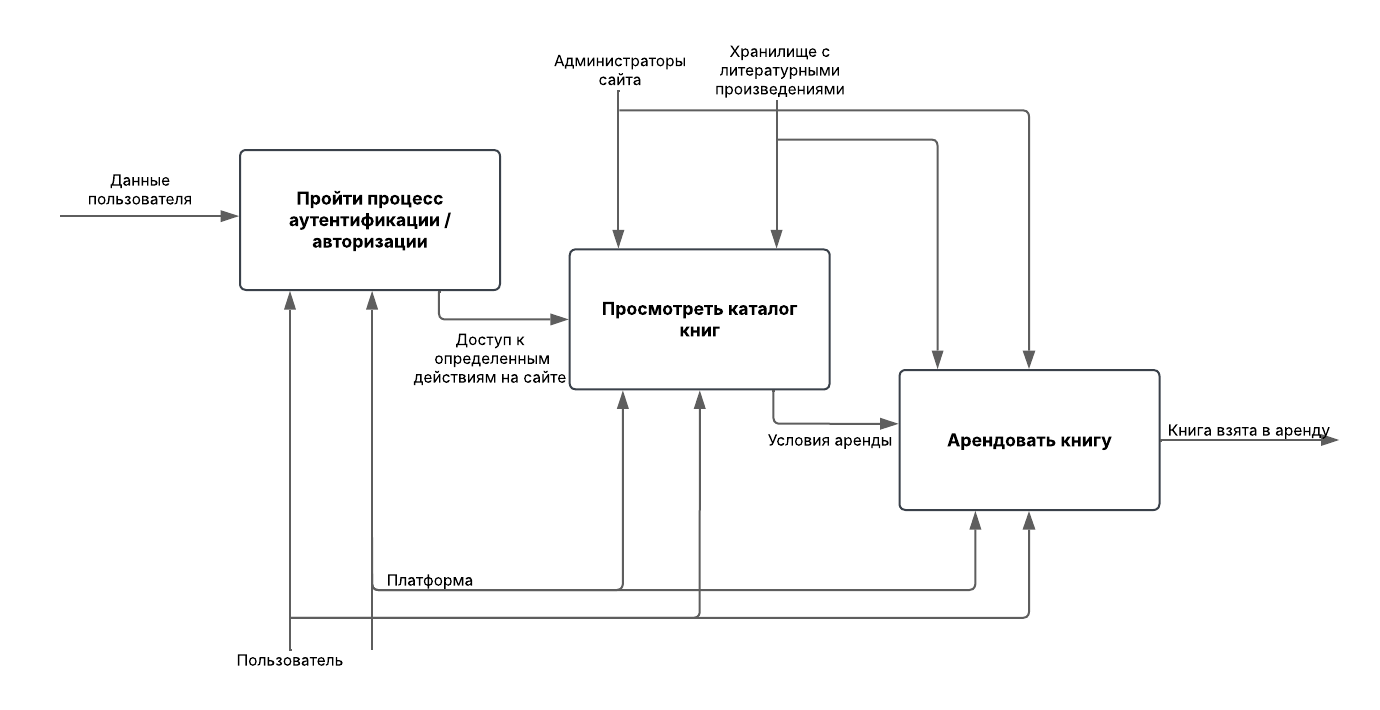


Рисунок 2.14 – Декомпозиция блока «Арендовать книгу»

# ПРОЕКТИРОВАНИЕ СИСТЕМЫ

## Структурная и функциональная схема **разрабатываемого программного обеспечения**

Структурная схема программной системы отображает ее состав, включая подсистемы и основные компоненты. Электронная система аренды литературных произведений включает в себя две подсистемы, представленные на рисунке 3.1.

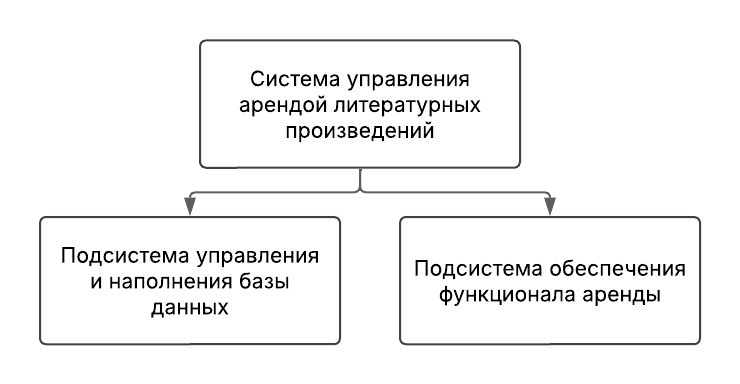


Рисунок 3.1 - Структурная схема

Однако для более детального представления о работе программного обеспечения и его взаимодействии как между внутренними элементами, так и с внешней средой, используется функциональная схема (рис. 3.2). Функциональная схема представляет взаимодействие компонентов ПО, описывает потоки данных, их состав, а также указывает используемые файлы и устройства. В отличие от структурных схем, функциональные схемы содержат больше информации о работе системы.

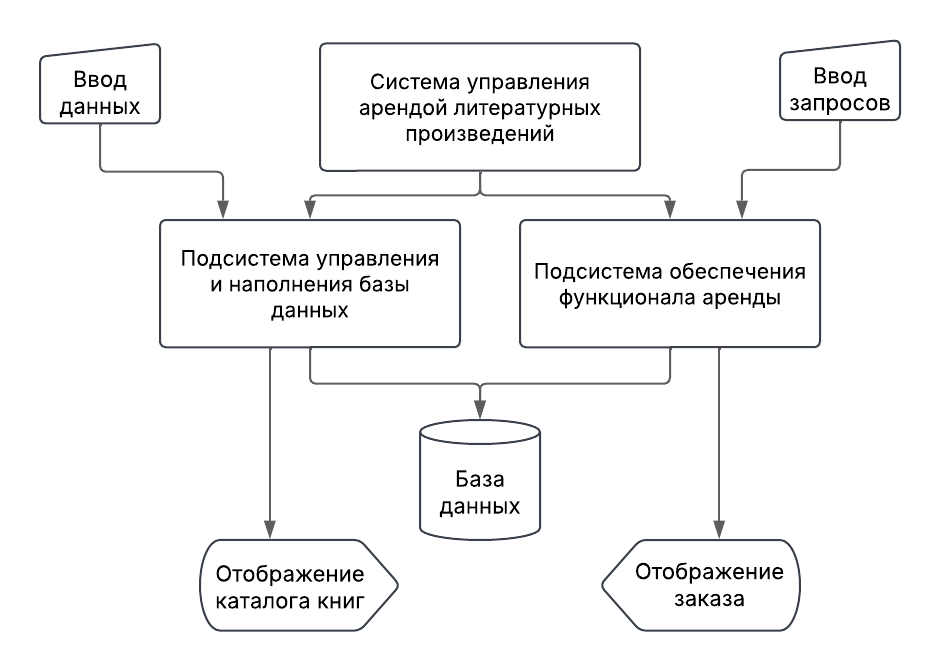


Рисунок 3.2 – Функциональная схема

## Проектирование технологии разработки приложения

Для разработки электронной платформы аренды литературных произведений была выбрана **клиент-серверная архитектура**. Все данные и логика хранятся на сервере, что упрощает управление бизнес-процессами и обслуживание системы в целом.

В клиент-серверной архитектуре приложение делится на две основные части: клиентскую, которая отвечает за взаимодействие с пользователем, и серверную, которая выполняет логику обработки запросов и работы с данными. Такой подход позволяет централизованно управлять данными, упрощает процесс обновлений и масштабирования, а также улучшает безопасность, так как серверная часть изолирована от клиента и может быть защищена от внешних угроз.

Приложение состоит из нескольких ключевых компонентов, каждый из которых выполняет свою специфическую функцию:

**Интерфейс пользователя** — будет реализован с использованием **JavaScript**, что позволяет создавать динамичные и интерактивные веб-страницы. Интерфейс пользователя будет обеспечивать возможность регистрации и входа на сайт, а также предоставлять функционал для просмотра каталога книг. Пользователи смогут фильтровать книги по различным критериям, таким как жанр, автор, язык и рейтинг. Кроме того, интерфейс будет включать возможность оформления заказа на аренду произведений, что позволит пользователю удобно и быстро выбрать и арендовать книгу на определенный срок.

**Серверная логика** — серверная часть приложения будет реализована с использованием **Java Spring Framework**. Этот фреймворк предоставляет мощные инструменты для создания надежных и масштабируемых приложений, включая поддержку работы с RESTful API, управление зависимостями, а также безопасность и обработку запросов. Использование Spring позволит эффективно обрабатывать запросы от пользователей, взаимодействовать с базой данных и обеспечивать логику аренды книг.

**База данных** — для хранения информации о пользователях, книгах, аренде и других данных будет использована **PostgreSQL**. Эта реляционная система управления базами данных обеспечивает высокую производительность, надежность и поддержку сложных запросов. PostgreSQL идеально подходит для работы с большим объемом данных и гарантирует целостность и консистентность информации.

Для обеспечения безопасности при авторизации пользователей был выбран метод аутентификации через **JWT (JSON Web Token).** Этот метод позволяет безопасно передавать информацию о пользователе между клиентом и сервером, минимизируя риски утечек данных. JWT-токен содержит зашифрованную информацию о пользователе, что позволяет серверу проверять его подлинность при каждом запросе без необходимости хранения сессий на сервере.

Для шифрования паролей пользователей выбран стандарт **Base-64,** который позволяет кодировать пароль перед его сохранением в базе данных, повышая уровень безопасности.

В рамках разработки приложения будет осуществляться написание **юнит-тестов** для проверки функциональности всех ключевых компонентов системы. Юнит-тестирование позволит гарантировать корректную работу каждой отдельной части приложения, таких как серверная логика, взаимодействие с базой данных и обработка пользовательских запросов. Тесты будут выполняться с использованием популярных инструментов для Java, таких как **JUnit**, что позволит автоматизировать процесс тестирования и выявлять ошибки на ранних этапах разработки.

Таким образом, проектирование технологии разработки приложения основывается на выборе современных и надежных технологий, что обеспечит высокую производительность, безопасность и масштабируемость системы.

## Диаграмма вариантов использования

Диаграмма вариантов использования является важным инструментом моделирования, который позволяет визуально представить, как пользователи взаимодействуют с системой. Она помогает определить функциональность программного обеспечения, выделить ключевые сценарии работы и показать роли пользователей.

Основным элементом диаграммы является пользователь, который взаимодействует с разрабатываемым ПО. Для него доступны следующие функции:

* просмотр каталога книг;
* оформление заказа, которое включает в себя определение условий аренды произведения;
* просмотр предыдущих заказов.

Диаграмма представлена на рисунке 3.3.

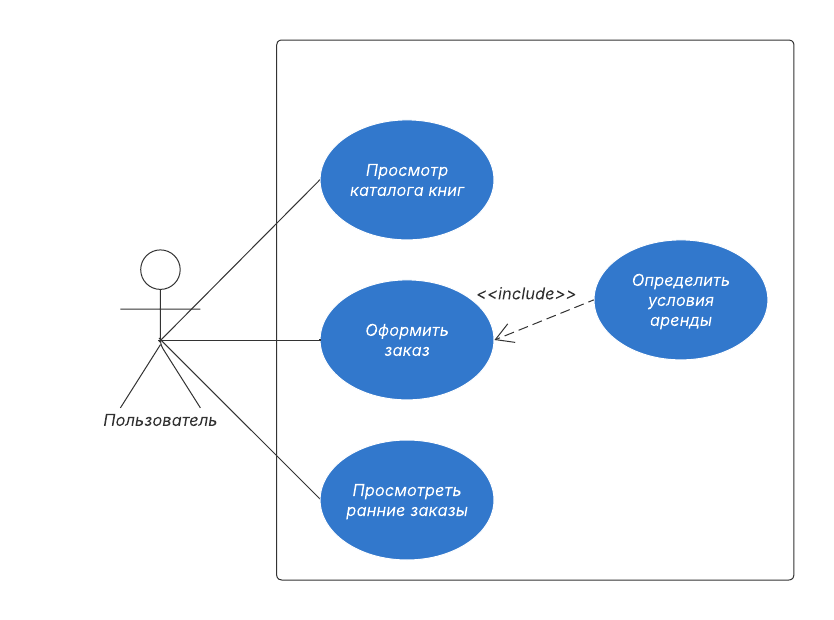


Рисунок 3.3 – Диаграмма вариантов использования

## Инфологическое моделирование предметной области

Инфологическое моделирование – это процесс создания концептуальной модели данных, которая описывает предметную область, не привязываясь к техническим деталям ее реализации. Оно играет важную роль в проектировании баз данных, так как позволяет определить структуру и взаимосвязи данных на высоком уровне абстракции, понятном как пользователям, так и разработчикам. Главная цель инфологического моделирования — создание четкой, структурированной и удобной для восприятия модели, которая максимально точно отражает реальный мир и требования пользователей.

Использование инфологического моделирования значительно упрощает разработку программного обеспечения. Во-первых, оно обеспечивает единое понимание структуры данных между всеми участниками проекта, включая заказчиков, аналитиков и программистов. Во-вторых, такая модель снижает риск ошибок на этапе проектирования за счет выявления избыточности данных, логических несоответствий и проблем с целостностью информации. В-третьих, концептуальная модель остается независимой от конкретной СУБД, что позволяет гибко адаптировать систему к разным платформам и требованиям.

Для построения инфологической модели используются ключевые элементы:

* сущность – объект реального мира, который обладает определенными характеристиками (например, «Книга», «Арендатор» в системе аренды литературных произведений);
* атрибут – свойство сущности, описывающее ее характеристики (например, у книги это «Название», «Автор», «Год издания»);
* связь – логическое отношение между двумя или более сущностями (например, связь «Аренда» между сущностями «Книга» и «Арендатор»).

Наиболее распространенным способом инфологического моделирования является модель «Сущность-Связь» (ER-модель), так как она проста в понимании и позволяет наглядно представить структуру данных. Качественно проработанная инфологическая модель делает последующие этапы проектирования базы данных и разработки ПО более эффективными, структурированными и понятными всем участникам процесса.

На основе анализа предметной области аренды книг с помощью электронной платформы были выделены следующие сущности и атрибуты:

**Пользователи** – содержит информацию о зарегистрированных пользователях сайта:

* **идентификатор** – уникальный номер пользователя;
* **электронная почта** – адрес e-mail пользователя (уникальное значение);
* **пароль** – зашифрованный пароль пользователя;
* **фамилия** – фамилия пользователя;
* **имя** – имя пользователя;
* **отчество** – отчество пользователя;
* **номер телефона** – контактный номер телефона пользователя (уникальное значение).

**Книги** – содержит информацию о литературных произведениях, доступных для аренды:

* идентификатор – уникальный номер книги;
* название – название книги (уникальное значение);
* автор – идентификатор автора книги (ссылка на таблицу авторы);
* жанр – идентификатор жанра книги (ссылка на таблицу жанры);
* язык – идентификатор языка книги (ссылка на таблицу языки);
* год публикации – год выхода книги;
* общее количество – общее число экземпляров книги;
* доступное количество – число экземпляров, доступных для аренды;
* стоимость аренды – цена за аренду книги в неделю;
* описание – краткая информация о книге;
* обложка – путь к изображению обложки книги.

**Авторы – содержит информацию об авторах произведений:**

* идентификатор – уникальный номер автора;
* фамилия – фамилия автора;
* имя – имя автора;
* отчество – отчество автора.

**Жанры – содержит информацию о жанрах произведений:**

* **идентификатор – уникальный номер жанра;**
* **название – наименование жанра.**

**Языки – содержит информацию о языках, на которых написаны произведения:**

* идентификатор – уникальный номер языка;
* название – наименование языка (уникальное значение);

Заказы (аренда книг) – содержит информацию о заказах:

* идентификатор – уникальный номер заказа;
* книга – идентификатор книги, взятой в аренду (ссылка на таблицу книги);
* пользователь – идентификатор пользователя, арендующего книгу (ссылка на таблицу пользователи);
* дата начала аренды – день, с которого пользователь берет книгу в аренду;
* дата окончания аренды – день, до которого пользователь может пользоваться книгой;

После опеределения сущностей и их атрибутов необходимо проанализировать связи между ними:

* связь между таблицами "Книги" и "Авторы" (М:1) – каждая книга может быть связана только с одним автором, но один автор может иметь несколько книг;
* связь между таблицами "Книги" и "Жанры" (М:1) – каждая книга принадлежит к одному жанру, но один жанр может включать множество книг. Удаление жанра приведет к удалению всех связанных с ним книг (ON DELETE CASCADE), а изменение жанра обновит его во всех связанных записях (ON UPDATE CASCADE);
* связь между таблицами "Книги" и "Языки" (М:1) – каждая книга написана на одном языке, но один язык может быть общим для множества книг;
* связь между таблицами "Аренда" и "Книги" (М:1) – каждая запись в таблице аренды относится к одной книге, но одна книга может быть арендована разными пользователями в разное время;
* связь между таблицами "Аренда" и "Пользователи" (М:1) – каждая запись аренды связана с одним пользователем, но один пользователь может арендовать несколько книг.

## Даталогическая модель предметной области

После анализа различных моделей данных для логического проектирования была выбрана реляционная модель, так как она наилучшим образом соответствует требованиям разрабатываемой информационной системы.

В реляционных базах данных даталогическое проектирование заключается в разработке схемы базы данных, представляющей собой совокупность схем отношений, отражающих объекты программного обеспечения и их семантические связи.

Переход от инфологической модели к реляционной осуществляется с использованием следующего алгоритма:

* каждая сущность преобразуется в соответствующее отношение;
* атрибут сущности становится атрибутом соответствующего отношения;
* первичный ключ сущности (PK) устанавливается как первичный ключ отношения, при этом его атрибуты должны быть обязательными (NOT NULL);
* в отношении, соответствующем подчиненной сущности, добавляются атрибуты основной сущности, входящие в его первичный ключ. Эти атрибуты становятся внешним ключом (FK);
* по умолчанию все атрибуты, не входящие в PK, являются необязательными;
* для категоризации сущностей могут применяться различные подходы;
* связи типа "многие ко многим" (М:М) должны быть раскрыты.

Следуя данному алгоритму, опишем каждую сущность датологической модели:

Жанры:

* идентификатор жанра – int, NOT NULL, PK;
* название жанра – varchar(100), NOT NULL.

Языки:

* идентификатор языка – int, NOT NULL, PK;
* название языка – varchar(255), UNIQUE.

Пользователи:

* идентификатор пользователя – int, NOT NULL, PK;
* электронная почта – varchar(255), UNIQUE;
* пароль – varchar(255), NOT NULL;
* фамилия – varchar(255), NOT NULL;
* имя – varchar(255), NOT NULL;
* отчество – varchar(255);
* номер телефона – int, UNIQUE.

Авторы:

* идентификатор автора – int, NOT NULL, PK
* фамилия – varchar(255), NOT NULL
* имя – varchar(255), NOT NULL
* отчество – varchar(255)

Книги:

* идентификатор книги – int, NOT NULL, PK;
* название – varchar(255), UNIQUE;
* автор – int, FK (ссылка на таблицу "Авторы");
* жанр – int, FK (ссылка на таблицу "Жанры");
* язык – int, FK (ссылка на таблицу "Языки");
* год публикации – int;
* общее количество экземпляров – int, DEFAULT 0;
* доступное количество экземпляров – int, DEFAULT 0;
* цена аренды (за неделю) – numeric(10,2);
* описание – text.
* путь к изображению обложки – varchar(500)

Аренда:

* идентификатор аренды – int, NOT NULL, PK;
* книга – int, FK (ссылка на таблицу "Книги");
* пользователь – int, FK (ссылка на таблицу "Пользователи");
* дата начала аренды – date;
* дата окончания аренды – date.

Исходя из приведенных отношений, построим даталогическую схему БД (рис. 3.4).

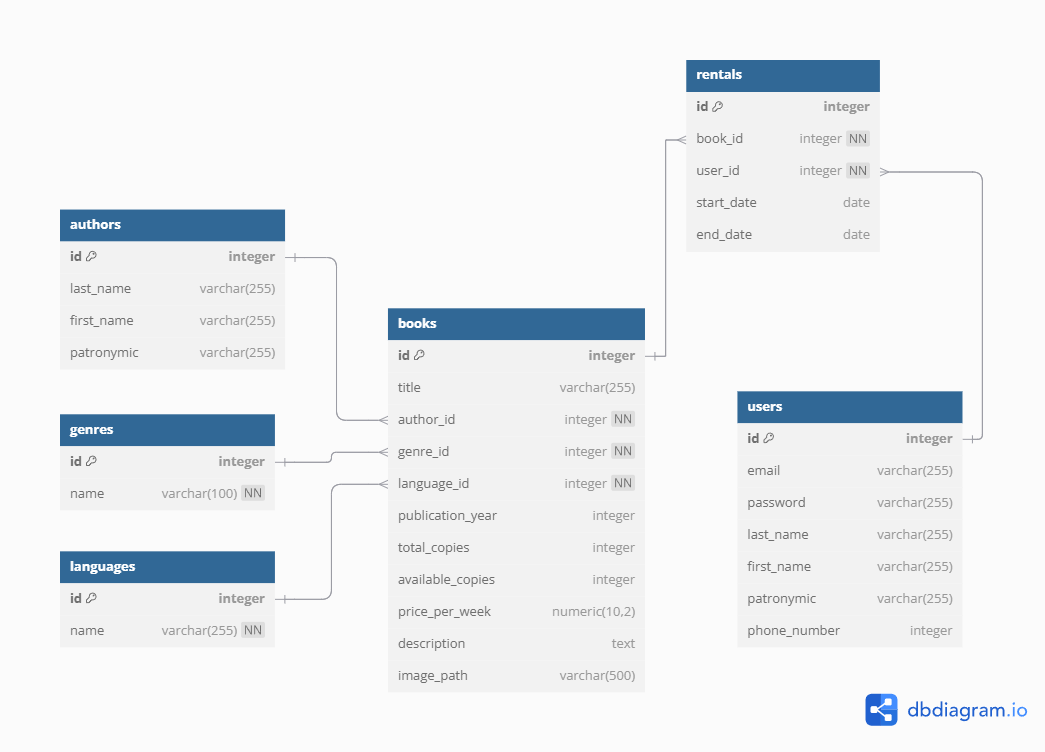


Рисунок 3.4 – Даталогическая модель БД

# ЗАКЛЮЧЕНИЕ

В ходе разработки электронной платформы для аренды литературных произведений были реализованы ключевые аспекты проектирования и создания веб-приложения, обеспечивающего удобный доступ пользователей к книгам в аренду. В качестве технологического стека использовались JavaScript для клиентской части, Java Spring для серверной логики и PostgreSQL в качестве системы управления базами данных. Такой выбор технологий позволил создать масштабируемое, надежное и безопасное решение.

В процессе разработки были проработаны механизмы регистрации и авторизации пользователей, формирования каталога книг, обработки запросов на аренду и управления платежами. Использование Spring Boot упростило разработку REST API, обеспечив эффективное взаимодействие между клиентом и сервером. PostgreSQL позволила реализовать структурированное хранение данных, гарантируя их целостность и поддержку сложных запросов. Для работы с базой данных применялись механизмы ORM (Hibernate), что повысило удобство работы с данными на уровне приложения.

Разработанная платформа обеспечивает автоматизацию процессов аренды книг, минимизируя необходимость ручного управления и исключая риск потери информации. Реализация механизма резервного копирования базы данных позволила повысить отказоустойчивость системы, а продуманная архитектура сервиса обеспечивает возможность дальнейшего масштабирования и расширения функционала.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Технология разработки программного обеспечения [Электронный ресурс] : учебное пособие для студентов специальности 1-08 01 01 Профессиональное обучение (по направлениям) / Белорусский национальный технический университет, Кафедра "Технология и методика преподавания" ; сост.: С. М. Азаров, А. А. Дробыш, А. Э. Дробинин. – Минск : БНТУ, 2020.
2. Электронный учебно-методический комплекс по учебной дисциплине «Компьютерные системы и сети» для специальности: I – 40 01 01 «Программное обеспечение информационных технологий» [Электронный ресурс] / Белорусский национальный технический университет, Кафедра «Программное обеспечение информационных систем и технологий» ; сост. С. В. Белова. – Минск : БНТУ, 2022.
3. <https://pravo.by/gosudarstvo-i-pravo/mestnoe-upravlenie-i-samoupravlenie-v-respublike-belarus/> – Национальный правовой Интернет-портал Республики Беларусь
4. <https://studfile.net/> – Файловый Интернет-архив для студентов