МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Инженерно-экономический факультет

Кафедра экономической информатики

Дисциплина: Современные технологии обработки экономической информации (с модулем Технологии автоматизации делопроизводства)

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

**Пояснительная записка**

к курсовому проекту

на тему

**Web-портал подбора и реализации книг**

БГУИР КП 1– 40 05 01-02 20 ПЗ

|  |  |  |
| --- | --- | --- |
| Студенты: |  | Е. П. Красовский  группа 772303 |
| Руководитель: |  | Е.Н. Унучек |

Минск 2020

СОДЕРЖАНИЕ

[Введение 3](#_Toc59002685)

[1 Описание работы web-портала подбора и реализации книг 4](#_Toc59002686)

[2 Описание процесса реализации книг в web-портала 6](#_Toc59002687)

[2.1 Описание методологии функционального моделирования IDEF0 6](#_Toc59002688)

[2.2 Отображение модели «Как есть» в нотации IDEF0 6](#_Toc59002689)

[2.3 Описание модели «Как будет» в нотации IDEF0 9](#_Toc59002690)

[3 Спецификация вариантов использования web-портала по продажи книг 15](#_Toc59002691)

[4 Информационная модель системы по продаже книг и её описание 17](#_Toc59002692)

[5 Обоснование выбора компонентов и технологий для реализации курсового проекта 19](#_Toc59002693)

[6 Модели представления системы по продаже книг и их описание 22](#_Toc59002694)

[7 Руководство по развёртыванию системы по продаже книг 27](#_Toc59002695)

[8 Результаты тестирования разработанной системы по продаже книг 28](#_Toc59002696)

[Заключение 38](#_Toc59002697)

[Список используемых источников 39](#_Toc59002698)

[Приложение А (обязательное) блок-схема алгоритма 40](#_Toc59002699)

Приложение Б (необязательное) листинг кода  [42](#_Toc59002703)

# Введение

Еще буквально несколько лет назад в нашей стране чтение было любимым временем препровождения для многих граждан (и детей, и взрослых). Но в последние годы интерес к книгам значительно поутих, даже библиотеки опустели. С одной стороны, это случилось из-за широкого внедрением компьютеров и планшетов в нашу жизнь, с другой стороны – из-за большой занятости у людей просто нет времени выбраться в магазин для выбора книги. Поэтому все чаше люди переходят на цифровые издания книг.

Именно поэтому и была выбрана тема Web-портал подбора и реализации книг.

Рекомендуемая стратегия: разработка MVP.

**MVP** (minimum viable product) – это минимально жизнеспособный продукт, который позволяет получить осмысленную обратную связь от пользователей, понять, что им нужно и не создавать то, что им неинтересно и за что они не готовы платить [1].

Целью данного курсового проекта является упростить процесс продажи книг и сократить трудозатраты, связанные с ним, для пользователя посредством упрощения количества действий необходимых для покупки, а так же что не маловажно выбрать себе ту самую книгу. Чтобы автоматизировать данный процесс произведена разработка системы по продаже книг на языках Java и JavaScript, предоставляющие удобные инструмента поиска, сортировки и многое другое.

Для достижения поставленной цели в курсовом проекте решаются следующие задачи:

* изучить предметную областьподбора и реализации книг и разработать модель**,** отражающую бизнес-процесс на предприятии;
* разработать и описать постановку задачи на разработку программного приложения;
* выполнить проектирование программного приложения, используя диаграммы на языке UML;
* разработать и описать алгоритм работы программного приложения;
* выполнить реализацию программного средства;
* предусмотреть исключительные ситуации и протестировать работу приложения на различных устройствах;
* разработать и описать руководство пользователю.

# Описание процесса ПОДБОРА И РЕАЛИЗАЦИИ КНИГ

* 1. **Общие подходы к организации процессов подбору и реализаций книг**
  2. **1. сравнительный анализ возможностей существующих автоматизированных систем по подбору и реализации книг**

Web-порталов подбора и реализации книг – сайт, торгующий товарами в интернете. Позволяет пользователям подобрать себе книгу по вкусу и скачать ее из сети Интернет.

В первую очередь, это позволяет сэкономить огромное количество времени. Не нужно добираться до магазина по пробкам, тратить средства на проезд, можно просто сидеть в удобном кресле, за чашкой чая, неторопливо щелкая при этом мышкой.

Так гораздо проще искать необходимое пособие, да и выбор книг в интернет-магазинах намного шире, а цены доступнее, так как продавец не платит за аренду помещения и, следовательно, не включает её в стоимость товара. А это, для многих любителей чтения, само собой, является весомым аргументом делать покупки именно через Интернет. Ведь цены на хорошие книжные издания порой очень кусаются.

Также есть возможность не торопиться с выбором, и не спеша прочитать отзывы о планируемой вами покупке, пообщаться с теми, кто уже прочитал ее и спросить совета у таких же прожжённых книгоманов. Многие читатели с радостью оставляют рецензии на прочитанное издание.

При открытии интернет-магазина может возникнуть ситуация, при которой продажи ни через неделю, ни через месяц после открытия не появляются. Основными проблемами являются отсутствие службы поддержки, сложности с заказом и связью.

Одним из способов избегания подобной ситуации является оказание как можно большего внимания клиентам, а именно, хорошо структурированная служба поддержки.

Также количество анкетных опросов на сайте должно быть сведено к минимуму, чтобы процесс заказа по времени занимал не больше трёх минут. Если нужно собрать о клиенте больше информации, то нужно попросить его ответить на несколько вопросов уже после оформления покупки.

Примером для разработки служит сайт Chitatel.by (рисунок 1.1). Сравнивая версию сегодняшнего сайта с тем, что уже реализовано в Chitatel.by, можно понять, что есть у нас, и чего нет у конкурента, что есть у него, однако нет у нас. Анализ функциональности поможет добиться того, чтобы на сайте были только полезные инструменты, которые помогут оптимизировать конверсию и, как следствие, увеличить выручку.

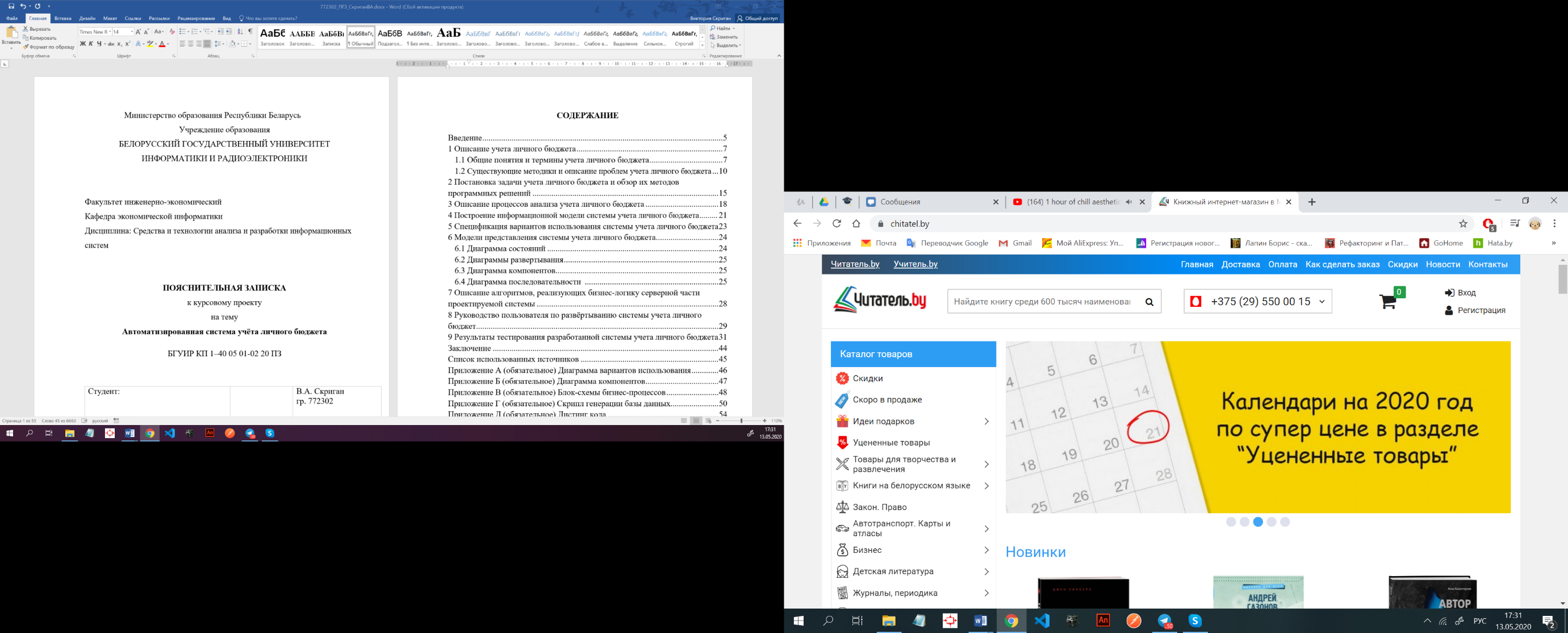


Рисунок 1.1 – Сайт Chitatel.by

Если потенциальному клиенту сложно найти кнопку для оформления заказа, какие-то конкретные группы товаров или поле поиска, то это отличный повод уйти с сайта и больше никогда на него не возвращаться. По данным Gomez, 88% клиентов в такой ситуации больше на сайт не вернутся. И этим все сказано.

Следовательно, при необходимости конкретного функционала, которого ещё нет на сайте, можно собрать обратную связь от пользователей и улучшить сервис.

# Описание процесса Реализации книг в WEB-портала

* 1. Описание методологии функционального моделирования IDEF0

IDEF (Integrated DEFinition) – методологии семейства ICAM (Integrated Computer-Aided Manufacturing) для решения задач моделирования сложных систем, позволяют отображать и анализировать модели деятельности широкого спектра сложных систем в различных разрезах. При этом широта и глубина обследования процессов в системе определяется самим разработчиком, что позволяет не перегружать создаваемую модель излишними данными [6].

Для отображения основных процессов автоматизированной системы по продаже книг используется диаграмма IDEF0.

Описание системы с помощью IDEF0 называется функциональной моделью. Функциональная модель предназначена для описания существу­ющих бизнес-процессов, в котором используются как естественный, так и графический языки. 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 (диаграммы декомпозиции). Затем каждая подсистема разбивается на более мелкие и так далее до достижения нужной степени подробности.

* 1. Отображение модели «Как есть» в нотации IDEF0

Для демонстрации модели AS-IS, или «Как есть» используется пример из реальной жизни, когда нужно отслеживать, как происходит процесс оказания услуг по удовлетворению спроса населения на книжные изделия. В результате чего, на контекстном уровне расположен блок «Книжный магазин». В качестве управления выступают «Накладные», «Прайс-лист», и «Законодательство РБ». На вход подаются «Список товаров для заказа у поставщика», «Информация о рынке», «Каталог товаров». Выполняет работу «Персонал» данного заведения.

Как результат – полученная от продажи товаров прибыль, удовлетворение потребности клиента и готовая финансовая и товарная документация.

На рисунке 2.1 изображён контекстный уровень, описанный выше.

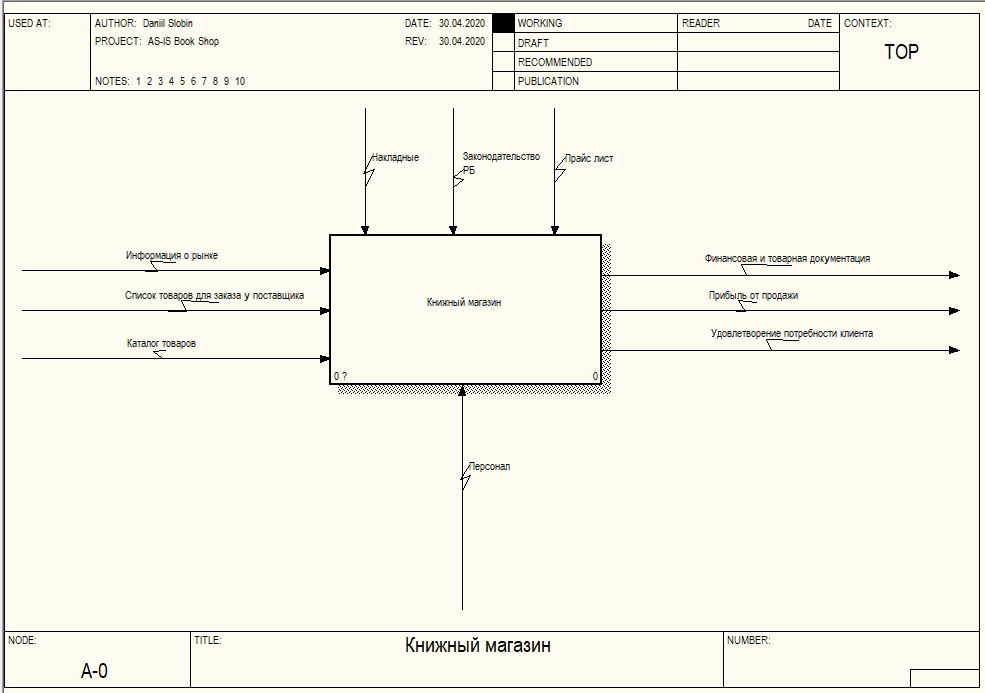


Рисунок 2.1 – Контекстный уровень диаграммы AS-IS

Далее контекстный уровень декомпозируется для более подробного описания бизнес-процессов, что, в свою очередь позволяет нам более подробно понять то, как функционирует магазин по продаже книг. На рисунке 2.2 изображена декомпозиция основного процесса.

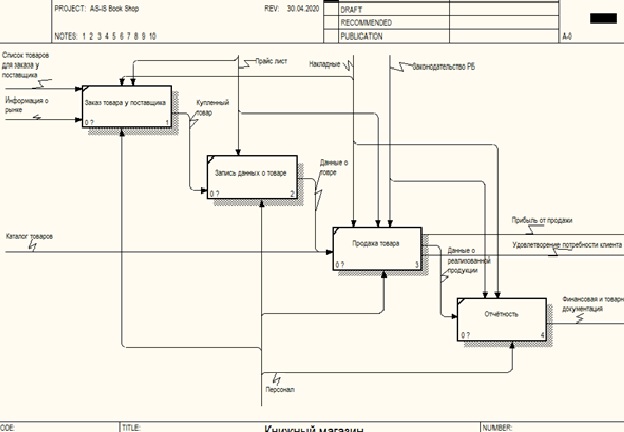


Рисунок 2.2 – Декомпозиция блока «Книжный магазин»

На изображении показано, что для функционирования книжного магазина необходимо четыре процесса.

Первым делом «Персонал» выполняет «Заказ товара у поставщика», для этого ему необходимо узнать «Информацию о рынке» и составить «Список товаров для заказа у поставщика». Регулирует данный процесс «Накладная» и «Прайс-лист». Результатом данного процесса является «Купленный товар».

После получения товара «Персонал» должен произвести «Запись данных о товаре». Данное действие регулирует «Прайс-лист» на товары и в итоге мы получаем готовые «Данные о товаре» и можем начинать его реализацию.

Процесс «Продажа товара» получает на вход «Каталог товаров», налаживание данного действия происходит при помощи двух регламентирующих данных: «Накладные» и «Законодательство РБ». В конечном итоге книжный магазин получает «Отчётность», благодаря которой на выходе можно получить «Финансовую и товарную документацию».

В связи с тем, что процессы, описанные выше, не автоматизированы, персоналу книжного магазина приходится вести всю документацию вручную. Что в свою очередь может привести к финансовым потерям предприятия, так как в ходе работы персонал из-за большого объёма информации начинает ошибаться, не успевать.

Данную проблему можно решить при помощи автоматизации процесса продажи книг. Для этой цели и был создан сайт Tzeench-shop. Прежде всего, создание сайта поможет уменьшить время и стоимость выполнения процесса, а вмести с этим и нагрузку на персонал.

В следующей главе будет приведён пример системы «TO-BE» или «Как будет».

* 1. Описание модели «Как будет» в нотации IDEF0

Для создания сайта была разработана диаграмма в нотации IDEF0 «Как будет» для того, чтобы можно было руководствоваться взвешенными решениями и видеть то, что создаётся.

На контекстном уровне расположен блок «Web-портал подбора и реализации книг». В качестве управления выступают «Данные о лицензиях», «Инструкция об использовании» «Продавец», и «Законодательство РБ». На вход подаются «Информация о рынке», «Заявки клиентов», «Прайс лист», «Каталог книг». Выполняет работу «Клиент», «ПО» и «Персонал» данного заведения. В конечном счёте, будут получены «Прибыль от продаж», «Реклама», полученная от удовлетворенных посетителей, а так же «Статистика» покупок и «Реализованный товар». На рисунке 2.3 отображён контекстный уровень диаграммы «TO-BE».

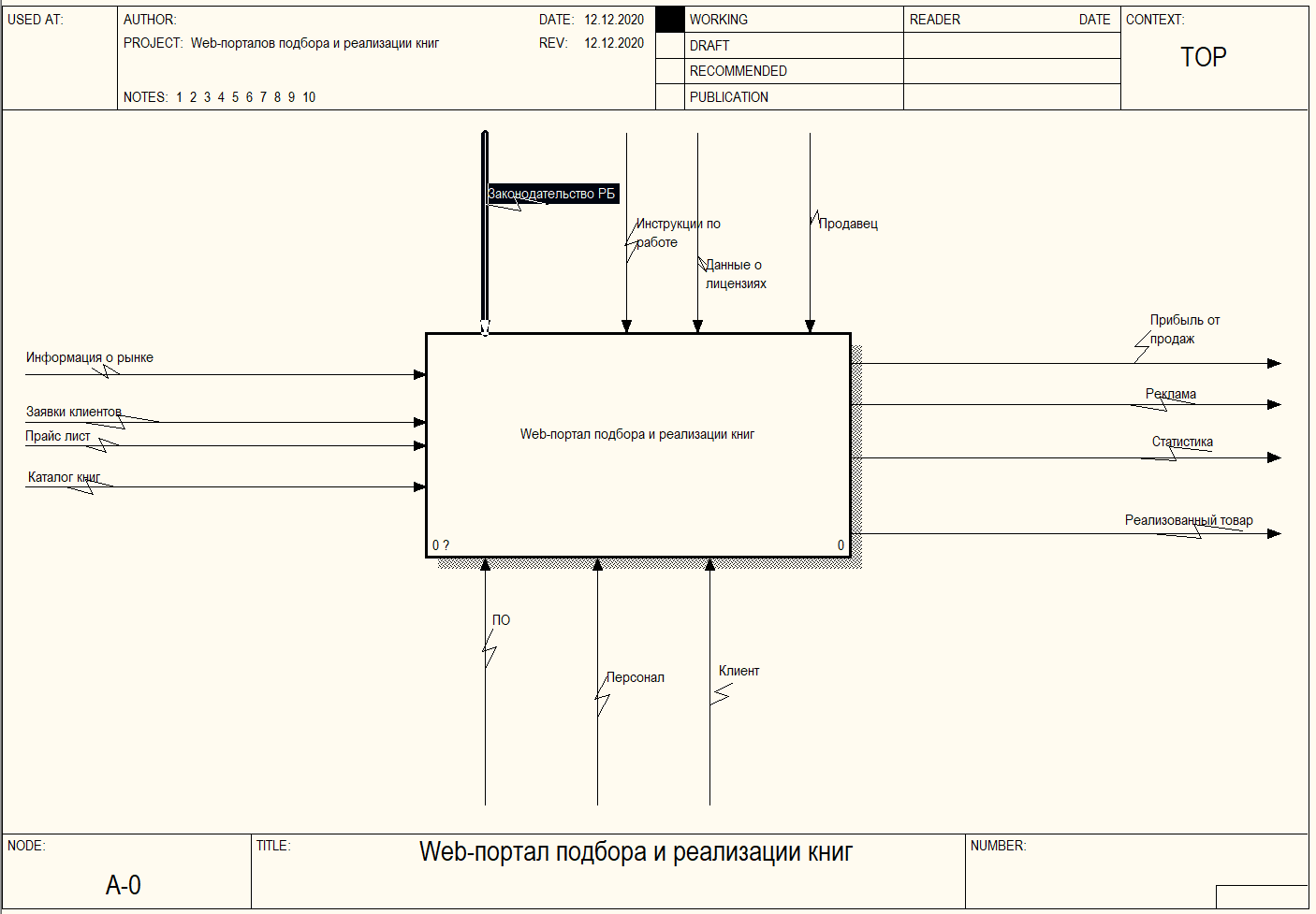


Рисунок 2.3 – Контекстный уровень диаграммы TO-BE

Декомпозиция блока «Книжный интернет-магазин» состоит из 4 бизнес- процессов (рисунок 2.4). Рассмотрим более подробно каждый из них.

Работа книжного магазина начинается с того, что необходимо произвести «Заказ лицензий на реализацию книг», для этого составляется список необходимых товаров, исходя из «Информация о рынке», «Заявки клиентов» и «Каталог книг» web-портала, для этого из базы берутся «Данные о лицензиях» и после этих действий «Персонал» производит заказ лицензий. Следующим действием является «Загрузка книг и данных о них на веб портал», так как теперь этот процесс автоматизирован, помимо персонала в нём ещё участвует «ПО».

Вместе с этим к следующему процессу добавляется «Клиент». Идет процесс «реализации книг», но об этом чуть позже, для этого используем «обновленный каталог», а также руководствуемся «инструкции по работе» и «законодательством РБ». Собственно, сам процесс происходит при участии «ПО», «Персонала» и «самого клиента»

Последним шагом на данном уровне декомпозиции является составление «Отчётности», которую регулирует «Законодательство РБ» и, благодаря которой можно узнать «Статистику» работы магазина.

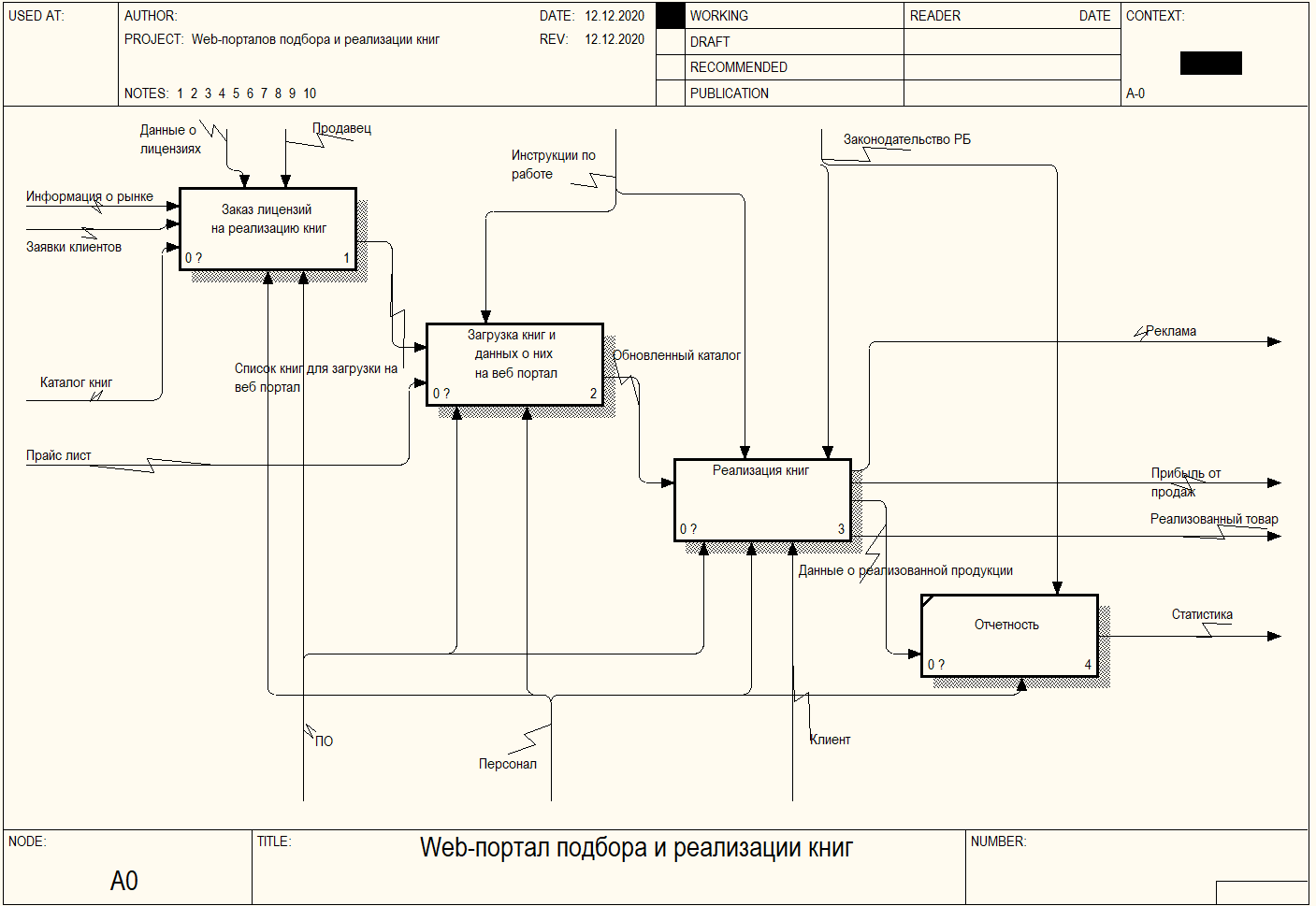


Рисунок 2.4 –Декомпозиция блока «Книжный интернет-магазин»

Далее рассмотрим процесс «Заказ товара у поставщика». Он состоит из четырёх мероприятий «Создание и заполнение документа», «Добавление товара в документ», «Отправление документа по электронной почте» и «Получение заказа от поставщика», в результате которых магазин получает «Список приобретённых товаров». На рисунке 2.5 вы можете ознакомиться с ними более подробно.

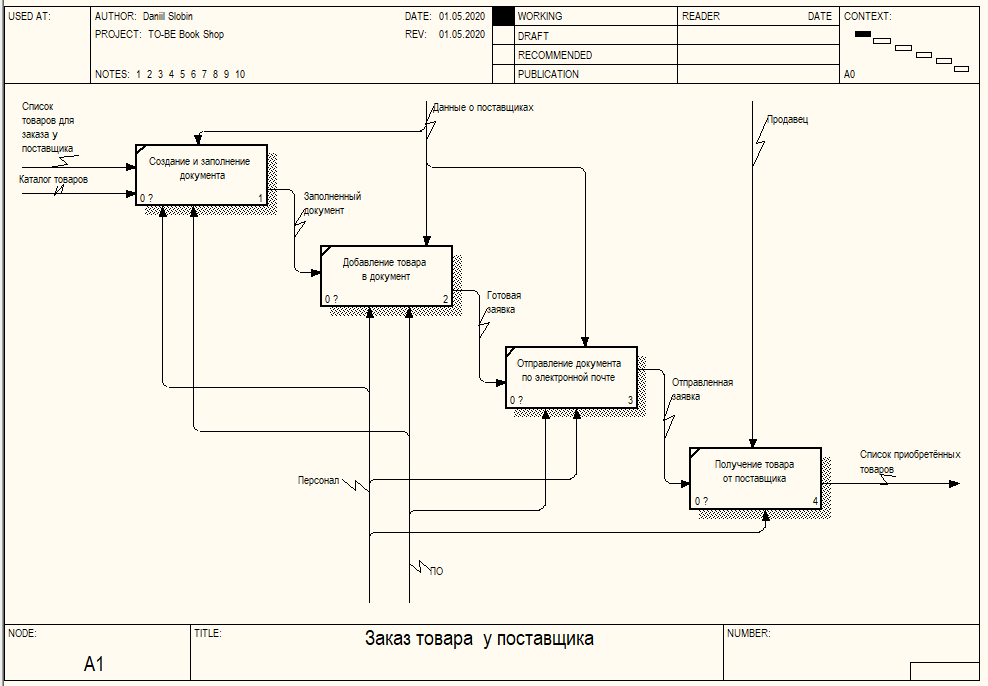


Рисунок 2.5 – Декомпозиция блока «Заказ товара у поставщика»

Процесс «Загрузки книг и данных по ним на веб портал» состоит из следующих этапов «Заполнение необходимых полей», «загрузки товара в корзину», «Создание новости о новинке». Все эти работы контролирует «Инструкции по работе», то есть магазин, осуществляющий деятельность по реализации книг. «Персонал» при помощи «ПО» выполняет все вышеназванные действия. Декомпозиция блока «Загрузки книг и данных по ним на веб портал» предоставлена на рисунке 2.6.

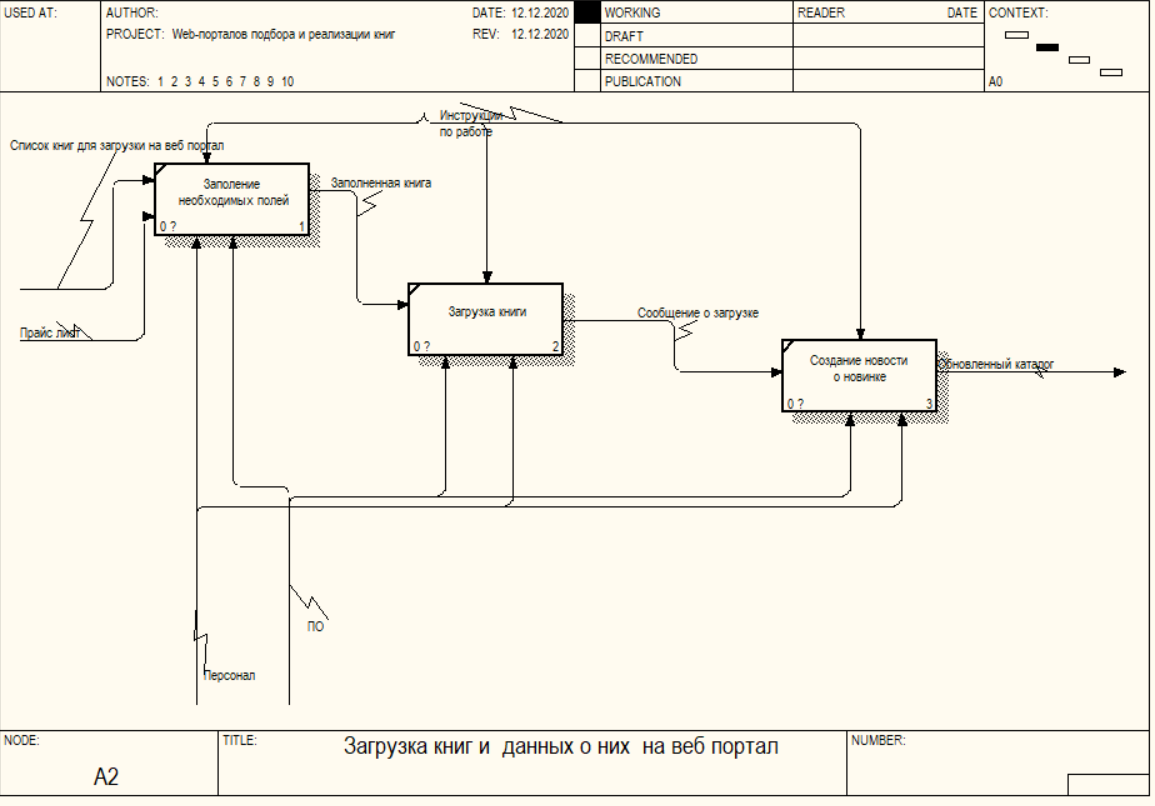


Рисунок 2.6 – Декомпозиция блока «Оформление заказа»

Последний, но не менее важный блок «Реализация», чтобы выполнить данное действие, необходимо исполнить четыре работы «Получить запрос от клиента», «Переход к выбранным книгам», «оплата» и «Загрузка книг клиенту».

Первым делом выполняется блок «Получить запрос от клиента». Для этого «Персонал» получает «Обновленный каталог». После чего «Клиент» на портале обращается к «Персоналу» или к другим пользователям при помощи «ПО», после чего они рекомендуют список из книг. Далее идёт «Переход к выбранным книгам», его выполняет «Клиент», при помощи «ПО». В данном блоке клиент смотрит, что за Книга, какие на нее отзывы, цена, рейтинг и решает нужна ли ему такая книга. После происходит «Оплата». Результатом данного действия является «Подтверждение оплаты». В конечном счёте «ПО» проводит «Загрузку книги клиенту» и на выходе получается «Реализованный товар» и «Данные о реализованной продукции». Все мероприятия данного процесса изображены на рисунке 2.7.

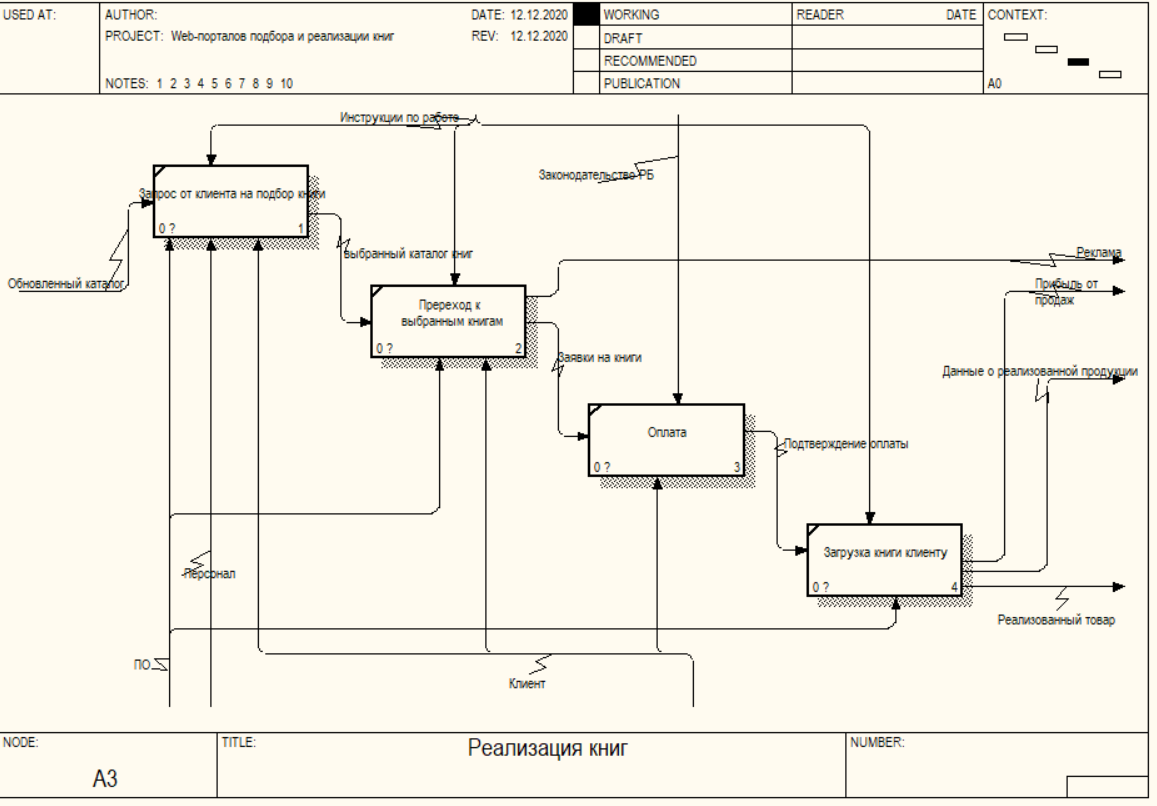


Рисунок 2.7 – Декомпозиция блока «Доставка»

Общая схема процесса, представленная на рисунке 2.3, раскрыта и демонстрирует, что каждый этап важен. Пропуск действий может привести к дополнительным временным и денежным ресурсам.

# 3 СПЕЦИФИКАЦИЯ вариантов использования WEB-портала по продажи книг

Основная цель создания любой программной системы – создание такого программного продукта, который помогает пользователю выполнять свои задачи. Для создания таких программ первым делом определяются требования, которым должна удовлетворять система.

Диаграмму вариантов использования необходимо строить во время изучения технического задания, она состоит из графической диаграммы, описывающей действующие лица и прецеденты, а также спецификации, представляющего собой текстовое описание конкретных последовательностей действий (потока событий), которые выполняет пользователь при работе с системой. Спецификация затем станет **основой для тестирования и документации**, а на следующих этапах проектирования она дополняется и оформляется в виде диаграммы (в рамках ICONIX используется диаграмма последовательности, но в UML для этого имеются также диаграммы деятельности). Кроме того, use-case диаграмма достаточно проста, чтобы ее мог понять заказчик, следовательно, вы **можете использовать ее для согласования ТЗ** (ведь диаграмма описывает функциональные требования к системе).

Между прецедентами может быть отношение включения (include) или расширения (extend), которые должны быть отражены на диаграмме вариантов использования. Может быть так же включено условие, при котором доступен тот или иной прецедент. На рисунке 3.1 представлена диаграмма использования.

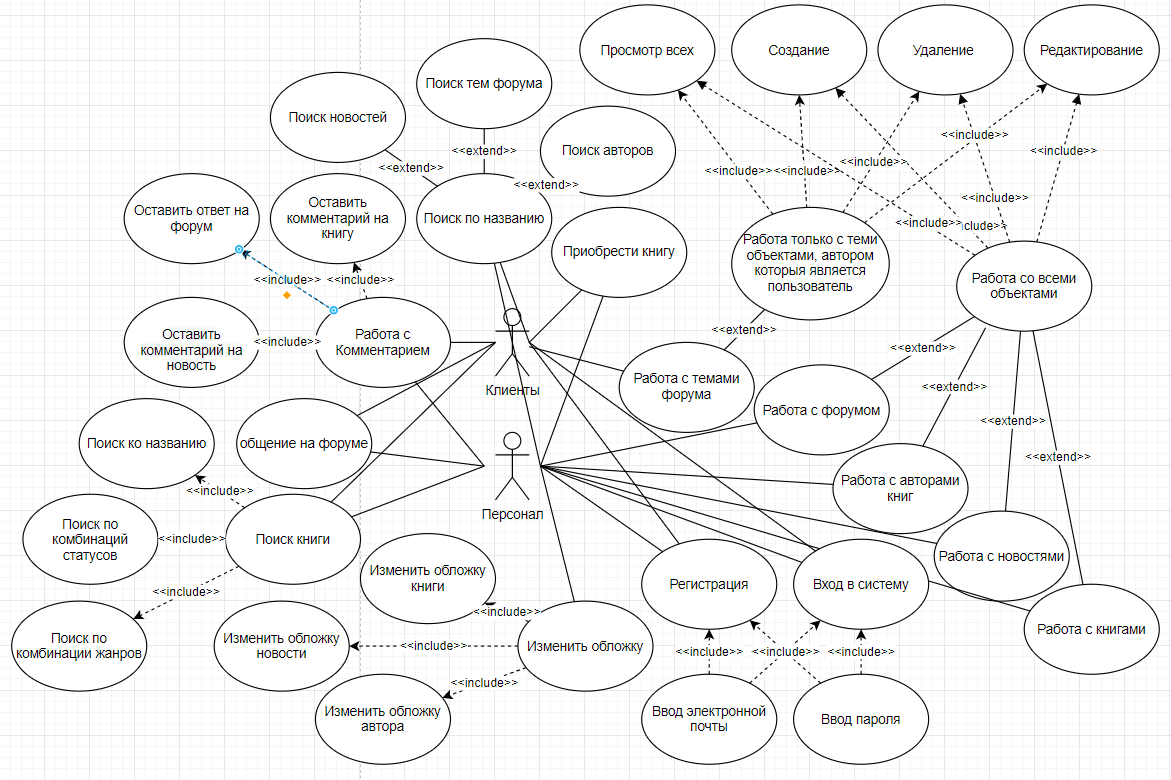


Рисунок 3.1 – Диаграмма вариантов использования системы

В приложении есть два актера, функционал которых отображены на рисунке 3.1.

Персонал может управлять списком книг, авторов, новостей, форумом, также производить действия удаления, редактирования и добавления, поиска различного рода, ещё ему доступна возможность удалять отзывы, которые оставляют пользователи, так как они могут нарушать какие-либо правила. Например, комментарий может содержать персональные данные, нецензурную лексику, оскорбления, сведения, порочащие честь и достоинство и так далее.

Пользователь (не администратор) обладает всем остальным функционалом, начиная от регистрации заканчивая изменением своих персональных данных и комментариями к книгам.

Для регистрации необходимо указать email, логин для гугл аккаунта. Зарегистрированному пользователю доступен вход (если его аккаунт не заблокирован администратором), после чего открывается домашняя страница сайта.

Пользователь может сделать заказ, посмотреть список доступных книг, оставить отзыв к интересующей его книге, и т.д. в соответствии с UML схемой.

# 4 ИНФОРМАЦИОННАЯ модель системы по продаже книг и её описание

Для хранения информации используется система управления базами данных MySQL. Концептуальная модель базы данных (дальше – БД) описывает сущности, их свойства и связи между ними.

**Сущность** (entity) – это реальный или представляемый тип объекта, информация о котором должна сохраняться и быть доступна. В диаграммах сущность представляется в виде прямоугольника, содержащего имя сущности. При этом имя сущности – это имя типа, а не некоторого конкретного экземпляра этого типа [2]. Примеры сущностей в данном проекте: users, books, reviews. Каждый экземпляр сущности (объект) должен быть отличим от любого другого экземпляра той же сущности.

В реальных СУБД модель данных, кроме атрибутов каждой сущности определяет также связи между сущностями. Связи формально определяются как ассоциации между участниками. Например, утверждение "пользователи покупают книги" указывает, что существует связь между сущностью "пользователи" и сущностью "книги".

Существует несколько типов связей между сущностями: "один к одному", "один ко многим" и "многие ко многим".

Записи об определенных параметрах каждой из сущностей называются атрибутами. Например, для сущности "книга", видимо, будет храниться информация об её наименовании, жанре, авторах и т.д.

Выбор нужного комплекта атрибутов – одна из самых больших проблем при проектировании баз данных. Очень часто в реальной базе данных нужный комплект атрибутов в итоге не хранится - просто по той причине, что пользователи не смогли сообщить в процессе сбора информации, что он действительно нужен. Иногда в базе, наоборот, попадают лишние атрибуты, заполнение которых требует дополнительного времени. Очень часто возникает проблема с форматом вводимых данных, например, на какие части делить адрес и что делать с нестандартными случаями.

Общее правило при выборе набора атрибутов: нужно начинать с результата и стараться упрощать модель, а не усложнять ее.

Данные хранятся в двумерных массивах (таблицах), то есть модель данных является реляционной. При этом значения в этой таблице должны быть соответствующе заданных типов данных. Отсутствие повторяющихся данных, использование уникальных идентификаторов, позволяющих определённо идентифицировать каждую из строк. Хранение данных – построчное. Использование по колоночного хранения позволяет ускорить поиск и фильтрацию данных, однако скорость записи уменьшается в разы и использование построчного хранения является оптимальным вариантом. Дальше будут рассмотрены сущности, используемые БД курсовой работы

Для полноценной работы приложения необходимы +- 25 таблиц базы данных.

На рисунке 4.1 изображена диаграмма IDEF1X, демонстрирующая связи сущностей БД между собой.

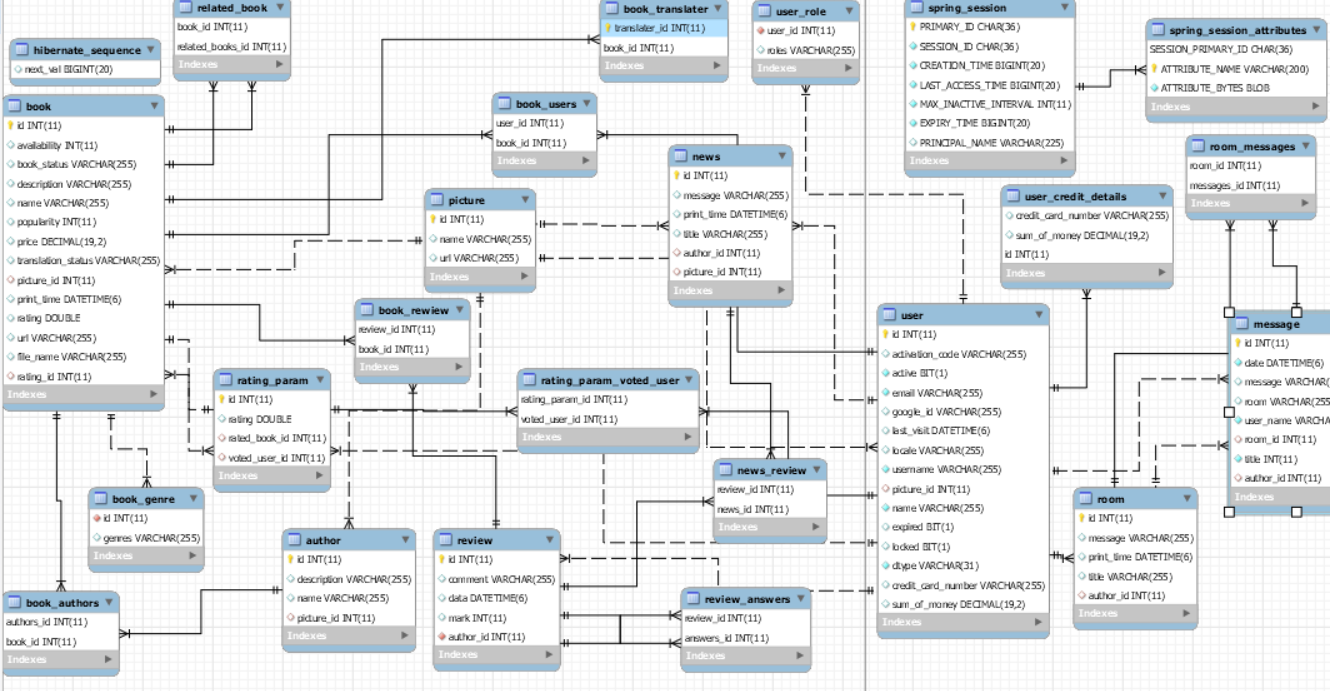


Рисунок 4.1 – Таблицы базы данных

Схема соответствует всему описанию, которое представлено выше и позволяет полностью реализовать спроектированную систему.

# 5 Обоснование выбора компонентов и технологий для реализации курсового проекта

Для достижения поставленной цели был выбран для бэкенд разработки язык Java с технологией Spring Boot и фрэймворк Angular для фронтенд разработки.

По данным МАРТ, в 2017 году доля интернет-торговли в розничном товарообороте Беларуси составила 2,8%. В денежном выражении это составило 1,1 млрд. рублей или $568 млн. На непродовольственные товары приходится 80,8% всех интернет-покупок жителей нашей страны. На продовольственные, соответственно, 19,2% [5].

За 2017 год 45% или 4,3 млн. белорусов купили какой-либо товар онлайн. Следовательно, разработка сайта должна быть в приоритете для заказчика, так как именно с неё будет получен максимальный эффект.

Главный плюс Java – принцип “написано однажды – работает везде”. Это значит, что ПО, написанное на одной платформе, будет запускаться и на других устройствах. В принципе, использование языка Java очень обширно: он используется для создания приложений для мобильных устройств, удаленных процессоров, беспроводных модулей, датчиков, да и в целом – практически любых электроустройств.

Также Java широко используется в электронной коммерции и в области вэб-приложений. Огромное количество RESTful сервисов было создано с использованием Spring MVC, Struts 2.0 и похожих фреймворков. Даже простейшие приложения, основанные на Servlet, JSP и Struts, достаточно популярны в различных государственных проектах. Многие вэб-приложения государственных, оздоровительных, страховых, образовательных, оборонительных и некоторых других отделений написаны на Java.

Spring – один из наиболее широко используемых фреймворков для разработки интерпрайз-приложений, обеспечивающий продуманную модель программирования и конфигурации. Целью создания данного фреймворка способствовало желание упростить разработку приложений на популярном в то время (да и сейчас) Java EE стеке технологий от компании Oracle, который на тот момент был очень сложен и не всегда удобен в использовании.

Использование Hibernate позволит создать ORM-модель между программными объектами и записями БД. Возможность строить автоматически запросы к БД сокращает время на извлечение данных. Он генерирует SQL вызовы и избавляет разработчика от ручной обработки результирующего набора данных и конвертации объектов, сохраняя приложение портативным во все SQL базы данных.

Для обеспечения хранения информации была выбрана база данных MySQL. Данная СУБД является решением для малых и средних приложений. Входит в состав серверов [WAMP](https://ru.wikipedia.org/wiki/WAMP), [AppServ](https://ru.wikipedia.org/w/index.php?title=AppServ&action=edit&redlink=1), [LAMP](https://ru.wikipedia.org/wiki/LAMP) и в портативные сборки серверов [Денвер](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BD%D0%B2%D0%B5%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0)), [XAMPP](https://ru.wikipedia.org/wiki/XAMPP), [VertrigoServ](https://ru.wikipedia.org/w/index.php?title=VertrigoServ&action=edit&redlink=1). Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Гибкость СУБД MySQL обеспечивается поддержкой большого количества типов таблиц: пользователи могут выбрать как таблицы типа MyISAM, поддерживающие полнотекстовый поиск, так и таблицы InnoDB, поддерживающие транзакции на уровне отдельных записей. Более того, СУБД MySQL поставляется со специальным типом таблиц EXAMPLE, демонстрирующим принципы создания новых типов таблиц. Благодаря открытой архитектуре и GPL-лицензированию, в СУБД MySQL постоянно появляются новые типы таблиц [3].

На протяжении всего процесса разработки приходится принимать массу решений – начиная от характеристик продукта и до технологических аспектов, которые в будущем повлияют на оперативность базы кода.

Angular может принимать решения и предлагает разработчикам параметры по умолчанию для таких вещей как сетевое соединение, административное управление состояниями, выбор языка, набор инструментов конфигурации. Эти параметры постоянно тестируются и корректируются, поэтому платформа Angular непрерывно и стабильно развивается. Эти параметры были созданы по принципу свободы выбора. В зависимости от потребностей конкретной организации Angular позволяет сменить их, если это нужно. Подтверждением этих слов является библиотека Material Design. Material Design задействовала преимущества последнего, самого лучшего Angular, но Angular позволяет компаниями создавать или выбирать любую библиотеку UI Library, которая им подходит.

Angular является идеальным подходом к созданию одностраничных приложений, в которых информация и представление распознаются в соответствии с дизайном Model-View-Controller («Модель-Представление-Контроллер»).

MVC – схема разделения данных приложения, пользовательского интерфейса и управляющей логики на три отдельных компонента: модель, представление и контроллер – таким образом, что модификация каждого компонента может осуществляться независимо [4].

Уточним, что термин «компонент» в данном случае не имеет никакой связи с компонентами некоторых популярных CMS или фреймворков, а компоненты Битрикса, например, вообще строятся из всех трёх составляющих MVC.

Схематичное изображение MVC на рисунке 5.1.

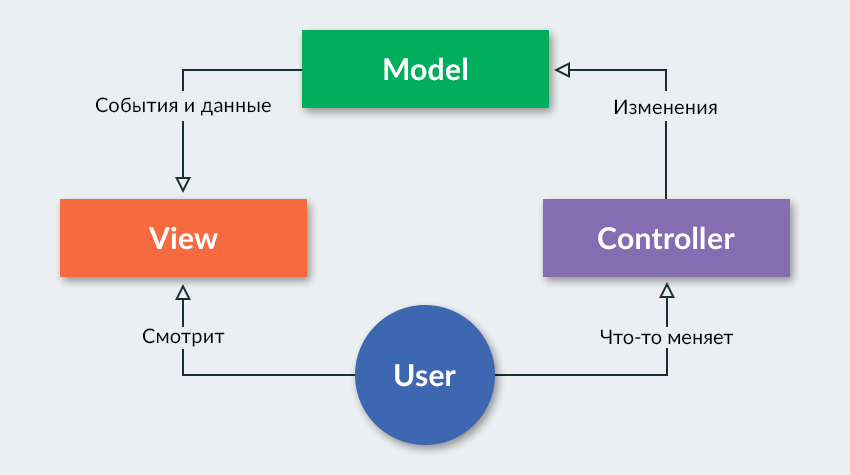


Рисунок 5.1. – Паттерн MVC

В приведённом определении под компонентом следует понимать некую отдельную часть кода, каждая из которых играет одну из ролей Контроллера, Модели или Представления, где Модель служит для извлечения и манипуляций данными приложения, Представление отвечает за видимое пользователю отображение этих данных (то есть, в применении к вебу, формирует отдаваемый сервером браузеру пользователя HTML/CSS), а Контроллер управляет всем этим оркестром.

# 6 Модели представления системы по продаже книг и их описание

UML – унифицированный язык моделирования (Unified Modeling Language) – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем.

UML проектирование имеет не мало плюсов, вот одни из них:

* возможность посмотреть на задачу с разных точек зрения;
* другим программистам легче понять суть задачи и способ ее реализации;
* диаграммы сравнительно просты для чтения после достаточно быстрого ознакомления с их синтаксисом.

Для того, чтобы разобраться, нужно ли именно вам использовать UML, необходимо рассмотреть основные диаграммы. Благодаря им складывается общая картина, дающая представление о возможностях выражения архитектурных идей в рамках бизнес-задач.

Все представленные ниже диаграммы связаны между собой. Комбинируя их, мы можем добиться необходимого уровня декомпозиции отдельно взятых задач.

Предлагаю познакомиться с одними из самых полезных и часто используемых диаграмм. Речь пойдет о диаграммах последовательности, состояний, деятельности и самой сложной из них – диаграмме классов.

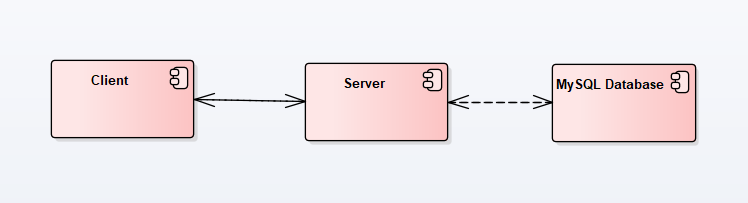


Рисунок 6.1 – Диаграмма компонентов

Диаграмма компонентов отражает разбиение программной системы на структурные компоненты и связи между ними. Объектами могут быть модули, исполняемые файлы, библиотеки, пакеты и прочее. На рисунке 6.1 отражена связь из трёх компонентов: когда приложению необходимо сделать запрос на сервер, тот делает запрос к базе данных, получает он неё ответ и возвращает приложению. Трёхуровневая архитектура позволяет увидеть ошибки, если таковые имеются, на нужном этапе без необходимости лезть в весь код.

На рисунке 6.2 изображена диаграмма последовательности.



Рисунок 6.2 – Диаграмма последовательности входа в приложение

На диаграмме последовательности основными объектами выступают Клиент, Сервер и окно База данных. Они нужны для того, чтобы показать, как происходит взаимодействие между ними при использовании приложения.

При запуске приложения пользователь попадает на страницу входа, на которой ему нужно ввести логин и пароль, далее эти данные отправляются на сервер. Сервер делает запрос по выбранному логину к БД, получает результат и сравнивает логины и пароли. Если такие данные были найдены, то сервер возвращает роль пользователя, логин и токен, а после этого пропускает пользователя на главную страницу. В обратном случае пользователю будет сообщено, что данные введены некорректно и ему необходимо снова ввести логин и пароль.

На рисунке 6.3 изображена диаграмма развёртывания, которая позволяет смоделировать физическое развёртывание артефактов на узлах.

Физическое представление программной системы не может быть полным, если отсутствует информация о том, на какой платформе и на каких вычислительных средствах она реализована. Конечно, если разрабатывается простая программа, которая может выполняться локально на компьютере пользователя, не задействуя никаких периферийных устройств и ресурсов, то в этом случае нет необходимости в разработке дополнительных диаграмм. Однако при разработке корпоративных приложений ситуация представляется совсем по-другому.

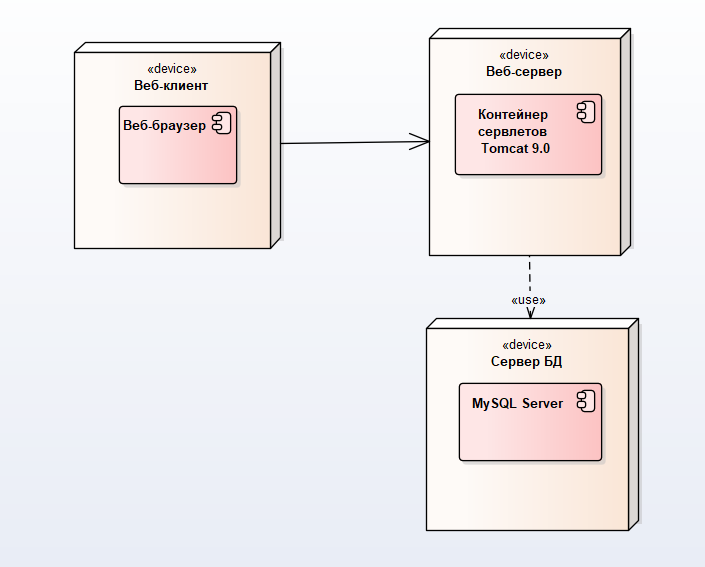


Рисунок 6.3 – Диаграмма развёртывания

Диаграмма развёртывания показывает связь веб-сервера с веб-браузером и сервером базы данных. При том, что база данных одна, сервер один, а браузеров может быть больше. То есть веб-сервер связан с Сервером БД связью один к одному, а с Веб-клиентом – один ко многим. На диаграмме могут быть указаны:

* узлы устройства;
* узлы среды выполнения.

Взаимодействие различных частей так же может быть указано (например, RMI, REST или JDBC).

Диаграмма состояний показывает диаграмму, моделирующую динамику систем (рисунок 6.4). На диаграмме показаны все стадии регистрации нового пользователя.

С приложением будет работать человек, а значит исключить ошибки невозможно. При авторизации приложение проверит данные на корректность ввода, например, поле логина и пароля не могут быть пустыми. Далее проводится проверка на совпадение с уже зарегистрированными пользователями. Если на каком-либо этапе произойдёт ошибка, администратору будет выведено сообщение об ошибке, и просьба о вводе новых данных. После прохождения проверок данные могут быть внесены в базу данных.

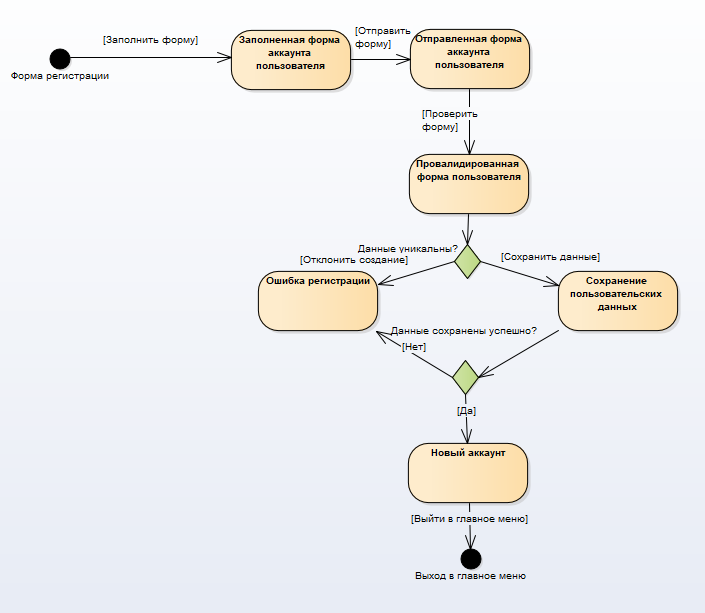


Рисунок 6.4 – Диаграмма состояний регистрации пользователя

Блок-схемы авторизации и фильтрации представлены на рисунках А.1 и А.2, соответственно.

Авторизация (рисунок А.1). Авторизоваться можно, пока не выключено приложение, отправляете данные на сервер, если данные корректны, сервер отправляет их в БД. В случае если такой пользователь есть, определяется роль пользователя и выводится соответствующее меню, в противном выводится ошибка авторизации.

Фильтрация (рисунок А.2). Пользователь вводит фильтры, форма проверяет их на валидность, если проверка успешна, данные можно отправить. После отправки происходит обработка формы, по результатам которой сервер отправляет запрос к СУБД. СУБД ищет в БД нужные данные и отправляет их на сервер, а тот в свою очередь к клиенту на вывод.

1. Руководство по РАзвёртыванию системы по продаже книг

Давайте разберем процесс развертывания системы. Запуск веб-сервера представлен на рисунке 7.1.

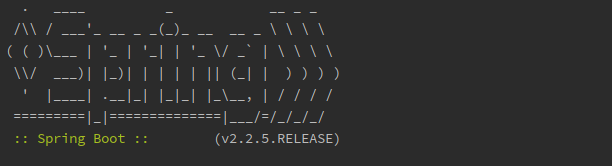


Рисунок 7.1 – Запуск веб-сервера

Для запуска веб-сервера необходимо ввести команду java –jar tzeentch-shop.jar через командную строку и запустить.

Для запуска клиентской части необходимо ввести доменный адрес в браузере.

# 8 РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ РАЗРАБОТАННОЙ СИСТЕМЫ ПО ПРОДАЖЕ КНИГ

После запуска приложения высвечивается окно, изображённое на рисунке 8.1. На данной странице можно сразу отправиться в каталог, а зачем еще нужен данный web-портал, использовать поиск, каталог и переход к популярным книгам и просто новинкам. Как бонус открыты последние новости и темы форума в кол-ве 4 ед.

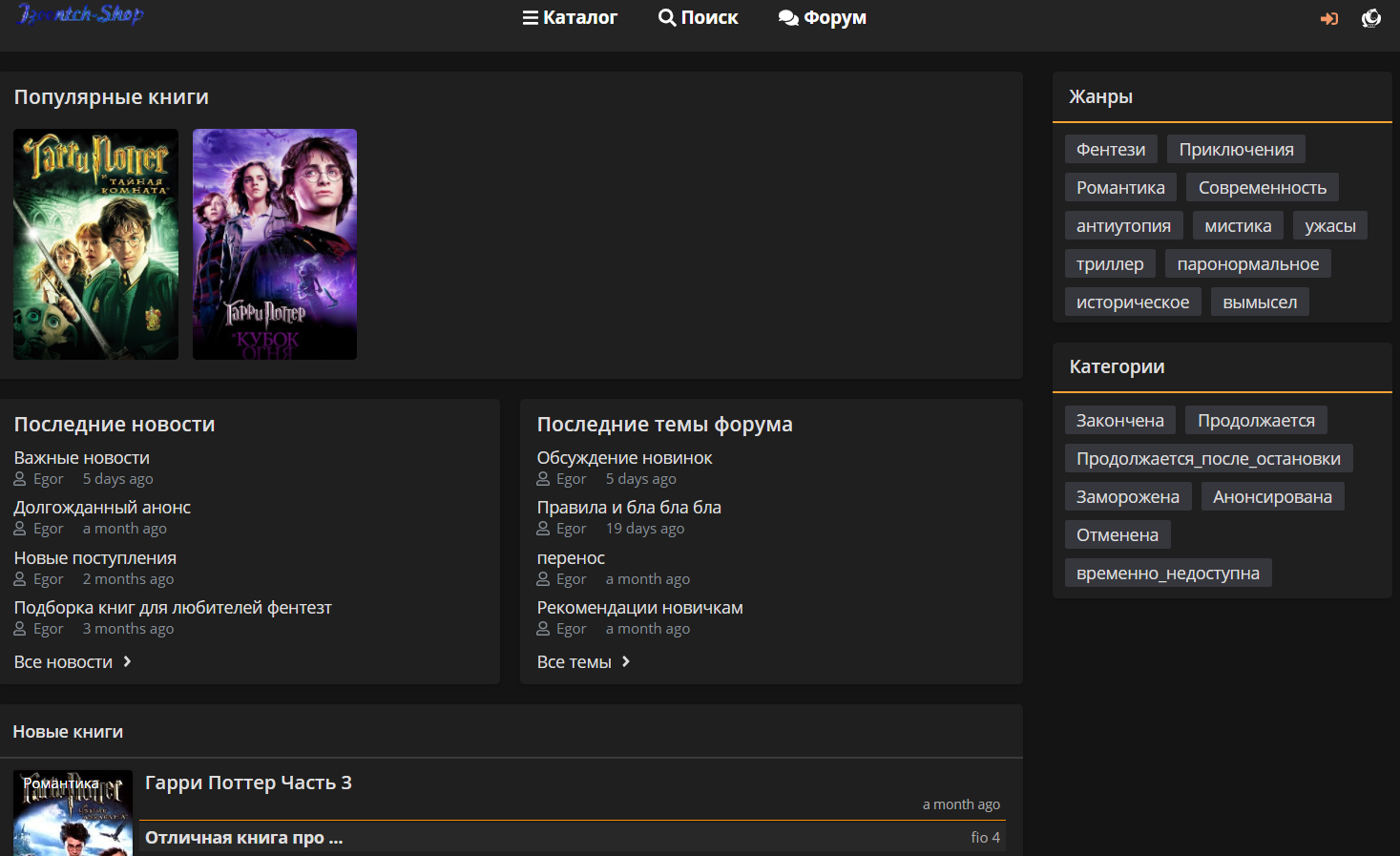
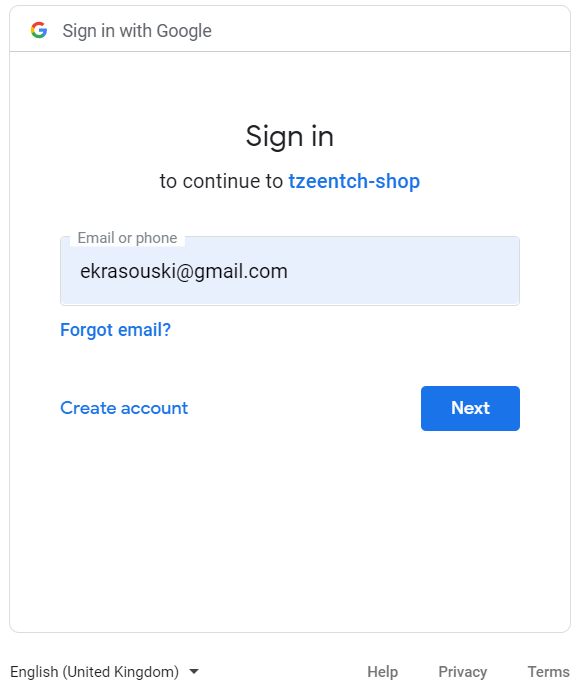


Рисунок 8.1 – Главная страница сайта

Чтобы иметь доступ ко всем возможностям системы, необходимо в ней зарегистрироваться. Страница регистрации изображена на рисунке 8.2.



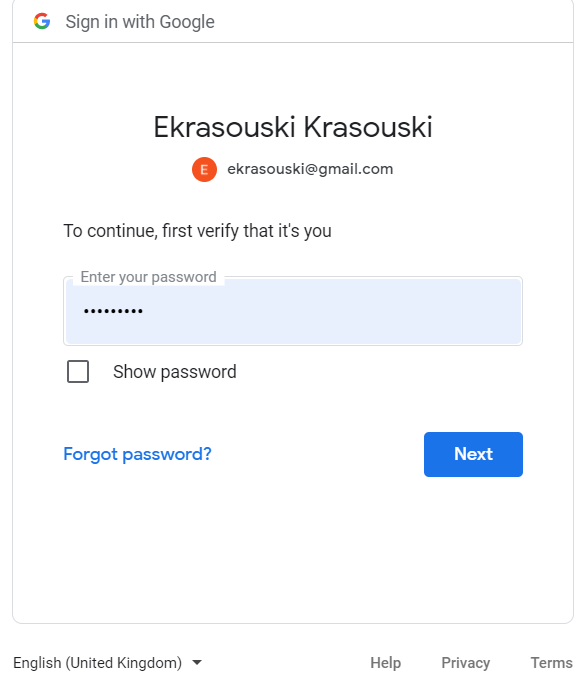


Рисунок 8.2 – Страница регистрации

Если пользователь уже зарегистрирован, после запуска приложения он может сразу вводить свои данные, которые вводил при регистрации, после чего он получит возможность оставлять отзывы на сайте, покупать книги и т.д. Авторизация отображена на рисунке 8.3.

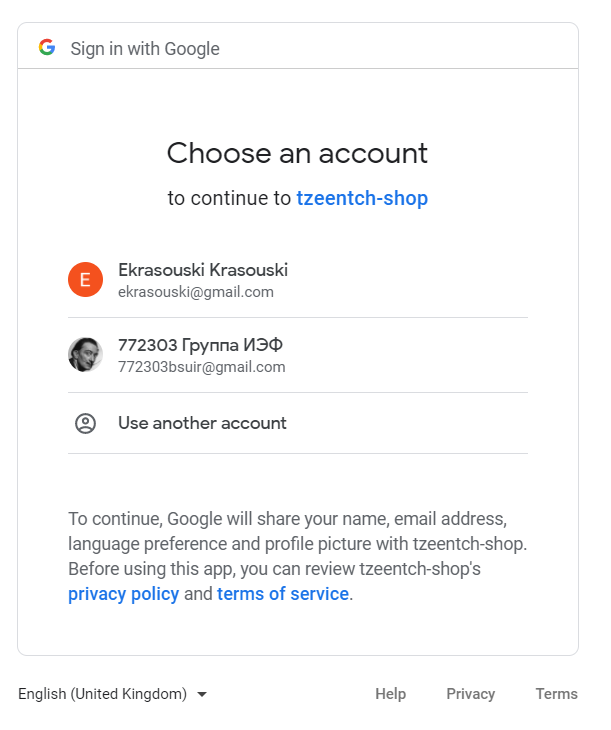


Рисунок 8.3 – Страница авторизации

При входе в систему под ролью администратора у пользователя есть возможность добавить новую книгу в базу. Данная возможность показана на рисунках 8.4 и 8.5.

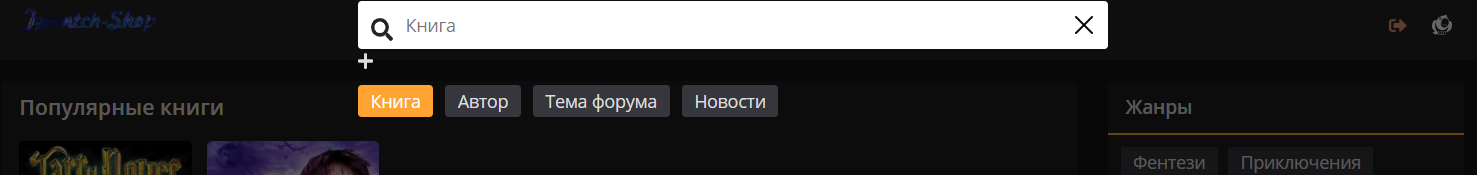


Рисунок 8.4 – Страница поиска и добавления книг администратором

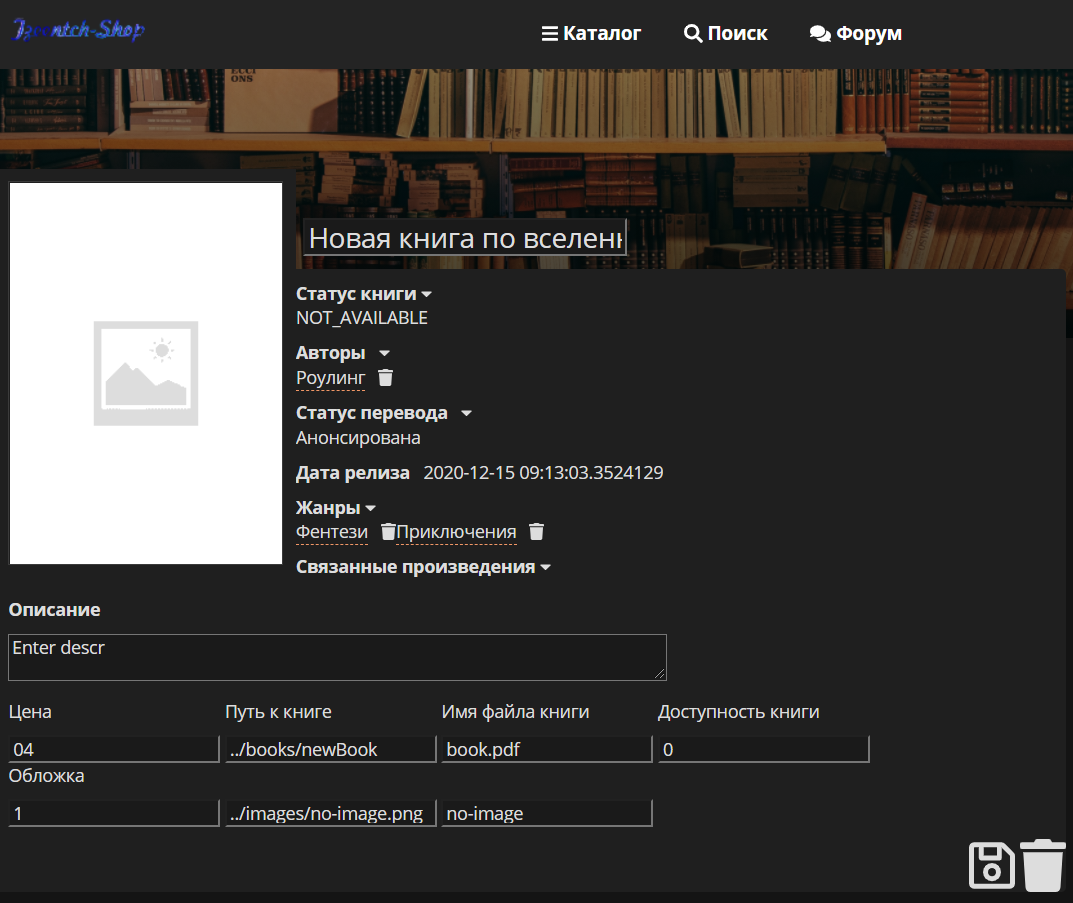


Рисунок 8.5 – Страница добавления книги

Как можно увидеть на рисунках 8.5 у администратора так же есть возможность удаления книг. Так же можно посмотреть все книги в каталоге (рисунок 8.8).

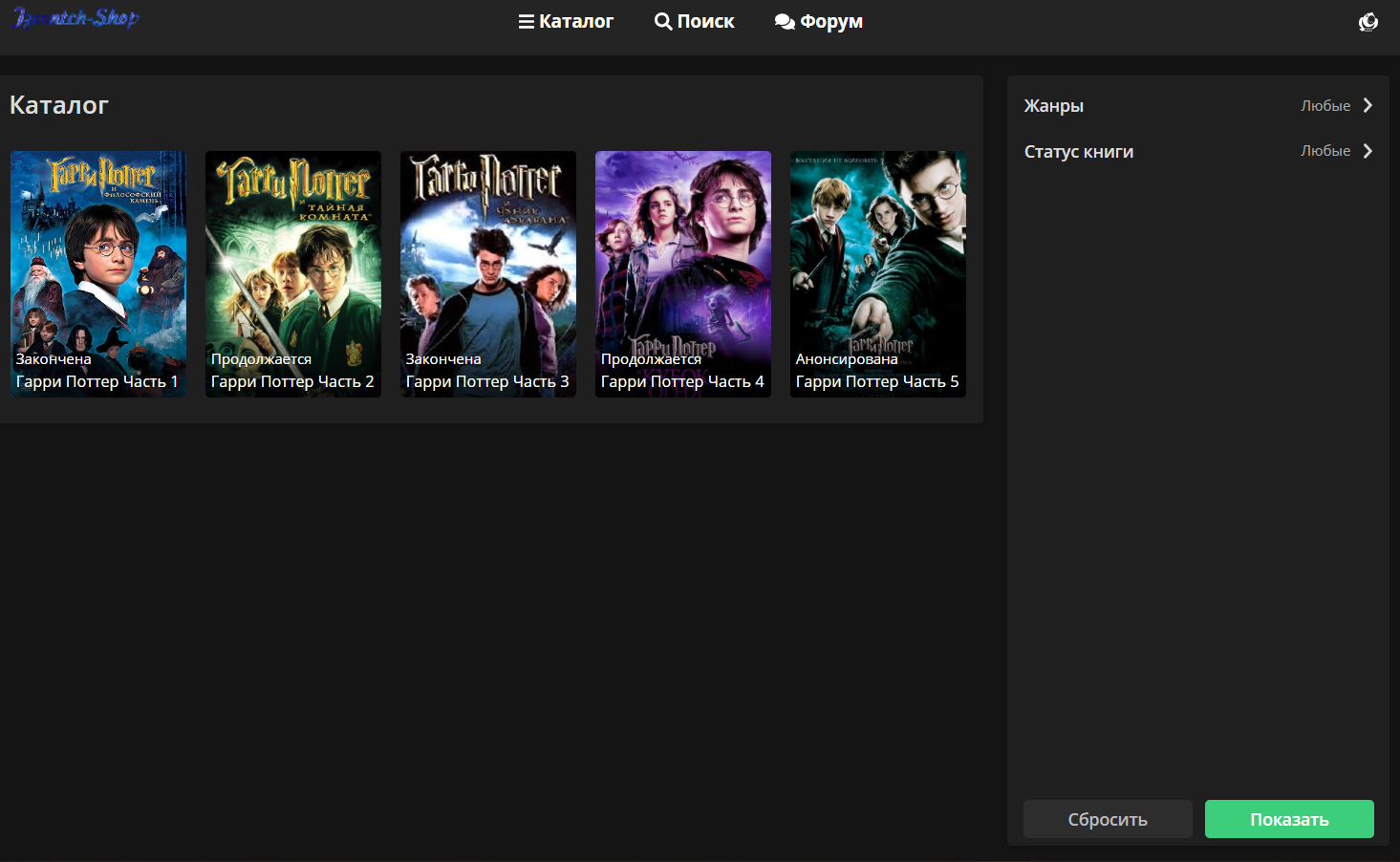


Рисунок 8.6 – Страница просмотра каталога

Еще доступны фильтры, представленные на рисунке 8.7 и 8.8

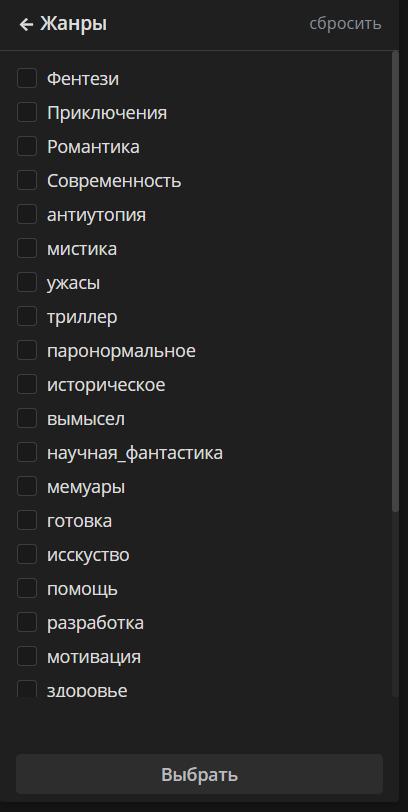


Рисунок 8.7 – Фильтр по жанрам

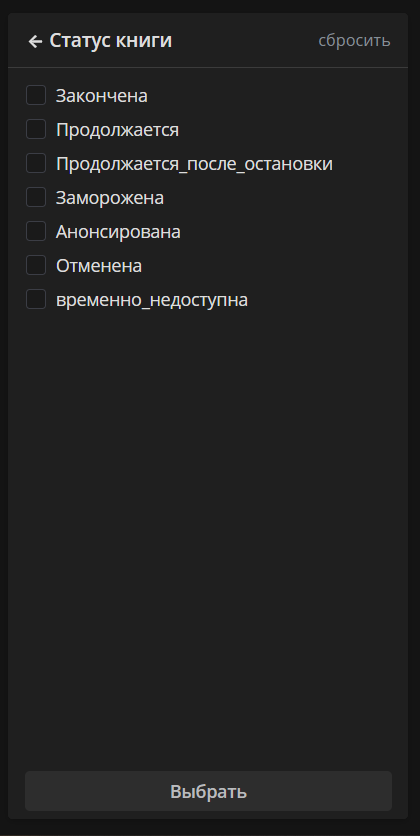


Рисунок 8.8 – Фильтр по статусам книги

У пользователя есть возможность найти интересующую его книгу (не через каталог) (рисунок 8.9), просмотр книги и возможность оставлять комментарии с возможностью сортировки последних, так же можно увидеть людей, который купили данное произведение и связанные произведения (рисунок 8.10).

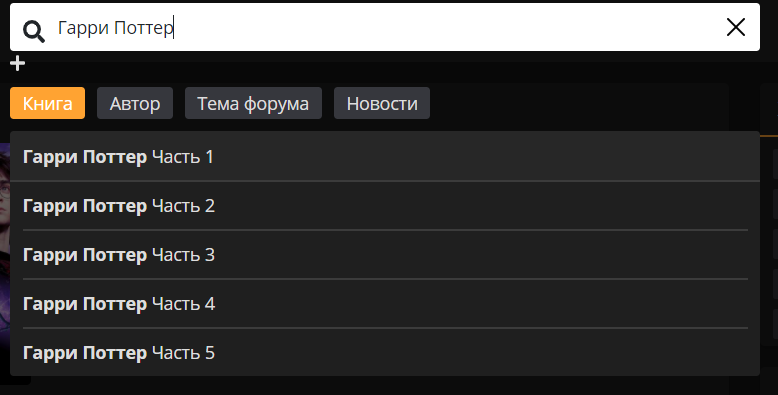


Рисунок 8.9 – Результат поиска книги

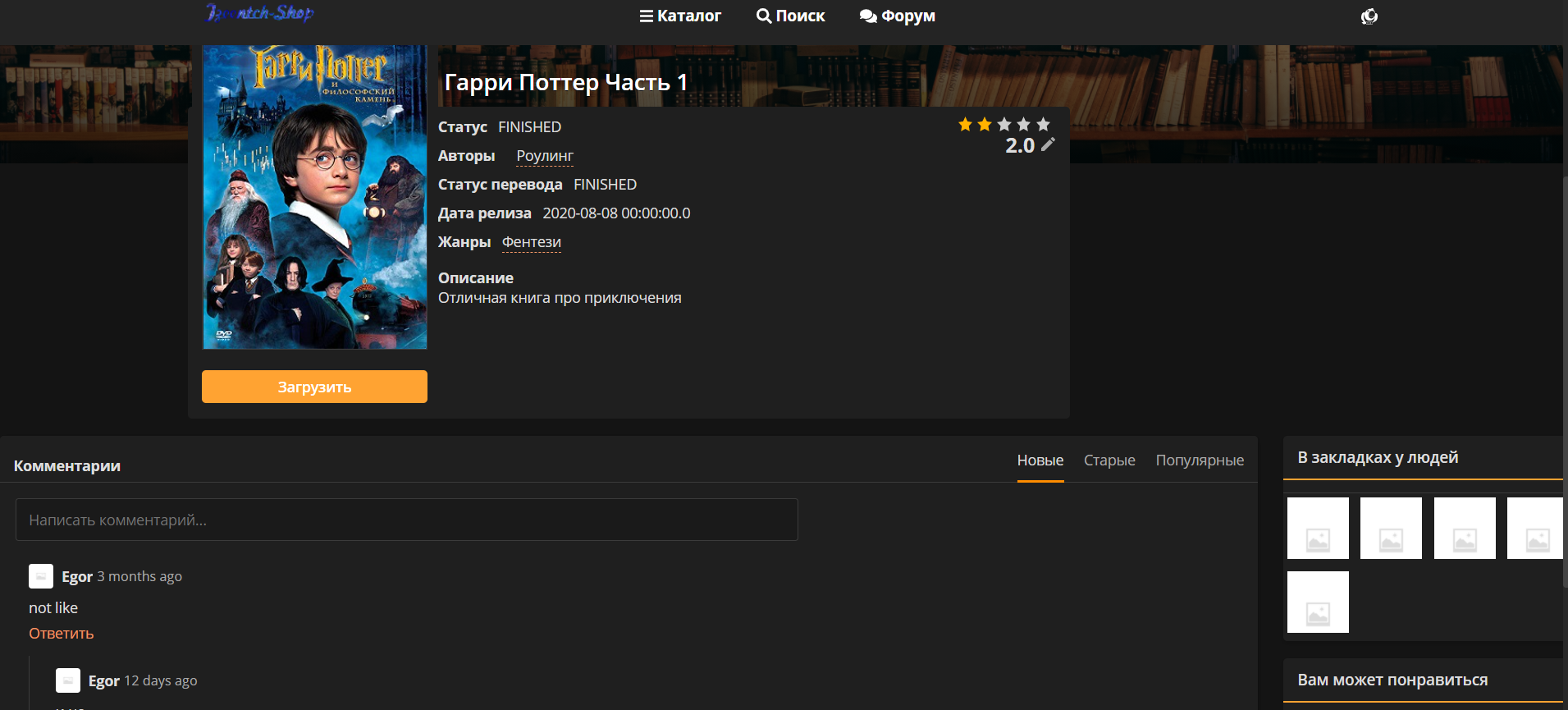


Рисунок 8.10 – Результат поиска книги

Теперь перейдем ко всем новостям, для этого на главной странице просто нажмем на соответствующую ссылку (рисунок 8.11).

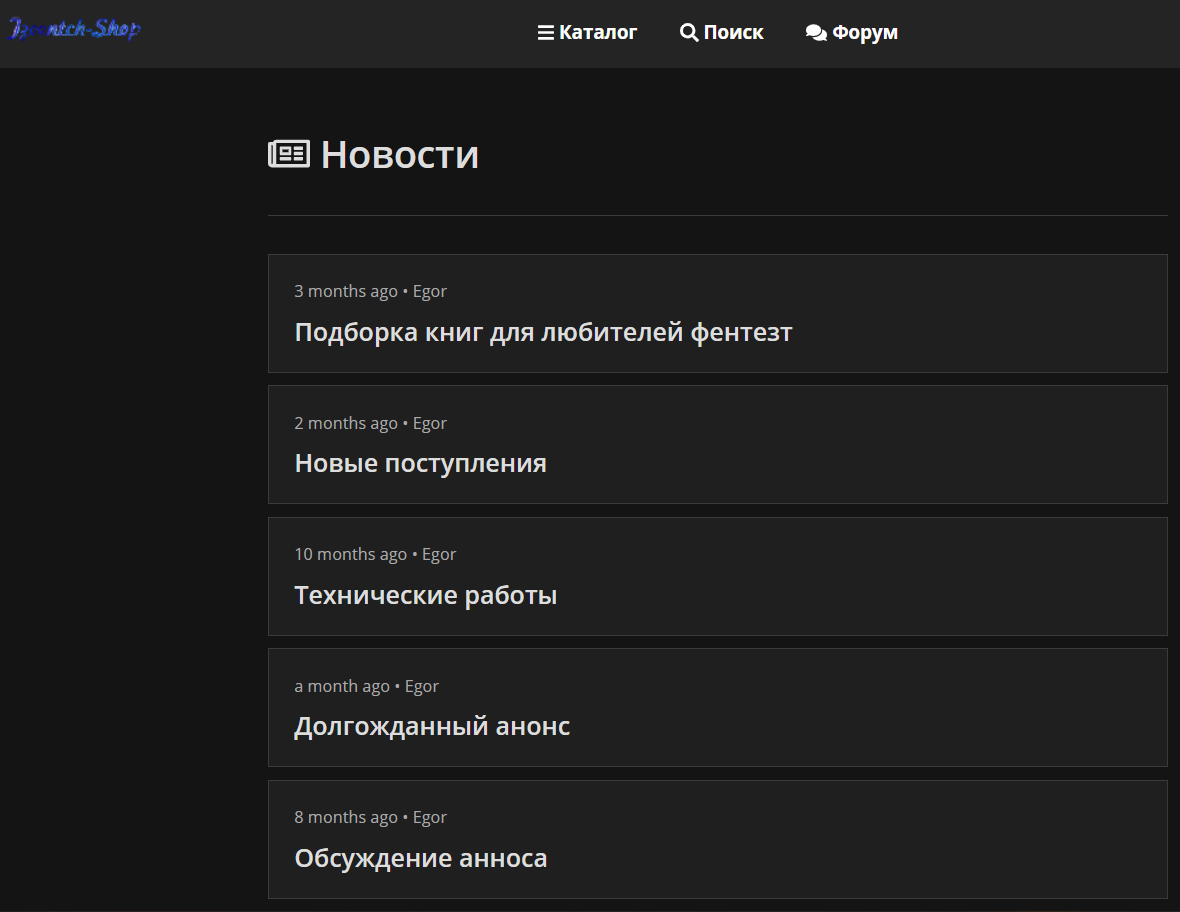


Рисунок 8.11 – Все новости

Далее изображена возможность пользователя по взаимодействию с форумом, на рисунке 8.12, изображен просмотр всех тем, а на рисунке 8.13 уже конкретная тема форума.

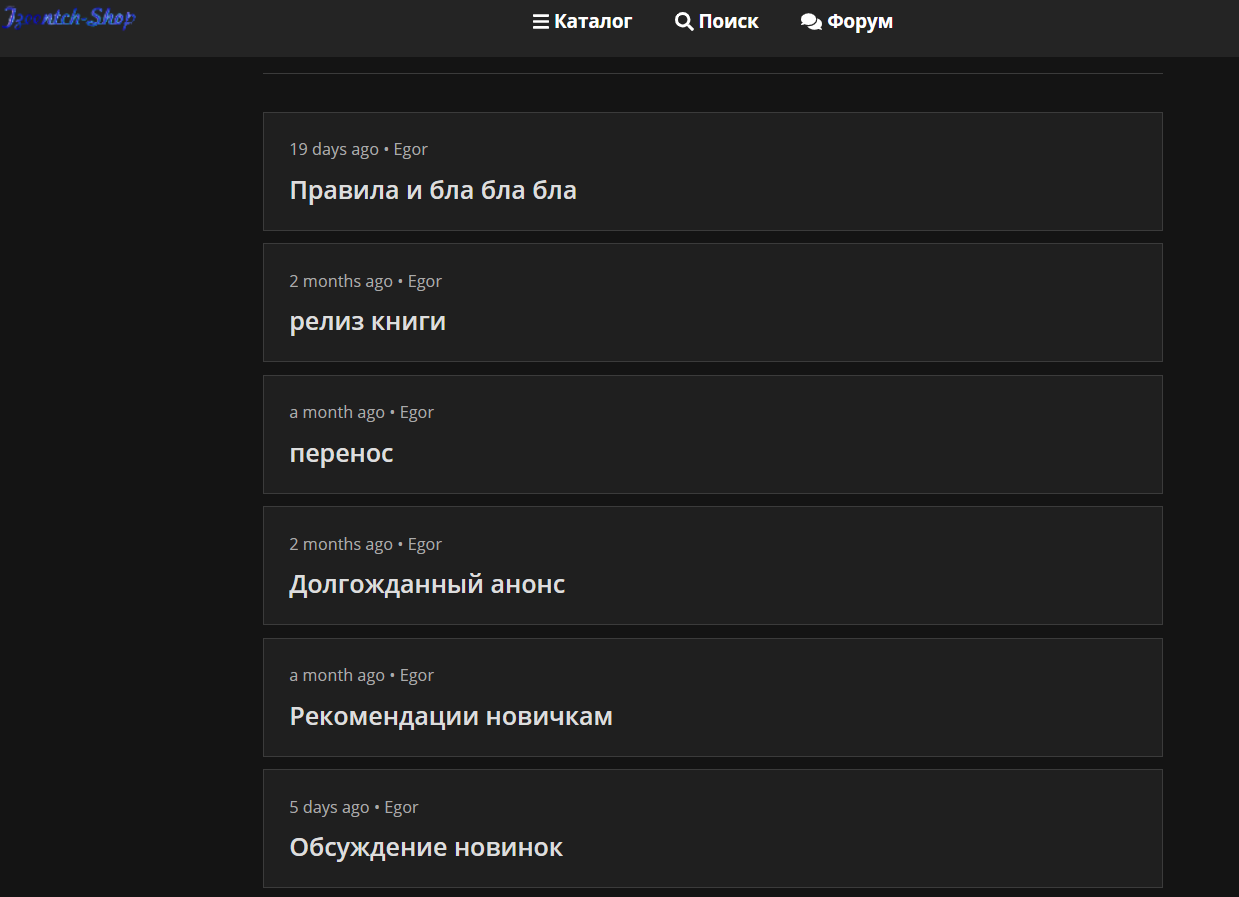


Рисунок 8.12 – Все темы форума

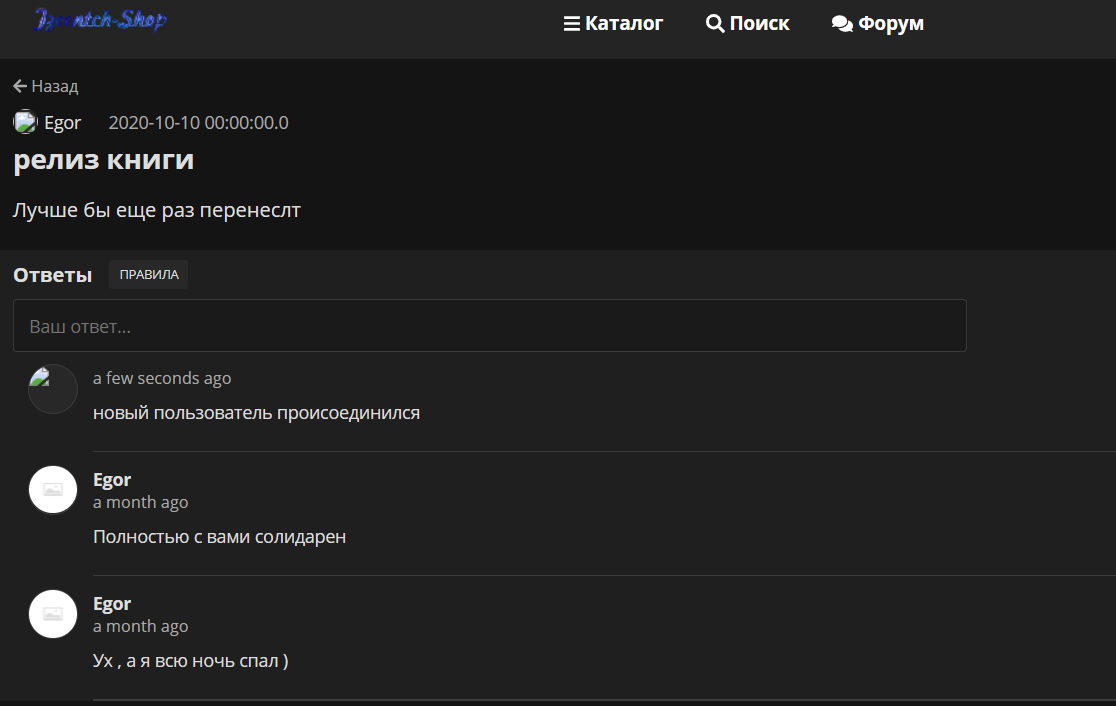


Рисунок 8.13 – Конкретная тема форума

Тестирование приложения производилось в различных браузерах. При нахождении багов, они устранялись, и функционал приложения проверялся снова. Ниже прикреплены скриншоты с описанием того, что на них изображено и, что они демонстрируют.

На рисунке 8.14 отображено сообщение, которое видит пользователь при неправильном вводе логина или пароля.

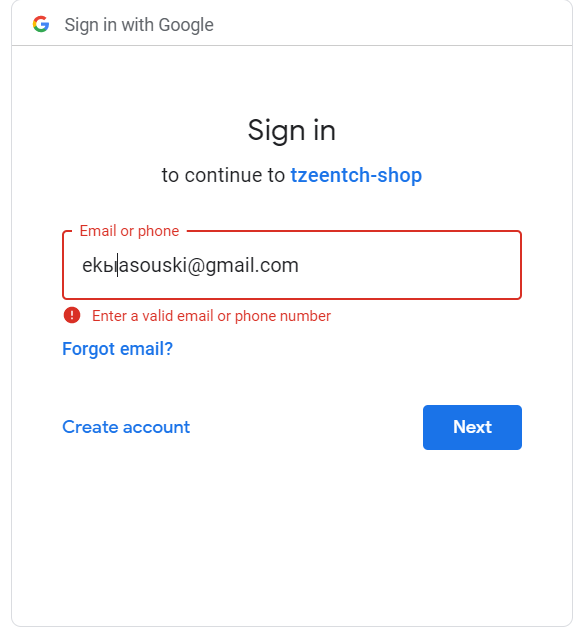


Рисунок 8.14 – Сообщение об ошибке при авторизации

Так же если пользователь ввел некорректные данные для авторизации, то выведется сообщение об ошибке (рисунок 8.17)

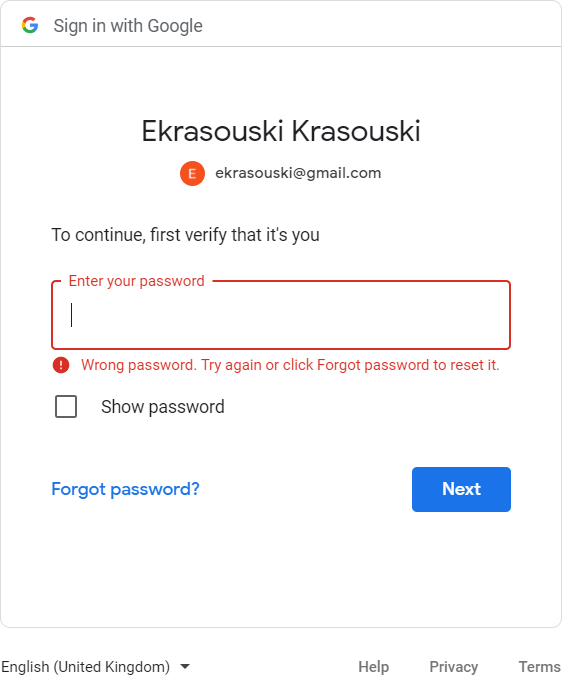


Рисунок 8.17 – Некорректные данные при авторизации

Таким образом, можно сделать вывод: большинство ошибок были продуманы и сделаны предупреждения, что говорит о высокой степени разработки программного приложения. Все эти предупреждения необходимы для того, чтобы пользователь, который впервые зашёл на сайт без труда понимал, что именно ему необходимо исправить при неправильном вводе. Данные подсказки помогают ускорить работу пользователей в поиске, добавлении и просмотре необходимой в данный момент информации.

# Заключение

При выполнении курсовой работы были исследованы процессы работы Web-портал, подбора и реализации книг. Приложение удовлетворяет основным характеристикам, которые были заявлены. Оно удобно в эксплуатации, целостно, конкретизировано в рамках системы по реализации книг.

Были разработаны пользовательские функции приложений: функции поиска, фильтрации и сортировки данных, функции авторизации, редактирования, проверки введенных данных и прочие функции бизнес логики, облегчающие исследованный процесс.

Были разработаны алгоритмы работы следующих функций: главного меню, поиска по заданному критерию и т.д.

Была предусмотрена обработка исключительных ситуаций с помощью функций проверки введенной информации.

Архитектура сервера и клиента выстроена таким образом, чтобы её можно было расширять впоследствии без ошибок и корректировок в уже существующий проект, так называемая масштабируемость. Логическое визуальное представление интерфейса повышает скорость изучения системы для её применения на практике.

Данное программное приложение обеспечивает надежность и защищенность данных, так как все сведения как о пользователях хранятся в базе данных, что не допускает несанкционированного доступа в систему. Были достигнуты основные цели проектирования курсовой работы. Тема проекта реализована в полной мере, в соответствии с функциональными возможностями аналогичных программ.

# Список используемых источников

[1] Stfalcon [Электронный ресурс]: – Электронные данные. – Режим доступа: <https://stfalcon.com/ru/blog/post/why-your-startup-needs-mvp>

[2] Основные понятия баз данных [Электронный ресурс]: – Электронные данные. – Режим доступа: <http://inf.susu.ac.ru/Klinachev/lc_sga_26.htm>.

[3] MySQL [Электронный ресурс]: – Электронные данные. – Режим доступа: <https://www.mysql.com/>

[4] Model-View-Controller – Википедия [Электронный ресурс]: – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/Model-View-Controller>

[5] Интернет-магазины Беларуси: подробная аналитика – E-pepper [Электронный ресурс]: – Электронные данные. – Режим доступа: <https://e-pepper.ru/news/internet-magaziny-belarusi-podrobnaya-analitika.html>

[6] IDEF – Википедия [Электронный ресурс]: – Электронные данные. – Режим доступа: <https://ru.wikipedia.org/wiki/IDEF>

[7] Baeldung [Электронный ресурс]. – Электронные данные. – Режим доступа: <https://www.baeldung.com/>

[8] Методы программирования. И.Н. Блинов, В.С. Романчик Минск: издательство «Четыре четверти», 2013. – 896 с.

[9] Java. Эффективное программирование. Джошуа Блох, издательство: «ДМК», 2018. – 294с.

# ПРИЛОЖЕНИЕ А

# (ОБЯЗАТЕЛЬНОЕ)

# БЛОК-СХЕМА АЛГОРИТМА

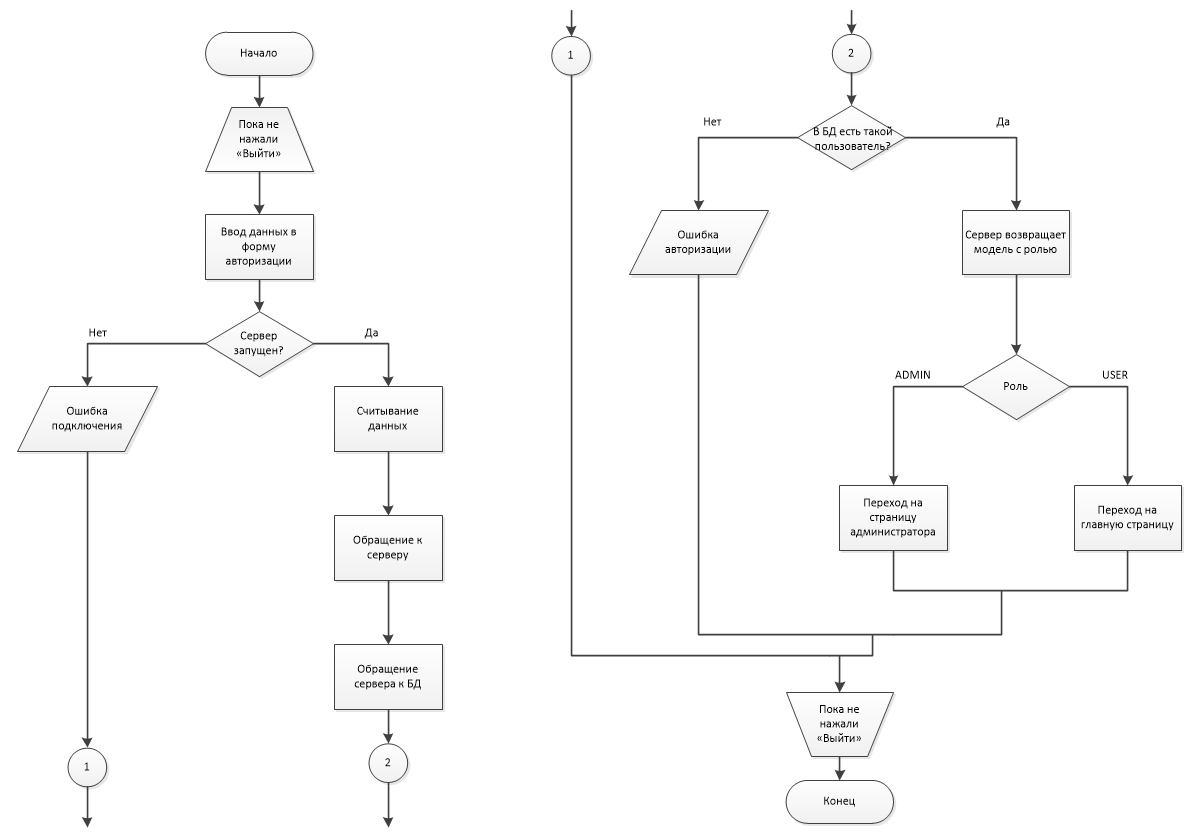


Рисунок A.1 – Схема авторизации в приложении



Рисунок А.2 – Блок-схема фильтрации

# ПРИЛОЖЕНИЕ Б

# (необязательное)

# Листинг кода

// класс книги

@Component  
@Entity  
@Table  
@Data  
@EqualsAndHashCode(of = {"id"})  
@ToString(exclude = {"usersThatBoughtIt", "relatedBooks"})  
@AllArgsConstructor  
@NoArgsConstructor  
public class Book implements Domain {  
 @Id  
 @GeneratedValue(strategy = GenerationType.AUTO)  
 @NotBlank(groups = {Exist.class})  
 @Null(groups = {New.class})  
 private Integer id;  
 private String name;  
 @ManyToOne(cascade = CascadeType.REFRESH, fetch = FetchType.EAGER)  
 private Picture picture;  
 @OnDelete(action = OnDeleteAction.CASCADE)  
 @ManyToMany(cascade = CascadeType.REFRESH, mappedBy = "writtenBooks", fetch = FetchType.EAGER)  
 private Set<Author> author;  
 @NotEmpty(groups = {New.class,Exist.class})  
// @ElementCollection(targetClass = BookStatus.class,fetch = FetchType.EAGER)  
 @CollectionTable(name = "book\_status", joinColumns = @JoinColumn(name = "id"))  
 @Enumerated(EnumType.STRING)  
 private BookStatus bookStatus;  
 @CollectionTable(name = "translation\_status", joinColumns = @JoinColumn(name = "id"))  
 @Enumerated(EnumType.STRING)  
 private TranslationStatus translationStatus;  
 @ElementCollection(targetClass = Genre.class, fetch = FetchType.EAGER)  
 @CollectionTable(name = "book\_genre", joinColumns = @JoinColumn(name = "id"))  
 @Enumerated(EnumType.STRING)  
 private Set<Genre> genres;  
 @ManyToMany(cascade = CascadeType.REFRESH, mappedBy = "books", fetch = FetchType.EAGER)  
 @JsonIgnore  
 private Set<User> usersThatBoughtIt;  
 private String description;  
 @ManyToMany(fetch = FetchType.EAGER)  
 @JoinTable(name = "Related\_Book",  
 joinColumns = @JoinColumn(name = "book\_id", referencedColumnName = "ID"),  
 inverseJoinColumns = @JoinColumn(name = "related\_Books\_id", referencedColumnName = "ID"))  
 private Set<Book> relatedBooks;  
  
 private Integer availability;  
 @Transient  
 private Double rating;  
  
 private Integer popularity;  
  
 private BigDecimal price;  
  
 private String url;  
 private String fileName;  
  
 @PastOrPresent(groups = {New.class, Exist.class})  
 private Timestamp printTime;  
  
 @Override  
 public Integer getId() {  
 return id;  
 }  
  
 public Double getRating() {  
 return ratingParamService.getAverageRating(this.id);  
 }  
  
 public void setRating(Double rating) {  
 this.rating = getRating();  
 }  
 @Transient  
 private static RatingParamService ratingParamService;  
  
 @Autowired  
 public void setRatingParamService(RatingParamService ratingParamService) {  
 Book.ratingParamService = ratingParamService;  
 }  
  
 @PostConstruct  
 public void init() {  
 System.out.println(Book.ratingParamService + "was init");  
 }  
}

//общий интерфейс классов сервисов  
public interface Service<T>{  
  
 Integer saveAndReturnId(T note);  
  
 public T save(T note);  
  
 public void delete(T note);  
  
 public void delete(int id);  
  
 public Optional<T> findById(Integer id);  
  
 public List<T> findAll(String property);  
  
 public List<T> findAll();  
  
 public List<T> search(List<SearchCriteria> params, Class<T> tClass);  
  
 public List<T> search(SearchCriteria[] params, Class<T> tClass);  
}

// контроллер книг

@Controller  
@RequestMapping("book")  
public class BookController extends CommonController<Book, BookDTO> {  
 @Autowired  
 private ServletContext servletContext;  
 private UserCreditService userCreditService;  
 private UserService userService;  
 private AuthorService authorService;  
  
 public BookController(BookService bookService, UserCreditService userCreditService, UserService userService, AuthorService authorService) {  
 this.service = bookService;  
 this.page = "book";  
 this.userCreditService = userCreditService;  
 this.userService = userService;  
 this.authorService = authorService;  
 }  
  
 @Override  
 public String getOneById(@RequestParam(name = "id") Integer id, Map<String, Object> model) throws IOException {  
 try {  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 User user = (User) auth.getPrincipal();  
 boolean isFound = false;  
 for (Book book : user.getBooks()) {  
 if (book.getId().equals(id)) {  
 isFound = true;  
 break;  
 }  
 }  
 model.put("Bought", isFound);  
 model.put("CanEdit", user.getAuthorities().contains(Role.ADMIN));  
 } catch (ClassCastException ex) {  
 model.put("Bought", false);  
 model.put("CanEdit", false);  
 }  
 Optional<Book> book = service.findById(id);  
 model.put("Book", book.orElse(new Book()));  
 if (book.isPresent() && book.get().getRelatedBooks() != null) {  
 model.put("RelatedBooks", JsonParser.mapToJson(book.get().getRelatedBooks()));  
 } else model.put("RelatedBooks", JsonParser.mapToJson(new LinkedList<Book>()));  
 if (book.isPresent() && book.get().getGenres() != null) {  
 model.put("Genres", JsonParser.mapToJson(book.get().getGenres()));  
 }  
 if (book.isPresent() && book.get().getAuthor() != null) {  
 model.put("Authors", JsonParser.mapToJson(book.get().getAuthor()));  
 }  
 try {  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 User user = (User) auth.getPrincipal();  
 if (user.getId() != null) {  
 model.put("login", true);  
 }  
 } catch (ClassCastException ex) {  
 model.put("login", false);  
 }  
  
 return page;  
 }  
  
 @Override  
 @PreAuthorize("hasAuthority('ADMIN')")  
 @GetMapping(path = "/change", consumes = MediaType.APPLICATION\_JSON\_VALUE)  
 public String changePage(@RequestParam(name = "id") Integer id, Map<String, Object> model) throws IOException {  
 Object book = fillDomain(model, id);  
 model.put("Authors", JsonParser.mapToJson(((Book) book).getAuthor()));  
 model.put("Genres", JsonParser.mapToJson(((Book) book).getGenres()));  
 model.put("AllGenre", JsonParser.mapToJson(Genre.values()));  
 model.put("AllBookStatus", JsonParser.mapToJson(BookStatus.values()));  
 model.put("AllTranslationStatus", JsonParser.mapToJson(TranslationStatus.values()));  
 model.put("RelatedBook", JsonParser.mapToJson(((Book) book).getRelatedBooks()));  
 model.put("AllBook", JsonParser.mapToJson(service.findAll()));  
 model.put("AllAuthors", JsonParser.mapToJson(authorService.findAll()));  
 try {  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 User user = (User) auth.getPrincipal();  
 if (user.getId() != null) {  
 model.put("login", true);  
 }  
 } catch (ClassCastException ex) {  
 model.put("login", false);  
 }  
 return page + "\_edit";  
 }  
  
 @GetMapping(path = "/catalog")  
 public String getAllPage(Map<String, Object> model,  
 @RequestParam(name = "genre", required = false) String genreString,  
 @RequestParam(name = "bookStatus", required = false) BookStatus bookStatus  
 ) throws IOException {  
  
 List<Book> nodes = new LinkedList<>();  
  
 if (genreString != null || bookStatus != null) {  
 final List<SearchCriteria> params = new ArrayList<>();  
 if (genreString != null) {  
 for (final Genre genre : Genre.values()) {  
 if (genre.getGenre().equals(genreString)) {  
 nodes = new LinkedList<>(((BookService) service).findAllByGenresContains(genre).orElse(new LinkedList<>()));  
 }  
 }  
 }  
 if (bookStatus != null) {  
 nodes = new LinkedList<>(((BookService) service).findAllByBookStatus(bookStatus));  
 }  
  
 } else {  
 nodes = service.findAll();  
  
 }  
 model.put("AllGenre", JsonParser.mapToJson(Genre.values()));  
 model.put("AllBookStatus", JsonParser.mapToJson(BookStatus.values()));  
 model.put("Books", JsonParser.mapToJson(nodes));  
 try {  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 User user = (User) auth.getPrincipal();  
 if (user.getId() != null) {  
 model.put("login", true);  
 }  
 } catch (ClassCastException ex) {  
 model.put("login", false);  
 }  
 return "All" + page;  
 }  
  
 @RequestMapping(path = "/download", method = RequestMethod.GET)  
 @PreAuthorize("hasAuthority('USER')")  
 public ResponseEntity<InputStreamResource> downloadFile1(  
 @RequestParam(value = "id") Integer id) throws IOException {  
  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 User user = (User) auth.getPrincipal();  
 UserCreditDetails creditDetails = userCreditService.findById(user.getId()).orElseThrow(NotFoundException::new);  
 user = userService.findById(creditDetails.getId()).orElseThrow(NotFoundException::new);  
 Book book = service.findById(id).orElseThrow(NotFoundException::new);  
 if (book.getAvailability().equals(0) || book.getBookStatus().name().equals(BookStatus.NOT\_AVAILABLE.name())) {  
 throw new BookNotAvailableException();  
 }  
 if (!user.getBooks().contains(book)) {  
 if (creditDetails.getSumOfMoney().subtract(book.getPrice()).longValueExact() < 0) {  
 throw new NotEnoughMoneyException();  
 }  
 creditDetails.setSumOfMoney(creditDetails.getSumOfMoney().subtract(book.getPrice()));  
 user.getBooks().add(book);  
 userService.save(user);  
 }  
  
 File file = new File(book.getUrl());  
 MediaType mediaType = MediaTypeUtils.getMediaTypeForFileName(this.servletContext, file.getName());  
 InputStreamResource resource = new InputStreamResource(new FileInputStream(file));  
 return ResponseEntity.ok()  
 // Content-Disposition  
 .header(HttpHeaders.CONTENT\_DISPOSITION, "attachment;filename=" + file.getName())  
 // Content-Type  
 .contentType(mediaType)  
 // Contet-Length  
 .contentLength(file.length()) //  
 .body(resource);  
  
 }  
  
 @Override  
 @PreAuthorize("hasAuthority('ADMIN')")  
 @GetMapping("/create")  
 public String createPage(@RequestParam(name = "name") String name, Map<String, Object> model)  
 throws IOException {  
 name = name.substring(0, 1).toUpperCase() + name.substring(1);  
 Picture picture = new Picture();  
 picture.setUrl("../images/no-image.png");  
 picture.setName("Enter name");  
 picture.setId(-1);  
 Book book = new Book();  
 book.setId(0);  
 book.setName("Enter name");  
 book.setPopularity(0);  
 book.setRating(0.0);  
 book.setDescription("Enter descr");  
 book.setPicture(picture);  
 book.setAuthor(new HashSet<>());  
 book.setRelatedBooks(new HashSet<>());  
 book.setPrintTime(Timestamp.valueOf(LocalDateTime.now()));  
 book.setBookStatus(BookStatus.NOT\_AVAILABLE);  
 book.setTranslationStatus(TranslationStatus.NOT\_SET);  
 book.setPrice(BigDecimal.ZERO);  
 book.setUsersThatBoughtIt(new HashSet<>());  
 book.setGenres(new HashSet<>());  
 book.setFileName("Enter file name");  
 book.setUrl("Enter Url");  
 book.setAvailability(0);  
 model.put(name, book);  
  
 model.put("Authors", JsonParser.mapToJson(book.getAuthor()));  
 model.put("Genres", JsonParser.mapToJson(book.getGenres()));  
 model.put("AllGenre", JsonParser.mapToJson(Genre.values()));  
 model.put("AllBookStatus", JsonParser.mapToJson(BookStatus.values()));  
 model.put("AllTranslationStatus", JsonParser.mapToJson(TranslationStatus.values()));  
 model.put("RelatedBook", JsonParser.mapToJson(book.getRelatedBooks()));  
 model.put("AllBook", JsonParser.mapToJson(service.findAll()));  
 model.put("AllAuthors", JsonParser.mapToJson(authorService.findAll()));  
 try {  
 Authentication auth = SecurityContextHolder.getContext().getAuthentication();  
 User user = (User) auth.getPrincipal();  
 if (user.getId() != null) {  
 model.put("login", true);  
 }  
 } catch (ClassCastException ex) {  
 model.put("login", false);  
 }  
 return page + "\_edit";  
 }  
  
  
}

// общий rest-controller

public class CommonRestController<T extends Domain, D extends DTO<T>> {  
 protected String page;  
  
 protected Service<T> service;  
  
  
 @GetMapping()  
 public String getOneById(@RequestParam(name = "id" , required = false ) Integer id , Map<String,Object> model) {  
 Object data = find(id);  
 model.put(data.getClass().getSimpleName(),data);  
 return page;  
 }  
  
  
 // @PostMapping(consumes = MediaType.APPLICATION\_JSON\_VALUE,produces = MediaType.APPLICATION\_JSON\_VALUE)  
 @PreAuthorize("hasAuthority('ADMIN')")  
 @PostMapping()  
 public T create(@Validated(New.class) @RequestBody D dto) throws IOException {  
 return service.save(dto.fromDTO());  
 }  
  
 @PutMapping(consumes = MediaType.APPLICATION\_JSON\_VALUE, produces = MediaType.APPLICATION\_JSON\_VALUE)  
 @PreAuthorize("hasAuthority('ADMIN')")  
 public T update(@Validated(Exist.class) @RequestBody D dto, @RequestParam(name = "id") Integer id) {  
 if (id != null && id > 0) {  
 find(id);  
 }  
 return service.save(dto.fromDTO());  
 }  
  
 @DeleteMapping(consumes = MediaType.APPLICATION\_JSON\_VALUE, produces = MediaType.APPLICATION\_JSON\_VALUE)  
 @PreAuthorize("hasAuthority('ADMIN')")  
 public String delete(@RequestParam(name = "id") Integer id) {  
 service.delete(id);  
 return "index";  
 }  
  
 protected T find(@PathVariable Integer id) {  
 return service.findById(id).orElseThrow(NotFoundException::new);  
 }  
  
 @ResponseStatus(HttpStatus.BAD\_REQUEST)  
 @ExceptionHandler(MethodArgumentNotValidException.class)  
 public Map<String, String> handleValidationExceptions(  
 MethodArgumentNotValidException ex) {  
 Map<String, String> errors = new HashMap<>();  
 ex.getBindingResult().getAllErrors().forEach((error) -> {  
 String fieldName = ((FieldError) error).getField();  
 String errorMessage = error.getDefaultMessage();  
 errors.put(fieldName, errorMessage);  
 });  
 errors.forEach((s, s2) -> System.out.println(s2));  
 return errors;  
 }  
}