Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Факультет информационных технологий и управления

Кафедра информационных технологий автоматизированных систем

|  |  |
| --- | --- |
|  | *К защите допустить*  Заведующий кафедрой ИТАС  \_\_\_\_\_\_\_\_\_\_\_\_А.А. Навроцкий |

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к дипломному проекту

на тему:

**АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА БИБЛИОТЕКИ**

БГУИР ДП 1-53 01 02 06 105 ПЗ

|  |  |
| --- | --- |
| Студент | В. В. Некрашевич |
| Руководитель | Е. Р. Павловская |
| Консультанты |  |
| *от кафедры ИТАС* | Е. Р. Павловская |
| *по экономической части* | Т. Л. Слюсарь |
| Нормоконтролер | Т. С. Боброва |
| Рецензент |  |

Минск 2022

**Реферат**

АВТОМАТИЗИРОВАННАЯ ИНФОРМАЦИОННАЯ СИСТЕМА БИБЛИОТЕКИ: дипломный проект / В. В. Некрашевич. – Минск : БГУИР, 2022, – п.з. – 60 с., чертежей (плакатов) – 6 л. формата А1.

**СОДЕРЖАНИЕ**

[Введение 6](#_Toc104970664)

[1 Описание и анализ объекта автоматизации 7](#_Toc104970665)

[1.1 Организационная структура библиотеки 7](#_Toc104970666)

[1.2 Анализ существующей библиотечной системы 9](#_Toc104970667)

[1.3 Обзор и анализ существующих аналогов 10](#_Toc104970668)

[1.4 Постановка задачи на дипломное проектирование 12](#_Toc104970669)

[2 Проектирование структуры информационной системы 14](#_Toc104970670)

[2.1 Выбранные методологии и средства проектирования 14](#_Toc104970671)

[2.2 Организационно-экономическая сущность задачи 15](#_Toc104970672)

[2.3 Информационное обеспечение 19](#_Toc104970673)

[2.4 Математическое и алгоритмическое обеспечение 28](#_Toc104970674)

[2.5 Организационное обеспечение 33](#_Toc104970675)

[3 Реализационная часть 35](#_Toc104970676)

[3.1 Техническое обеспечение 35](#_Toc104970677)

[3.2 Программное обеспечение 39](#_Toc104970678)

[3.3 Руководство пользователя 46](#_Toc104970679)

[4 Технико-экономическое обоснование разработки и использования автоматизированной информационной системы библиотеки 51](#_Toc104970680)

[4.1 Описание разрабатываемой автоматизированной системы 51](#_Toc104970681)

[4.2 Расчёт затрат на разработку автоматизированной системы 52](#_Toc104970682)

[4.3 Расчёт экономического эффекта 56](#_Toc104970683)

[4.4 Расчёт экономической эффективности 58](#_Toc104970684)

[Заключение 60](#_Toc104970685)

[Список использованных источников 61](#_Toc104970686)

[Приложение А (справочное) Листинг функции расчёта коэффициента](#_Toc104970687)

[корреляции 62](#_Toc104970687)

[Приложение Б (справочное) Листинг настройки *jwt* безопасности 63](#_Toc104970688)

Введение

Постоянное развитие техники и науки делает возможным существенно упростить и ускорить многие привычные процессы. В настоящее время повсеместно внедряются автоматизированные технологии.

Автоматизация активно применяется в самых разных сферах жизни. Крупные библиотеки используют автоматизированные информационные системы для заказа, покупки, каталогизации и распространения книг. Небольшие библиотеки часто используют лишь некоторые из этих возможностей, не редко учёт книг и работа с читателями ведётся практически вручную.

Целью данного дипломного проекта является разработка автоматизированной информационной системы библиотеки. Под автоматизацией понимается повышение производительности и эффективности труда библиотекарей, направленное на устранение однообразных и трудоемких операций, ускорение процесса выполнения работ и оказания услуг.

Использование системы позволяет читателям осуществлять поиск книг, оставлять отзыв, бронировать желаемую книгу, просматривать историю книговыдачи. Помимо перечисленных задач, также поставлена задача создания удобного пользовательского интерфейса, доступного любому читателю, имеющему персональный компьютер.

Для привлечения новых читателей поставлена задача разработки рекомендательной системы книг с использованием коллаборативной фильтрации. В качестве меры фильтрации выбран коэффициент корреляции Пирсона.

Для написания автоматизированной информационной системы библиотеки используются веб-технологии. Клиентская часть включает в себя язык гипертекстовой разметки, каскадную таблицу стилей и язык программирования для написания сценариев. Серверная часть включает в себя программную платформу *Node*.*js* с подключенной реляционной базой *PostgreSQL*.

Дипломный проект выполнен самостоятельно, проверен в системе «Антиплагиат» [1]. Процент оригинальности соответствует норме, установленной кафедрой. Цитирования обозначены ссылками на публикации, указанные в «Списке полезных источников».

1. Описание и анализ объекта автоматизации
   1. Организационная структура библиотеки

Библиотека – это учреждение культуры, организующее сбор, хранение и общественное пользование произведениями печати и другими документами. Библиотеки систематически занимаются сбором, хранением, пропагандой и выдачей читателям произведений печати, а также информационно-библиографической работой, являются общедоступным источником знаний и основной базой для самообразования.

Совокупность протекающих в библиотеке частных процессов, представляет собой единый процесс. Именно в этом процессе происходит объединение сотрудников библиотеки, а его содержание оказывает определяющее воздействие на организационное построение библиотеки и ее структурных единиц.

Организационную структуру библиотеки можно определить как целостную, упорядоченную совокупность относительно самостоятельных подразделений, реализующих закрепленные за ними функции и цели, наделенных соответствующей.

Среди организационных структур библиотек наибольшее распространение получила функционально-технологическая структура. Ее основу составляют три главных раздела библиотечного труда: отдел обслуживания читателей, отдел комплектования и информационно-издательский отдел.

Основной целью отдела обслуживания читателей является помощь читателем и посетителям в пользовании библиотекой. Сотрудники отдела выдают читательские билеты, знакомят новых посетителей с услугами и структурой библиотеки, помогает найти нужную информацию.

Основными задачами отдела являются:

* регистрация новых читателей;
* информирование о правилах пользования библиотекой, имеющихся библиотечных услугах, программе культурных мероприятий и библиотечных фондах;
* анализ читательского спроса и заказ недостающих изданий;
* выдача книг на дом из фонда абонемента;
* выдача книг из хранения в читальные залы библиотеки;
* решение конфликтных ситуаций.

Основная цель деятельности отдела комплектования – комплектование полноценного, универсального по составу единого книжного фонда, способного формировать и удовлетворять общеобразовательные, культурные и профессиональные запросы всех групп читателей. Отдел комплектования, обработки и организации единого фонда занимается формированием и обработкой фонда перед его передачей в библиотеки.

Информационно-издательский отдел осуществляет издательскую и полиграфическую деятельность, направленную на подготовку и выпуск научно-вспомогательной библиографии, информационных, научно-методических материалов в области библиотечного дела, библиографии, книговедения и смежных наук.

Организационная структура библиотеки представлена на рисунке 1.1.

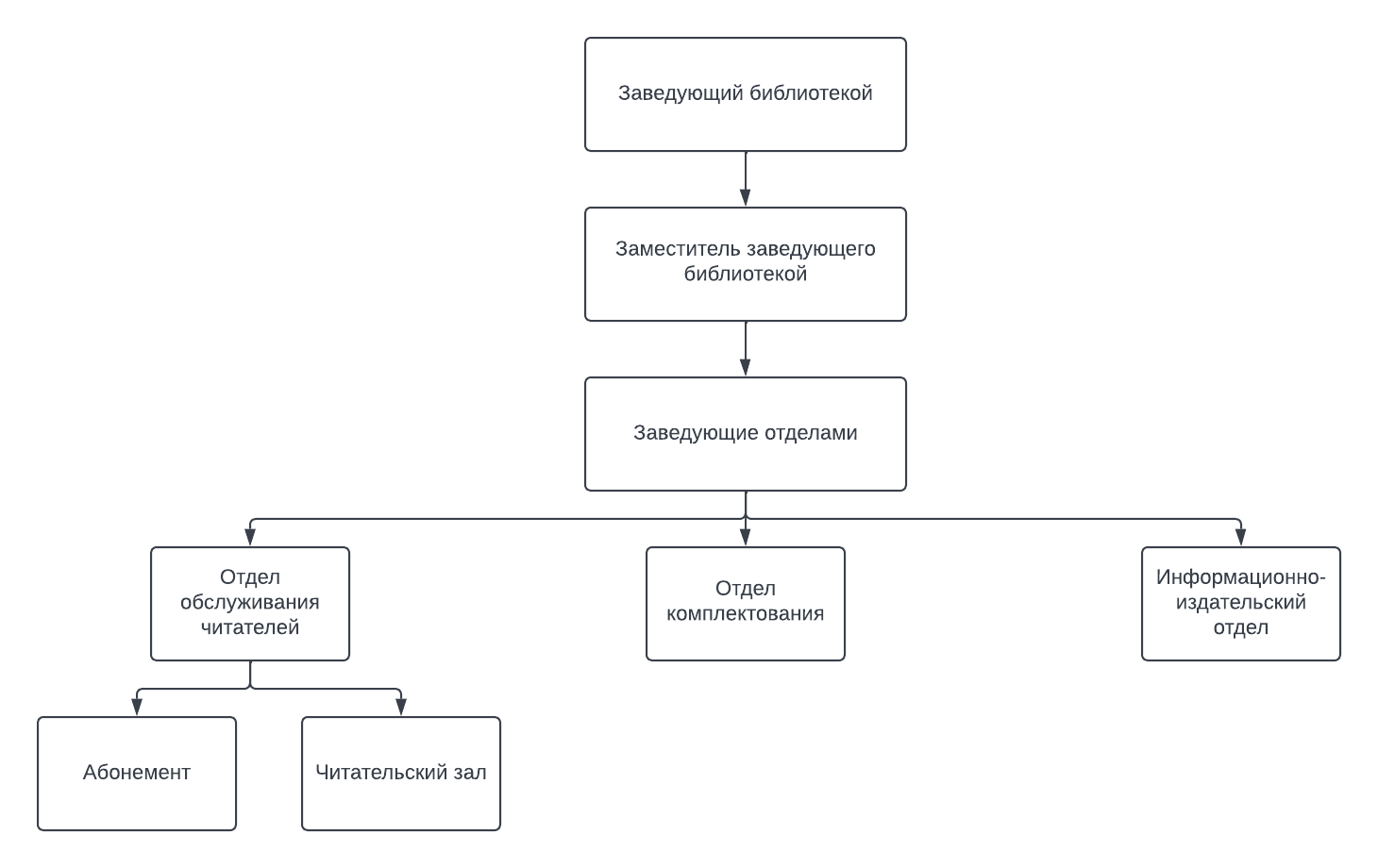


Рисунок 1.1 – Организационная структура библиотеки

Помимо администрации, обязательными элементами существования структуры библиотеки являются библиотекари. Библиотекарь – это сотрудник, который принимает участие в комплектовании библиотечного фонда, создании каталогов, справочно-информационных фондов, популяризации литературы и чтения и обслуживает посетителей библиотеки. Последнее включает в себя и помощь в выборе книг и периодических журналов, и заполнение необходимых документов (в бумажном и/или электронном виде), и проведение тематических мероприятий.

Читатели библиотеки имеют право получать книги и другие источники информации на всех пунктах выдачи библиотеки (абонемент и читальный зал). Читатели, приходящие на пункт выдачи, обязаны иметь при себе читательский билет.

В библиотеку приходит много книг из различных издательств. Каждая книга, хранящаяся в библиотеке, имеет следующие параметры:

* название книги;
* уникальный шифр *ISBN*;
* издательство;
* автор;
* ­количество страниц;
* аннотация.

Каждая книга может присутствовать в нескольких экземплярах. Все книги различаются по своему уникальному шифру – *ISBN*.

В библиотеке ведется картотека читателей. О каждом читателе заносятся следующие сведения:

* фамилия, имя, отчество;
* дата рождения;
* номер паспорта;
* адрес;
* контактный телефон.

Каждому читателю присваивается номер читательского билета.

* 1. Анализ существующей библиотечной системы

В классическом понимании библиотечная информационная система представляет собой систему планирования ресурсов, которые используются для отслеживания библиотечных фондов.

Информационная система состоит из реляционной базы данных, специализированного программного обеспечения, которое взаимодействует с базой данных, и пользовательского интерфейса. Главным элементом традиционной информационной системы библиотеки является электронный каталог.

Каждый читатель (посетитель) и издание имеют уникальный идентификатор в базе данных, которая позволяет системе отслеживать деятельность библиотеки.

Система включает следующие основные функции:

* ведение автоматизированного электронного книжного журнала;
* списание книг по причинам: ветхость, утеря, неактуальность;
* регистрация читателей в библиотечной системе;
* автоматизированное формирование отчетов;
* поиск по заданным критериям.

Минус системы в том, что она не поддерживает технологию публичного онлайн доступа к каталогам и информационным ресурсам и доступна только библиотекарям. Также система не имеет функций по анализу и подбору литературы. Учитывая перечисленные недостатки можно сделать вывод о том, что система неактуальна и не соответствует современным требованиям.

* 1. Обзор и анализ существующих аналогов

Автоматизированная информационно-библиотечная система МАРК-*SQL* представляет собой интегрированную систему, предназначенную для автоматизации информационных технологий в области библиотечного дела, и обеспечивает реализацию полных технологических циклов от заказа литературы до выдачи ее читателю. МАРК-*SQL* предназначена для использования в библиотеках различного уровня и принадлежности.

Система разрабатывалась на основе принципа открытых систем и базируется на архитектуре «клиент-сервер». Система функционирует под *WINDOWS NT*/2000 – серверная часть, для клиента используется *WINDOWS* 95/98/2000/*NT Workstation*.

Основные функции системы реализованы в виде модульной структуры, реализован дружественный графический интерфейс, осуществляется поддержка диалогового режима работы.

МАРК-*SQL* устанавливается на неограниченное количество рабочих мест и поставляется на компакт-диске. В комплект поставки входит документация в электронном и печатном виде.

Система является устаревшим программным обеспечением, поскольку компания *Microsoft* официально прекратила поддержку своих старых операционных систем. Большинство современных ноутбуков не имеют встроенных дисководов, это делает установку МАРК-*SQL* недоступной. Также следует отметить отсутствие доступа к системе с любого персонального компьютера.

1*С*:Библиотека – программа для автоматизации деятельности библиотек любого типа и назначения. Продукт позволяет автоматизировать рабочие процессы библиотеки, в зависимости от ее назначения, типа, состава фондов. Есть конфигурации для школы, ВУЗа, колледжа.

К основным функциям системы можно отнести комплектование библиотечного фонда, каталогизацию, обслуживание читателей, формирование отчётных и учётных документов.

Основным недостатком системы является низкая безопасность и защищённость информации. Благодаря широкой распространённости программ 1*С* в сети интернет существуют тысячи способов незаконного доступа и кражи данных. Количество хакерских приёмов для взлома постоянно обновляется и увеличивается.

МегаПро – решение комплексной автоматизации информационно-библиотечной деятельности, построения библиотечных сетей, эффективного управления информационными ресурсами и организации доступа к ним на основе веб-технологий. Система построена на основе веб-технологий (*IIS*, .*Net* *Framework*, *ASP*.*NET*, *MVC*), поэтому все основные инструменты управления каталогами и базами данных доступны в браузере.

Недостатком «МегаПро» можно назвать зависимость этой системы от технологий *Microsoft*. Неизвестно, будет ли развиваться в будущем линейка продуктов *IIS*: корпорация из Редмонда делает ставку на *Azure*. И если развитие сервера *IIS* или технологии .*Net* будет заморожено, пользователи «МегаПро» рискуют в какой-то момент остаться наедине с багами последних версий серверных продуктов *Microsoft* и в какой-то момент будут вынуждены мигрировать, например, на *Linux* или *FreeBSD*.

Характеристики сравниваемых информационных систем приведены в табличной форме (см. таблицу 1.1).

Таблица 1.1 – Сравнение существующих аналогов

|  |  |  |  |
| --- | --- | --- | --- |
| Функциональные возможности | Существующие автоматизированные информационные системы библиотеки | | |
| МАРК-*SQL* | 1С:библиотека | МегаПро |
| Доступ к системе онлайн | Нет | Нет | Да |
| Подбор необходимой литературы | Да | Нет | Да |
| Получение данных к электронным материалам | Для зарегистрированных пользователей | Только для библиотекарей | Да |

* 1. Постановка задачи на дипломное проектирование

Целью данной работы является проектирование автоматизированной информационной системы библиотеки. Система предназначена для повышения эффективности работы обслуживающего персонала библиотеки и быстрого обслуживания читателей за счёт услуги автоматизированного поиска книги, сокращения времени поиска и бронирования необходимой читателю книги.

Система работает в двух режимах, в зависимости от типов пользователей: в режиме администратора и зарегистрированного пользователя. Права администратора предоставляются каждому работнику библиотеки.

Режим работы системы с правами администратора обеспечивает следующие возможности:

* создание, загрузка, редактирование и сохранение базы данных читателей (ФИО, паспортные данные, адрес, номер читательского билета, контактный телефон);
* создание, загрузка, редактирование и сохранение базы данных книг (название, *ISBN*, жанр, автор, издательство, количество страниц, количество экземпляров в библиотеке, аннотация);
* подтверждение бронирования книги для читателя;
* изменение истории книговыдачи;
* многоаспектный поиск данных по различным признакам (поиск по названию, поиск по автору, поиск по *ISBN*, поиск по жанру) и их сочетаниям.

Режим работы системы с правами зарегистрированного пользователя обеспечивает следующие возможности:

* регистрация и авторизация в автоматизированной системе;
* многоаспектный поиск данных по различным признакам (поиск по названию, поиск по автору, поиск по *ISBN*, поиск по жанру) и их сочетаниям;
* отслеживание личной истории книговыдачи (название книги, автор, дата выдачи);
* добавление рейтинга и рецензии для прочитанных ранее книг;
* бронирование желаемых книг для последующего получения в библиотеке;
* использование рекомендательной системы для подбора книг;

Конечной стадией реализации проекта должна стать веб-система, которая предоставляет доступ к работе с ней для всех пользователей, имеющих доступ к сети интернет. Пользователю должна быть предоставлена возможность регистрации и авторизации, а также возможность осуществления различного вида запросов. Все данные пользователей должны хранится в базе данных и иметь высокий уровень безопасности от несанкционированного доступа к ним.

Разрабатываемая автоматизированная информационная система должна иметь положительный экономический эффект.

1. Проектирование структуры информационной системы
   1. Выбранные методологии и средства проектирования

Методология проектирования составляет основу проекта любой информационной системы. Методология реализуется через конкретные технологии и поддерживающие их стандарты, методики и инструментальные средства, которые позволяют выполнять все процессы жизненного цикла.

Существует две основных методологии проектирования:

* методология структурного проектирования;
* методология объектно-ориентированного проектирования.

Сущность структурного подхода заключается в ее декомпозиции на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны.

Методологии структурного проектирования, используемые при построении автоматизированной информационной системы библиотеки:

1. *DFD* – общепринятое сокращение от англ. *data flow diagrams* – диаграммы потоков данных. Методология графического структурного анализа, описывающая внешние по отношению к системе источники и адресаты данных, логические функции, потоки данных и хранилища данных, к которым осуществляется доступ.
2. *IDEF*0 предписывает построение иерархической системы диаграмм. Сначала проводится описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная декомпозиция (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.
3. *IDEF*3 – способ описания процессов с использованием структурированного метода, позволяющего эксперту в предметной области представить положение вещей как упорядоченную последовательность событий с одновременным описанием объектов, имеющих непосредственное отношение к процессу.

Для реализации методологий структурного проектирования выбрано *CASE*-средство *BPWin* – мощный инструмент моделирования, который используется для анализа, документирования и реорганизации сложных бизнес-процессов. Модели *BPwin* дают основу для оценки влияния тех или иных событий, а также описывают взаимодействие процессов и потоков информации в организации.

Центральным понятием объектно-ориентированного подхода к проектированию является класс. Класс – это выделение из окружающего мира некой сущности, для которой определены атрибуты (свойства) и операции (действия, которые сущность производит над окружающими объектами). Объект – это конкретная реализация некоторой сущности. В объекте инкапсулируется некоторая часть приложения, которая может представлять собой процесс, группу данных или какую-либо более сложную сущность.

Для графического описания объектно-ориентированной методологии используется *UML* – унифицированный язык моделирования. *UML* является языком широкого профиля, созданным для определения, визуализации, проектирования и документирования программных систем. Он состоит из девяти различных диаграмм, каждая из которых позволяет моделировать отдельные статические или динамические аспекты процесса.

Для применения принципов объектно-ориентированного моделирования и языка *UML* используется *CASE*-средство *Rational Rose.* Инструменты моделирования *Rational Rose* позволяют разработчикам создавать целостную архитектуру процессов предприятия, сохраняя все взаимосвязи и управляющие воздействия между различными уровнями иерархии.

* 1. Организационно-экономическая сущность задачи

Разрабатываемая в дипломном проекте автоматизированная система библиотеки предназначена для повышения эффективности работы обсуживающего персонала библиотеки и более быстрого обслуживания читателей. Система позволяет читателям забронировать желаемое издание и сократить время поиска литературы.

Для общего описания системы и её взаимодействия с внешней средой строится контекстная диаграмма. Контекстная диаграмма представляет собой общее описание работы автоматизированной системы библиотеки и ее взаимодействия с внешними элементами или данными.

Для автоматизированной информационной системы библиотеки спроектирована контекстная диаграмма, представленная на рисунке 2.1.

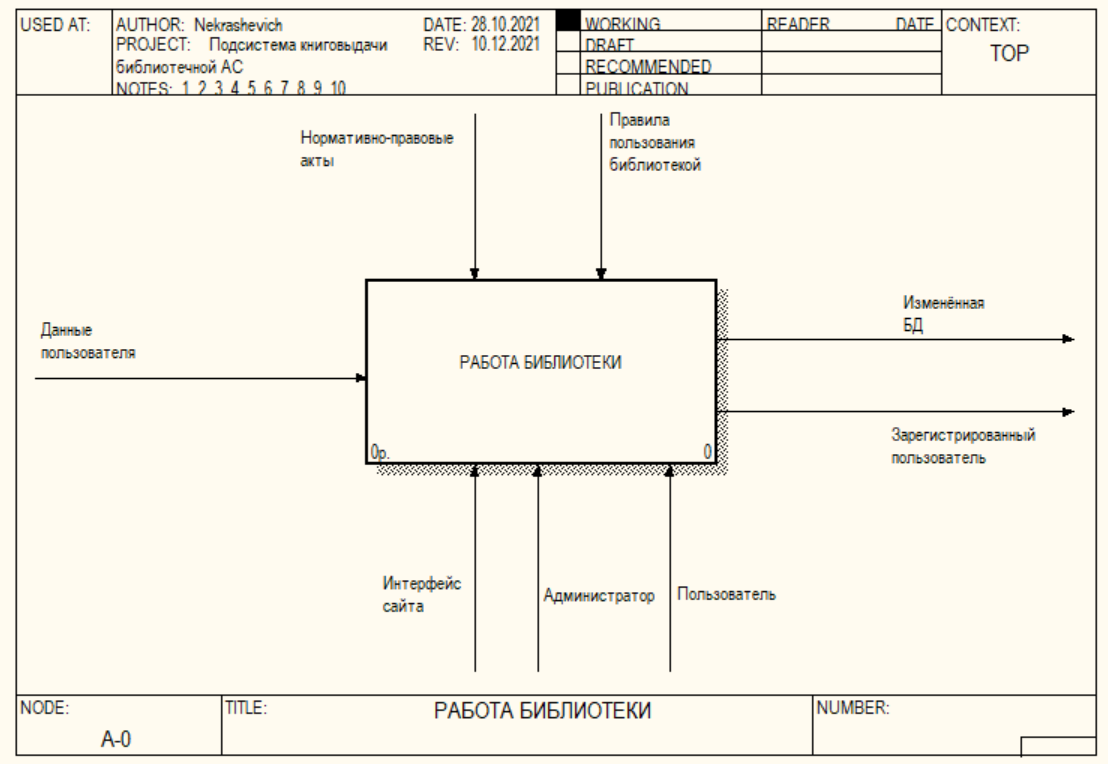


Рисунок 2.1 – Контекстная диаграмма автоматизированной информационной системы библиотеки

На приведённой контекстной диаграмме представлены виды входной и выходной информации системы, а также механизмы и элементы управления.

На вход поступает информация о персональных данных пользователя на основе его роли в системе.

В качестве выходной информации служат следующие объекты:

* зарегистрированный пользователь – представляет собой обработанную и записанную, исходя из введённых данных, информацию о новом пользователе;
* изменённая БД – представляет собой базу данных библиотеки, изменённую после взаимодействия пользователей с системой.

В качестве механизмов служат следующие элементы:

* администратор – работник библиотеки, имеющий неограниченные права и администрирующий сервис;
* пользователь – пользователь, который имеет возможность менять информацию о себе и управлять данными для входа в систему;
* интерфейс сайта – совокупность средств, при помощи которых пользователь взаимодействует с системой.

Элементы управления представлены следующими составляющими:

* нормативно-правовые акты – включают в себя нормативно-правовые документы, регулирующие деятельность библиотек;
* правила пользования библиотекой – свод правил, описывающих взаимоотношения пользователей и библиотеки в процессе библиотечного, информационного и справочно-библиографического обслуживания.

После построения контекстная диаграмма детализируется с помощью диаграммы декомпозиции первого уровня. На этой диаграмме отображаются функции системы, которые должны быть реализованы в рамках основной функции.

При построении диаграммы декомпозиции исходную функцию разбили на четыре функциональных блока:

* вход в систему – функциональный блок, отвечающий за предоставление доступа к системе, формирование персональных данных для незарегистрированных пользователей, проверку и верификацию данных для входа в систему;
* взаимодействие с каталогом – функциональный блок, отвечающий за основное взаимодействие пользователя с системой и включающий в себя возможность отправки пользовательских запросов;
* поиск книг – функциональный блок, осуществляющий реализацию поисковых запросов;
* бронирование книг – функциональный блок, позволяющий пользователям бронировать желаемые книги.

Диаграмма декомпозиции первого уровня для рассматриваемой задачи приведена на рисунке 2.2.

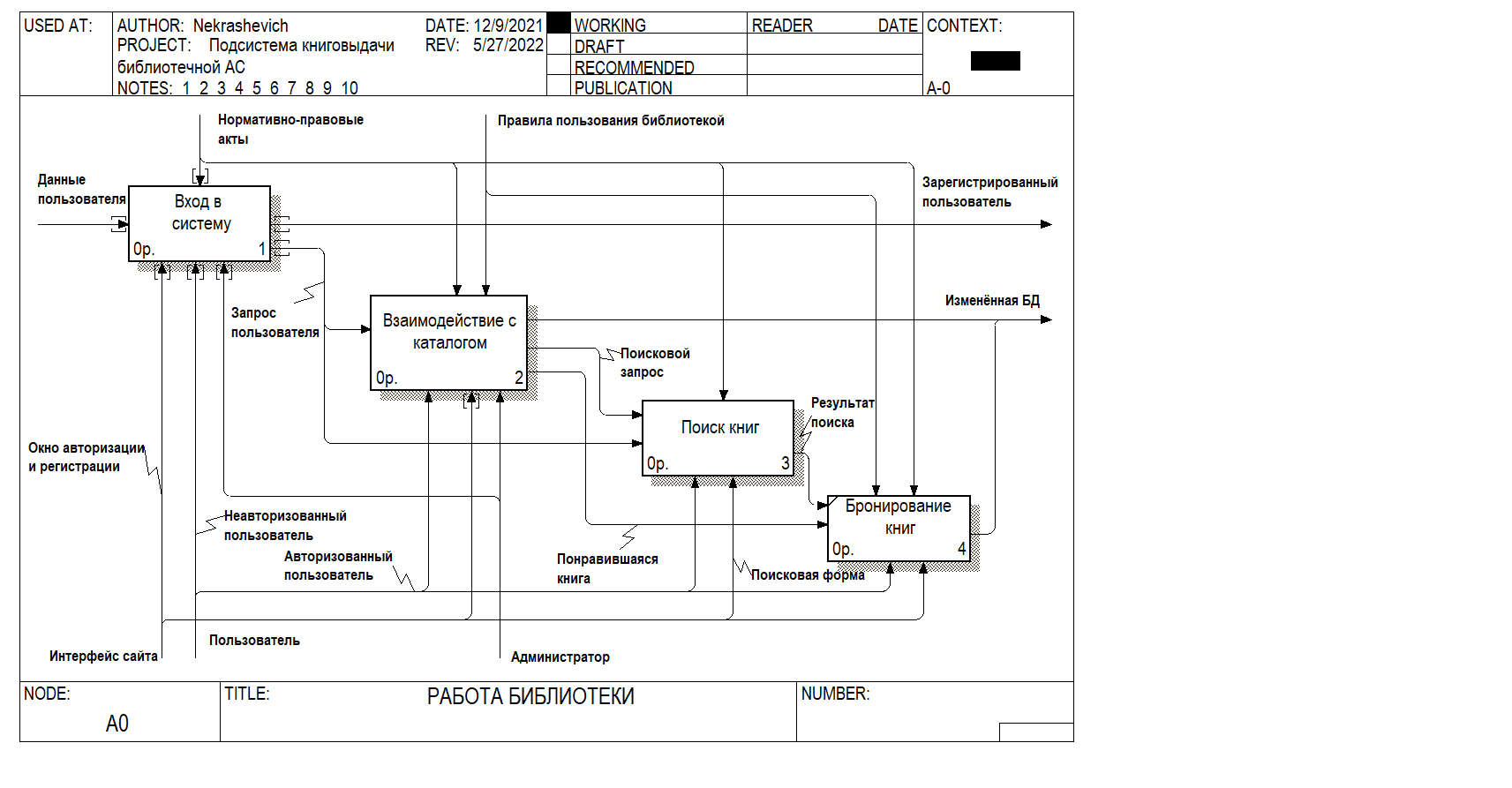


Рисунок 2.2 – Диаграмма декомпозиции первого уровня

Для более подробного описания блока взаимодействия с каталогом строится диаграмма декомпозиции второго уровня, включающая в себя следующие функциональные блоки:

* посмотреть рекомендации – функциональный блок, позволяющий пользователю просматривать личные книжные рекомендации, предоставленные автоматизированной системой ;
* открыть поисковую форму – функциональный блок, предоставляющий пользователю возможность взаимодействовать с поисковой формой сайта и отправлять поисковые запросы;
* отследить историю книговыдачи – функциональный блок, отвечающий за предоставление пользователю информации об истории книговыдачи;
* добавить отзыв - функциональный блок, отвечающий за возможность добавления пользователем отзыва на ранее прочитанную книгу.

Входными данными диаграммы является запрос пользователя, выходными – изменённая база данных, поисковой запрос и понравившаяся книга.

Диаграмма декомпозиции для блока взаимодействия с каталогом, представлена на рисунке 2.3.

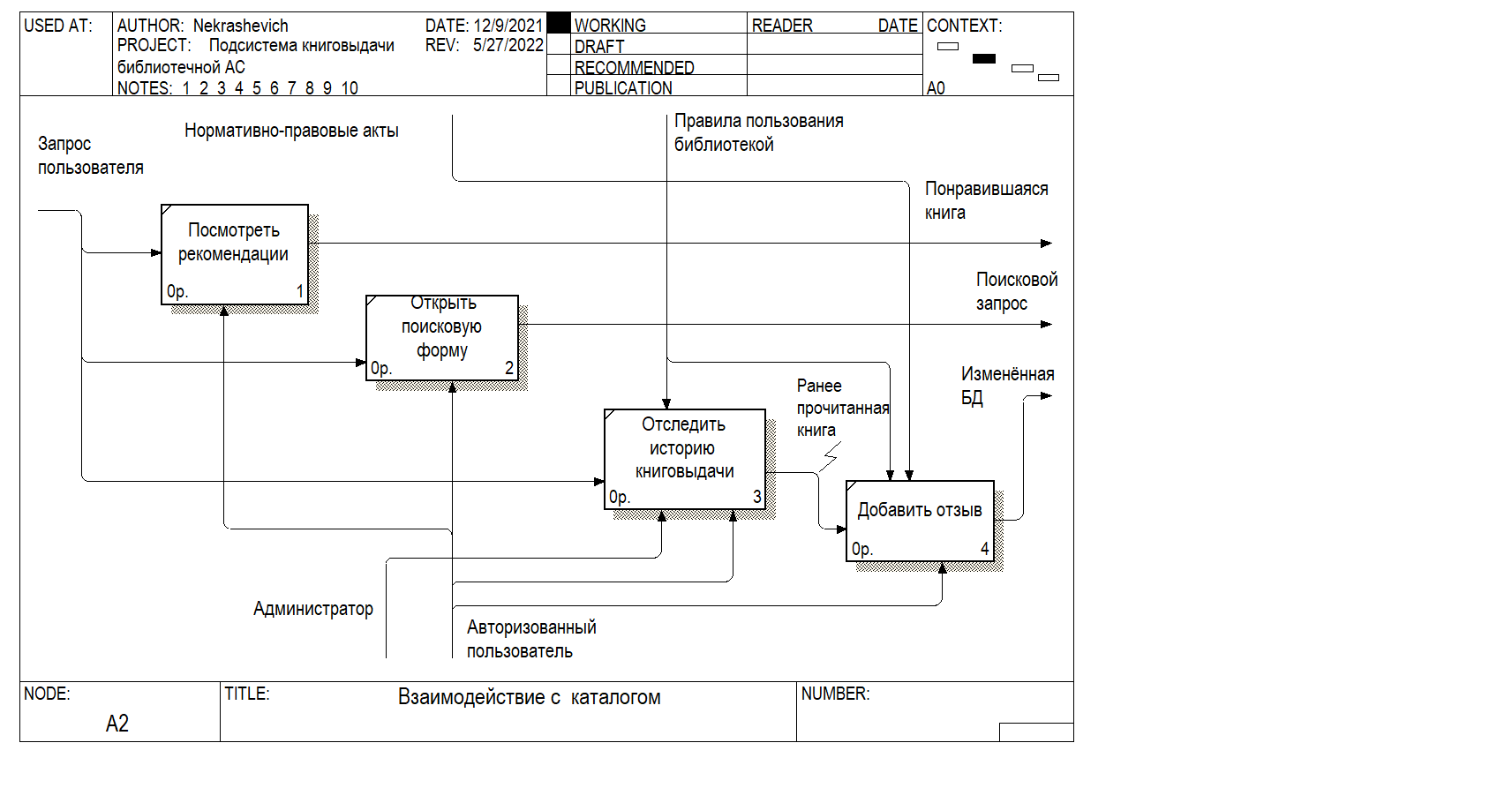


Рисунок 2.3 – Диаграмма декомпозиции блока взаимодействия с каталогом

* 1. Информационное обеспечение

**2.3.1** Базой данных называется организованная в соответствии с определенными правилами и поддерживаемая в памяти компьютера совокупность сведений об объектах, процессах, событиях или явлениях, относящихся к некоторой предметной области, теме или задаче. Она организована таким образом, чтобы обеспечить информационные потребности пользователей, а также удобное хранение этой совокупности данных, как в целом, так и любой ее части.

Реляционная база данных представляет собой множество взаимосвязанных таблиц (сущностей), каждая из которых содержит информацию об объектах определенного типа. Каждая строка таблицы содержит данные об одном объекте, а столбцы таблицы содержат различные характеристики этих объектов – атрибуты.

Нормальная форма – требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами.

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа (ПК). Для соблюдения этой нормальной формы во всех таблицах создаются внешние ключи, а их сочетание составляет уникальный индекс. Таким образом, невозможна функциональная зависимость не ключевого атрибута от части первичного ключа.

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут не транзитивно зависит от первичного ключа. Правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Исходя из функций и задач системы и требований трёх нормальных форм, выделяются следующие сущности базы данных: пользователи (*users*), читательские билеты (*cards*), забронированные книги (*booking*), истории книговыдачи (*rentals*), книжные отзывы (*reviews*), книги (*books*), авторы (*authors*), жанры (*genres*), администраторы (*admins*). Дополнительные сущности, служащие для реализации связи многие-ко-многим: книги и авторы (*booksAuthors*), книги и жанры (*booksGenres*).

Перечень сущностей с указанием их назначения, атрибутов и ключей представлен в табличной форме (см. таблицу 2.1).

Таблица 2.1 – Сущности базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Содержимое | Атрибуты | Ключи |
| Пользователи | Информация о зарегистриро-ванных пользователях системы | Идентификатор пользователя, электронная почта, логин, пароль, идентификатор читательского билета | Идентификатор пользователя (первичный),  идентификатор читательского билета (вторичный) |
| Читательские билеты | Информация о читателях библиотеки | Идентификатор читательского билета, ФИО, дата рождения, номер телефона, адрес | Идентификатор читательского билета (первичный) |
| Забронированные книги | Информация о бронировании книг | Идентификатор бронирования, дата бронирования,  идентификатор читательского билета, идентификатор книги, статус | Идентификатор бронирования (первичный),  идентификатор читательского билета (вторичный), идентификатор книги (вторичный) |
| Истории книговыдачи | Информация о выдаче книг читателям | Идентификатор книговыдачи,  идентификатор читательского билета, идентификатор книги,  дата выдачи,  дата возврата | Идентификатор книговыдачи (первичный),  идентификатор читательского билета (вторичный), идентификатор книги (вторичный) |
| Книжные отзывы | Оценки и рецензии на прочитанные книги | Идентификатор отзыва, рецензия, оценка, идентификатор книговыдачи | Идентификатор отзыва (первичный),  идентификатор книговыдачи (вторичный) |

Продолжение таблицы 2.1

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Содержимое | Атрибуты | Ключи |
| Книги | Данные о книгах | Идентификатор книги, название, *ISBN*, обложка, количество страниц, количество, аннотация | Идентификатор книги (первичный) |
| Авторы | Данные об авторах | Идентификатор автора, ФИО, дата рождения, биография | Идентификатор автора (первичный) |
| Жанры | Жанры книг | Идентификатор жанра, жанр | Идентификатор жанра(первичный) |
| Администраторы | Информация о работниках библиотеки | Идентификатор администратора, электронная почта, логин, пароль, ФИО, дата рождения, телефонный номер, адрес | Идентификатор администратора (первичный) |
| Книги и авторы | Вспомогатель-ное, для создания связи «многие-ко-многим» | Идентификатор книги, идентификатор автора | Идентификатор книги (вторичный), идентификатор автора (вторичный) |
| Книги и жанры | Вспомогатель-ное, для создания связи «многие-ко-многим» | Идентификатор книги, идентификатор жанра | Идентификатор книги (вторичный), идентификатор жанра(вторичный) |

Каждой сущности соответствует таблица базы данных, а каждому атрибуту – поле таблицы. Ниже приводится описание атрибутов сущностей и их характеристика.

Описание атрибутов сущности «Пользователи» (*users*) представлено в таблице 2.2.

Таблица 2.2 – Атрибуты сущности «Пользователи»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Идентификатор пользователя (*IDuser*) | *bigserial* | Уникальный ключ |
| Электронная почта (*email*) | *varchar*(100) | Электронная почта пользователя |
| Логин (*login*) | *varchar*(100) | Персональное имя для входа в систему |
| Пароль (*password*) | *varchar*(50) | Пароль в формате *BCrypt* |
| Идентификатор читательского билета (*IDcard*) | *int* | Идентификатор связанного читательского билета |

Описание атрибутов сущности «Читательские билеты» (*cards*) представлено в таблице 2.3.

Таблица 2.3 – Атрибуты сущности «Читательские билеты»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Идентификатор читательского билета (*IDuser*) | *bigserial* | Уникальный ключ |
| Фамилия имя отчество (*fullName*) | *varchar*(100) | Фамилия имя отчество |
| Дата рождения (*dateOfBirth*) | *date* | Дата рождения |
| Номер телефона (*phoneNumber*) | *varchar*(20) | Номер телефона |
| Адрес (*address*) | *varchar*(100) | Адрес |

Описание атрибутов сущности «Забронированные книги» (*booking*) представлено в таблице 2.4.

Таблица 2.4 – Атрибуты сущности «Забронированные книги»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Идентификатор бронирования (*IDbooking*) | *bigserial* | Уникальный ключ |
| Дата брони (*bookingDate)* | *date* | Дата бронирования книги |
| Статус | *boolean* | Статус бронирования |
| Идентификатор читательского билета (*IDcard*) | *int* | Идентификатор связанного читательского билета |
| Идентификатор книги (*IDbook*) | *int* | Идентификатор связанной книги |

Описание атрибутов сущности «Истории книговыдачи» (*rentals*) представлено в таблице 2.5.

Таблица 2.5 – Атрибуты сущности «Истории книговыдачи»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Идентификатор книговыдачи (*IDrent*) | *bigserial* | Уникальный ключ |
| Идентификатор читательского билета (*IDcard*) | *int* | Идентификатор связанного читательского билета |
| Идентификатор книги (*IDbook*) | *int* | Идентификатор связанной книги |
| Дата выдачи (*takeDate*) | *date* | Дата получения книги читателем |
| Дата возврата (*returnDate*) | *date* | Дата возврата книги читателем в библиотеку |

Описание атрибутов сущности «Книжные отзывы» (*reviews*) представлено в таблице 2.6.

Таблица 2.6 – Атрибуты сущности «Книжные отзывы»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Идентификатор отзыва (*IDreview*) | *bigserial* | Уникальный ключ |
| Рецензия (*review*) | *text* | Текст отзыва на прочитанную книгу |
| Оценка (*rating*) | *int* | Оценка прочитанной книги |
| Идентификатор книговыдачи (*IDrent*) | *int* | Идентификатор связанной книговыдачи |

Описание атрибутов сущности «Книги» (*books*) представлено в таблице 2.7.

Таблица 2.7 – Атрибуты сущности «Книги»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Идентификатор книги (*IDbook*) | *bigserial* | Уникальный ключ |
| Название (*title*) | *varchar*(100) | Название книги |
| *ISBN* (*ISBN*) | *varchar*(20) | Уникальный номер книжного издания |

Продолжение таблицы 2.7

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Обложка (*cover*) | *text* | Ссылка на изображение обложки |
| Количество страниц (*pages*) | *int* | Количество страниц в книге |
| Количество (*amount)* | *int* | Количество книг, доступных в библиотеке |
| Аннотация (*annotation*) | *text* | Краткое описание |

Описание атрибутов сущности «Авторы» (*authors*) представлено в таблице 2.8.

Таблица 2.8 – Атрибуты сущности «Авторы»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Идентификатор автора (*IDauthor*) | *bigserial* | Уникальный ключ |
| Фамилия имя отчество (*fullName*) | *varchar*(100) | Фамилия имя отчество |
| Дата рождения (*dateOfBirth*) | *date* | Дата рождения автора |
| Биография (*biography*) | *text* | Краткая биография автора |

Описание атрибутов сущности «Жанры» (*genres*) представлено в таблице 2.9.

Таблица 2.9 – Атрибуты сущности «Жанры»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Идентификатор жанра (*IDgenre*) | *bigserial* | Уникальный ключ |
| Жанр (*genre*) | *varchar*(50) | Название жанра |

Описание атрибутов сущности «Администраторы» (*admins*) представлено в таблице 2.10.

Таблица 2.10 – Атрибуты сущности «Администраторы»

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип данных | Описание |
| Идентификатор администратора (*IDadmin*) | *bigserial* | Уникальный ключ |

Продолжение таблицы 2.10

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибут | Тип данных | | Описание |
| Электронная почта (*email*) | *varchar*(100) | | Электронная почта пользователя |
| Логин (*login*) | *varchar*(100) | | Персональное имя для входа в систему |
| Пароль (*password*) | *varchar*(50) | | Пароль в формате *BCrypt* |
| Фамилия имя отчество (*fullName*) | | *varchar*(100) | Фамилия имя отчество | |
| Дата рождения (*dateOfBirth*) | | *date* | Дата рождения | |
| Номер телефона (*phoneNumber*) | | *varchar*(20) | Номер телефона | |
| Адрес (*address*) | | *varchar*(100) | Адрес | |

Между сущностями можно установить одну из следующих связей: многие ко многим, один ко многим, один к одному.

Если нескольким записям из одной таблицы соответствует несколько записей из другой таблицы, то такая связь называется «многие ко многим» и организовывается посредством связывающей таблицы.

В типе связей один ко многим одной записи первой таблицы соответствует несколько записей в другой таблице.

Связь один к одному образуется, когда ключевой столбец (идентификатор) присутствует в другой таблице, в которой тоже является ключом либо свойствами столбца задана его уникальность.

Связи в сущностях определяются с учётом трёх нормальных форм.

Связь многие ко многим установлена между сущностями «книги» и «авторы», «книги» и «читательские билеты», «книги» и «жанры».

Связь один к одному установлена между таблицами «читательские билеты» и «пользователи», «истории книговыдачи» и «книжные отзывы».

Сущности «забронированные книги» и «истории книговыдачи» реализуют связь многие ко многим между сущностями «книги» и «читательские билеты».

Сущность «книги и авторы» реализует связь многие ко многим между сущностями «книги» и «автора».

Сущность «книги и жанры» реализует связь многие ко многим между сущностями «книги» и «жанры».

На рисунке 2.3 представлена *ER*-диаграмма спроектированной базы данных.

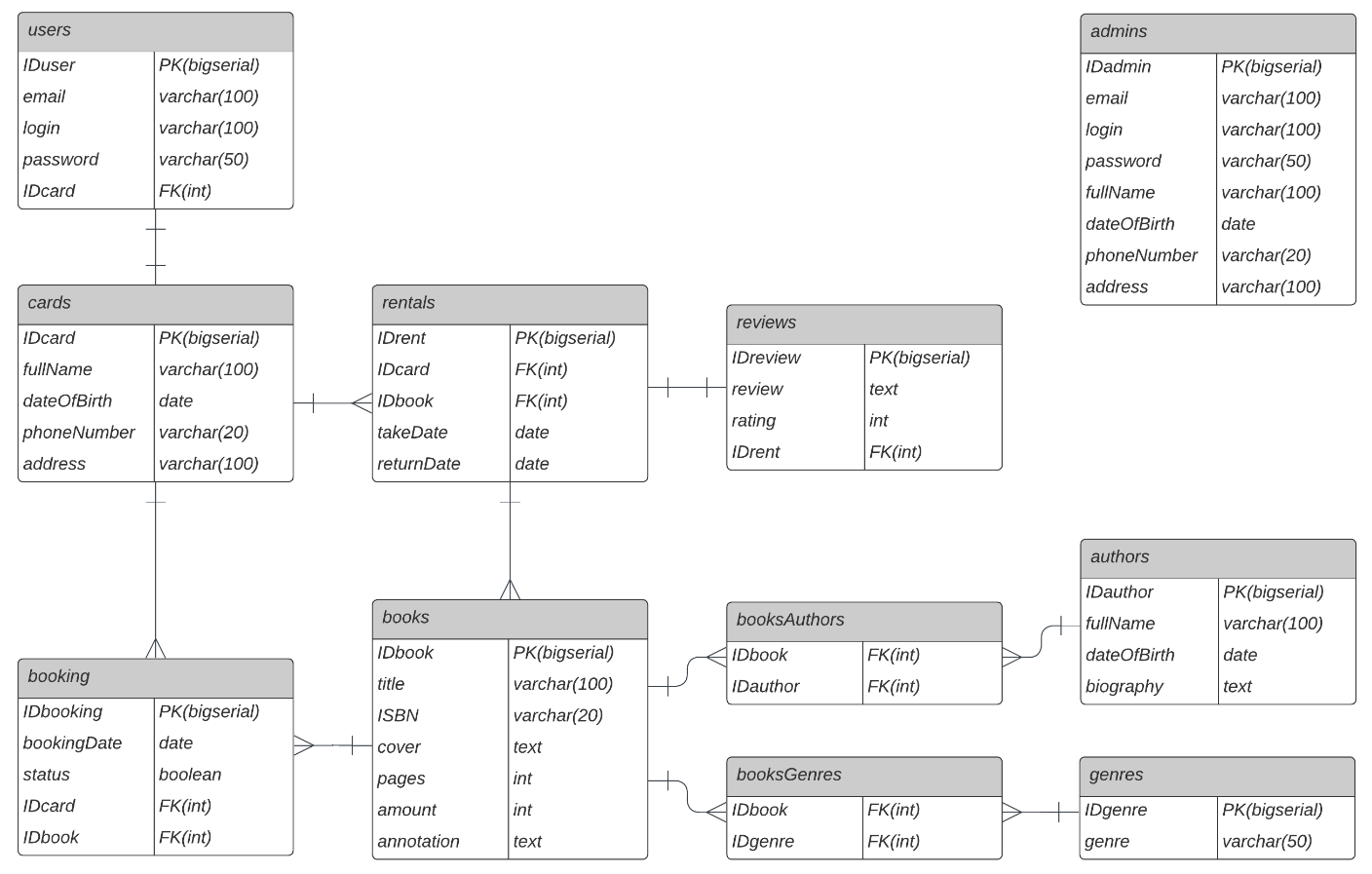


Рисунок 2.4 – *ER*-диаграмма базы данных библиотеки

**2.3.2** *DFD*-диаграммы создаются для моделирования существующего процесса движения информации. Для изображения входных и выходных данных, точек хранения информации и путей ее передвижения в таких диаграммах используются следующие элементы:

* работы – любые процессы, которые ведут к изменению информации и созданию выходных данных;
* хранилища данных – файлы или репозитории, где хранится информация для последующего использования, например, базы данных или формы заявки на участие;
* внешние ссылки – внешние системы, из которых поступает или куда направляется информация в результате взаимодействия с изображаемой системой;
* стрелки – маршруты, по которым информация перемещается между внешними сущностями, процессами и хранилищами данных.

Диаграммы *DFD* дополняют то, что уже отражено в модели *IDEF*0, поскольку они описывают потоки данных, позволяя проследить, каким образом происходит обмен информацией как внутри системы между функциональными блоками, так и системы в целом с внешней информационной средой.

*DFD*-диаграмма разработана для описания процесса обработки информации при входе в систему. Её внешний вид представлен на рисунке 2.4.

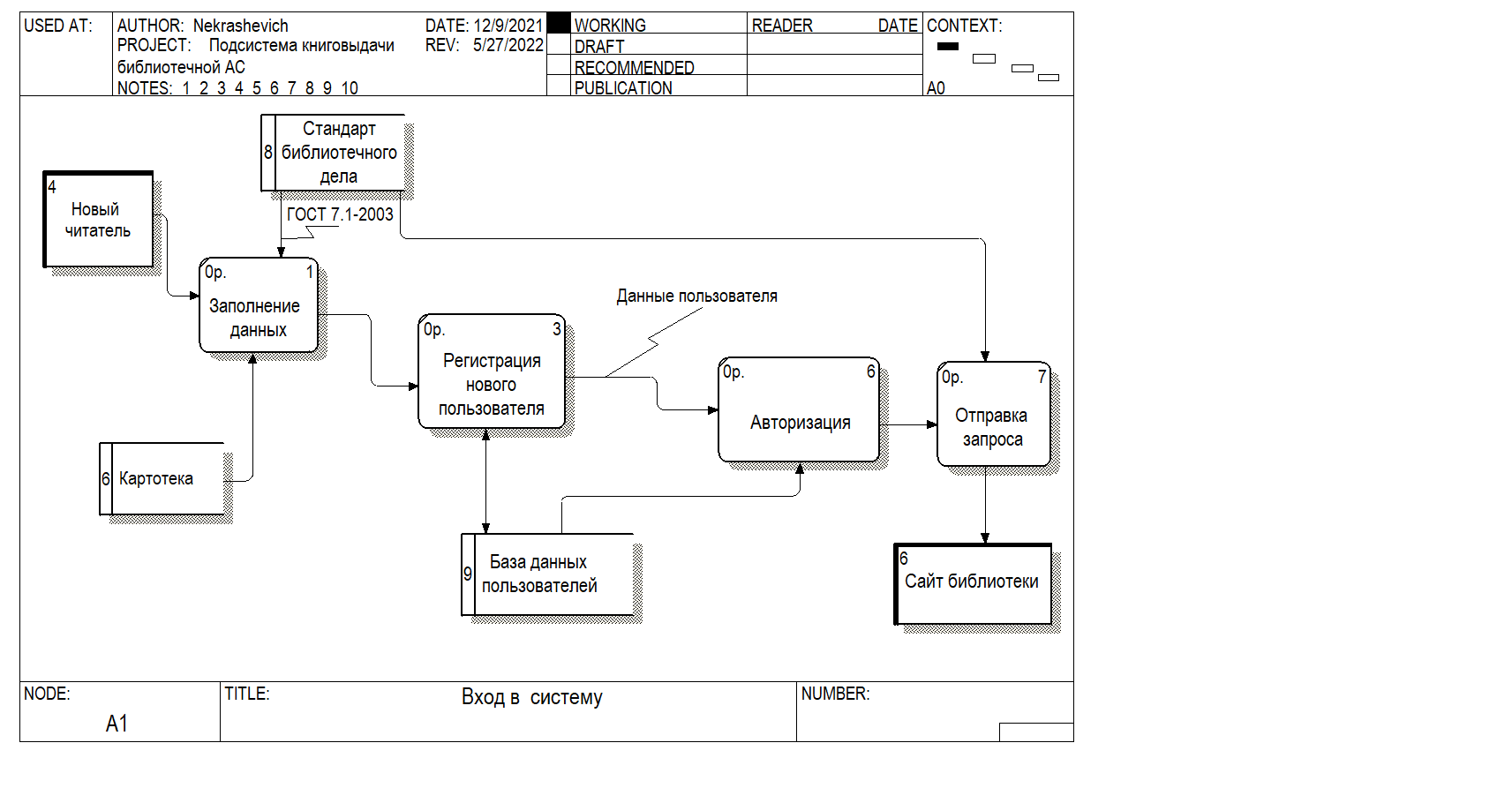


Рисунок 2.5 – *DFD*-диаграмма процесса входа в систему

На данной диаграмме присутствуют следующие работы:

* заполнение данных – заполнение новым пользователем данных, необходимых для регистрации;
* регистрация нового пользователя – занесение данных о новом пользователе в базу данных библиотеки;
* авторизация – ввод данных, необходимых для взаимодействия пользователя с системой;
* отправка запроса – получение информации о пользователе и предоставление ему возможности отправлять запросы.

Внешними ссылками диаграммы являются:

* новый читатель – незарегистрированный пользователь системы;
* сайт библиотеки – приёмник информации о пользователе.

Хранилища данных, представленные на диаграмме:

* стандарт библиотечного дела, предписывающий правила библиографических записей;
* картотека, хранящая информацию о читательских билетах;
* база данных с информацией о пользователях.
  1. Математическое и алгоритмическое обеспечение

**2.4.1** Рекомендательная система — комплекс алгоритмов, программ и сервисов, задача которого предсказать, что может заинтересовать того или иного пользователя. В основе работы лежит информация о профиле человека и иные данные.

Существует четыре типа рекомендательных систем:

* Коллаборативная фильтрация (*collaborative filtering*);
* Основанные на контенте (*content-based*);
* Основанные на знаниях (*knowledge-based*);
* Гибридные (*hybrid*).

В автоматизированной информационной системе библиотеки используется коллаборативная фильтрация с применением коэффициента корреляции.

Методы коллаборативной фильтрации основаны на истории оценок как самого пользователя, так и других.

В основе коллаборативной фильтрации книг лежит идея о том, что интерес к определенной книге – показатель того, что пользователю также будет интересна и другая книга.

Чтобы получить рекомендации, система коллаборативной фильтрации книг связывает между собой две основные сущности: пользователи и книги. Простейший способ связи – явно указанный пользователем рейтинг.

Используя данные об оценке книг пользователями, получается матрица «книга-количество оценок». Каждый столбец содержит все оценки пользователей для конкретной книги. Вид матрицы представлен в табличном виде (см. таблицу 2.11).

Таблица 2.11 – Вид матрицы «книга-количество оценок»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Книга 1 | Книга 2 | … | Книга *n* |
| Оценка «1» | Количество оценок «1» для «Книги 1» | Количество оценок «1» для «Книги 2» | … | Количество оценок «1» для «Книги *n*» |
| Оценка «2» | Количество оценок «2» для «Книги 1» | Количество оценок «2» для «Книги 2» | … | Количество оценок «2» для «Книги *n*» |
| Оценка «3» | Количество оценок «3» для «Книги 1» | Количество оценок «3» для «Книги 2» | … | Количество оценок «3» для «Книги *n*» |

Продолжение таблицы 2.11

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Книга 1 | Книга 2 | … | Книга *n* |
| Оценка «4» | Количество оценок «4» для «Книги 1» | Количество оценок «4» для «Книги 2» | … | Количество оценок «4» для «Книги *n*» |
| Оценка «5» | Количество оценок «5» для «Книги 1» | Количество оценок «5» для «Книги 2» | … | Количество оценок «5» для «Книги *n*» |

Схожесть предпочтений пользователя с другими пользователями считается при помощи коэффициента корреляции.

Коэффициент корреляции – это статистическая мера, которая вычисляет силу связи между двумя наборами данных. Значения коэффициента корреляции находятся в диапазоне от -1.0 до 1.0. Если вычисленное число больше 1.0 или меньше -1.0, то это свидетельствует о наличии погрешности в измерении корреляции.

Коэффициент корреляции высчитывается по формуле 2.1. Код написания функции расчёта коэффициента корреляции приведён в приложении А.

|  |  |  |
| --- | --- | --- |
|  |  | (2.1) |

где *n* – длина столбца матрицы; *x* – набор оценок пользователей для книги из личного рейтинга; *y* – набор оценок пользователей для сравниваемой книги.

Коэффициент корреляции высчитывается для каждой книги пользователя, оценка которой больше четырёх, и поочерёдно всех оставшихся книг из матрицы подобия. В результате для каждого пользователя составляется индивидуальный массив объектов, содержащий идентификатор сравниваемых книг и их коэффициент корреляции по отношению к книгам с высокими оценками от этого пользователя.

Рекомендательная система будет отображать книги из массива, чей коэффициент корреляции по модулю больше либо равен семи.

В случае, если пользователь не выставляет оценки, его рекомендации будут состоять из 15 наиболее популярных книг среду других пользователей. Популярной считается книга, имеющая наибольший средний рейтинг.

Блок-схема алгоритма рекомендательной системы представлена на рисунке 2.6.

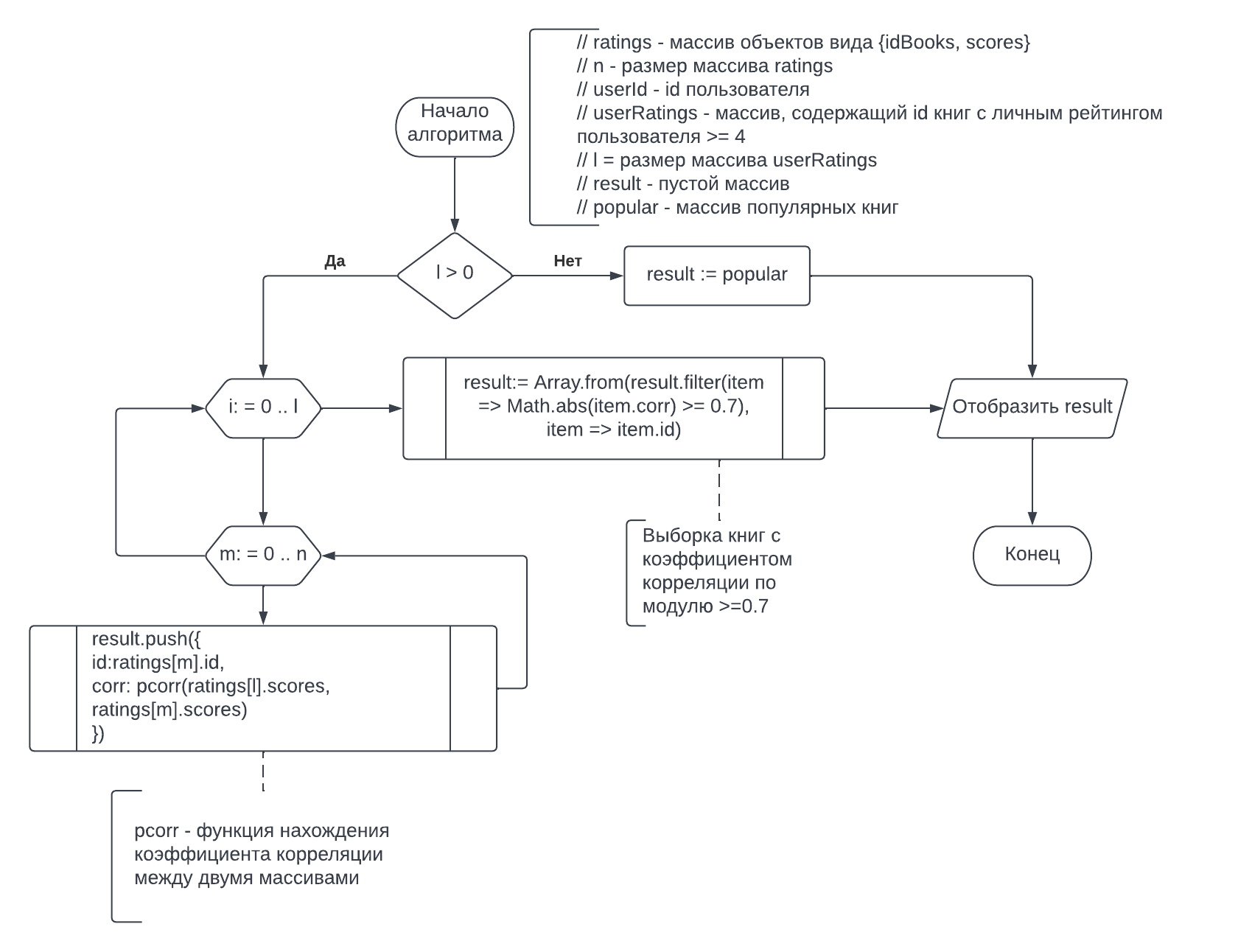


Рисунок 2.6 – Блок-схема алгоритма подбора рекомендаций

**2.4.2** *IDEF*3-диаграмма позволяет описать процесс как упорядоченную последовательность событий. Диаграмма использует следующие типы объектов:

* работа – изображается прямоугольником с прямыми углами и имеет имя, обозначающее процесс действия;
* стрелка – показывают взаимоотношение работ;
* перекрёсток – окончание одной работы может служить сигналом к началу нескольких работ, или же одна работа для своего запуска может ожидать окончания нескольких работ.

Методология *IDEF*3 применятся для описания сценария поиска книг. Функциональный блок начинает своё выполнение на моменте, когда пользователь отправляет поисковой запрос. После отправки запроса, в зависимости от введённых данных, система начнёт поиск по одной или нескольким категориям: поиск по названию книги, поиск по автору, поиск по I*SBN*, поиск по жанру. В зависимости от полученного из базы данных ответа система выведет ошибку или сформирует конечный результат. В случае формирования результата система либо выведет сообщение об отсутствии книги, либо отобразит запрашиваемые в поиске книги.

Внешний вид описанной *IDEF*3-диаграммы представлен на рисунке 2.7.

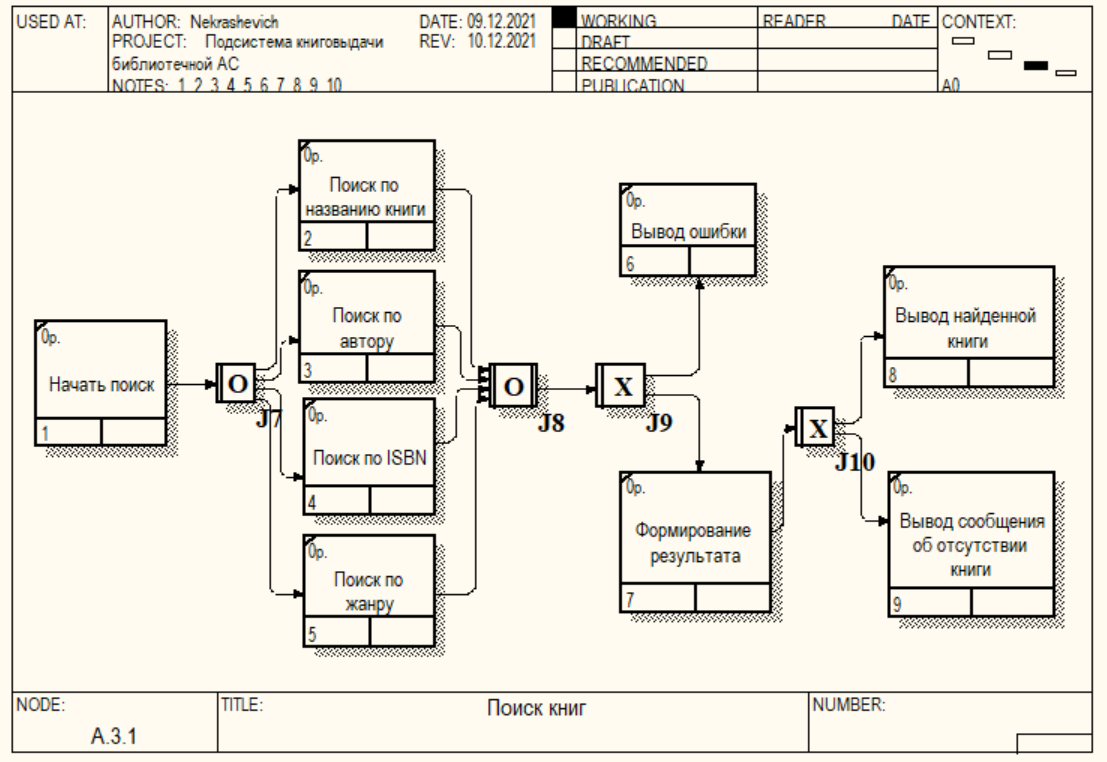


Рисунок 2.7 – *IDEF*3-диаграмма сценария поиска книг

**2.4.3** Диаграммы активностей являются представлением алгоритмов неких действий (активностей), выполняющихся в системе. Эта диаграмма представляет собой блок-схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой.

Диаграммы деятельности состоят из ограниченного количества фигур, соединённых стрелками. Основные фигуры диаграммы:

1. Прямоугольники с закруглениями – обозначают действия (операция).
2. Узел управления – это абстрактный узел действия, которое координирует потоки действий.
3. Ромбы – представляют собой решения. Узел решения предназначен для определения правила ветвления и различных вариантов дальнейшего развития сценария. В точку ветвления входит ровно один переход, а выходит два или более.
4. Широкие полосы – начало (разветвление) и окончание (схождение) ветвления действий. Узел объединения имеет два и более входящих узла и один исходящий.
5. Чёрный круг – начало процесса (начальный узел). Начальный узел деятельности является узлом управления, в котором начинается поток (или потоки) при вызове данной деятельности извне.
6. Чёрный круг с обводкой – окончание процесса (финальный узел). Конечный узел деятельности (или конечное состояние деятельности) является узлом управления, который останавливает все потоки данной диаграммы деятельности. На диаграмме может быть более одного конечного узла.

Стрелки идут от начала к концу процесса и показывают потоки управления или потоки объектов (данных).

Для графического отображения алгоритма авторизации построена диаграмма активностей для прецедента «Авторизоваться», представленная на рисунке 2.8.

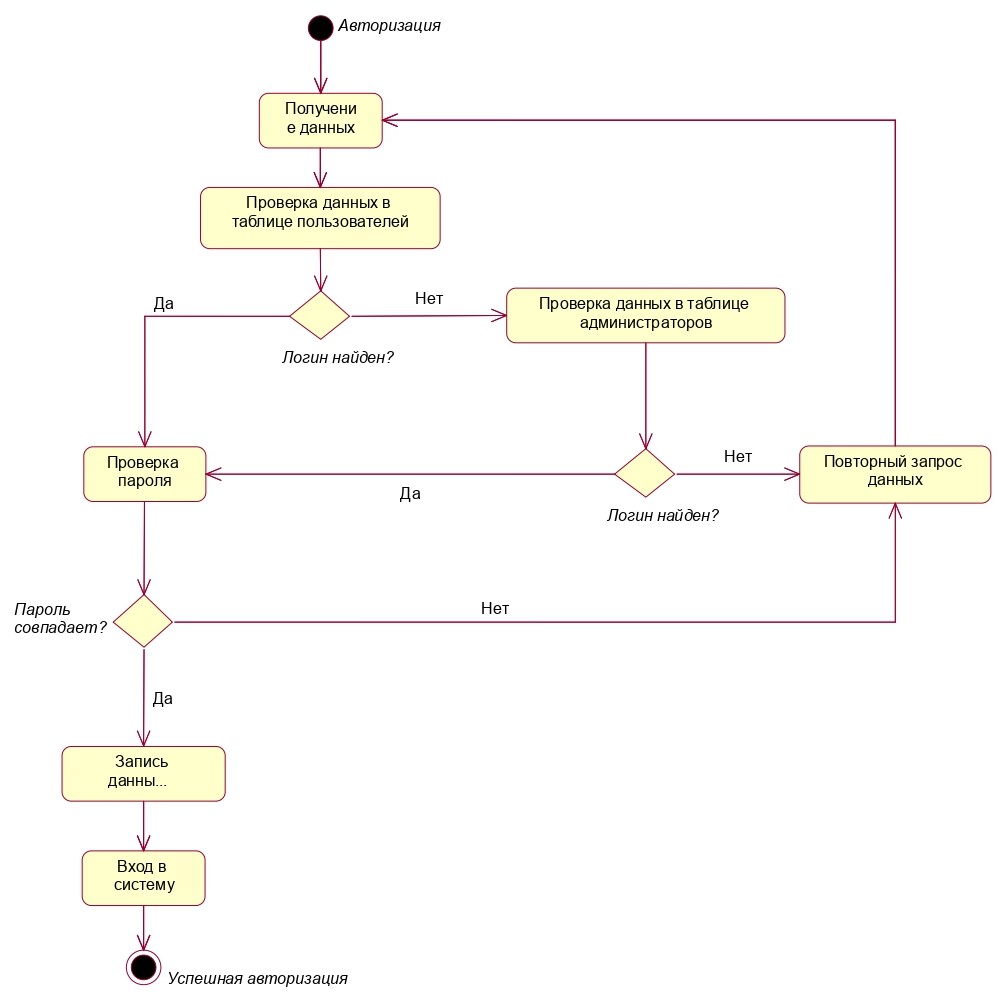


Рисунок 2.8 – диаграмма активностей для прецедента «Авторизоваться»

В зависимости от того, в какой таблице был найден пользователь, система при входе присвоит соответствующие права доступа. Если данные пользователя найдены в таблице пользователей, в куки браузера запишется статус «*user*». Если данные пользователя найдены в таблице администраторов, в куки браузера запишется статус «*admin*».

* 1. Организационное обеспечение

Для безопасной аутентификации пользователей, в автоматизированной информационной системе используется *JSON Web Token* (*JWT*). Токены создаются сервером, подписываются секретным ключом и передаются клиенту, который в дальнейшем использует данный токен для подтверждения своей личности.

После того, как пользователь прошел авторизацию в системе, указав свой логин и пароль, система выдает ему два токена: *access token* и *refresh token*. Для взаимодействием с системой, пользователь вместе с запросом передает свой *access token*. Сервер, получив его, проверяет, что он действительный, вычитывает полезные данные из него и, таким образом, может идентифицировать пользователя.

Токен разделен на три основные группы: заголовок, полезные данные и сигнатура, разделенные между собой точкой.

Заголовок – это первая часть токена. Он закодирован в *Base*64.

Полезные данные представляют собой некоторый набор информации о пользователе. В случае данной системы, полезными данными является идентификатор пользователя.

Сигнатура является секретной фразой, хранящейся на стороне сервера. И представляет собой захешированный набор данных.

*Access token* имеет короткий срок жизни, всего один день. Как только время выйдет, пользователю снова придётся проходить авторизацию. Чтобы этого избежать, пользователю также выдаётся *refresh token*. *Refresh token* имеет длительное время жизни – месяц, но он одноразовый и служит исключительно для обновления *access token* пользователя.

Схема аутентификации в таком случае выглядит следующим образом:

* пользователь проходит процедуру аутентификации и получает от сервера *access token* и *refresh token*;
* при обращении к ресурсу пользователь передает в запросе свой *access token*, на основе которого сервер идентифицирует и авторизует клиента;
* при истечении *access token* клиент передает в запросе свой *refresh token* и получает от сервера новые *access token* и *refresh token*;
* при истечении *refresh token* пользователь заново проходит процедуру аутентификации.

Блок-схема работы *jwt* безопасности показана на рисунке 2.9.

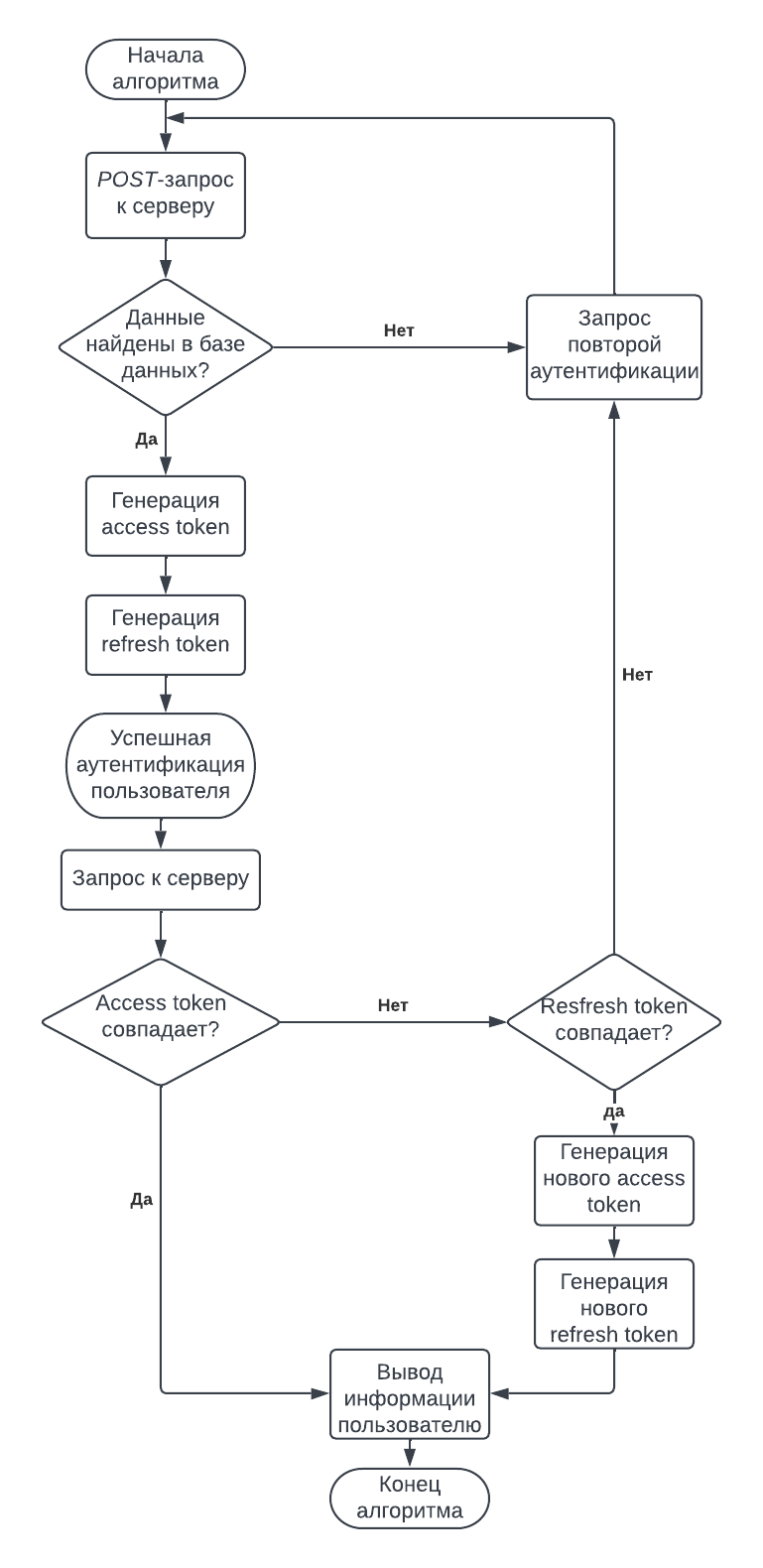


Рисунок 2.9 – Блок-схема работы *jwt* безопасности

Токены доступа пользователя хранятся в куки браузера.

Для дополнительной безопасности пользователей их пароли в базе данных хранятся с использованием криптографической функции *bcrypt*.

Код обращения пользователя к *jwt* токенам через сервер приведён в приложении Б.

1. Реализационная часть
   1. Техническое обеспечение

Веб-сайт – это набор веб-страниц связанных общей тематикой. Веб-сайт находится на одном сервере (хостинге) и принадлежит одному владельцу.

Веб-страница строится на основе языка разметки гипертекста. Официальное название этого языка *HTML* (*Hyper Text Markup Language*).

Для отражения содержимого на веб-страницы созданы и служат каскадные таблицы стилей – *CSS*.

В создании динамических страниц, на помощь разработчикам «приходят» языки скриптов. Самый популярный язык *JavaScript*.

С помощью этих трех языков можно создать любую, даже самую сложную, веб – страницу, а созданные веб-страницы собрать в веб-сайт.

**3.1.1** *HTML* (*Hypertext Markup Language*) – это код, который используется для структурирования и отображения веб-страницы и её контента. Например, контент может быть структурирован внутри множества параграфов, маркированных списков или с использованием изображений и таблиц данных.

Веб-браузеры получают *HTML* документ от сервера по протоколам *HTTP/HTTPS* или открывают с локального диска, далее интерпретируют код в интерфейс, который будет отображаться на экране монитора.

Элементы *HTML* являются строительными блоками *HTML* страниц. С помощью *HTML* разные конструкции, изображения и другие объекты такие как интерактивная веб-формы могут быть встроены в отображаемую страницу. *HTML* предоставляет средства для создания заголовков, абзацев, списков, ссылок, цитат и других элементов. Элементы *HTML* выделяются тегами, записанными с использованием угловых скобок. Некоторые теги могут напрямую вводить контент на страницу, другие – окружать и оформлять текст внутри себя и включать другие теги в качестве подэлементов. Браузеры не отображают *HTML*-теги, но используют их для интерпретации содержимого страницы.

Изначально язык *HTML* задуман и создан как средство структурирования и форматирования документов без их привязки к средствам воспроизведения (отображения). В идеале, текст с разметкой *HTML* должен без стилистических и структурных искажений воспроизводиться на оборудовании с различной технической оснащённостью. Однако современное применение *HTML* очень далеко от его изначальной задачи. С течением времени основная идея платформонезависимости языка *HTML* принесена в жертву современным потребностям в мультимедийном и графическом оформлении.

**3.1.2** *CSS* (*Cascading Style Sheets*) – формальный язык описания внешнего вида документа (веб-страницы), написанного с использованием языка разметки *HTML*.

*CSS* используется создателями веб-страниц для задания цветов, шрифтов, стилей, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки *CSS* являлось отделение описания логической структуры веб-страницы (которое производится с помощью *HTML* или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка *CSS*). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

Кроме того, *CSS* позволяет представлять один и тот же документ в различных стилях или методах вывода, таких как экранное представление, печатное представление, чтение голосом (специальным голосовым браузером или программой чтения с экрана) или при выводе устройствами, использующими шрифт Брайля.

**3.1.3** *JavaScript* – мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили. Является реализацией спецификации *ECMAScript*.

*JavaScript* обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

*JavaScript* является объектно-ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс-ориентированными языками. Кроме того, *JavaScript* имеет ряд свойств, присущих функциональным языкам – функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания – что придаёт языку дополнительную гибкость.

Свойства, присущие *JavaScript*:

* объекты с возможностью интроспекции;
* функции как объекты первого класса;
* автоматическое приведение типов;
* автоматическая сборка мусора;
* анонимные функции.

**3.1.4** *React* (иногда *React.js* или *ReactJS*) – *JavaScript*-библиотека с открытым исходным кодом для разработки пользовательских интерфейсов. React может использоваться для разработки одностраничных и мобильных приложений. Его цель – предоставить высокую скорость, простоту и масштабируемость.

Основная цель *React* – минимизировать ошибки, возникающие при разработке пользовательских интерфейсов. Это достигается за счёт использования компонентов – автономных логических фрагментов кода, которые описывают часть пользовательского интерфейса. А уже эти компоненты объединяются для создания полноценного пользовательского интерфейса. *React* абстрагирует большую часть работы по визуализации, оставляя возможность сосредоточиться на дизайне.

*React* может отвечать за одну кнопку, несколько частей или же весь пользовательский интерфейс приложения.

**3.1.5** *Node* (или более формально *Node.js*) – кросплатформенная среда исполнения с открытым исходным кодом, которая позволяет разработчикам создавать всевозможные серверные инструменты и приложения используя язык *JavaScript*. Среда исполнения предназначена для использования вне контекста браузера (т.е. выполняется непосредственно на компьютере или на серверной ОС). Таким образом, среда исключает *API*-интерфейсы *JavaScript* для браузера и добавляет поддержку более традиционных *OS API*-интерфейсов, включая библиотеки *HTTP* и файловых систем.

*Node* разработан для оптимизации пропускной способности и масштабируемости в веб-приложениях и очень хорошо справляется со многими распространёнными проблемами веб-разработки (например, веб-приложения реального времени).

Код написан на «обычном старом *JavaScript*», а это означает, что затрачивается меньше времени при написании кода для браузера и веб-сервера связанное с «переключением технологий» между языками.

Менеджер пакетов *Node* (*NPM*) обеспечивает доступ к сотням тысяч многоразовых пакетов. Он также имеет лучшее в своём классе разрешение зависимостей и может также использоваться для автоматизации большинства инструментов построения.

**3.1.6** *Express* – самый популярный веб-фреймворк для *Node*. Он предоставляет следующие механизмы:

1. Написание обработчиков для запросов с различными *HTTP*-методами в разных *URL*-адресах (маршрутах).
2. Интеграцию с механизмами рендеринга «*view*», для генерации ответов, вставляя данные в шаблоны.
3. Установка общих параметров веб-приложения, такие как порт для подключения, и расположение шаблонов, которые используются для отображения ответа.

В традиционных динамических веб-сайтах, веб-приложение ожидает *HTTP*-запроса от веб-браузера (или другого клиента). Когда запрос получен, приложение определяет, какое действие необходимо выполнить на основе *URL* шаблона и, возможно, связанной информации, содержащейся в данных *POST* или *GET*. В зависимости от того, что требуется, *Express* может затем читать или записывать данные из/в базы данных или выполнять другие задачи, в соответствии с полученным запросом. Затем приложение возвращает ответ в веб-браузер, зачастую динамически создавая *HTML* страницу для отображения браузером, вставляя извлечённые данные в заполнители *HTML* шаблона.

*Express* предоставляет методы, позволяющие указать, какая функция вызывается для конкретного *HTTP* запроса (*GET*, *POST*, *SET*), и *URL* шаблон ("*Route*"), а также методы, позволяющие указать, какой механизм шаблона ("*view*") используется, где находятся шаблоны файлов и какой шаблон использовать для вывода ответа.

**3.1.7** *PostgreSQL* – это популярная свободная объектно-реляционная система управления базами данных. *PostgreSQL* базируется на языке *SQL* и поддерживает многочисленные возможности.

Преимущества *PostgreSQL*:

* поддержка БД неограниченного размера;
* мощные и надёжные механизмы транзакций и репликации;
* расширяемая система встроенных языков программирования и поддержка загрузки C-совместимых модулей;
* наследование;
* легкая расширяемость.

*PostgreSQL* не просто реляционная, а объектно-реляционная СУБД. Это даёт ему некоторые преимущества над другими *SQL* базами данных с открытым исходным кодом.

Существует обширный список типов данных, которые поддерживает Постгрес. Кроме числовых, с плавающей точкой, текстовых, булевых и других ожидаемых типов данных (а также множества их вариаций), *PostgreSQL* может похвастаться поддержкой *uuid*, денежного, перечисляемого, геометрического, бинарного типов, сетевых адресов, битовых строк, текстового поиска, *xml*, *json*, массивов, композитных типов и диапазонов, а также некоторых внутренних типов для идентификации объектов и местоположения логов.

* 1. Программное обеспечение

**3.2.1** Диаграмма развертывания – это тип *UML*-диаграммы, которая показывает архитектуру исполнения системы, включая такие узлы, как аппаратные или программные среды исполнения, а также промежуточное программное обеспечение, соединяющее их.

Диаграммы развертывания обычно используются для визуализации физического аппаратного и программного обеспечения системы. Использование диаграмм развёртывания позволяет понять, как система будет физически развернута на аппаратном обеспечении.

Диаграммы развертывания помогают моделировать аппаратную топологию системы по сравнению с другими типами *UML*-диаграмм, которые в основном описывают логические компоненты системы.

*Back-end* часть подсистемы книговыдачи библиотечной АС расположена на веб-сервере *Express*, который при помощи дополнительного пакета модулей подключается к базе данных, расположенной в СУБД *PostgreSQL*. Пользователь может взаимодействовать с серверной частью, используя любой из доступных браузеров. Для реализации клиент-серверных запросов используется *REST*-архитектура.

Для визуализации инфраструктуры, на которой развёрнуто веб-приложение автоматизированной информационной библиотеки, построена диаграмма, показанная на рисунке 3.1.

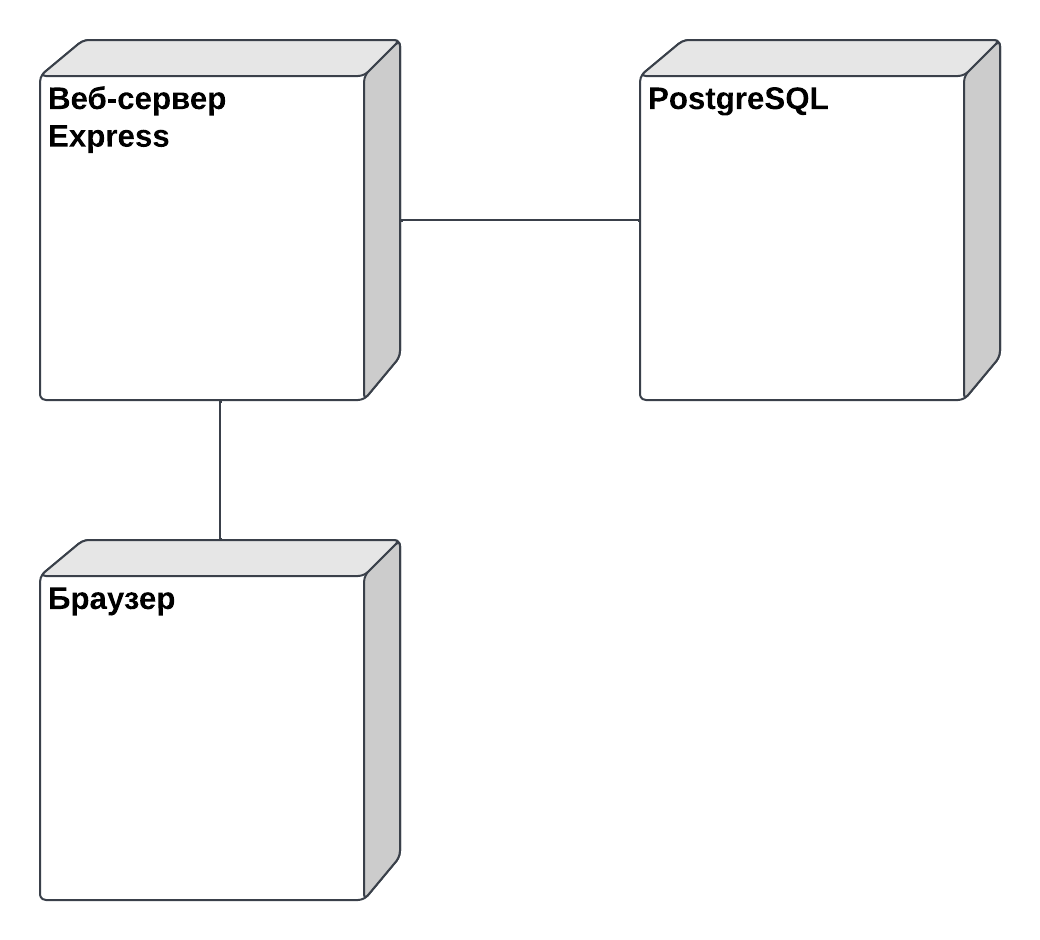


Рисунок 3.1 – Диаграмма развёртывания

**3.2.2** Диаграмма вариантов использования (сценариев поведения, прецедентов) в *UML* является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью вариантов использования. При этом актером называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация функций, которые система предоставляет актеру.

Вариант использования обозначается на диаграмме эллипсом, внутри которого содержится его описание, обозначающее выполнение какой-либо операции или действия.

Связи между актерами и вариантами отображаются с использованием отношений четырех видов:

* ассоциации – семантическое отношение между двумя и более элементами;
* обобщения – отображение иерархических взаимосвязей между различными элементами;
* включения (зависимость со стереотипом «*include*») – поведение одного прецедента включается в некоторой точке в другой прецедент в качестве составного компонента;
* расширения (зависимость со стереотипом «*extend*») – отражает возможное присоединение одного варианта использования к другому в некоторой точке.

Для корректного подробного описания работы автоматизированной информационной системы библиотеки, построена диаграмма вариантов использования, актёрами являются пользователь, неавторизированный пользователь администратор.

Администратору предоставляется возможность одобрить бронирование, запрошенное пользователем, оформить возврат книги, добавить в базу данных автора или книгу. Администратор и пользователь связаны между собой отношением обобщения. Это значит, что администратору доступны все варианты использования пользователя.

Пользователю доступны следующие варианты использования: забронировать книгу, просмотреть рекомендации, просмотреть личную историю книговыдачи и добавить отзыв. Вариант использования «добавить отзыв» доступен только после просмотра личной истории книговыдачи. Таким образом, пользователь может поставить оценку книге только после её прочтения.

Неавторизованному пользователю доступна авторизация, поиск в каталоге и просмотр информации об авторах или книгах. Неавторизированный пользователь и пользователь связаны между собой отношением обобщения.

Спроектированная диаграмма прецедентов представлена на рисунке 3.2.

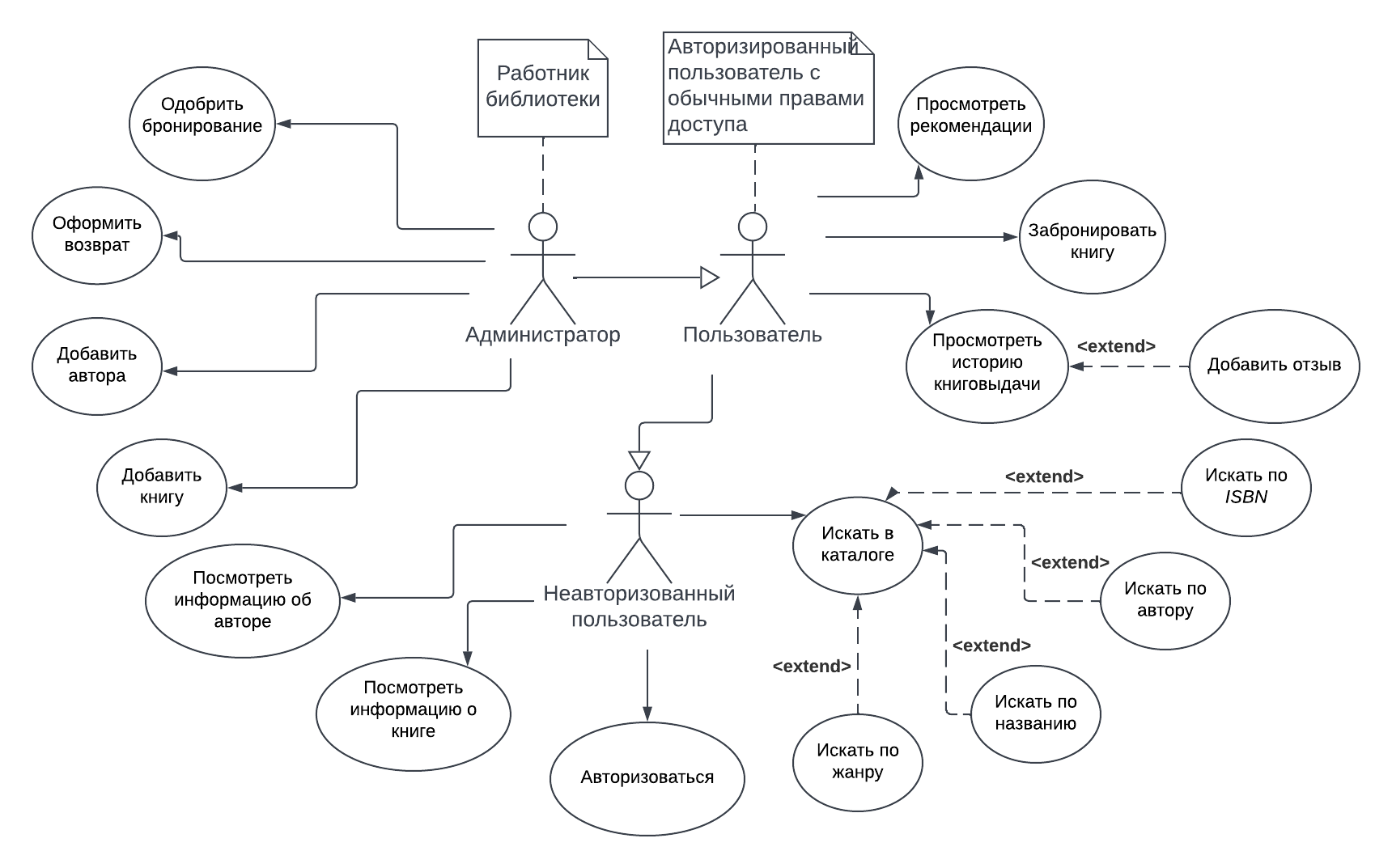


Рисунок 3.2 – Диаграмма вариантов использования

**3.2.3** Диаграммы последовательности используются для уточнения диаграмм прецедентов, более детального описания логики сценариев использования.

Диаграммы последовательностей обычно содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями.

Объекты обозначаются прямоугольниками с подчеркнутыми именами (чтобы отличить их от классов), сообщения (вызовы методов) – линиями со стрелками, возвращаемые результаты – пунктирными линиями со стрелками.

Прямоугольники на вертикальных линиях под каждым из объектов показывают время жизни объектов.

Для автоматизированной информационной системы библиотеки спроектированы диаграммы последовательностей для сценариев «добавление отзыва» и «бронирование книги», представленные на рисунках 3.3 и 3.4 соответственно.

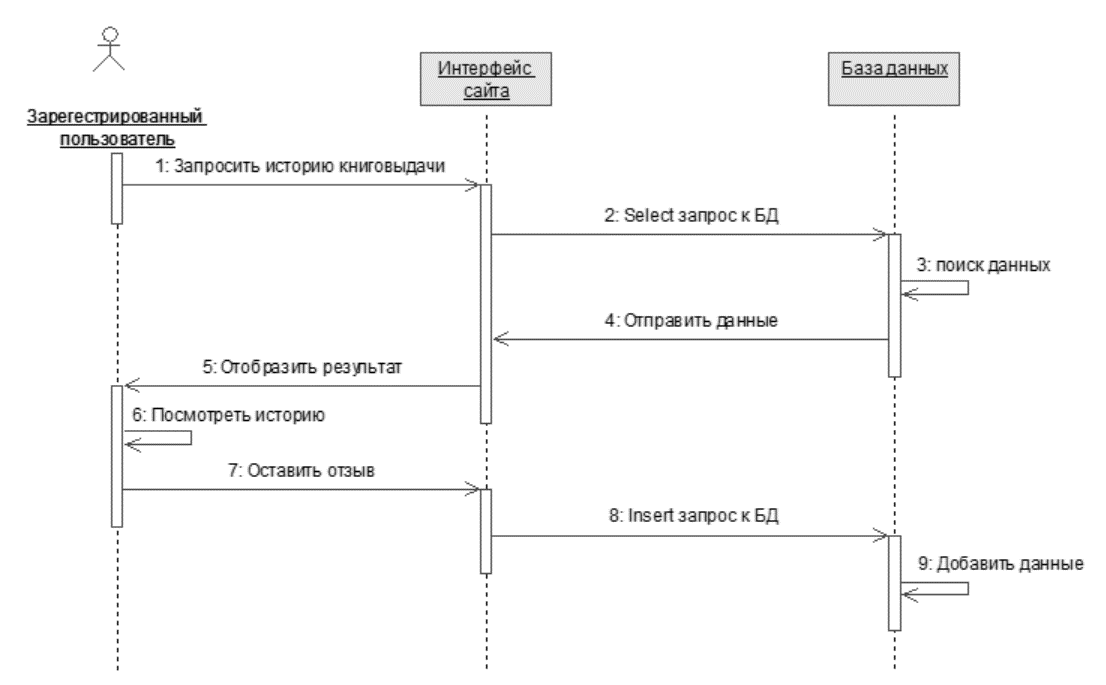


Рисунок 3.3 – Диаграмма последовательности сценария «Добавить отзыв»

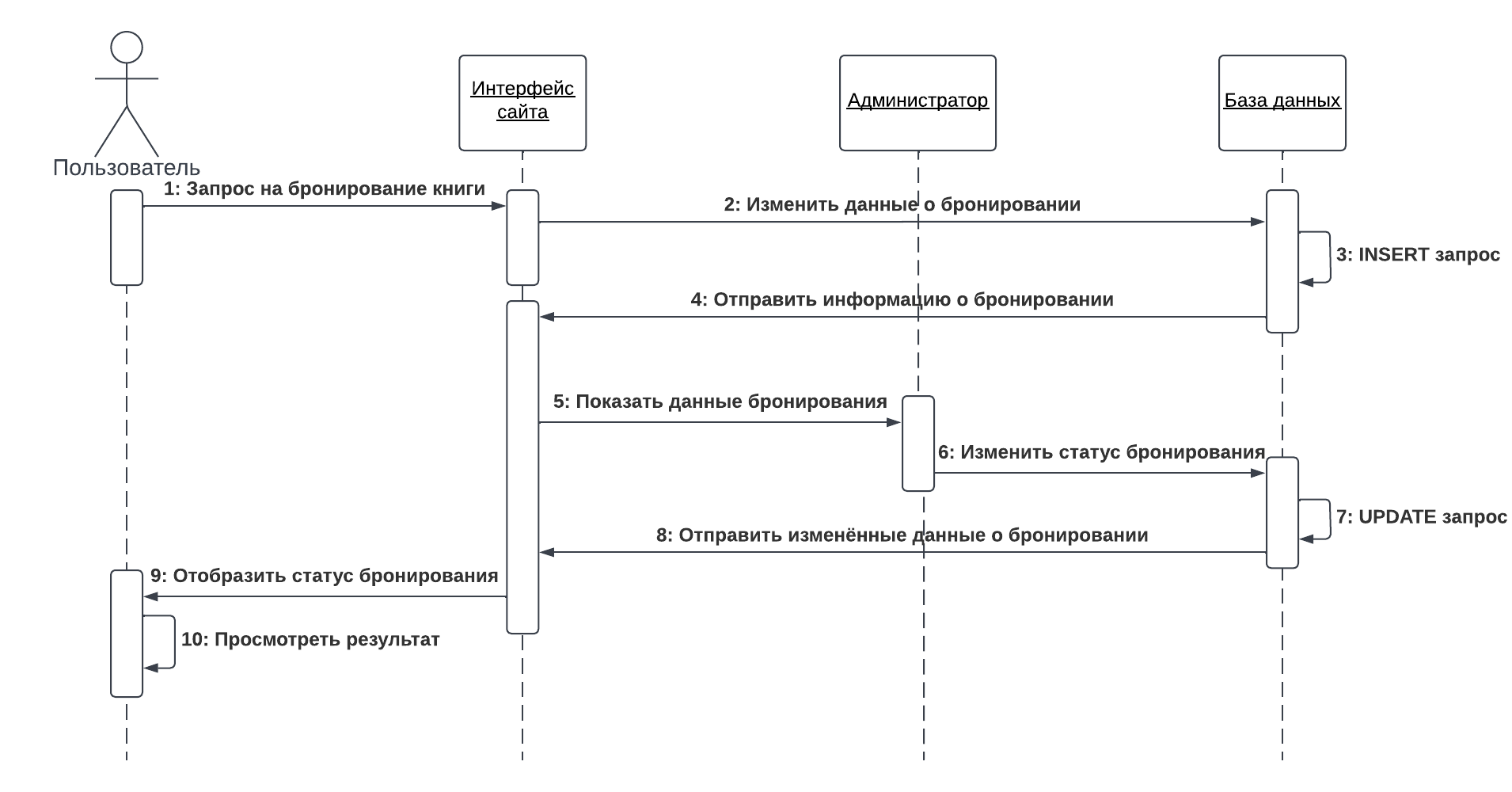


Рисунок 3.4 – Диаграмма последовательности сценария «Забронировать книгу»

**3.2.4** Для того чтобы сайт обладал полным функционалом, необходимо настроить подключение к базе данных. Для этого используется следующий фрагмент кода:

*const {Sequelize} = require('sequelize');*

*require('dotenv').config();*

*const sequelize = new Sequelize(process.env.DB\_NAME, process.env.DB\_USERNAME, process.env.DB\_PASS, {*

*host: process.env.DB\_HOST,*

*dialect: "postgres",*

*define: {*

*"timestamps": false*

*}}*

*);*

*module.exports = sequelize;*

В настройках используются данные, импортируемые из файла .*env*, который содержит пример всех переменных, необходимых для настройки серверной части системы. Содержимое файла следующее:

*DB\_NAME = name*

*DB\_USERNAME = username*

*DB\_PASS = pass*

*DB\_HOST = localhost*

*PORT = 3001*

*JWT\_KEY = JWT\_KEY*

*JWT\_REFRESH\_KEY = JWT\_REFRESH\_KEY*

Для обращения к базе данных и получения данных через *POST* и *GET* запросы, используется маршрутизация и метод *Route*, применимый к каждому необходимому набору информации. Для определения каждого из роутов существует отдельный файл настроек. Ниже приведён код, служащий для определения роутов авторизации пользователя:

*const express = require("express");*

*const router = express.Router();*

*const bcrypt = require("bcrypt");*

*const Users = require("../models/Users");*

*const { generateAccessToken, generateRefreshToken } = require("../JWT");*

*router.post("/login", async (req, res) => {*

*const { login, pass } = req.body;*

*const query = await Users.findOne({ where: { login: login } });*

*if (!query) {*

*return res.status(400).json({ error: "User doesn't exist" });*

*}*

*const user = query.dataValues;*

*const dbPassword = user.pass;*

*bcrypt.compare(pass, dbPassword).then((match) => {*

*if (!match) {*

*return res*

*.status(400)*

*.json({ error: "Wrong username and password combination" });*

*}*

*const accessToken = generateAccessToken(user);*

*const refreshToken = generateRefreshToken(user);*

*const response = {*

*isLogged: true,*

*accessToken,*

*refreshToken,*

*};*

*res.send(response);*

*});*

*});*

*module.exports = router;*

Основная настройка серверной стороны системы расположена в файле *index.js*. Файл содержит конфигурацию для каждого роута, а также осуществляет подключение сервера к базе данных. Фрагмент кода из файла приведён ниже:

*const express = require("express");*

*const cors = require("cors");*

*require("dotenv").config();*

*const projectsRouter = require("./routes/projectsRouter");*

*const signupRouter = require("./routes/signupRouter");*

*const loginRouter = require("./routes/loginRouter");*

*const refreshTokenRouter = require("./routes/refreshTokenRouter");*

*const createCompanyRouter = require("./routes/createCompanyRouter");*

*const app = express();*

*app.use(express.json());*

*app.use(cors());*

*app.use(express.static("public"));*

*app.use("/", booksRouter);*

*app.use("/", signupRouter);*

*app.use("/", loginRouter);*

*app.use("/", refreshTokenRouter);*

*app.use("/", createBookRouter);*

*app.use("/", createAuthorRouter);*

*app.use("/", recommendation);*

*app.listen(process.env.PORT, () => {*

*console.log("Server running on port 3001");*

*});*

Для рендеринга содержимого сайта на стороне клиента используется следующий код:

*import React from "react";*

*import ReactDOM from "react-dom";*

*import "./index.css";*

*import App from "./App";*

*import { Provider } from "react-redux";*

*import reportWebVitals from "./reportWebVitals";*

*import { HashRouter } from "react-router-dom";*

*import { store } from "./redux/store";*

*ReactDOM.render(*

*<React.StrictMode>*

*<HashRouter>*

*<Provider store={store}>*

*<App />*

*</Provider>*

*</HashRouter>*

*</React.StrictMode>,*

*document.getElementById("root")*

*);.getElementById('root')*

*);*

Для выдачи пользователю нового *refresh token* используется фрагмент кода, представленный ниже:

*import axios from "axios";*

*import { API\_URL } from "./apiURL";*

*export async function refreshToken() {*

*const refreshToken = localStorage.getItem("refreshToken");*

*try {*

*const response = await axios({*

*url: `${API\_URL}/token`,*

*method: "POST",*

*headers: { "Content-Type": "application/json" },*

*data: { refreshToken: refreshToken },*

*});*

*const accessToken = await response.data;*

*localStorage.setItem("accessToken", accessToken);*

*return true;*

*} catch (error) {*

*return false;*

*}*

*}*

* 1. Руководство пользователя

Для использования системы пользователю необходимо авторизоваться. Для этого необходимо нажать иконку «Войти», расположенную вверху сайта. Внешний вид верхней панели показан на рисунке 3.2.



Рисунок 3.2 – Верхняя панель сайта

После нажатия кнопки пользователю будет предложена форма входа, представленная на рисунке 3.3.

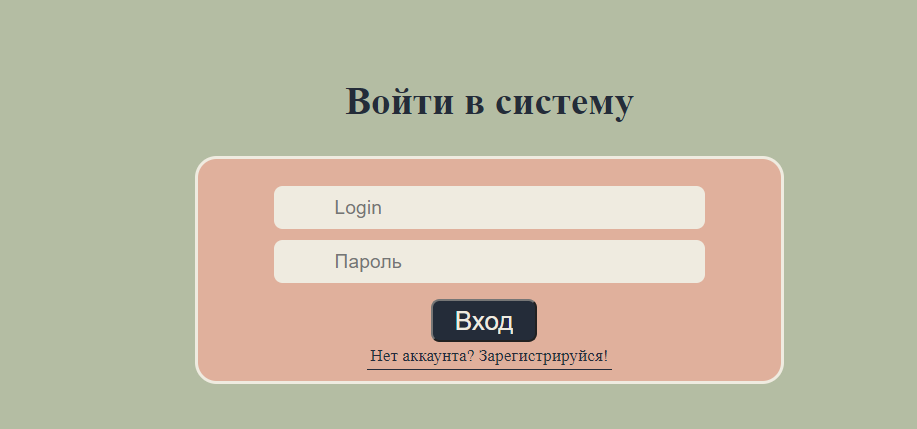


Рисунок 3.3 – Форма входа в систему

Если пользователь заходит на сайт впервые, ему необходимо нажать зарегистрироваться. Для открытия формы регистрации необходимо нажать на текст «Нет аккаунта? Зарегистрируйся!». Внешний вид формы регистрации показан на рисунке 3.4.

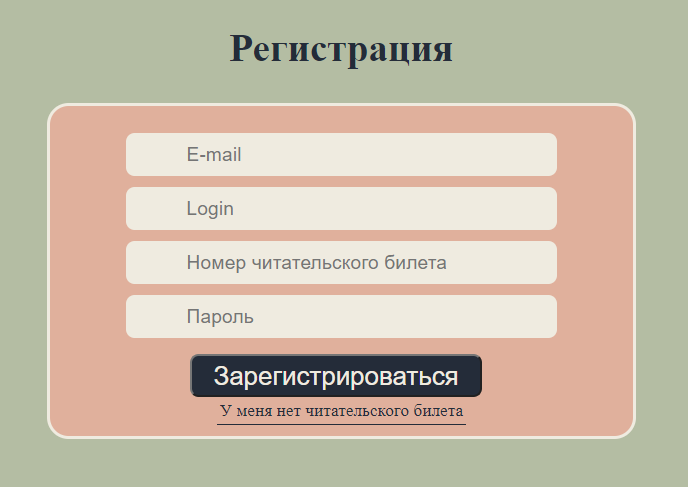


Рисунок 3.4 – Форма регистрации

После заполнения необходимых полей и успешной авторизации, пользователю становится доступен весь функционал системы. В верхней панели появляется возможности поиска по нескольким категориям. Обновлённый вид панели представлен на рисунке 3.6.

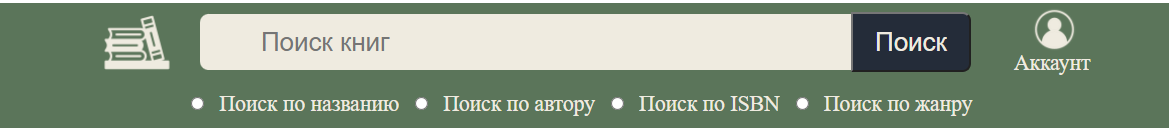


Рисунок 3.6 – Верхняя панель после авторизации

Воспользовавшись поиском, пользователь получит на экране все найденные в базе данных книги. Результат поиска книг по жанру «биография» представлен на рисунке 3.7.

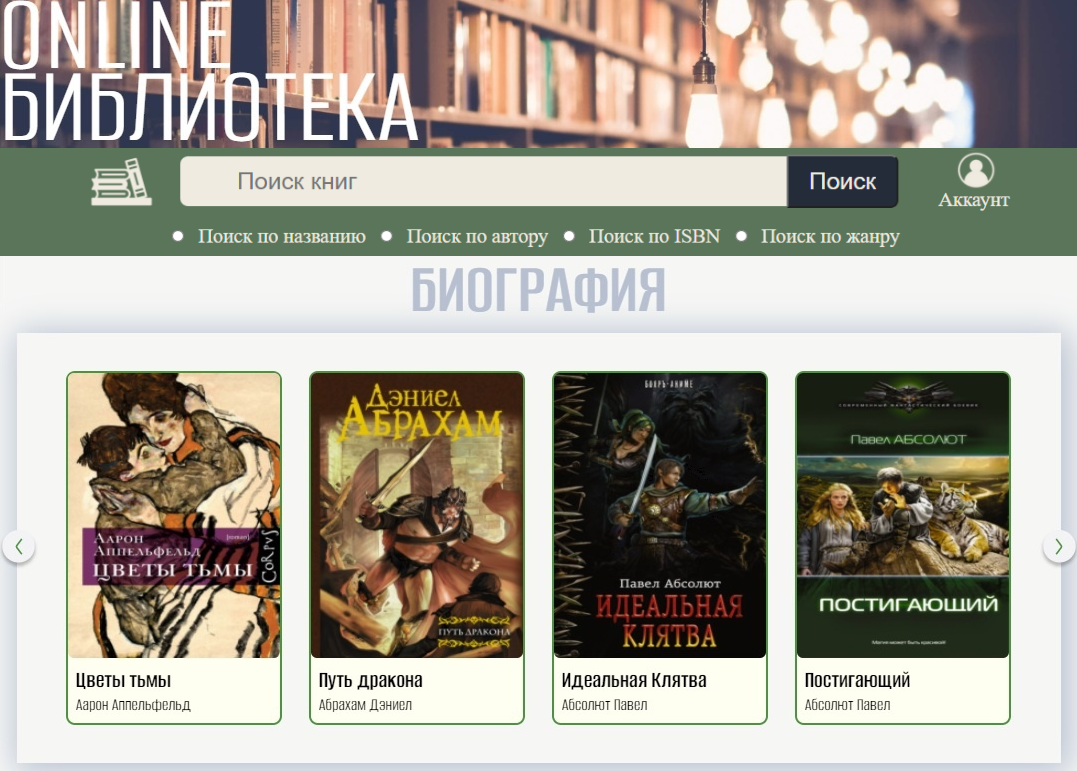


Рисунок 3.7 – Результат работы формы поиска

При нажатии кнопки «Аккаунт» в верхней панели сайта пользователь переходит в свой личный кабинет. В личном кабинете ему доступны несколько вариантов использования: «забронировать книгу», «посмотреть историю книговыдачи», «предложить добавить книгу».

При нажатии кнопки «Забронировать книгу», отображается часть сайта с ранее найденной пользователем книгой. Внешний вид сайта представлен на рисунке 3.8.

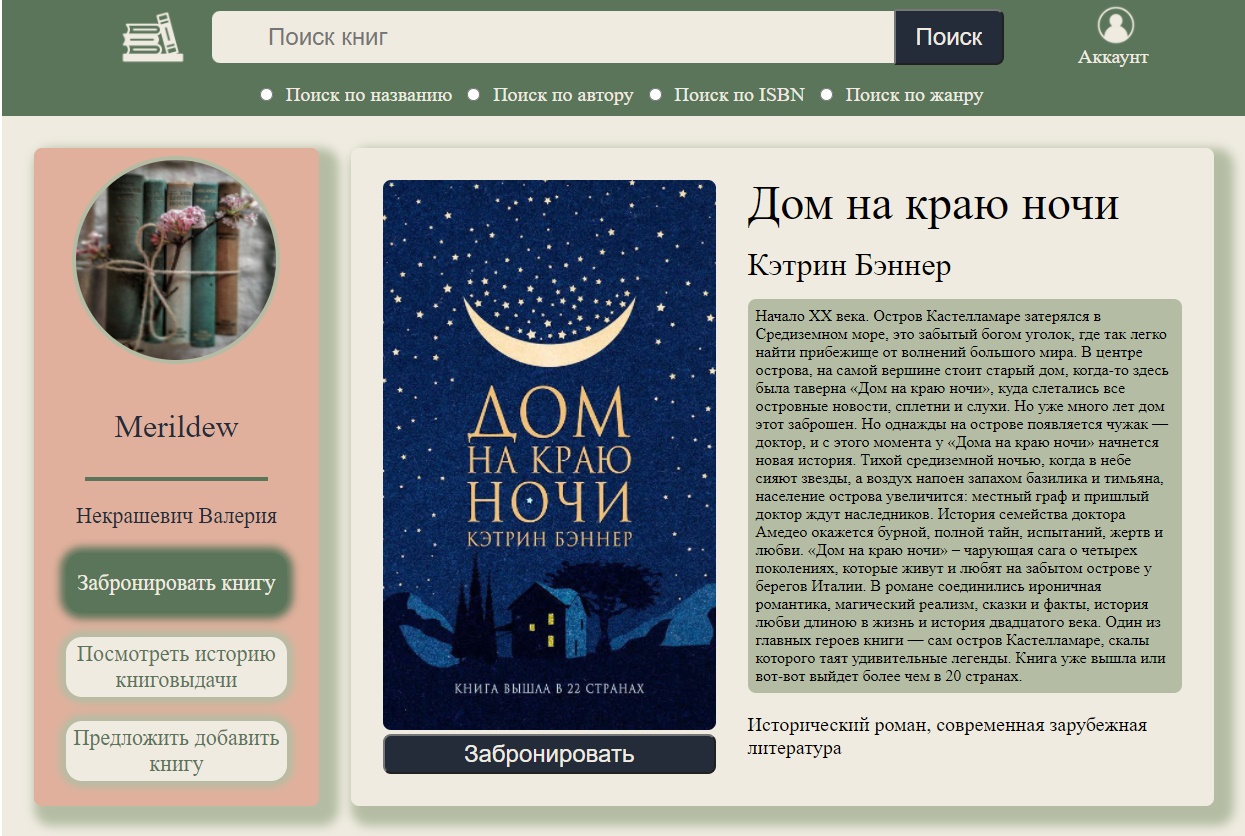


Рисунок 3.8. – Панель бронирования книг

При нажатии кнопки «Посмотреть историю книговыдачи», на экран выводится последняя история книговыдачи пользователя. Обновлённый внешний вид сайта представлен на рисунке 3.9.

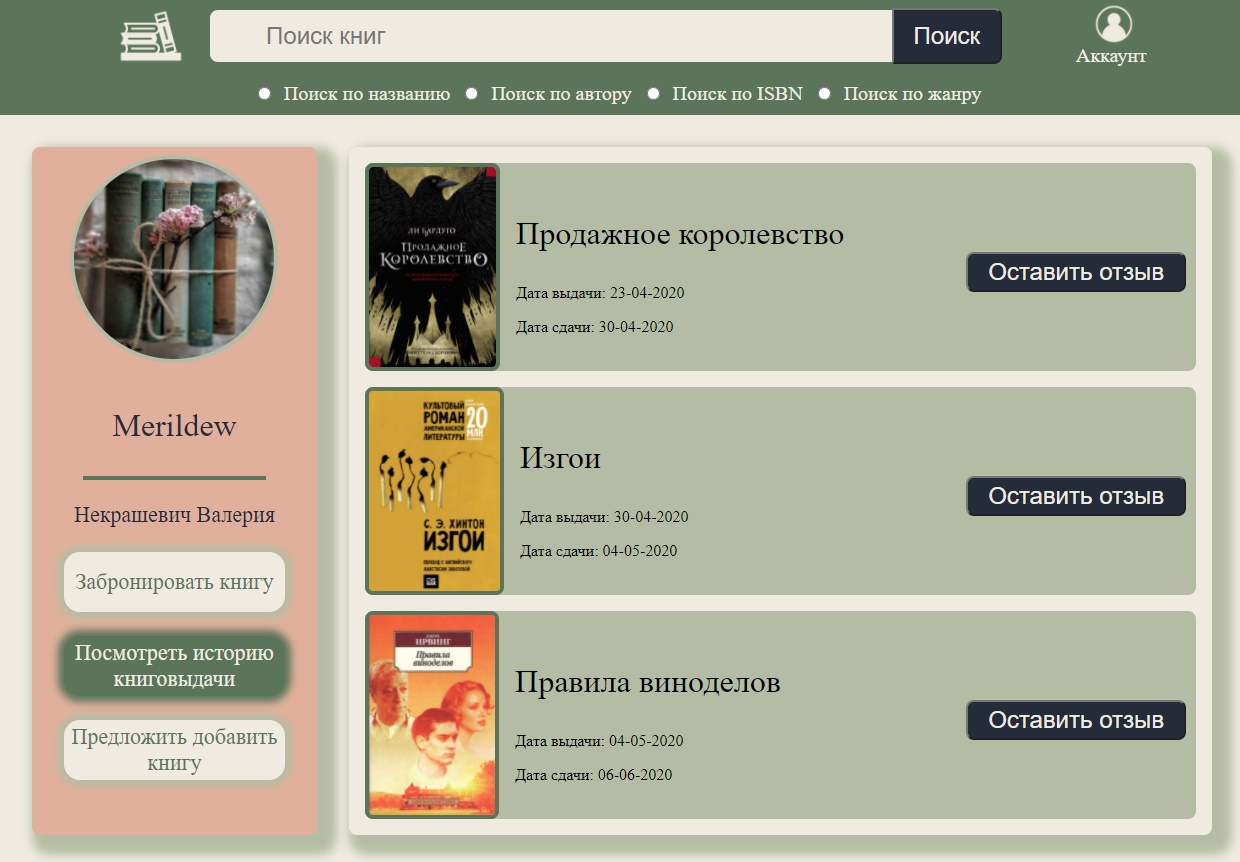


Рисунок 3.9 – Отображение истории книговыдачи

При нажатии кнопки «Добавить книгу» на экран выводится форма добавления книги. Внешний вид формы представлен на рисунке 3.10.

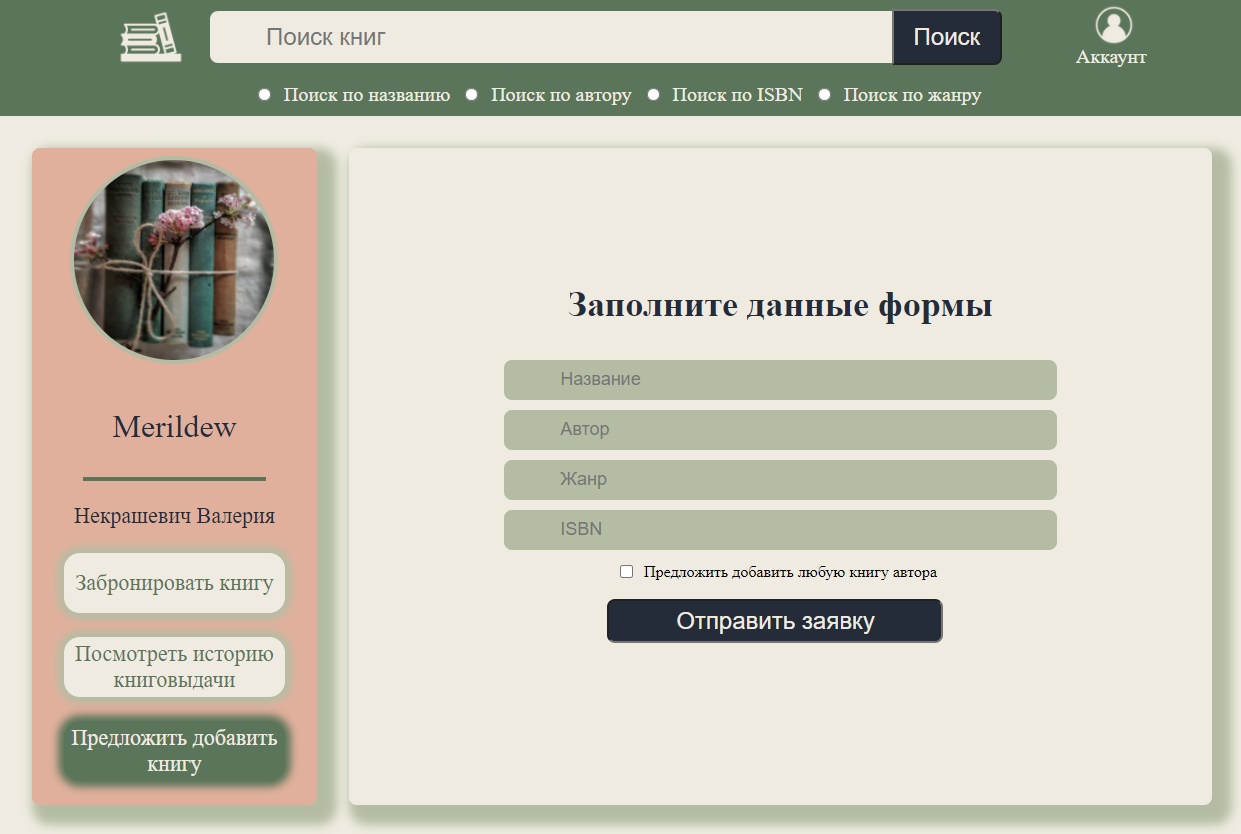


Рисунок 3.10 – Форма для добавления новой книги

1. Технико-экономическое обоснование разработки и использования автоматизированной информационной системы библиотеки
   1. Описание разрабатываемой автоматизированной системы

Задача любой библиотеки состоит в предоставлении литературы читателям для получения новых знаний, расширения кругозора и развития грамотности. Достичь эффективности процесса предоставления литературы можно за счёт внедрения автоматизированных технологий.

Разработанная в дипломном проекте автоматизированная система предназначена для повышения эффективности работы обсуживающего персонала библиотеки и более быстрого обслуживания читателей. Система позволяет читателям забронировать желаемое издание и сократить время поиска литературы.

Функционал автоматизированной информационной системы библиотеки включает в себя следующие возможности:

* получение электронного читательского билета библиотеки, регистрация и авторизация в автоматизированной системе;
* многоаспектный поиск данных по различным признакам (поиск по названию, поиск по автору, поиск по *ISBN*, поиск по жанру) и их сочетаниям;
* отслеживание личной истории книговыдачи;
* добавление рейтинга и рецензии для прочитанных ранее книг;
* просмотр рекомендаций;
* бронирование желаемых книг для последующего получения в библиотеке.

Внедрение автоматизированной системы позволяет сократить затраты библиотеки на обслуживание пользователей библиотечного фонда, освободить сотрудников от рутинного труда и сократить штат библиотеки. Показатель экономической эффективности библиотеки возрастает за счёт оптимизации расходов на персонал и снижения неоправданных расходов на оплату труда.

Разработанная автоматизированная информационная система может использоваться библиотеками любого типа. Целесообразно использование системы в библиотеках с большим библиотечным фондом, которым необходимо привлечение новых читателей.

* 1. Расчёт затрат на разработку автоматизированной системы
     1. **Затраты на основную заработную плату команды разработчиков**

Для реализации проекта необходима команда разработчиков со следующим составом:

* *Frontend*-разработчик отвечает за создание клиентской части пользовательского интерфейса;
* *backend*-разработчик занимается программно-административной частью,  внутренним содержанием системы, серверными технологиями — базой данных, архитектурой, программной логикой.;
* *UX*/*UI*-дизайнер разрабатывает интерфейс приложения;
* копирайтер создаёт текстовый контент;
* *data scientist* отвечает за работу с данными изанимается разработкой системы рекомендаций;
* *project manager* отвечает за управление проектом и связь с заказчиком.

Расчёт затрат на основную заработную плату разработчиков осуществляется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.1) |

где – коэффициент премий и иных стимулирующих выплат; *n* – количество исполнителей, занятых разработкой программного средства; – часовая заработная плата исполнителя *i*-й категории, руб.; – трудоемкость работ, выполняемых исполнителем *i*-й категории, ч.

Количество рабочих часов в месяце составляет 168 часов. Срок разработки данного программного продукта – один месяц. Коэффициент премий и иных стимулирующих выплат равен единице.

Расчёт зарплат на основную заработную плату осуществляется в табличной форме (см. таблицу 4.1).

Таблица 4.1 – Расчёт основной заработной платы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, руб. | Часовой оклад, руб. | | Трудоёмкость работ, ч. | Итого, руб. |
| *Frontend*-разработчик | 2025 | 12 | | 168 | 2025 |
| *Backend*-разработчик | 1680 | 10 | | 120 | 1200 |
| *UX*/*UI*-дизайнер | 1512 | | 9 | 35 | 315 |
| Копирайтер | 1344 | | 8 | 20 | 160 |
| *Data Scientist* | 2352 | | 14 | 80 | 1120 |
| *Project Manager* | 3000 | | 17,8 | 168 | 3000 |
| Итого | | | | | 7820 |
| Премия и иные стимулирующие выплаты | | | | | 0 |
| Всего затрат на основную заработную плату разработчиков | | | | | 7820 |

* + 1. **Затраты на дополнительную заработную плату команды разработчиков**

Расчёт затрат на дополнительную заработную плату разработчиков осуществляется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.2) |
|  |  |  |

где – норматив дополнительной заработной платы ( = 15%).

Дополнительная заработная плата составит:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

* + 1. **Отчисления на социальные нужды**

Отчисления на социальные нужды определяются в соответствии с действующими законодательными актами по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.3) |

где ‒ норматив отчислений в ФСЗН и Белгосстрах ( = 34,6 %).

|  |  |
| --- | --- |
|  |  |

* + 1. **Прочие расходы**

Прочие затраты включаются в себестоимость разработки автоматизированной системы в процентах от затрат на основную заработную плату разработчикам по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.4) |

где ‒ норматив прочих расходов ( = 30 %).

* + 1. **Общая сумма затрат**

Общая сумма затрат на разработку программного продукта находится по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.5) |

* + 1. **Плановая прибыль**

Плановая прибыль, которая закладывается в цену программного средства рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.6) |

где – рентабельность затрат на разработку программного средства ( = 30%).

* + 1. **Отпускная цена**

Отпускная цена рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.7) |

Формирование цены на основе затрат осуществляется в табличной форме (см. таблицу 4.2).

Таблица 4.2 – Формирование цены программного средства на основе затрат

|  |  |  |
| --- | --- | --- |
| Наименование статьи затрат | Расчет по формуле | Значение, руб. |
| 1. Основная заработная плата разработчиков | Формула (4.1) | 7280 |
| 2. Дополнительная заработная плата разработчиков | Формула (4.2) | 1173 |
| 3. Отчисления на социальные нужды | Формула (4.3) | 2924,74 |
| 4. Прочие расходы | Формула (4.4) | 2184 |
| 5. Общая сумма затрат на разработку | Формула (4.5) |  |
| 6. Плановая прибыль, включаемая в цену программного средства | Формула (4.6) |  |
| 7. Отпускная цена программного средства | Формула (4.7) |  |

Таким образом, при помощи методики формирования цен на основе затрат, определена цена разрабатываемой автоматизированной информационной системы библиотеки, равная рублей. Данная цена получается при условии, что над системой работает команда из шести человек (см. таблицу 4.1).

* 1. Расчёт экономического эффекта
     1. **Расчёт экономического эффекта для организации-разработчика**

Полученная автоматизированная система реализовывается организацией разработчиком по отпускной цене, сформированной на основе затрат на разработку (см. таблицу 4.2).

Экономический эффект, полученный организацией-разработчиком, в виде прироста чистой прибыли от его разработки, определяется по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.8) |

где ‒ прибыль, включаемая в цену программного средства, руб.; ‒ставка налога на прибыль, в соответствии с действующим законодательством равная 18%.

Таким образом, организация-разработчик получает чистую прибыль в размере рублей.

* + 1. **Расчёт экономического эффекта для организации-заказчика**

Экономическим эффектом в результате использования автоматизированной информационной системы библиотеки является прирост чистой прибыли, полученный за счёт экономии затрат на заработную плату с начислениями на заработную плату служащих в связи с сокращением их численности.

Экономия на заработной плате и начислениях на заработную плату в результате сокращения численности работников рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.9) |

где *n* – категории работников, высвобождаемых в результате внедрения программного средства; ‒ численность работников *i*-й категории, высвобожденных после внедрения программного средства, чел.; – годовая заработная плата высвобожденных работников *i*-й категории после внедрения программного средства, руб.

Расчёт экономии на заработной плате и начислениях на заработную плату в результате сокращения численности работников осуществляется в табличной форме (см. таблицу 4.3).

Таблица 4.3 – Расчёт экономии на заработной плате

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Категория работников | Численность высвобожденных работников, чел. | Годовая заработная плата, руб. | , % | , % | Итого, руб. |
| Ведущий библиотекарь | 0 | 9600 | 15 | 34.6 | 0 |
| Главный библиотекарь | 0 | 7800 | 0 |
| Библиотекарь I квалификационной категории | 1 | 6600 | 10216,14 |
| Библиотекарь II квалификационной категории | 3 | 5640 | 26190,5 |
| Итого | | | | | 36406,608 |

Экономическим эффектом при использовании программного средства является прирост чистой прибыли, полученной за счет экономии на текущих затратах предприятия, который рассчитывается по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.10) |

где – прирост текущих затрат связанных с использованием автоматизированной системы.

включают в себя затраты на интернет-трафик и составляют 480 рублей (40 рублей в месяц).

Таким образом, использование автоматизированной информационной системы библиотеки позволяет сократить четверых сотрудников и получить чистую прибыль в размере рублей.

* 1. Расчёт экономической эффективности
     1. **Расчёт экономической эффективности для организации-разработчика**

Для организации-разработчика программного средства оценка экономической эффективности разработки осуществляется с помощью расчета простой нормы прибыли по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.11) |

Норма прибыли дает возможность оценить доходность бизнеса, которая имеет место после осуществления всех связанных с ним расходов и вычета причитающихся к уплате налогов. Полученный показатель экономической эффективности считается приемлемым, а значит, организации-разработчику выгодно заняться проектом разработки автоматизированной информационной системы библиотеки.

* + 1. **Расчёт экономической эффективности для организации-заказчика**

Инвестиции в автоматизированную информационную систему библиотеки окупаются менее, чем за год, поскольку сумма инвестиций меньше суммы годового экономического эффекта. Таким образом, оценка экономической эффективности инвестиций в разработку системы осуществляется с помощью расчета рентабельности инвестиций (*Return on Investment, ROI*) по формуле:

|  |  |  |
| --- | --- | --- |
|  |  | (4.12) |

где ‒ прирост чистой прибыли, полученной от использования разработанной автоматизированной системы по индивидуальному заказу, руб.; ‒ цена автоматизированной системы, рассчитанная на основе затрат, руб.; ‒ ставка налога на добавленную стоимость в соответствии с законодательством (), %.

Рентабельность инвестиций равна:

Полученный показать экономической эффективности является приемлемым и отражает высокую результативность заказанной работы. Таким образом, на основании полученных данных можно сделать вывод, что разработка системы библиотеки по индивидуальному заказу целесообразна и имеет коммерческий потенциал.

Помимо экономического эффекта, использование автоматизированной информационной системы библиотеки также сопровождается социальным эффектом. Удалённый доступ к библиотеке способствует привлечению новых читателей, в результате которого происходит повышение грамотности населения. За счёт сокращения рутинной работы библиотекарей и упрощения взаимодействия с каталогом книг происходит повышение производительности труда и квалификации работников библиотеки.

Заключение

На основе анализа объекта автоматизации, обоснована актуальность темы курсовой работы, определены цели и основные задачи.

В процессе проектирования подсистемы книговыдачи библиотечной АС использовались структурная и объектно-ориентированная методологии. Для описания системы в виде модели «*As to be*» использовались диаграммы *IDEF*0, *IDEF*3, *DFD* – структурной методологии, и *UML*-диаграммы последовательности, классов и вариантов использования – объектно-ориентированной.

Логика функционирования системы и её элементов графически представлена в виде диаграммы активностей и диаграммы состояний.

Для корректного информационного обеспечения системы построена база данных, содержащая всю необходимую для функционирования сайта информацию и соответствующая требуемым нормальным формам архитектуры баз данных.

Рассмотрены все основные виды обеспечения системы: техническое, программное и организационное. Определен набор технологий для реализации полного функционала. Спроектирован и разработан удобный и понятный пользовательский интерфейс.

В ходе проведения курсовой работы углублены, закреплены и конкретизированы теоретические знания в области автоматизации процессов, приобретены навыки по проектированию.

Список использованных источников

[1] Антиплагиат. Обнаружение зависимостей [Электронный ресурс]. – Электронные данные. – *https://www.antiplagiat.ru/*.

[2] Вендров, А.М. Объектно-ориентированный анализ и проектирование c использованием языка *UML* и *Rational* *Rose /* А.М. Вендров– М. : Издательский отдел факультета ВМиК МГУ, 2000. – 54 с.

[3] Хенли, Дж. Автоматизированная библиотека и информационные системы / Дж. Хенли – М. : Мир, 1974. – 122 с.

[4] Алёшин, Л.И. Автоматизация в библиотеке / Л.И. Алёшин – М. : Профиздат, 2001 – 144 с.

[5] Колесов, Ю.Б. Объектно-ориентированное моделирование систем / Ю.Б. Колесов – СПб. : Издательство СПбГПУ, 2004. – 240 с.

[6] Дейт, К. Дж. Введение в системы баз данных / К. Дж. Дейт – М. : Вильямс, 2017. – 1328 с.

[7] *Habr SQL*. Нормализация отношений. Шесть нормальных форм [Электронный ресурс]. – Электронные данные. – Режим доступа: *https*://*habr*.*com*/*post*/254773/, свободный.

[8] Нейл, Т. Проектирование веб-интерфейсов / Т. Нейл, Б. Скотт – М. : *O’Reilly*, 2014. – 352с.

[9] Макконнелл, С. Совершенный код. Мастер класс / С. Макконнелл – СПб. : БХВ-Петербург, 2017. – 896c.

[10] Доманов, А.Т. СТП 01-2017. Стандарт предприятия. Дипломные проекты(работы). Общие требования. / А. Т. Доманов, Н. И. Сорока – Минск : БГУИР, 2017. – 170 с.

Приложение А  
(справочное)  
Листинг функции расчёта коэффициента корреляции

*const pcorr = (x, y) => {*

*let sumX = 0,*

*sumY = 0,*

*sumXY = 0,*

*sumX2 = 0,*

*sumY2 = 0;*

*const minLength = x.length = y.length = Math.min(x.length, y.length),*

*reduce = (xi, idx) => {*

*const yi = y[idx];*

*sumX += xi;*

*sumY += yi;*

*sumXY += xi \* yi;*

*sumX2 += xi \* xi;*

*sumY2 += yi \* yi;*

*}*

*x.forEach(reduce);*

*return (minLength \* sumXY - sumX \* sumY) / Math.sqrt((minLength \* sumX2 - sumX \* sumX) \* (minLength \* sumY2 - sumY \* sumY));*

*};*

*pcorr([arr], [arr]);*

Приложение Б  
(справочное)  
Листинг настройки *jwt* безопасности

*const jwt = require("jsonwebtoken");*

*require("dotenv").config();*

*const generateAccessToken = (payload) => {*

*return jwt.sign(payload, process.env.JWT\_KEY, { expiresIn: "15s" });*

*};*

*const generateRefreshToken = (payload) => {*

*return jwt.sign(payload, process.env.JWT\_REFRESH\_KEY, {*

*expiresIn: "7d",*

*});*

*};*

*const validateToken = (req, res, next) => {*

*let token = req.headers["authorization"];*

*token = token.split(" ")[1];*

*if (!token || token === "null") {*

*return res.status(403).json({ error: { message: "User not Authenticated!" } });*

*}*

*try {*

*const validToken = jwt.verify(token, process.env.JWT\_KEY);*

*if (validToken) {*

*return next(); }*

*} catch (err) {*

*return res.status(401).json({ error: err });*

*}*

*};*

*const validateRefreshToken = async (req, res) => {*

*const refreshToken = req.body.refreshToken;*

*if (!refreshToken)*

*return res.status(403).json({ error: { message: "Refresh token is empty" } });*

*const decoded = jwt.decode(refreshToken);*

*const query = await Users.findOne({*

*where: { login: decoded.login }, });*

*const user = query.dataValues;*

*try {*

*const validToken = jwt.verify(refreshToken, process.env.JWT\_REFRESH\_KEY);*

*if (validToken) {*

*const accessToken = generateAccessToken(user);*

*res.json(accessToken); }*

*} catch (err) {*

*return res.status(401).json({ error: err });*

*}};*

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Обозначение* | | | | | *Наименование* | *Дополнительные сведения* | | | | | |
|  | | | | | *Текстовые документы* |  | | | | | |
|  | | | | |  |  | | | | | |
| *БГУИР ДП 1-53 01 02 06 105 ПЗ* | | | | | *Пояснительная записка* | *63 с.* | | | | | |
|  | | | | | *Отзыв руководителя* |  | | | | | |
|  | | | | | *Рецензия* |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | | *Графические документы* |  | | | | | |
|  | | | | |  |  | | | | | |
| *ГУИР.000000.001 ПД* | | | | | *Схема базы данных* | *Формат А1* | | | | | |
|  | | | | |  |  | | | | | |
| *ГУИР.000000.002 ПД* | | | | | *Диаграмма вариантов* |  | | | | | |
|  | | | | | *использования* | *Формат А1* | | | | | |
|  | | | | |  |  | | | | | |
| *ГУИР.000000.003 ПЛ* | | | | | *Блок-схема рекомендательной* |  | | | | | |
|  | | | | | *системы* | *Формат А1* | | | | | |
|  | | | | |  |  | | | | | |
| *ГУИР.000000.004 ПЛ* | | | | | *Диаграмма декомпозиции* |  | | | | | |
|  | | | | | *первого уровня* | *Формат А1* | | | | | |
|  | | | | |  |  | | | | | |
| *ГУИР.000000.005 ПЛ* | | | | | *Диаграмма* |  | | | | | |
|  | | | | | *последовательности сценария* |  | | | | | |
|  | | | | | *добавления отзыва* | *Формат А1* | | | | | |
|  | | | | |  |  | | | | | |
| *ГУИР.000000.006 ПЛ* | | | | | *Диаграмма* |  | | | | | |
|  | | | | | *последовательности сценария* |  | | | | | |
|  | | | | | *бронирования книг* | *Формат А1* | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  | | | | |  |  | | | | | |
|  |  |  |  |  | *БГУИР ДП 1-53 01 02 06 105 Д1* | | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| *Изм.* | *Л.* | *№ докум.* | *Подп.* | *Дата* | *Автоматизированная информационная система библиотеки*  *Ведомость дипломного  проекта* | | *Лит* | | | *Лист* | *Листов* |
| *Разраб.* | | *Некрашевич* |  |  |  | *Т* |  | *63* | *63* |
| *Провер.* | | *Павловская* |  |  | *Кафедра ИТАС*  *гр. 820604* | | | | |
| *Т.контр.* | | *Павловская* |  |  |
| *Н.контр.* | | *Боброва* |  |  |
| *Утв.* | | *Навроцкий* |  |  |