Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет инженерно-экономический

Кафедра экономической информатики

Дисциплина «Программирование сетевых приложений»

|  |  |
| --- | --- |
|  | «К ЗАЩИТЕ ДОПУСТИТЬ» |
|  | Ассистент  \_\_\_\_\_\_\_\_\_\_\_\_\_ Ю. В. Сильванович |
|  | \_\_\_.\_\_\_\_.2023 |

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

к курсовому проекту

на тему:

**«ОНЛАЙН-ПЛАТФОРМА ДЛЯ ПОДБОРА И ПОКУПКИ КНИГ»**

БГУИР КП 1-40 01 01-02 023 ПЗ

|  |  |
| --- | --- |
|  | Выполнил студент группы 072304  Козаченко Станислав Андреевич  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |
|  | Курсовой проект представлен на проверку \_\_.\_\_ 2023  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись студента) |

Минск 2023

**СОДЕРЖАНИЕ**

[Перечень условных обозначений, символов и терминов 6](#_Toc128910340)

[Введение 8](#_Toc128910341)

[1 Анализ и моделирование предметной области программного средства 9](#_Toc128910342)

[1.1 Описание предметной области 9](#_Toc128910343)

[1.2 Разработка функциональной модели предметной области 13](#_Toc128910344)

[1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований 18](#_Toc128910345)

[1.4 Разработка информационной модели предметной области 20](#_Toc128910346)

[1.5 Модели представления программного средства и их описание 23](#_Toc128910347)

[2 Проектирование и конструирование программного средства 27](#_Toc128910348)

[2.1 Постановка задачи 27](#_Toc128910349)

[2.2 Архитектурные решения 27](#_Toc128910350)

[2.3 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства 27](#_Toc128910351)

[2.4 Проектирование пользовательского интерфейса 27](#_Toc128910352)

[2.5 Обоснование выбора компонентов и технологий для реализации программного средства 27](#_Toc128910353)

[2.6 Методы и средства, используемые для обеспечения безопасности данных 27](#_Toc128910354)

[3 Тестирование и проверка работоспособности программного средства 28](#_Toc128910355)

[4 Руководство по развёртыванию и использованию программного средства 29](#_Toc128910356)

[4.1 Руководство по развёртыванию 29](#_Toc128910357)

[4.2 Руководство пользователя 29](#_Toc128910358)

[Заключение 30](#_Toc128910359)

[Список использованных источников 31](#_Toc128910360)

[Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат» 32](#_Toc128910361)

[Приложение Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику 33](#_Toc128910362)

[Приложение В (обязательное) Листинг скрипта генерации базы данных 34](#_Toc128910363)

[Ведомость документов курсового проекта 35](#_Toc128910364)

# ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ, СИМВОЛОВ И ТЕРМИНОВ

|  |  |
| --- | --- |
| БД (база данных) | – представленная в объективной форме совокупность самостоятельных материалов, систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины |
| Информационно-коммуникационные технологии | – процессы, методы поиска, сбора, хранения, обработки, предоставления, распространения информации и способы осуществления таких процессов и методов |
| Информационная система | – система, предназначенная для хранения, поиска и обработки информации, и соответствующие организационные ресурсы (человеческие, технические, финансовые), которые обеспечивают и распространяют информацию |
| Нормальная форма | – свойство отношения в реляционной модели данных, характеризующее его с точки зрения избыточности, потенциально приводящей к логически ошибочным результатам выборки или изменения данных |
| Среда выполнения | – вычислительное окружение, необходимое для выполнения компьютерной программы и доступное во время выполнения компьютерной программы |
| СУБД (система управления базами данных) | – совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных |
| ТЗ (техническое задание) | – документ, содержащий требования заказчика к объекту разработки, определяющий порядок и условия её проведения |
| *API* (*application programming interface*) | – описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой |
| *IDE* (*Integrated development environment*) | – комплекс программных средств, используемый программистами для разработки программного обеспечения |
| *IDEF* | – методология функционального моделирования (англ. *function modeling*) и графическая нотация, предназначенная для формализации и описания бизнес-процессов |
| *Java* | – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией *Sun* *Microsystems* |
| *Microsoft SQL Server* | – система управления реляционными базами данных (СУБД), разработанная корпорацией *Microsoft* |
| *SQL* (*structured query language*) | – язык структурированных запросов, декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных |
| *MySQL Server* | – свободная реляционная система управления базами данных |
| *UML* (*Unified Modeling Language*) | – язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования бизнес-процессов, системного проектирования и отображения организационных структур |
| IntelliJ IDEA | – интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains. |
|  |  |
|  |  |
|  |  |

# ВВЕДЕНИЕ

В современном мире существует большая конкуренция между различными направлениями в сфере торговли. Для большинства компаний продажа товаров и услуг через интернет является основным способом найти новых поставщиков и клиентов. Также наличие соответствующего онлайн-сервиса является конкурентным преимуществом. Это не обошло стороной и книжные магазины.

Кто-то может сказать, что в век информационных технологий бумажные книги потеряли свое положение на торговом рынке. Однако это утверждение будет не верно. Так как некоторые книги нельзя скачать с интернет ресурсов бесплатно, так же кто-то все еще предпочитает бумажный формат книги, а также не отменены бумажные книги в школьных заведениях. Из этого следует, что бумажные книги все еще популярны.

Все магазины находятся в борьбе за покупателя, пытаясь увеличить свой ассортимент, предоставляя покупателю огромное количество различных товаров, они забывают о том, сколько клиент готов потратить своего драгоценного времени выбирая из представленного ассортимента то что его действительно интересует.

Зачастую приходя в книжный магазин, покупатель теряется в предоставляемом ему выборе товара. Даже несмотря на то, что магазины стараются разложить все товары согласно каким-либо критериям, или книги по жанрам, не всегда взгляд клиента совпадает с порядком сортировки магазина. В таком случае покупатель вынужден бродить по магазину в поисках нужной ему книги. Не справляясь самостоятельно, он вынужден обратиться за помощью к сотрудникам магазина, и в этом случае покупатель сталкивается с рядом других проблем:

* в магазине не оказывается сотрудников консультантов;
* сидящий на кассе сотрудник отказывается предоставить помощь в связи с тем, что это не его обязанности;
* кассир занят обслуживанием других покупателей.

В таких случаях клиенту приходится тратить свое драгоценное время на ожидания помощи, либо если не может себе этого позволить, он вынужден покинуть магазин и остаться без нужного ему товара, либо уйти к конкурентам, в надежде что там он найдет нужный ему товар. В любом случае магазин теряет потенциального клиента, а значит и свой доход.

Целью данного курсового проекта является повышение эффективности обслуживания клиентов книжного магазина за счет разработки онлайн-сервиса, позволяющего выполнять все функции, интересующие клиента.

Для решения поставленной цели в данном курсовом проекте решаются следующие задачи:

* рассмотреть предметную область книжного магазина;
* описать постановку задачи на разработку программного продукта;
* выполнить реализацию клиент-серверного приложения с подключением к базе данных;
* реализовать клиентскую часть приложения, с удобным интерфейсом для пользователя;
* разработать и описать руководство пользователя;
* протестировать полученное программное средство и убедиться, что оно корректно реализует свою бизнес-логику.

Таким образом данная информационная система позволит осуществить быстрое и надёжное взаимодействие клиента с услугами книжного магазина.

# Анализ и моделирование предметной области программного средства

## 1.1 Описание предметной области

Современная издательская система сложилась в последнее десятилетие прошлого века. На ее структуру и особенности функционирования оказали большое влияние самые различные внутренние и внешние факторы, важнейшим из которых является изменение социально-политических условий развития общества на основе демократизации различных сфер его жизнедеятельности.

Демократизация сферы массовой информации и книгоиздания проходила на фоне неизбежно связанных с ней экономических процессов, обеспечивших переход на рельсы рыночной экономики, когда издательства, формировавшиеся в этих условиях, сразу становились самостоятельными хозяйствующими субъектами нарождающегося книжного рынка.

Серьезным фактором развития издательского бизнеса стало возрастающее развитие и применение новых информационных технологий. Правовые и экономические аспекты книжного дела получили научно-технологическую основу, что позволило буквально за несколько лет проделать путь, на преодоление которого раньше понадобились бы десятилетия.

Наконец, формированию новой издательской системы в значительной степени способствовала глобализация информационного пространства, встраивание отечественного книгоиздания в мировое издательское сообщество. Это дало возможность изучать богатый международный опыт и более продуктивно использовать его в отечественной практике.

Книжный магазин занимается деятельностью по продаже и закупке книг, а значит, в магазине работают сотрудники, которые выполняют весь процесс, от закупок книг у каких-либо поставщиках, с которыми они заключают договора, и продажей этих книг клиентам книжного магазина.

На рисунке 1.1 представлена организационная структура книжного магазина.

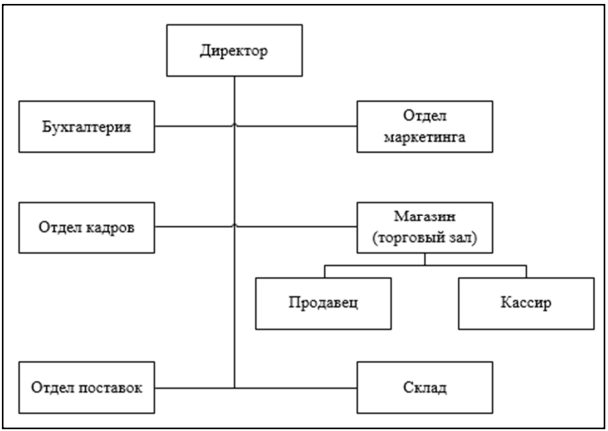


Рисунок 1.1 – Организационная структура книжного магазина

На рисунке 1.1 представлены основные подразделения книжного магазина такие как: директор, бухгалтерия, отдел маркетинга, отдел кадров, магазин (торговый зал), кассир, продавец, отдел поставок, склад. Каждое подразделение выполняет определенные функции.

Директор занимает руководящую должность, обеспечивая выполнение предприятия всех обязанностей, связанных договором с поставщиком. Выполняет вопросы, касающиеся финансовой стороны, контролирует деятельность и выполнение своих должностных обязанностей остальных подразделений, назначает руководителей подразделений. Обеспечивает законную деятельность предприятия.

Бухгалтерия выполняет работу по бухгалтерскому учету, отражает движение денежных средств, составляет платежные поручения, оплачивает аренду, расплачивается с поставщиками и контролирует вовремя были ли получены денежные средства.

Отдел маркетинга занимается изучением рынка потребителя, анализом деятельности конкурентов, изучением интересов и спрос покупателей, мониторинг места на рынке фирмы.

Отдел кадров занимается подбором компетентных рабочих кадров, отвечающие требованиям предприятия, составлением рабочего и отпускного графика, мониторингом рабочих по выполнению своих обязанностей, анализом текучести кадров.

Магазин (торговый зал: продавец, кассир) – продавец расставляет товар согласно установленному порядку, размещает цену, информирует покупателей о проходящих скидках или акциях. Предоставляет помощь клиентам в поиске товара. Кассир, подчитывает окончательную сумму товара для покупателя, производит расчеты с поставщиками.

Отдел поставок занимается анализом рынка поставщиков, участвует в выборе поставщиков, производит закупки, а также контролирует доставку.

Склад осуществляет принятие, размещение и хранение товара. Предоставляет товаров торговый зал продавцу для дальнейшей его продажи потребителю.

## 1.2 Разработка функциональной модели предметной области

Представим деятельность онлайн-платформы по подбору и покупке книг в виде набора взаимосвязанных функциональных блоков с помощью наглядного графического языка IDEF0.

IDEF0 — методология функционального моделирования**.** Описание системы с помощью IDEF0 называется функциональной моделью. Для передачи информации о конкретной системе источником графического языка является сама методология IDEF0.

Методология IDEF0 предписывает построение иерархической системы диаграмм — единичных описаний фрагментов системы. Сначала проводит­ся описание системы в целом и ее взаимодействия с окружающим миром (контекстная диаграмма), после чего проводится функциональная деком­позиция — система разбивается на подсистемы и каждая подсистема опи­сывается отдельно. Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

На рисунке 1.2 представлена контекстная диаграмма системы.

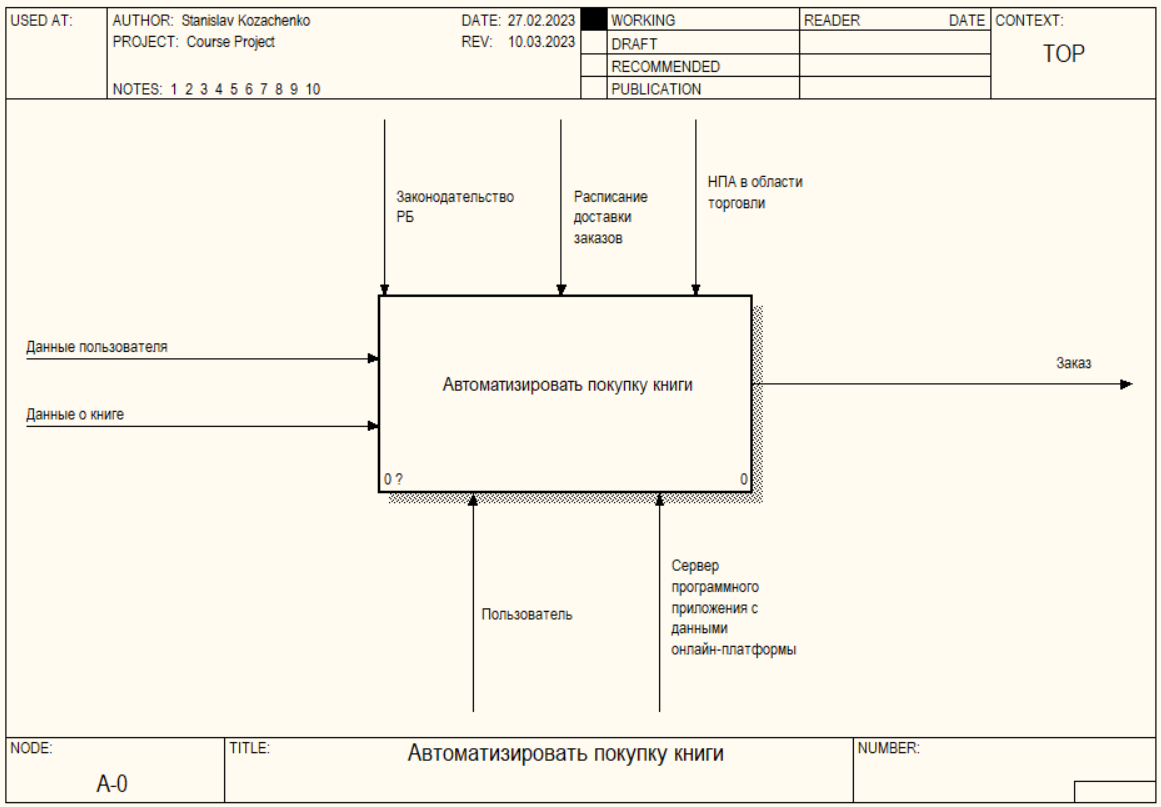


Рисунок 1.2 – Контекстная диаграмма

В качестве входных данных будут использоваться данные пользователя, а также данные о книге.

Суть деятельности онлайн-платформы по подбору и покупке книг заключается в подборе и покупке книг, поэтому выходными данными будет являться информация о запрашиваемой книге. Деятельность выполняется пользователем и сервером программного приложения с данными о книгах.

Проведем декомпозицию контекстной диаграммы, разбив ее на следующие процессы (рисунок 1.3):

* авторизоваться;
* ввести критерии поиска книги;
* найти подходящие книги;
* оформить заказ.

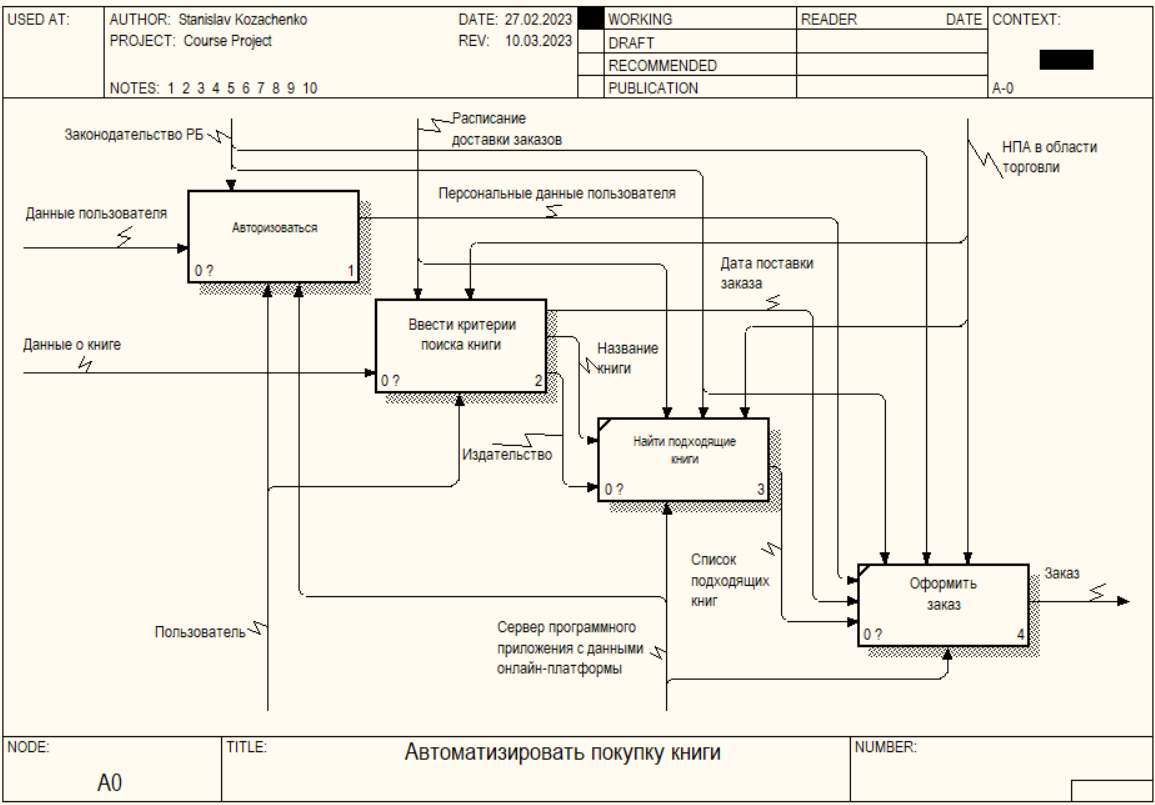


Рисунок 1.3 – Декомпозиция контекстной диаграммы

Клиенты авторизуются в программе. Те, которые успешно прошли авторизацию, могут по необходимым критериям найти нужную книгу. Из списка подходящих книг пользователь выбирает нужную и оформляет заказ.

Процесс авторизации состоит из следующих этапов (рисунок 1.4):

* ввести логин;
* ввести пароль;
* проверить наличие пользователя в системе;
* подтвердить авторизацию.

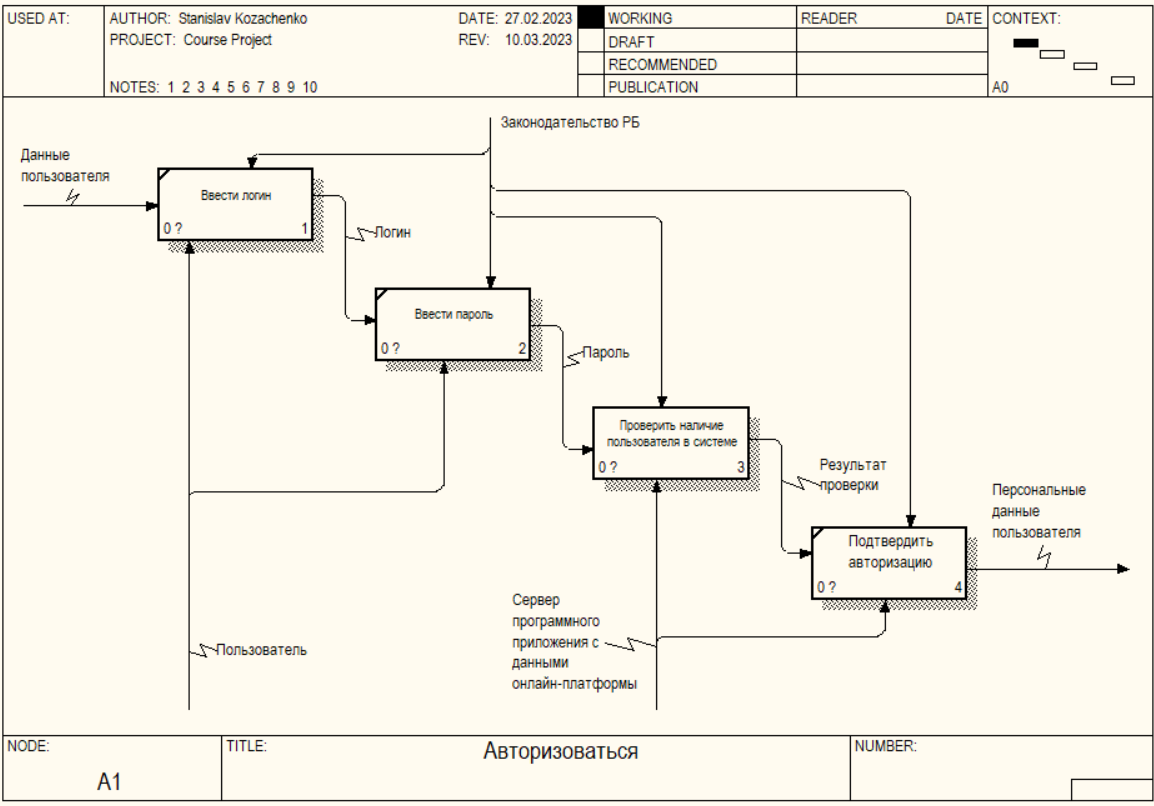


Рисунок 1.4 – Декомпозиция блока «Авторизоваться»

После ввода логина и пароля, сервер программного приложения проверяет наличие пользователя в системе и в случае положительного результата получает персональную информацию пользователя.

Процесс проверки наличия пользователя в системе состоит из следующих этапов (рисунок 1.5):

* сформировать данные пользователя;
* проверить совпадение данных пользователя с данными базы данных онлайн-платформы;
* сформировать результат проверки.

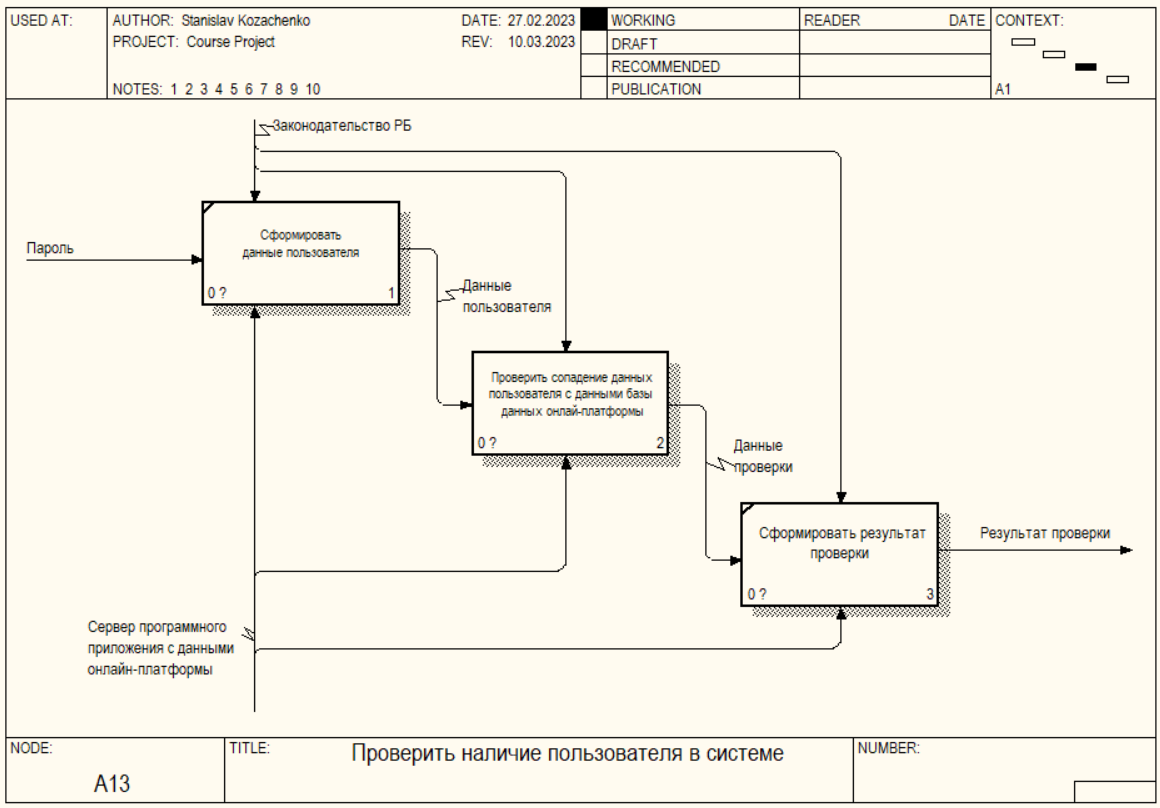


Рисунок 1.5 – Декомпозиция блока «Проверить наличие пользователя в системе»

Процесс проверки совпадения данных пользователя с данными базы данных онлайн-системы состоит из следующих этапов (рисунок 1.6):

* считать данные пользователя;
* считать данные базы данных онлайн-платформы;
* сравнить данные.

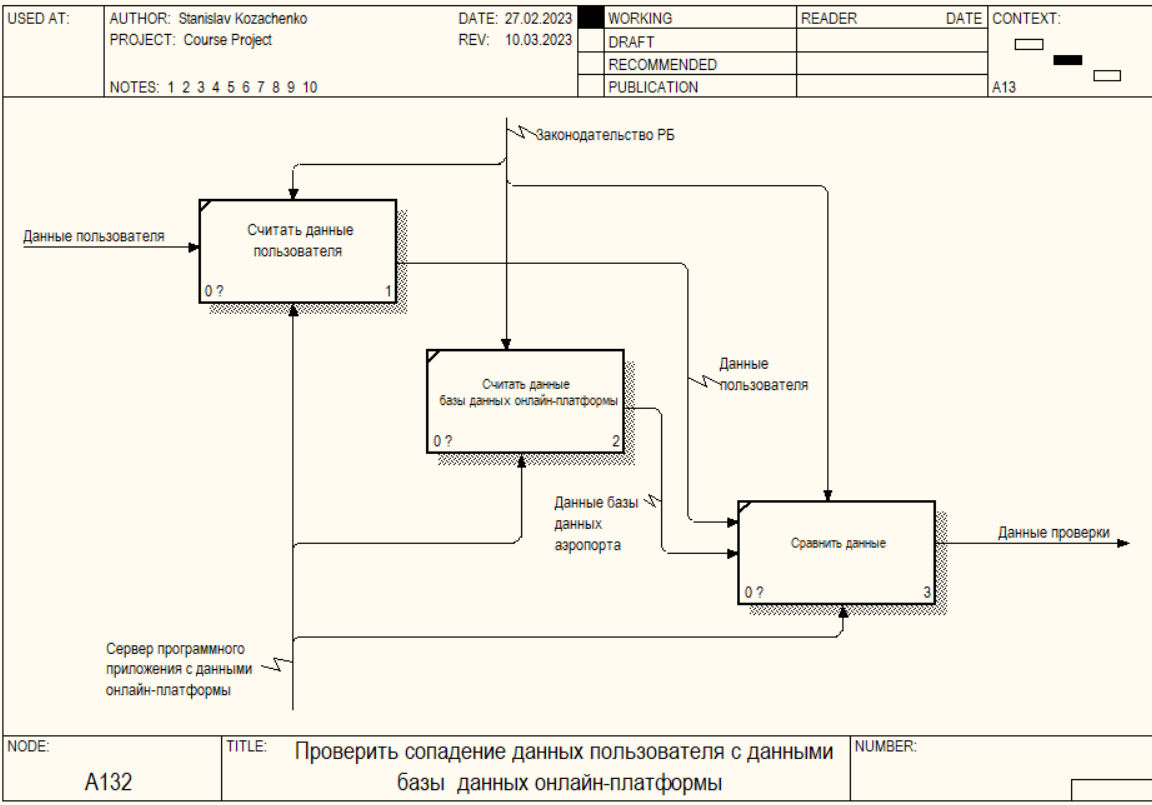


Рисунок 1.6 – Декомпозиция блока «Проверить совпадение данных пользователя с данными базы данных онлайн-платформы»

Процесс поиска полета по введенному критерию состоит из следующих этапов (рисунок 1.7):

* ввести издательство;
* ввести название книги;
* ввести дату поставки.

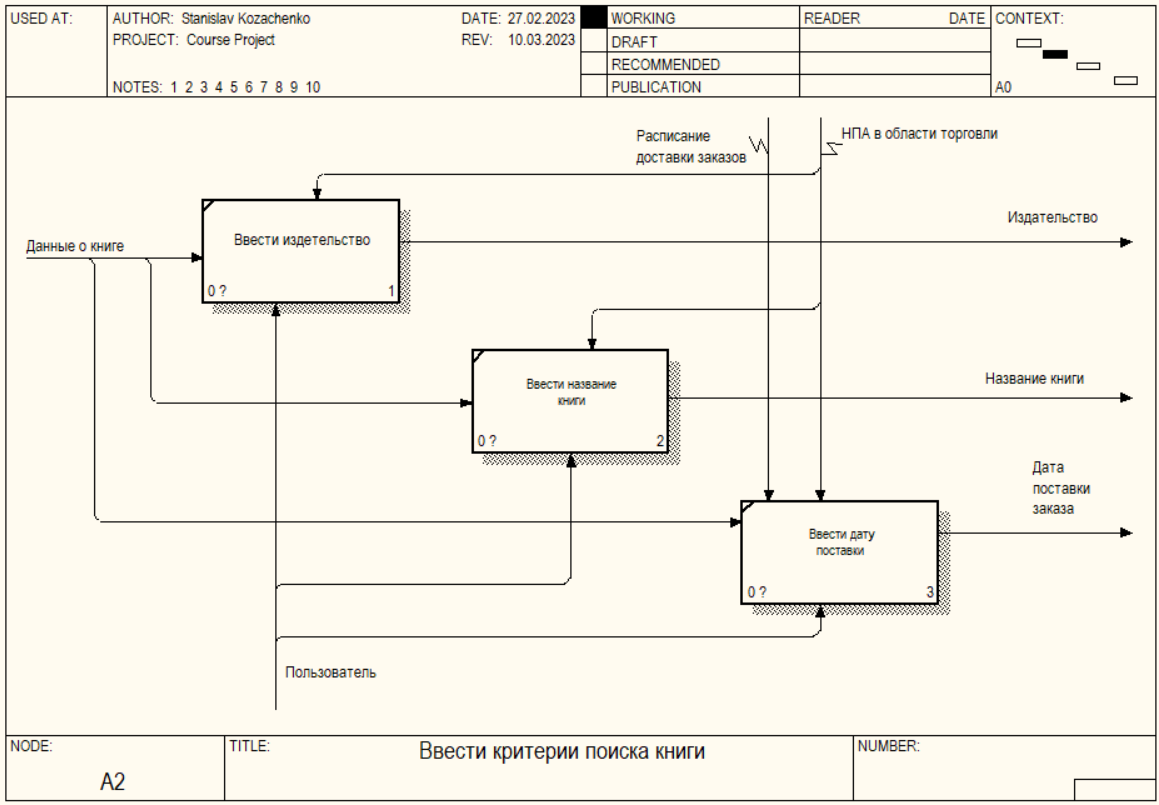


Рисунок 1.7 – Декомпозиция блока «Ввести критерии поиска книги»

Перед проектированием программного средства следует определить, какие требования должны быть предъявлены к нему, так как невыполнение некоторых требований, которые были сформированы еще на ранней стадии, говорит о том, что разработанный сервис не сможет эффективно и с максимальной пользой использоваться, как было задумано изначально.

## 1.3 Анализ требований к разрабатываемому программному средству. Спецификация функциональных требований

Для разработки программного средства необходимо понять принцип работы онлайн-платформы.

Так как у каждого клиента имеются личные данные, необходимо спроектировать базу данных, которая осуществляла бы их хранение в электронном виде. Это позволит облегчить поиск информации для администратора онлайн-платформы, а также позволит клиентам получить доступ к информации в любое время.

Для корректной работы необходимо предусмотреть возможность добавления, удаления и редактирования информации в базе данных. Для хранения информации будет использован *MYSQL* *Server*. Подключение к нему будет осуществляться при авторизации и в зависимости от полученных прав пользователь, будет получать определённый набор возможностей для выполнения своей работы.

Сотрудник магазина (администратор) должен иметь доступ ко всей информации, а также должен иметь возможность проводить любые операции с ней и получать отчёты.

Клиент (пользователь) будет иметь доступ к данным, позволяющим осуществлять операции по подбору и покупке книг.

Диаграмма вариантов использования описывает взаимоотношения и зависимости между группами вариантов использования и действующих лиц, участвующих в процессе.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых 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. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

Диаграмма вариантов использования данной программы представлена на рисунке 1.8.

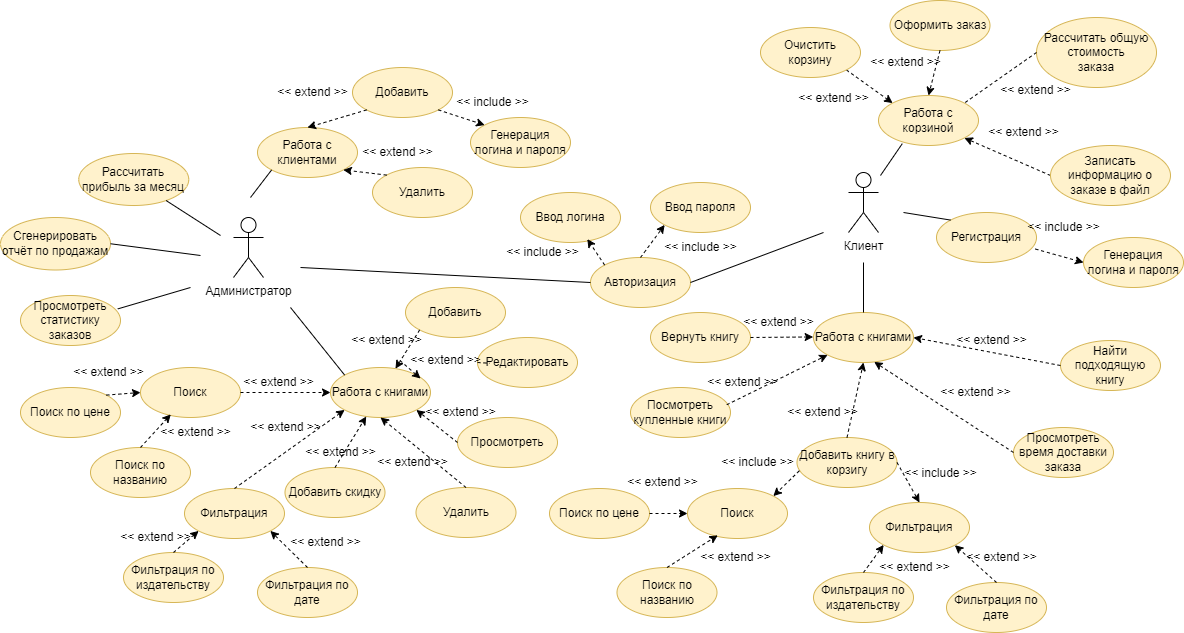


Рисунок 1.8 – Диаграмма вариантов использования

На диаграмме видно, что для работы всем пользователям необходимо войти в систему (ввести логин и пароль), а новым клиентам зарегистрироваться.

Администратор может добавлять, удалять пользователей, осуществлять работу с книгами (добавление, редактирование, поиск, удаление, фильтрация), а также просматривать статистику покупок.

Пользователь может просмотреть и выбрать книги, работать с необходимыми данными, рассчитывать полную стоимость заказа, добавлять, возвращать книги, просматривать купленные книги, а также получать данные о заказе в файл.

## 1.4 Разработка информационной модели предметной области

При проектировании системы было принято решение использовать следующие сущности:

* *Users*;
* *Books*;
* *Transactions*;
* *Authors*;
* *Publishers*;
* *Orders*;
* *OrderItem.*

Графическое отображение информационной модели приведено на диаграмме рисунке 1.9.

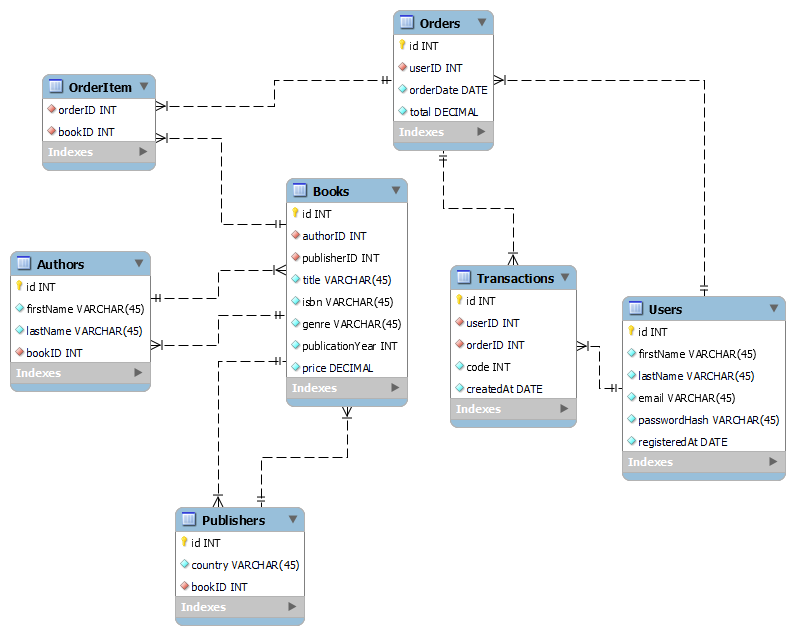


Рисунок 1.9 – Информационная модель системы

Ниже представлено подробное описание всех сущностей, входящих в модель.

Сущность *Users* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер пользователя;
* *firstName* – отвечает за хранение имени пользователя;
* *lastName* – хранит фамилию пользователя;
* *email* – хранит почту пользователя;
* *passwordHash* – хранит хэш пароля;
* *registeredAt* – хранит дату регистрации пользователя.

Сущность *Books* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный книги;
* *authorID* – отвечает за хранение идентификатора автора;
* *publisherID* – отвечает за хранение идентификатора издателя;
* *title* – хранит название книги;
* *isbn* – хранит ISBN книги;
* *genre* – хранит жанр книги;
* *publicationYear* – хранит год издания книги;
* *price* – хранит стоимость книги.

Сущность *Transactions* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер транзакции;
* *userID* – хранит идентификатор пользователя;
* *orderID* – хранит идентификатор заказа;
* *code* – хранит статус-код транзакции;
* *createdAt* – хранит дату создания транзации.

Сущность *Authors* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер автора;
* *firstName* – отвечает за хранение имени автора;
* *lastName* – хранит фамилию автора;
* bookID – хранит идентификатор книги.

Сущность *Publishers* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер издательства;
* *country* – хранит страну издательства;
* bookID – хранит идентификатор книги.

Сущность *Orders* содержит в себе следующие атрибуты:

* *id* – хранит идентификационный номер заказа;
* *userID* – хранит идентификатор пользователя;
* *orderDate* – хранит данные о дате заказа;
* *total* – хранит итоговую сумму заказа.

Сущность *OrderItem* содержит в себе следующие атрибуты:

* *orderID* – отвечает за хранение идентификатора заказа;
* *bookID* – хранит идентификатор книги.

Таким образом, база данных приведена к третей нормальной форме, поскольку у каждой таблицы имеется всего один первичный ключ, а каждое не ключевое поле не транзитивно зависит от первичного ключа, то есть изменив значение в одном столбце не потребуется изменение в другом столбце.

*SQL*-скрипт для генерации базы данных приведен в приложении В.

## 1.5 Модели представления программного средства и их описание

**1.5.1** Диаграмма состояний. Диаграмма состояний представляет собой состояния, соединенные переходами. Переходы могут сработать в результате заданного в качестве условия перехода события — это может быть истечение заданного таймаута, получение диаграммой состояний сообщения, выполнение заданного логического условия и т.д. Срабатывание перехода приводит к переходу управления диаграммы состояний в то состояние, в которое ведет этот переход. Состояния могут быть иерархическими, то есть содержать другие состояния и переходы.

На рисунке 1.10 изображена диаграмма состояний для разрабатываемого программного средства. Данный вид диаграмм используется для моделирования поведения интерфейса, классов, коопераций. Они заостряют внимание на поведении объекта, которое зависит от последовательности событий.

Каждый объект системы, обладающий определенным поведением, может находится в определенных состояниях, переходить из состояния в состояние, совершая при этом определенные действия в процессе реализации сценария поведения объекта.

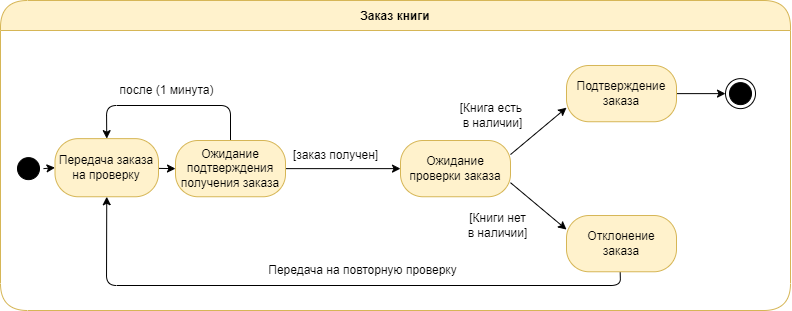


Рисунок 1.10 – Диаграмма состояний

Данная диаграмма состояний представлена со стороны сервера. Сервер считывает информацию для оформления заказа, после чего происходит проверка.

При подтверждении введенных данных происходит проверка заказа на предмет наличия книги, где в положительном случае происходит подтверждение заказа, а в случае ошибки осуществляется передача заказа на повторную проверку.

**1.5.2** Диаграмма последовательности. Диаграмма последовательности – [UML-диаграмма](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%B0%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0_(UML)), на которой для некоторого набора объектов на единой временной оси показан жизненный цикл объекта и взаимодействие актеров (действующих лиц) информационной системы в рамках [прецедента](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%86%D0%B5%D0%B4%D0%B5%D0%BD%D1%82_(UML)).

Диаграмма последовательности представляет собой тип диаграммы взаимодействия, потому что она описывает, как и в каком порядке группа объектов работает вместе.

Эти диаграммы используются разработчиками программного обеспечения и бизнес-специалистами для понимания требований к новой системе или для документирования существующего процесса. Диаграммы последовательности иногда называют диаграммами событий или сценариями событий.

Диаграмма последовательности поиска полета показана на рисунке 1.11.

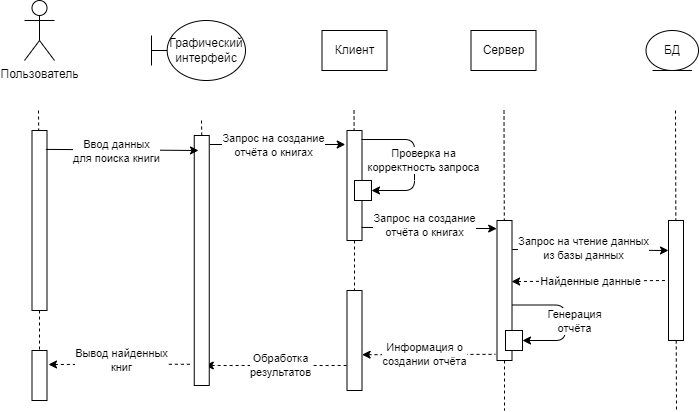


Рисунок 1.11 – Диаграмма последовательности поиска книги

Первоначально пользователь производит ввод данных, взаимодействуя с графическим интерфейсом программы, далее происходит отправка запроса программе клиента с последующей его обработкой на предмет корректности введенной информации. После получения ответа от клиентской программы данные отправляются на сервер, далее в базу данных. Завершающим этапом является вывод найденных книг.

**1.5.3** Диаграмма развертывания. 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. Используя его, вы можете понять, как система будет физически развернута на аппаратном обеспечении.

В контексте унифицированного языка моделирования (UML) диаграмма развертывания относится к семейству структурных диаграмм, поскольку она описывает аспект самой системы. В этом случае схема развертывания описывает физическое развертывание информации, генерируемой программным обеспечением на аппаратных компонентах.

Данная диаграмма представлена на рисунке 1.12.

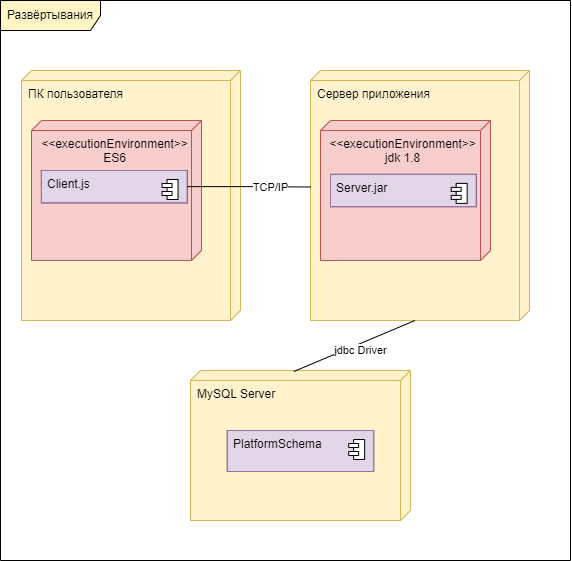


Рисунок 1.12 – Диаграмма развёртывания

Таким образом, диаграмма развертывания предназначена для визуализации элементов и компонентов системы, существующих лишь на этапе ее исполнения (*runtime*), к которым относятся исполнимые файлы, динамические библиотеки, таблицы базы данных. Те компоненты, которые не используются на этапе исполнения (например, исходные тексты программ), на диаграмме не показываются.

В первой главе данного курсового проекта были рассмотрены и проанализированы предметная область, а также главный процесс курсового проекта – осуществить учет товаров в магазине электротоваров. Также были построены основные UML-диаграммы, которые помогут в дальнейшем правильно спроектировать приложение при написании кода. Была разработана и информационная модель, что помогло правильно спроектировать базу данных.

# Проектирование и конструирование программного средства

## 2.1 Постановка задачи

В данном курсовом проекте была сформулирована цель, и определены задачи для её достижения. Однако важно определить главную задачу, чтобы в дальнейшем, ориентируясь на неё, достигнуть наилучших результатов. Главной задачей является создание легкого для понимания пользовательского интерфейса, реализованного для работы с услугами онлайн-платформы по подбору и покупке книг.

Проанализировав деятельность онлайн-платформы, можно уверенно сказать, что разработка данного программного продукта является весьма актуальной.

Приложение должно быть выполнено в архитектуре клиент-сервер с организацией взаимодействия с базой данных на объектно-ориентированном языке Java.

Java обладает большим количеством инструментов для создания приложений клиент-серверного типа с графическим пользовательским интерфейсом. Можно отметить, что важным достоинством выбранного языка программирования является надежность.

Суть клиент-серверной архитектуры заключается в следующем. Клиентские процессы посылают запросы серверному процессу, посылающему обратно результаты этих запросов. Взаимодействие между клиентским и серверным процессами представляет собой совместный транзакционный обмен, в котором активность исходит от клиента, а сервер реагирует на эту активность.

К преимуществам данной модели взаимодействия относится ее безопасность. Все данные хранятся на сервере, который, как правило, защищён гораздо лучше большинства клиентов. На сервере проще обеспечить контроль полномочий, чтобы разрешать доступ к данным только клиентам с соответствующими правами доступа. Также стоит упомянуть, что в приложениях клиент-сервер большое внимание уделяется созданию на клиентской стороне пользователю удобного и понятного интерфейса.

В данном программном средстве будет разрешена работа как администратору, так и обычному пользователю.

В возможности администратора будет входить управление учётными записями пользователей, а также добавление и редактирование данных о действиях пользователей.

Так как обычными пользователями будут являться клиенты онлайн-платформы, то для пользователей данного типа будет реализована возможность добавления нового заказа с выбранными книгами, отслеживание статуса заказа, а также его оплата.

Также для решения поставленных задач можно вывести следующие требования к приложению:

– многопоточность сервера. Для того, чтобы с приложением могли работать сразу несколько пользователей;

– приложение должно запускаться без использования интегрированных средств разработки;

– данные и интерфейс должны быть на русском языке.

Необходимо отладить и протестировать разработанное приложение для того, чтобы была гарантия качественной работы в дальнейшем.

## 2.2 Обоснование выбора компонентов и технологий для реализации программного средства

Программное приложение должно быть написано на объектно-ориентированном языке *Java (с серверной стороны),* и библиотеки *React (с клиентской стороны)*, а также использовать архитектуру «клиент-сервер». Java обладает большим количеством инструментов для создания приложений клиент-серверного типа с графическим пользовательским интерфейсом. Можно отметить, что важным достоинством выбранного языка программирования является надежность.

Суть клиент-серверной архитектуры заключается в следующем. Клиентские процессы посылают запросы серверному процессу, посылающему обратно результаты этих запросов. Взаимодействие между клиентским и серверным процессами представляет собой совместный транзакционный обмен, в котором активность исходит от клиента, а сервер реагирует на эту активность.

Клиент-серверная архитектура – взаимодействие двух самостоятельных процессов – клиента и сервера, которые могу выполняться как на одном, так и на разных компьютерах, обмениваясь данными по сети (см. рисунок 2.1). Серверы – процессы, которые реализуют определенную службу (например, службу файловой системы или БД), и принимают запросы.

Клиенты – процессы, которые запрашивают службы у серверов с помощью отправки запросов и последующего ответа от него.

Таким образом, архитектура «клиент-сервер» позволяет эффективно распределить работу между клиентскими и серверными частями системы: приложения, которые работают на стороне клиента не читают записи базы данных «напрямую», а посылают запросы на сервер, где они обрабатываются, результаты обработки отсылаются назад клиенту, что сокращает потоки информации.



Рисунок 2.1 – Архитектура «клиент-сервер»

Написать программу было выбрано в IDE под названием «Intelij idea». IntelliJ IDEA – интегрированная среда разработки программного обеспечения для многих языков программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains.

Среда разработки также предоставляет функцию множественного выделения, позволяющую вносить в код правки в нескольких местах одновременно. Платформа предоставляет инструменты для работы с базами данных и SQL файлами, включая удобный клиент и редактор для схемы базы данных.

Создание UI проводилось с помощью библиотеки *React*, которая дает возможность написания динамичного, реактивного клиентского приложения.

MySQL Workbench – приложение для работы с языком SQL, полный набор взаимодействия, редактирования, и изменения SQL.

AllFusion Process Modeler 7 – программа для построения диаграмм IDEF. Данный инструмент содержит удобную структуру декомпозиций, которая помогает быстро создать нужные диаграммы.

Таким образом, разработанная программа упростит работу пользователю по выбору наиболее подходящего издательства и покупке необходимых книг, тем самым повысив качество работы онлайн-платформы.

## 2.3 Архитектурные решения

При разработке приложения были использованы следующие паттерны: паттерн *Repository*, а также паттерн *MVC*.

## Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой.

Пример использования шаблона *Null Object* представлен на рисунке 2.2.

## 

## Рисунок 2.2 – Применение паттерна Repository

Образец MVC расшифровывается как Pattern-View-Controller Pattern. Этот шаблон используется для разделения проблем приложения.

*Модель* – модель представляет объект, несущий данные. Он также может иметь логику для обновления контроллера, если его данные изменяются.

*Представление* – представление представляет собой визуализацию данных, содержащихся в модели.

*Контроллер* – контроллер действует как на модель, так и на вид. Он контролирует поток данных в объект модели и обновляет представление при каждом изменении данных. Он сохраняет вид и модель отдельно.

Пример использования данного паттерна в программе показан на рисунках 2.3 – 2.4.



Рисунок 2.3 – Представление модели пользователя

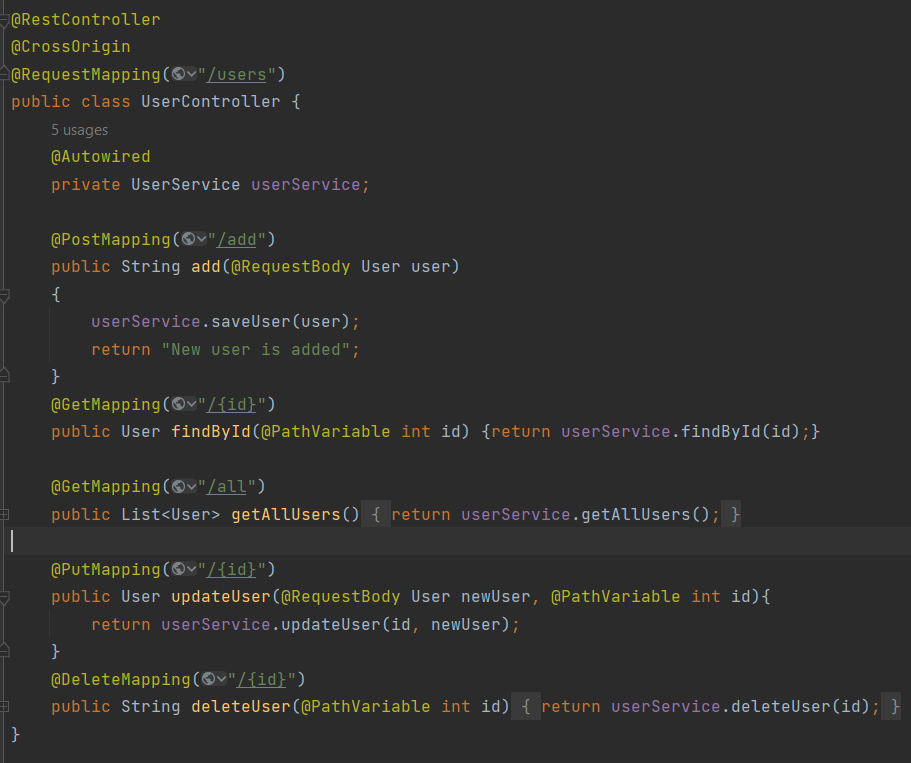


Рисунок 2.4 – Контроллер пользователя

Таким образом в данном пункте были продемонстрированы использованные паттерны проектирования.

## 2.4 Описание алгоритмов, реализующих бизнес-логику разрабатываемого программного средства

Алгоритм работы программы представлен на рисунках 2.5 и 2.6.

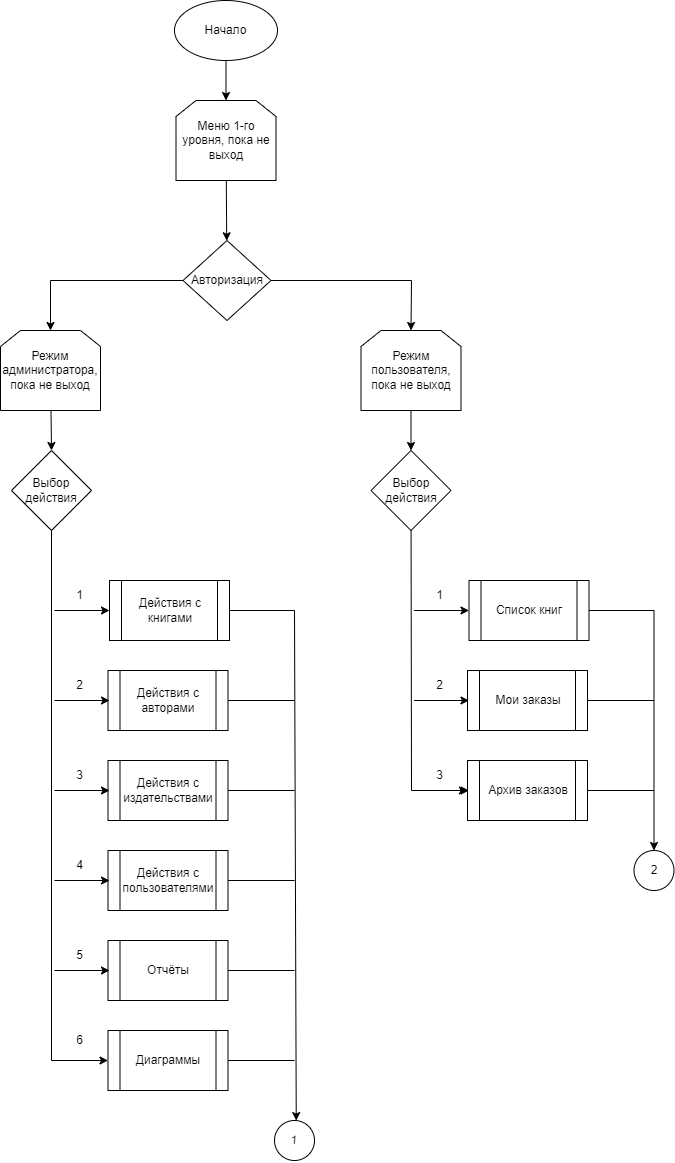


Рисунок 2.5 – Алгоритм работы программы

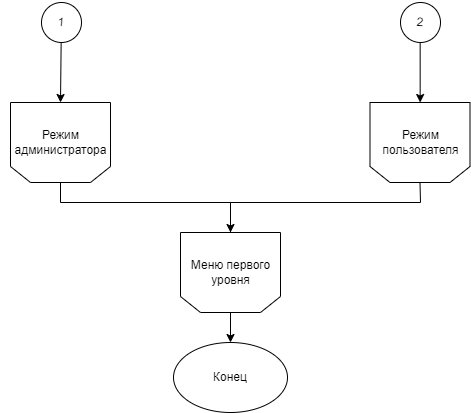


Рисунок 2.6 – Алгоритм работы программы

При входе в программу требуется ввести логин и пароль и в зависимости от введенной информации, в базе данных находится соответствующая роль и открывается нужное меню.

Если авторизация не прошла успешно, то на экран выводится соответствующее сообщение и можно попытаться авторизоваться снова.

У каждой роли есть свой индивидуальный функционал.

У администратора он более широкий. Помимо стандартных операций он может просмотреть и сформировать отчет о заказах, также все записи, которые хранятся в таблицах базы данных можно добавлять, изменять, удалять.

Пользователю доступно взаимодействие с данными, обеспечивающими функционал по действиям с книгами, а также с данными о заказах. Также пользователь может просматривать время доставки и личную информацию.

Важнейшим процессом в моделируемой системе является алгоритм поиска книг. На рисунке 2.7 представлена схема данного алгоритма.

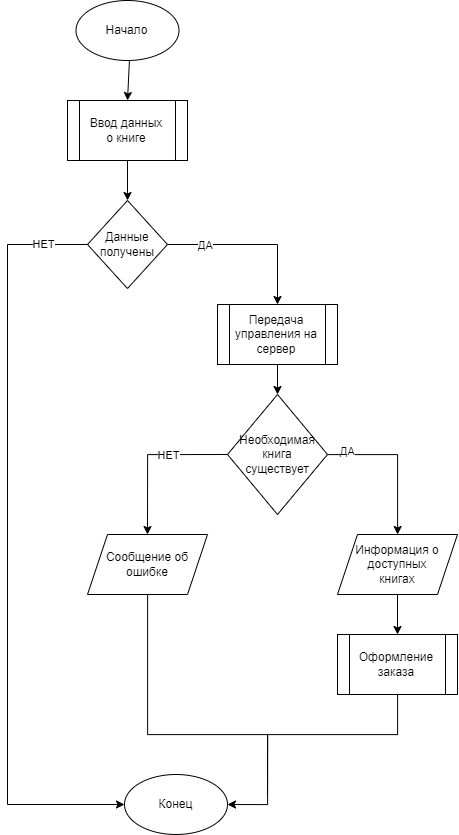


Рисунок 2.7 – Схема алгоритма поиска книги

Алгоритмы и их схемы – это два разных инструмента, используемых в компьютерном программировании. Алгоритм представляет собой пошаговый анализ процесса, в то время как схема алгоритма объясняет шаги программы в графическом виде.

Таким образом, данные схемы алгоритмов наглядно иллюстрируют и помогают лучше понять описываемые алгоритмы.

## 2.5 Проектирование пользовательского интерфейса

Пользовательский интерфейс – это система средств для взаимодействия пользователя с компьютером, основанная на представлении всех доступных пользователю системных объектов и функций в виде графических компонентов экрана (окон, значков, меню, кнопок, списков и т.п.).

При этом, в отличие от интерфейса командной строки, пользователь имеет произвольный доступ (с помощью клавиатуры или указательного устройства ввода) ко всем видимым экранным объектам, а на экране реализуется модель мира в соответствии с некоторой метафорой и осуществляется прямое манипулирование.

Одним из требований к хорошему графическому интерфейсу программной системы является концепция «предсказуемости», чтобы система работала предсказуемо, чтобы пользователь заранее интуитивно понимал, какое действие выполнит программа после получения его команды.

Приложение имеет оконный интерфейс в рамках окна браузера, разработанный с использованием *React*. Главное окно администратора (рисунок 2.8) содержит вкладки, предоставляющие доступ к различным категориям сущностей, участвующих в бизнес-процессе – книги, авторы, издательства, заказы. Все вкладки организованы однотипно, область окна разделяется на две части. В первой располагается меню для работы с приложением. Во второй части отображается тот блок, который был вызван для отображения.

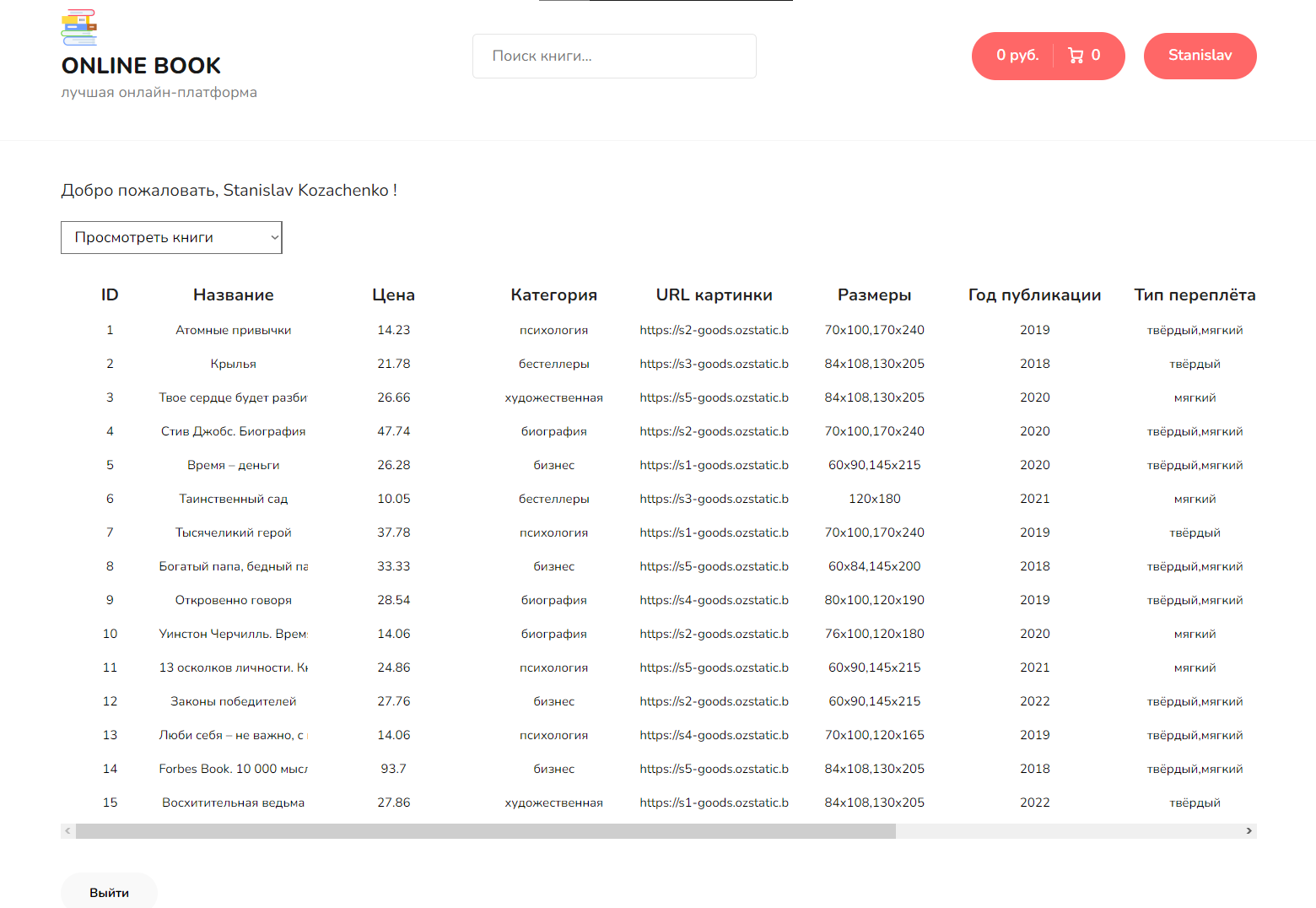


Рисунок 2.8 – Главное окно администратора

В приложении также предусмотрено окно авторизации (проверка имени и пароля пользователя программы). Данное окно служит для идентификации пользователя программы в целях предоставления ему определенных прав в работе с программой.

## 2.6 Методы и средства, используемые для обеспечения безопасности данных

В приложении был реализован алгоритм хеширования PJW-32. Алгоритм разработана Питером Вэйнбергером для общего применения. Не является криптографической хеш-функцией. Пример реализации представлен на рисунке 2.9.

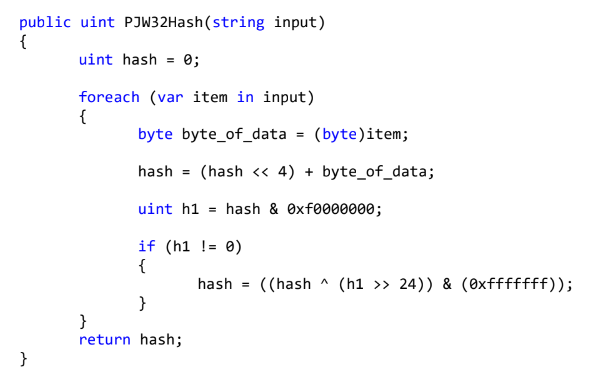


Рисунок 2.9 – Алгоритм шифрования PJW-32

Пример реализации представлен на рисунке 2.10.



Рисунок 2.10 – Пример реализации алгоритма PJW-32

# ТЕСТИРОВАНИЕ И ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММНОГО СРЕДСТВА

Тестирование – это процесс выполнения программы, целью которого является выявление ошибок.

Для того чтобы убедиться, что программное средство может корректно выполнять работу и устойчиво к ошибкам пользователя было проведено тестирование, отражающие все возможные исключительные ситуации.

Во-первых, было рассмотрена возможность не корректной авторизации пользователя, а именно:

* попытка входа без указания данных;
* ввод неверных данных.

Программа реагирует на все ситуации, и пользователю возвращается окно с указанием ошибки. Иллюстрация работы программы приведена на рисунках 3.1-3.2.

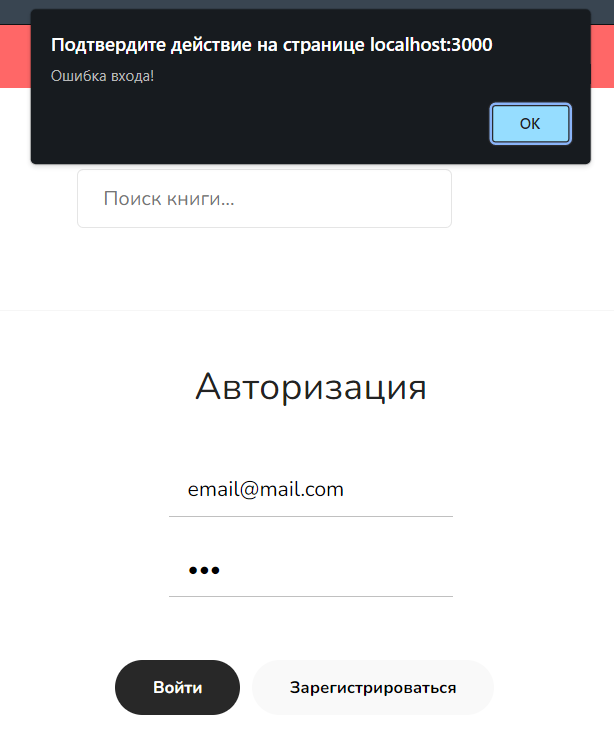


Рисунок 3.1 – Попытка ввода неверных данных

## 

## Рисунок 3.2 – Попытка ввода без указания всех данных

Далее была протестирована работа во время прохождения регистрации пользователя. Программа реагирует на ошибки при регистрации, а именно: регистрация пользователя под логином, который уже существует, попытка регистрации без ввода логина/пароля. Программа реагирует на все ситуации и возвращает пользователю сообщение об ошибке. Действие программы отражено на рисунках 3.3 – 3.5.

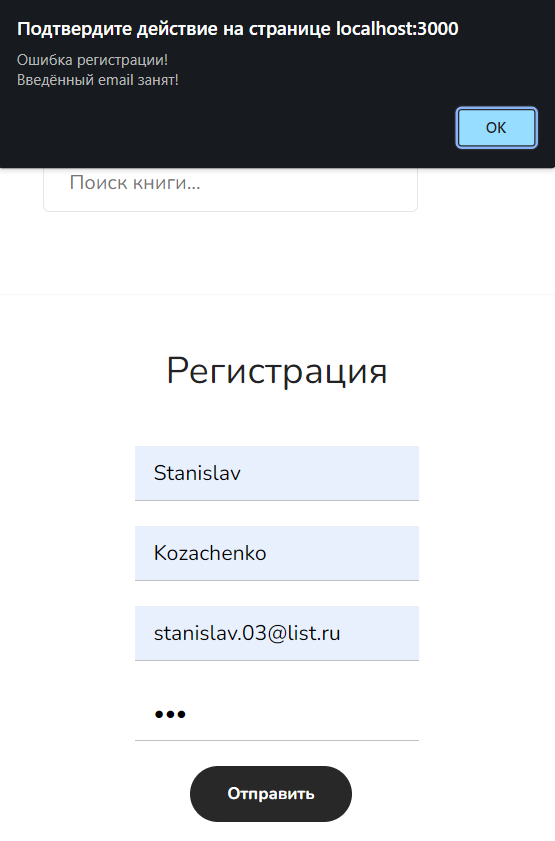


Рисунок 3.3 – Попытка регистрации существующего логина

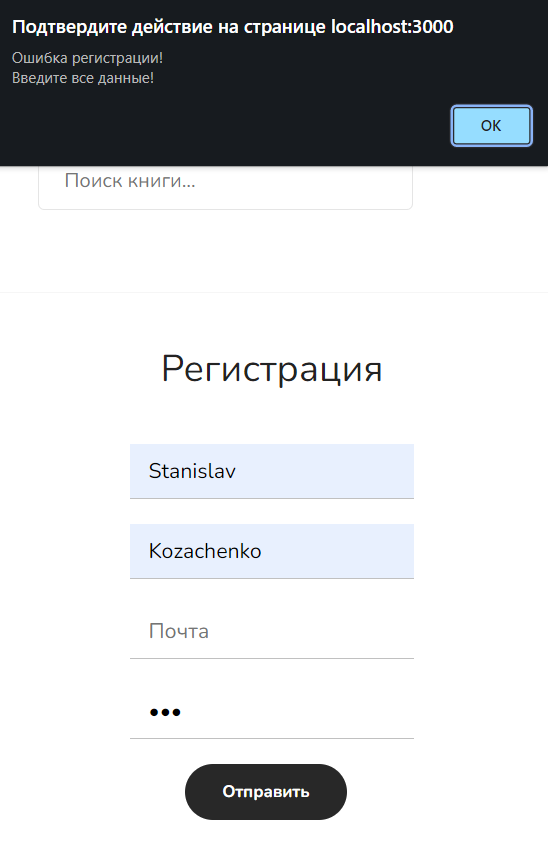
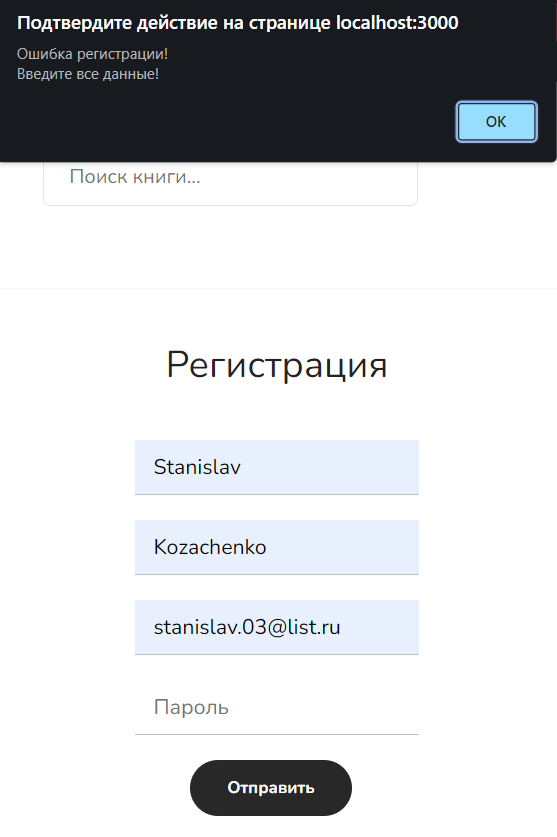


Рисунок 3.4 – Попытка регистрации без ввода логина



## Рисунок 3.5 – Попытка регистрации без ввода пароля

Проведем тестирование функционала системы при добавлении данных. При добавлении новых данных, если не заполнить одно из обязательных полей, программа вернет ошибку пользователю.

Пример действия программы приведен на рисунке 3.6.

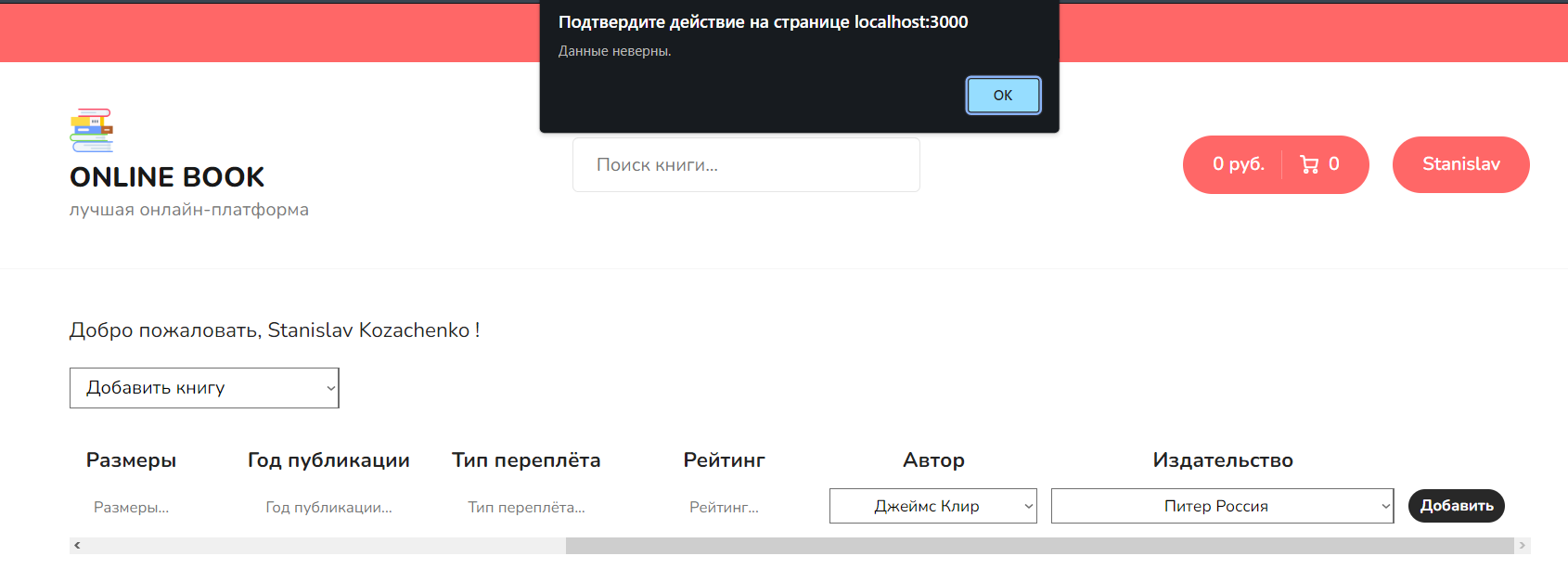


Рисунок 3.6 – Добавление записи с пустыми полями ввода

При удалении данных также осуществляется проверка правильности ввода данных. Пример действия программы приведён на рисунке 3.7.

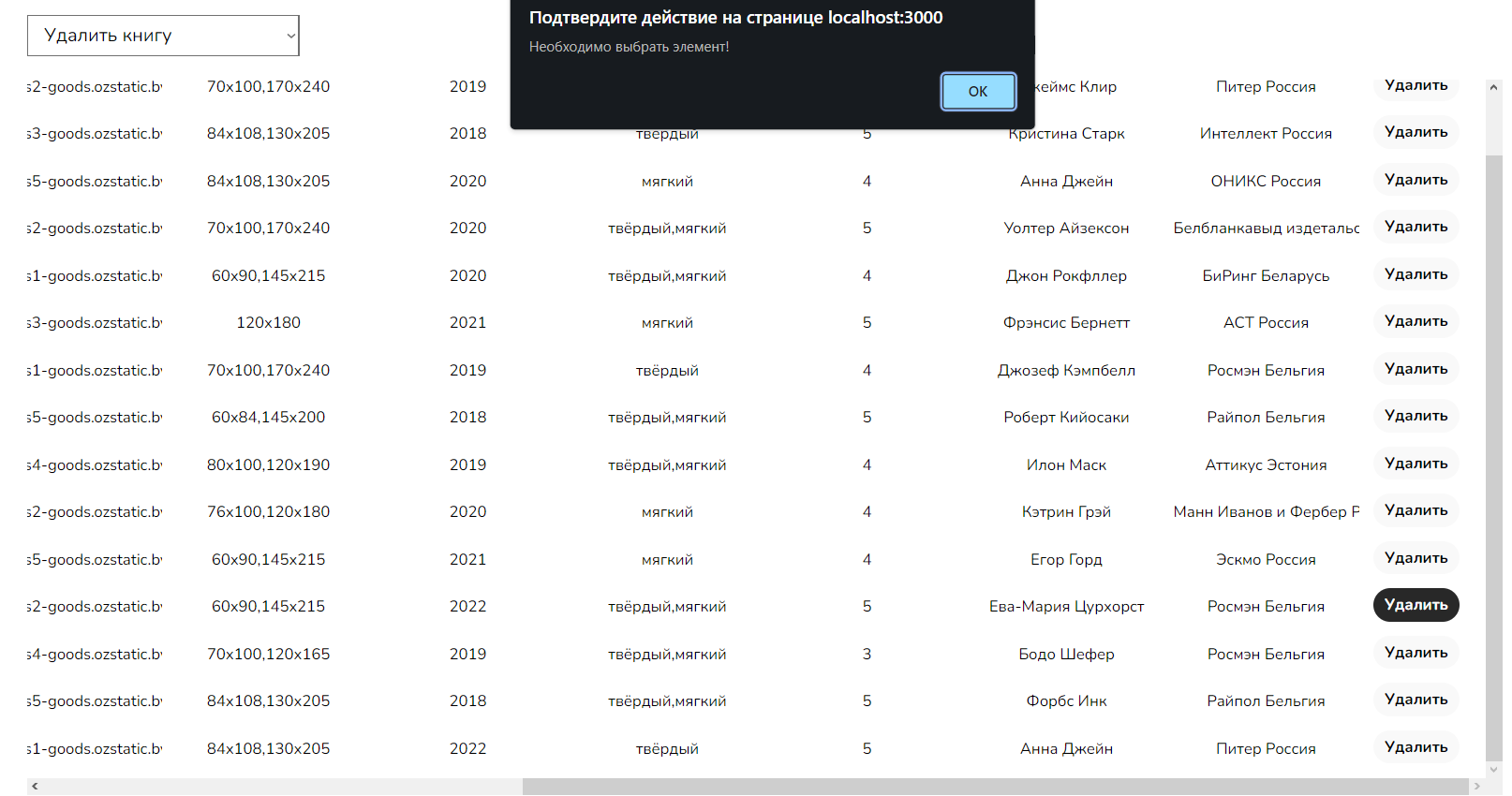


Рисунок 3.7 – Попытка удаления несуществующей книги

## По результатам тестирования можно сделать вывод, что разработанное программное средство удовлетворяет функциональным требованиям и функции выполняются корректно.

# РУКОВОДСТВО ПО РАЗВЁРТЫВАНИЮ И ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

## 4.1 Руководство по развёртыванию

## 4.2 Руководство пользователя

# ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была исследована работа книжного магазина, изучены механизмы заказа книг. Было реализовано программное приложение с понятным и удобным интерфейсом, позволяющее автоматизировать процесс покупки книг. Приложение дает возможность добавлять и удалять авторов и издателей, производить поиск необходимых книг, заказывать книги, и обладает защитами от исключительных ситуаций.

Результатом является программное средство, реализованное на объектно-ориентированном языке Java, обеспечивающее простую и удобную работу с информацией, а также с использованием библиотеки React, обеспечивающую создание пользовательских интерфейсов. Наличие нескольких ролей позволяет разделить полномочия пользователей. Программное средство автоматизирует процесс заказа книг, упрощая работу сотрудников онлайн-платформы.

При разработке было учтено возможное возникновение ошибок при работе с информацией и реализованы исключения, позволяющие избежать некорректного ввода данных.

Для использования данной программы не требуются высококвалифицированные специалисты, так как работа с ней не требует специальных навыков.

В программе предусмотрены возможности по расширяемости и добавлении дополнительного функционала, что может быть полезно при разработке гораздо более сложной системы на основе данного приложения.

Подводя итоги, можно утверждать, что цель, поставленная при создании проекта, была достигнута: данное приложение упростит работу с информацией, а также поможет повысить качество работы онлайн-платформы по подбору и покупке книг, оказывая ему программную поддержку.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

# Приложение А (обязательное) Отчет о проверке на заимствования в системе «Антиплагиат»

# Приложение Б (обязательное) Листинг кода алгоритмов, реализующих основную бизнес-логику

# Приложение В (обязательное) Листинг скрипта генерации базы данных

CREATE DATABASE `bookshop` /\*!40100 DEFAULT CHARACTER SET utf8mb3 \*/ /\*!80016 DEFAULT ENCRYPTION='N' \*/;

SELECT \* FROM bookshop.orders;

CREATE TABLE `authors` (

`firstName` varchar(45) NOT NULL,

`lastName` varchar(45) NOT NULL,

`id` int NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=15 DEFAULT CHARSET=utf8mb3;

CREATE TABLE `books` (

`id` int NOT NULL AUTO\_INCREMENT,

`authorId` int NOT NULL,

`publisherId` int NOT NULL,

`title` varchar(45) NOT NULL,

`sizes` varchar(45) NOT NULL,

`category` int NOT NULL,

`publicationYear` int NOT NULL,

`price` double NOT NULL DEFAULT '0',

`types` varchar(45) NOT NULL,

`imageUrl` varchar(255) NOT NULL,

`rating` int NOT NULL,

PRIMARY KEY (`id`),

KEY `authorId\_idx` (`authorId`),

KEY `publisherId\_idx` (`publisherId`),

CONSTRAINT `authorId` FOREIGN KEY (`authorId`) REFERENCES `authors` (`id`),

CONSTRAINT `publisherId` FOREIGN KEY (`publisherId`) REFERENCES `publishers` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=16 DEFAULT CHARSET=utf8mb3;

CREATE TABLE `orderitem` (

`id` int NOT NULL AUTO\_INCREMENT,

`orderId` int NOT NULL,

`bookId` int NOT NULL,

PRIMARY KEY (`id`),

KEY `FK\_orderItem\_book\_idx` (`bookId`),

KEY `FK\_orderItem\_order` (`orderId`),

CONSTRAINT `FK\_orderItem\_book` FOREIGN KEY (`bookId`) REFERENCES `books` (`id`),

CONSTRAINT `FK\_orderItem\_order` FOREIGN KEY (`orderId`) REFERENCES `orders` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=18 DEFAULT CHARSET=utf8mb3;

CREATE TABLE `orders` (

`id` int NOT NULL AUTO\_INCREMENT,

`userId` int NOT NULL,

`orderDate` date NOT NULL,

`total` double NOT NULL,

`code` int NOT NULL,

PRIMARY KEY (`id`),

KEY `FK\_order\_user\_idx` (`userId`),

Продолжение приложения В

CONSTRAINT `FK\_order\_user` FOREIGN KEY (`userId`) REFERENCES `users` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8mb3;

CREATE TABLE `publishers` (

`id` int NOT NULL AUTO\_INCREMENT,

`country` varchar(45) NOT NULL,

`name` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=12 DEFAULT CHARSET=utf8mb3;

CREATE TABLE `transactions` (

`id` int NOT NULL AUTO\_INCREMENT,

`orderId` int NOT NULL,

`createdAt` date NOT NULL,

`userId` int NOT NULL,

PRIMARY KEY (`id`),

KEY `FK\_transaction\_order\_idx` (`orderId`),

CONSTRAINT `FK\_transaction\_order` FOREIGN KEY (`orderId`) REFERENCES `orders` (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=18 DEFAULT CHARSET=utf8mb3;

CREATE TABLE `users` (

`id` int NOT NULL AUTO\_INCREMENT,

`firstName` varchar(45) NOT NULL,

`lastName` varchar(45) NOT NULL,

`email` varchar(45) NOT NULL,

`passwordHash` varchar(45) NOT NULL,

`registeredAt` date NOT NULL,

`role` int NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=4 DEFAULT CHARSET=utf8mb3;

# ВЕДОМОСТЬ ДОКУМЕНТОВ КУРСОВОГО ПРОЕКТА

# 

## 

## 

## 

## 